Swarm Intelligence manuscript No.
(will be inserted by the editor)

An Advanced ACO Algorithm Implementing Boundary Search for Constrained Numerical Optimization Problems

Guillermo Leguizamón[1] **and Carlos A. Coello Coello**[2]

[1] LIDIC - Universidad Nacional de San Luis

Ejército de los Andes 950

San Luis, ARGENTINA 5700

legui@unsl.edu.ar

[2] Evolutionary Computation Group (EVOCINV)

Computer Science Department

CINVESTAV-IPN

Av. IPN No. 2508. Col. San Pedro Zacatenco

México D.F. 07300, MÉXICO

ccoello@cs.cinvestav.mx

Abstract The boundary approach as a constraint-handling technique that can be considered a suitable alternative when facing constrained numerical optimization problems with active constraints. The definition of *ad hoc* or more general boundary operators is relevant in the area of numerical optimization for approaching the region between the feasible and infeasible search space. Although the success of the boundary approach will mainly depend on the solutions representation and the respective exploration operators, it is also an important issue the provided search engine for applying the boundary approach. In this paper we proposed an advanced search engine implementing the boundary approach based on a new Ant Colony Optimization algorithm for continuous problem ($ACO_{\mathbb{R}}$). The paper describes the adaptation of $ACO_{\mathbb{R}}$ to incorporate the boundary approach for constrained numerical optimization problems and includes an experimental study to determine the impact of the parameter setting on the behavior of the proposed ACO algorithm for constrained optimization problems. In addition, the performance of the modified $ACO_{\mathbb{R}}$ is compared against a former and a simpler

version of an ACO algorithm for continues problems, and the Stochastic Ranking, a well known method for constrained evolutionary optimization.

1 Introduction

The Ant Colony Optimization (ACO) metaheuristic has been extensively applied to solving plenty of combinatorial optimization problems. The ACO metaheuristic (Corne et al. [5], Dorigo and Stüztle [6]) includes a class of different algorithms derived from the main concepts involved in it, i.e., algorithms which design is based on the behavior of the ant colonies. These algorithms involve a colony of artificial ants that aim to find good solutions to a problem by cooperating among them. The cooperation is indirectly achieved by *stigmergy*, that is, by indirect communication mediated by the environment. For example, the Ant System (AS) was the first example of an ant colony optimization algorithm to be proposed in the literature. However, AS was not competitive with state-of-the-art algorithms for the TSP, the problem to which the original AS was applied. Accordingly, several improvements were proposed to the original version of AS, many of which were especially designed to deal with the TSP problem. The most important improvements are[1]: AS with an *elitist strategy* for updating the pheromone trail levels, AS$_{rank}$ (a rank-based version of Ant System), $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS), and the Ant Colony System (ACS), however, all of them were originally designed to operate on combinatorial optimization problems (including some dynamic versions) according to the main formulation of the ACO metaheuristic [6].

On the other hand, one of the first ACO extensions to operate on continuous spaces can be found in Bilchev et al. [3] in which the whole search space is discretized in order to represent a finite number of search directions. This approach was validated using a small set of constrained problems. Since then, several other researchers have proposed schemes to apply the ACO algorithm to continuous search spaces. However, all of these approaches only deal with unconstrained optimization problems. For example, Ling et al. [22] report a general proposal to apply an ACO algorithm in a continuous convex domain space without including any experimental results. The proposal involves the application of adaptive crossover and mutation operators based on the relative fitness of the solutions. Lei et al. [19, 20] adapt the ACO metaphor for continuous spaces by dividing the search space into n subregions (i.e., a discrete view as proposed by Bilchev). The approach was applied to a set of uni-

[1] A details description of these instances of the ACO metaheuristic can be found in [6].

variable multimodal functions defined on an unconstrained search space where each subregion corresponds to a subinterval of the variable. Initially, each ant is assigned to the respective search interval. As the search process continues, each ant shifts the middle point of its interval according to the quality of the solution found. In this way, overlapped search regions will arise as the ants focus the search on common promising subregions of the search space. The pheromone trail distribution on interval i is given by a unimodal function T_i (bell shaped) reaching a higher peak as the quality of solution x_i increases. Function T_i represents the learning experience of the algorithm in order to explore/exploit different subregions of the search space. Thus, the decision regarding the interval in which an ant can deposit its pheromone is proportional to the amount of pheromone trail on the respective intervals. Although this work is limited to a few unconstrained continuous problems, it could be an interesting approach to be extended for constrained problems. Finally, it is worth remarking that this algorithm is applied as a complementary step after a genetic algorithm has found some promising subregions of the search space.

More recently, Dreo et al. [7] proposed a new manner of designing ACO algorithms in continuous spaces by introducing the concept of *heterarchy* and communication channels. The approach is tested on only one problem (multimodal unconstrained function) and designed considering that the pheromone trail is not the only way of applying indirect communication among the ants. Instead, they apply the concept of dense heterarchy as a manner of explaining the behavior of some insect species for which the communication is achieved through either indirect or direct communication channels with well-defined properties. The prominent characteristic of this approach is represented by the possibility of using two complementary communication channels, either indirect channels[2] to promote exploration or direct channels to promote exploitation of the search space according to the solutions previously evaluated. Using a similar approach, Monmarché et al. [24] presented an ACO algorithm called API which implements a parallel search in the solution space by conforming hunter sites (points in the search space) according to the quality of the solutions. These sites can be moved (translated) during the search process (exploration) by applying local search on the hunter site. The API algorithm was applied with promising results to a set of well known unconstrained continuous functions.

[2] According to the authors, this concept is similar to that used with Particle Swarm Optimization in [14] and to path-relinking [10].

On the other hand, Pourtakdoust et al. [25] propose an extension of an ant colony system to continuous optimization which is purely pheromone based. To explore the search space, the algorithm uses a normal probability distribution to model a relationship among the parameters, the aggregation of ants around the food source (best so far point) and the distance of a particular point from the food source. Thus, the more the distance between the point and the food source, the less the pheromone intensity. The pheromone update is achieved in each iteration by updating the food source and the aggregation factor. In particular, the aggregation factor is obtained considering the overall distance between all the points found and the food source and the corresponding objective values. The experimental study includes the De Jong's standard testbed functions (i.e., unconstrained problems) and an experimental comparison with API [24] and a GA. An extension of the ACO metaheuristic to continuous domains and applied to continuous and mixed discrete-continuous problems is presented by K. Socha [28] which follows the original conception on the ACO approach in regards of the way the solutions are built, i.e., incrementally. The solutions are built by using a probability density distribution (PDF). At step i each ant generates a random number according to a mixture of normal kernels of PDFs $P^i(x_i)$ defined on the interval $a_i \leq x_i \leq b_i$, i.e., a multimodal PDF aimed at considering several subregions of that interval at the same time. The pheromone maintenance is implemented following three alternatives: positive update, negative update, and a variant of a negative update called dissolving instead of the classical evaporation. The experimental study involves a set of continuous unconstrained problems and the results are better than other ACO algorithms and competitive with respect to some other non-ACO algorithms. In another recent work by Socha et al. [29], the former ideas proposed by Socha [28] regarding continuous domains are extensively presented and details concerning implementation issues are given through the $\text{ACO}_{\mathbb{R}}$ algorithm. The experimental study considered a test suite of several unconstrained continuous optimization problems. In addition, an analysis of the behavior of $\text{ACO}_{\mathbb{R}}$ is presented regarding the impact on its performance of its main parameters: q and ξ.

In Leguizamón et al. [18] a new proposal for constraint-handling technique is implemented in an ACO algorithm for continuous problems based in the former works by Bilchev et al. [3]. The Leguizamón et al.'s work introduced a more general boundary approach for solving nonlinear constrained problems which was presented as a possible extension of the ACO algorithms for continuous search spaces. The boundary approach under the ACO metaheuristic showed to be competitive with other state-of-the-art algorithm with respect to nonlinear

problems with active constraints. It is also worth noting that the boundary approach has been studied from the perspective of evolutionary computation. For example in Michalewicz et al. [23] the efficiency of this approach was shown by using two constrained optimization problems: Keane's function (also known as $G02$) [13] and another function with one equality constraint (also known as $G03$). For these cases, it was possible to define *ad hoc* genetic operators that fit perfectly the boundary of the feasible region. However, this sort of approach is impractical in an arbitrary problem with many constraints, and it is therefore necessary to define a more general approach for boundary search which can be as robust as possible to deal with different types of constraints. Similarly, in Schoenauer et al. [27] some evolutionary operators capable of exploring a general surface of dimension $n-1$ (n is the number of variables) for the following three test cases: function $G03$ and two additional functions which represent respectively a constrained versions of the two original (unconstrained) functions proposed by Baluja [1]. On the other hand, Wu et al. [31] proposed a GA for the optimization of a water distribution system, which is a highly constrained optimization problem. The proposed approach co-evolves and self-adapts two penalty factors in order to guide and preserve the search towards the boundary of the feasible search space. However, the Wu et al.'s work does not involve any explicit boundary operator.

The present work is adopts one of the more recent ACO extensions for continuous search spaces and shows how the boundary approach could be included in a more advanced search engine based on the ACO metaheuristic. More specifically we adopted the $\text{ACO}_{\mathbb{R}}$ algorithm proposed by Socha et al.[29].

The remainder of this paper is organized as follows. Section 2 describes the formulation of the general nonlinear optimization problems and some features of these problems that could be exploited when some conditions are met. In addition, a general formulation of the boundary approach (see [18, 17]) is presented. The two ACO algorithms, $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(B)}$ which is the search engine formerly used to study the applicability of our proposed boundary approach, and $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$, the more advanced search engine based in $\text{ACO}_{\mathbb{R}}$; are presented in Section 3. The test problems and experimental results are presented and analyzed in Section 4. Finally, our conclusions and some possible paths for future research are provided in Section 5.

2 The Boundary Search Approach

The general nonlinear programming problem whose aim is to find $\mathbf{x}$ so as to optimize:

$$f(\mathbf{x}) \quad \mathbf{x} = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$$

where $\mathbf{x} \in \mathcal{F} \subset \mathcal{S}$. The set $\mathcal{S} \subset \mathbb{R}^n$ defines the search space and sets $\mathcal{F} \subseteq \mathcal{S}$ and $\mathcal{U} = \mathcal{S} - \mathcal{F}$ define the *feasible* and *infeasible* search spaces, respectively. The search space $\mathcal{S}$ is defined as an *n*-dimensional rectangle in $\mathbb{R}^n$ (domains of variables defined by their lower and upper bounds):

$$l(i) \leq x_i \leq u(i) \text{ for } 1 \leq i \leq n$$

whereas the feasible set $\mathcal{F}$ is defined by the intersection of $\mathcal{S}$ and a set of additional $m \geq 0$ constraints:

$$g_j \leq 0, \text{ for } j = 1, \ldots, q \quad \text{and} \quad h_j = 0 \text{ for } j = q+1, \ldots, m.$$

At any point $\mathbf{x} \in \mathcal{F}$, the constraints g_k that satisfy $g_k(\mathbf{x}) = 0$ are called the active constraints at $\mathbf{x}$. Equality constraints h_j are active at all points of $\mathcal{F}$. It is worth remarking that plenty of problems formulated as above include active constraints at the best known or optimal solutions. For example, for problems with at least one equality constraint h_j, the respective optimal solution will lay on the region defined by $h_j(\mathbf{x}) = 0$. Furthermore, for many problems, the best solutions may lay on the boundary between the feasible and infeasible search space of some inequality constrains, i.e., the region defined by $g_j(\mathbf{x}) = 0$. When those conditions are met for a particular problem, the design of *ad hoc* operators or approaches that explore the search space focusing on the boundary region (according either to the equality and/or inequality constraints) can be a suitable alternative for including in a specific search engine or metaheuristic.

In the following we first explain how the boundary region can be approached given a specific search space; more precisely, the n-dimensional space $\mathbb{R}^n$. Then, we also describe the manner in which this search space can be explored assuming a hypothetical search engine and exploration operators. Afterwards, we present in detail the proposed technique that takes advantage of the boundary approach to explore some specific regions of the boundary of the feasible search space.

2.1 Approaching the boundary

We describe here a general boundary approach (proposed in [18,17]) which is based on the notion that each point $\mathbf{b}$ of the boundary region can be represented by means of two different points $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{x}$ is some feasible point and $\mathbf{y}$ is some infeasible one, i.e., $(\mathbf{x}, \mathbf{y})$ can represent one point lying on the boundary by applying a "binary search" on the straight line connecting the points $\mathbf{x}$ and $\mathbf{y}$ (when considering an equality constraint, $\mathbf{z} \in \mathcal{F}$ iff $h(\mathbf{z}) \leq 0$; otherwise, $\mathbf{z} \in \mathcal{U}$). Figure 1 shows a hypothetical search space including the

feasible (shadowed area) and infeasible regions. We can identify four points lying on the boundary $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$, and $\mathbf{b}_4$ which are respectively obtained from $(\mathbf{x}_1, \mathbf{y}_1)$, $(\mathbf{x}_2, \mathbf{y}_2)$, $(\mathbf{x}_3, \mathbf{y}_3)$, and $(\mathbf{x}_4, \mathbf{y}_4)$.

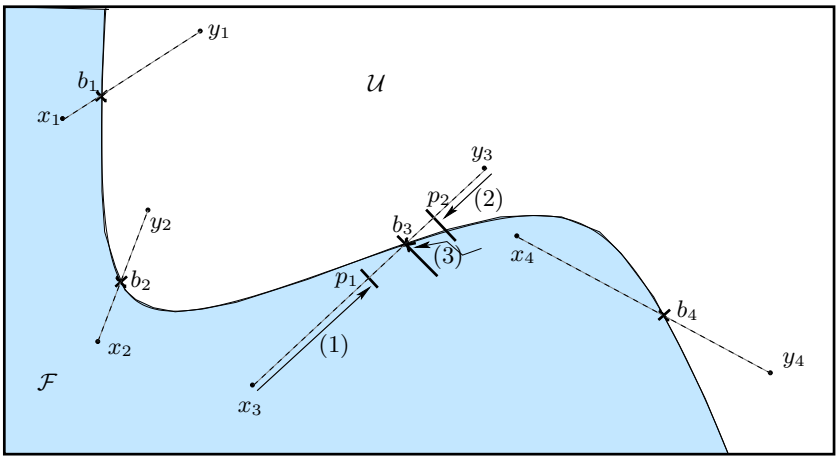


Fig. 1 Given one feasible and one infeasible point, the respective point lying on the boundary can be easily reached by using a simple binary search. In this way, the each point on the boundary can be reached from at least a pair of points $(\mathbf{x}, \mathbf{y})$ with $\mathbf{x} \in \mathcal{F}$ and $\mathbf{y} \in \mathcal{U}$.

The binary search applied to each pair of points $(\mathbf{x}, \mathbf{y})$ is achieved following the steps described in function BS (see Algorithm 1). For example, a possible application of this process can be seen in Figure 1 where we adopt the pair of points $(\mathbf{x}_3, \mathbf{y}_3)$ from which we obtain the point $\mathbf{b}_3$, which lies on the boundary. The first step (labeled (1)) indicates that the first mid point found is infeasible. Consequently, the left side of the straight line ($\mathbf{x}_3$) is moved to point $\mathbf{p}_1$. In the next step (labeled (2)) we consider the points $\mathbf{p}_1$ and $\mathbf{y}_3$ as extreme points for which the mid point is the feasible point $\mathbf{p}_2$. Thus, the new feasible point or right extreme of the line is now the point $\mathbf{p}_2$. Finally, the last point generated is $\mathbf{b}_3$ which can be either lying on or close to the boundary. Condition ((dist_to_boundary($\mathbf{m}$) $\leq \delta$) AND Feasible($\mathbf{m}$)) defines a threshold to stop the process of approaching the boundary. However, the second part of this condition (i.e., "Feasible($\mathbf{m}$)") it is only applied when considering an inequality constraint. In this way, function BS guarantees that $\mathbf{m}$ is in the feasible side regarding the corresponding inequality constraint under consideration. It is worth noticing that parameters $\mathbf{x}$ and $\mathbf{y}$ are local to BS, i.e., function BS behaves as a decoder of the pair of feasible and infeasible points passed as parameters. Therefore, the number of "mid_points_between" $\mathbf{x}$ and $\mathbf{y}$ before approaching the boundary within a

distance less that δ is given by $log_2(r)$ where $r = (dist(\mathbf{x}, \mathbf{y})))/\delta$. Thus, the closer to the boundary, the larger $log_2(r)$.

Algorithm 1 BS(**x**,**y**: real vector): real vector

```
1: m: real vector;
2: repeat
3:    m = mid_point_between(x, y);
4:    if Is_on_Boundary(m) then
5:       return m; { m is a point lying on the boundary }
6:    end if
7:    if Feasible(m) then
8:       x = m;
9:    else
10:      y = m;
11:   end if
12: until (dist_to_boundary(m)≤ δ) AND (Feasible(m));
13: return m; {The closest point to the boundary according to δ }
```

2.2 Exploring the boundary region

So far, we have shown how a point lying on the boundary $\mathbf{b}$ can be represented through a pair of points $(\mathbf{x}, \mathbf{y})$ with $\mathbf{x} \in \mathcal{F}$ and $\mathbf{y} \in \mathcal{U}$. Now we need to consider the exploration of the search space which, according to our proposal, can be defined as $\mathcal{G} = \{(\mathbf{x}, \mathbf{y}) | \mathbf{x} \in \mathcal{F} \subset \mathbb{R}^n \wedge \mathbf{y} \in \mathcal{U} \subset \mathbb{R}^n\}$, that is, the set of pair of points $(\mathbf{x}, \mathbf{y})$ as described above. This space can be considered a *genotype space* as known in the area of evolutionary computation. Since each point from $\mathcal{G}$ represents a point on the boundary, it is necessary the application of the decoder represented by function BS (see Algorithm 1) to obtain the respective *phenotype*, i.e., the "gene expression" of $(\mathbf{x}, \mathbf{y}) \in \mathcal{G}$. Thus, the set $\mathcal{B} = \{\mathbf{b} | \mathbf{b} = BS(\mathbf{x}, \mathbf{y})\}$ is conformed by the set solutions on the boundary. Each solution in this set is evaluated by function ϕ, which represents a measure of solutions quality and gives as result an element of set $\mathbb{E} = \{e \in \mathbb{R} | e = \phi(\mathbf{b})\}$. Figure 2 displays the respective spaces and how they are related with each other by the application of functions BS and ϕ, respectively.

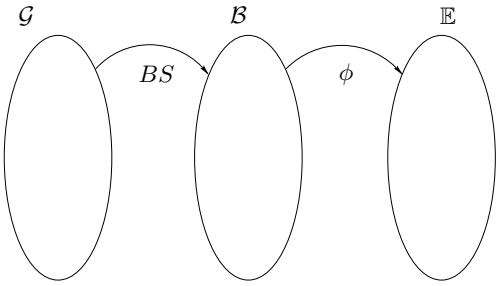


Fig. 2 The search or genotype space ($\mathcal{G}$), phenotype space ($\mathcal{B}$), and space $\mathbb{E}$, and the respective connection through the decoder BS and function evaluation ϕ.

From the above described, is clear that the search engine must deal with the exploration of space $\mathcal{G}$. Figure 3 shows a set of hypothetical points in $\mathcal{G}$, a problem constraint and the respective points on the boundary. In the third pair of points (from left to right) is represented a possible exploration region for $\mathbf{x}_3$ and $\mathbf{y}_3$ (it should be noticed that the shape and size of the exploration area could vary when considering different search engines and/or operators). In this case, the projection of the extreme sides of the exploration areas on the boundary (zig-zag line), represents the covered area on the boundary of points $\mathbf{x}_3$ and $\mathbf{y}_3$ regarding a possible exploration area. For example, from the perspective of evolutionary algorithms, it can be created a population of individu-

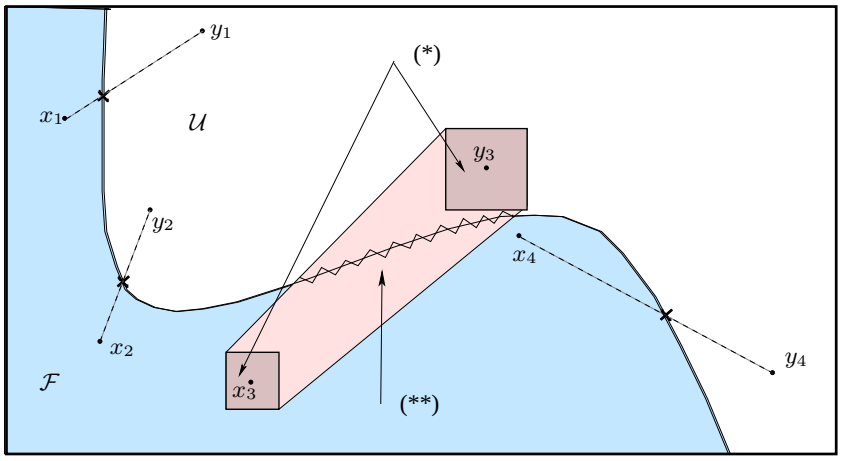


Fig. 3 A set of hypothetical points in $\mathcal{G}$, a problem constraint and the respective points on the boundary where (*) indicates the possible exploration regions for $\mathbf{x}_3$ and $\mathbf{y}_3$ and (**) indicates the respective points on the boundary region based on possible perturbations of $\mathbf{x}_3$ and $\mathbf{y}_3$.

als where each one of then represent an element of set $\mathcal{G}$. Therefore, suitable operators to be chosen could be any qualified crossover and/or mutation operators for floating-point representations. A similar approach can be

adopted if using another search engine suitable for exploring continuous spaces, e.g., particle swarm optimization, differential evolution, immune systems, etc. However, from the perspective of the ACO metaheuristic the possibilities are more limited. In this work we will show at least two alternative for the ACO metaheuristic in the next sections.

2.3 Focusing on the problem constraints

It is important to remember that we are assuming active constraints at the global optimum to proceed with this method where the search is always performed "indirectly" on the boundary of the space defined by some of the problem constraints. The simplest case to apply the boundary approach is when the problem has only one constraint which could be either an equality or an inequality constraint. Let us suppose that the problem includes only one constraint, let us say h, then the search engine should proceed by generating a set of elements of set $\mathcal{G}$. After that, the exploration of $\mathcal{G}$ by the search engine will indirectly and exclusively explore the region defined by $h(x)$, i.e., all solutions generated will be feasible without requiring any *ad-hoc* boundary operator. In section 4 we will show for a test problem with only one constraint ($G25$) a particular distribution of points in space $\mathcal{G}$ and the respective points on the boundary region through the execution of the proposed algorithm in this work.

On the other hand, when facing the typical situation in which we have more than one constraint, it is necessary to define an appropriate policy to explore the boundary as efficiently as possible. One possibility is to explore in turn the boundary of each constraint. The selection of the constraints to search for can be determined using different methods. If the problem includes at least one equality constraint, such equality constraints are the most appropriate candidates to be selected first. However, a possible search engine could keep focused on a particular constraints over the whole run or may be change from one problem constraint to another depending on a particular condition. In our previous work [18] we defined a simple condition based on a parameter called t_c which counts the number of iterations the algorithm focuses in a particular constraint. However, more complex condition could be considered, for example, taking into account the population deversity or the degree in which some problem constraints are being violated. In this work, as will be explained in a further section, we adopted the parameter t_c to control the time when the algorithm should focused on a different problem constraint.

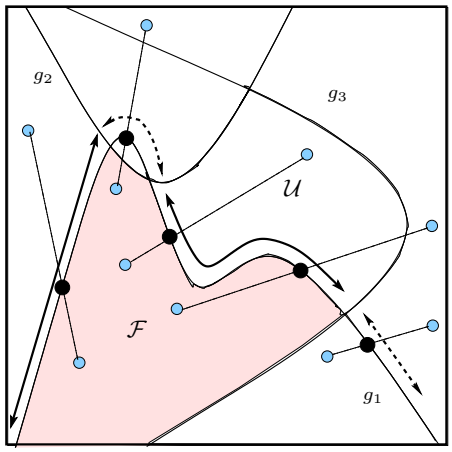


Fig. 4 Feasible search space defined by 3 inequality constraints. The search proceeds on the boundary of constraint g_1.

As an illustrative example, Figure 4 shows a hypothetical search space determined by three inequality constraints. Let's suppose that the search proceeds starting on constraint g_1. If the visited points are on the boundary of $\mathcal{F}$, these points will also satisfy the remaining problem constraints (filled line in Figure 4). However, the exploration of the boundary with respect to constraint g_1 will eventually produce points violating constraints g_2 and g_3 (dotted line in 4). One of the simplest methods to deal with this situation is the application of a penalty function for the infeasible solutions. In addition, if g_1 is active at the global optimum, the method will focus the search on the boundary in order to restrict the explored regions of the whole search space. Note however, that other (more sophisticated) constraint-handling techniques can also be adopted. For example, it could be considered the inclusion of the Stochastic Ranking approach [26] to make the comparisons among the solutions generated[16] and thus avoiding the inclusion and tuning of any penalty factor for solutions evaluation.

3 Boundary Approach in ACO algorithms

In this section we present two ACO algorithms, the $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(B)}$ and $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$ which respectively implement the boundary search approach as explained above. The first algorithm, $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(B)}$ is based on the Bilchev's proposal [2] and was first presented in Leguizamón et al. [18,17]. The second algorithm, called here $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$, is an extended version of $\text{ACO}_{\mathbb{R}}$ (Socha et al. [29]) for dealing with constrained continuous optimization problems. In this case, we present a possible implementation of the boundary approach following the main characteristics of $\text{ACO}_{\mathbb{R}}$ regarding the way in which is explored a continuous search space.

3.1 A summary of the main characteristics of ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$

A possible design to apply the ACO approach in continuous search problems is by discretizing the continuous search space in some way. In ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ we used a discrete structure to represent a set of different points spread on the search space. These points are called *directions*, following Bilchev et al.'s proposal in which the continuous search space is discretized in the so-called search directions. Each one of these search directions was represented through a reference point in the search space. The discrete structure is then related to a trail pheromone structure used in the ant algorithm proposed for representing the desirability of exploring on a particular search direction. For further details see [3]. In ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$, the discrete structure is similar, except for the way in which the directions are represented. Our discrete structure can be seen as a set $\{d_1, d_2, ..., d_k\}$, where k is a parameter for the number of directions. Each direction d_l is represented as a pair of two real n-dimensional vectors, i.e., $d_l = (\mathbf{x}_l, \mathbf{y}_l)$, from which new points are generated by the ants allocated in direction l. As an example, Figure 5 shows $k = 4$ search directions (i.e., the 4 pair of points) and the corresponding 4 points on the boundary which they respectively represent. The 4 points on the boundary are the result of the corresponding application of function BS on the 4 hypothetical directions.

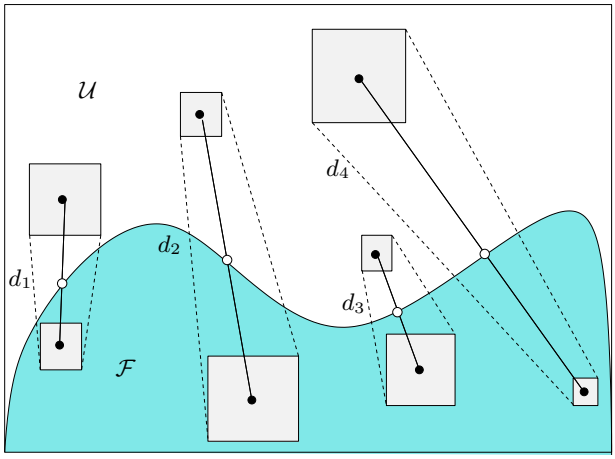


Fig. 5 A 2-dimensional search space with $k = 4$ possible search directions. Notice that each direction include a hypothetical exploration area on $\mathcal{G}$ and the respective covered area on the boundary.

A general outline of the ACO algorithm is shown in Figure 2. It is worth remarking that the original proposal [3] for ACO in continuous domains is used to proceed with the local exploration after a genetic algorithm has finished with the global search. However, ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ is in charge of performing the entire search

process. More precisely, our ACO algorithm starts with a set of k directions $d = (\mathbf{x}, \mathbf{y})$ randomly generated with $\mathbf{x} \in \mathcal{F}$ and $\mathbf{y} \in \mathcal{U}$.

Algorithm 2 Outline of the ACO algorithm based on Bilchev's proposal

1: $t = 0$
2: initialize $A(t)$; evaluate $A(t)$;
3: **while** stop condition not met **do**
4: $t = t + 1$
5: update_trail;
6: reallocate_ants $A(t)$;
7: evaluate $A(t)$;
8: **end while**

The ACO algorithm displayed in Figure 2 works as follows: `initialize A(t)` "distributes" N_a ants on the k directions, where $N_a > k$ in order to allocate one or more ants to the same direction. Each ant allocated in a direction i generates a new solution via the mutation-like operator applied to the pair of points $(\mathbf{x}_l, \mathbf{y}_l)$ representing the initial reference points on direction l; `evaluate A(t)` obtains the objective value for the new points generated; *update_trail* is in charge of accumulating pheromone trial in each direction proportionally to the quality of the objective function values found in the corresponding direction, i.e., $\tau_l = (1 - \rho) \cdot \tau_l + \Delta\tau_l$ where $\Delta\tau_l$ is a value proportional to the best objective value on direction d and $0 \leq \rho \leq 1$ is the pheromone trail evaporation rate; *reallocate_ants A(t)* redistributes the population of ants on the k directions, proportionally to the accumulated pheromone trail values. Thus, the ants on direction $l \in \{1, \ldots, k\}$ are in charge of searching in the neighborhood of the respective boundary feasible point on direction l. The new reference point on direction l for the next iteration is the best solution found in direction l.

The main characteristics of our ACO algorithm include two abstraction levels:

1. *individual search*: involves the strategy followed by each ant to search in its neighborhood. In our case, a mutation-like operator ψ, such as $\psi(\mathbf{x}, \mathbf{y}) = (\mathbf{x}', \mathbf{y}')$ where (the same applies to $\mathbf{y}'$):

$$\mathbf{x}' = (x_1, \ldots, x'_i, \ldots, x_n) \text{ where } i \text{ is a random number from } \{1, \ldots, n\}$$

and,

$$x'_i = \begin{cases} x_i + (u(i) - x_i) \times R & \text{if } r > 0.5 \\ x_i - (x_i - l(i)) \times R & \text{otherwise} \end{cases}$$

where r is a random number in the range $[0..1]$ and $0 \leq R \leq 1$ is considered to define the extent of the search interval with respect to each variable. Parameter R starting at value 1 will vary down to 0 on each iteration as described below.

2. *cooperation*: involves information exchange among the ants in order to guide the search to certain regions of the search space. This information is represented by the pheromone trial structure (τ) where τ_l represents the accumulation of pheromone trail on direction l, i.e., the algorithm's learning experience to be applied to favor the promising regions of the search space. The distribution of the ants on the different directions is achieved by the formula:

$$P_l(t) = \frac{\tau_l(t)}{\sum_{h=1}^{k} \tau_h(t)} \tag{1}$$

The changes on the values of ratio R, involved in our mutation operator, controls the extent of the search interval for each dimension and can be implemented as $\Delta_R(t) = R(1 - r^{(1-t/T_{max})})$ where r is a random number in the range $[0..1]$ and T_{max} is the maximum number of iterations. Consequently, the value $\Delta_R(t)$ falls in the range $[0..R]$ and gets closer to 0 as the elapsed number of iterations t increases.

Finally, it is worth noting the rationale behind the pheromone trail: "the accumulated pheromone trail will decrease on directions that produce low-quality solutions due to the effects of the evaporation process focusing the ants' attention on more promising regions of the feasible search space". In order to avoid premature convergence of the algorithm, a potentially useful direction can remain as an alternative search region by bounding with lower and upper values the amount of pheromone trail in each direction following the principle of the $\mathcal{MM}$AS algorithm.

3.2 The proposed algorithm ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ for Boundary Approach

In this section we describe the design of ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ algorithm which implements the boundary search. The search engine involved in ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ is based on the ACO$_{\mathbb{R}}$ algorithm presented in [29]. Before explaining the implementation of ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$, we first describe briefly the main characteristics of ACO$_{\mathbb{R}}$ as it was proposed and tested in [29] on unconstrained continuous optimization benchmark problems.

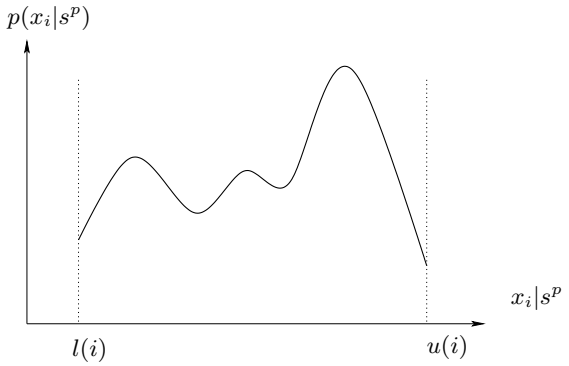


Fig. 6 A continuous probability density function $p(x|s^p)$ where $x_i \in [l(i), u(i)]$, and s^p is a partial solution under construction (see [29] for further details).

Taking into account that the ACO metaheuristic works by incrementally building the solutions according to a biased (by pheromone trail) probabilist choice of solutions components, the ACO$_{\mathbb{R}}$ algorithm was designed aiming at obtaining a set of *probability density functions* (PDFs). Each PDF is obtained from the search experience and is used to incrementally build a solution $\mathbf{x} \in \mathbb{R}^n$ considering in turn each component x_i ($\forall i \ldots n$). Figure 6 represents a hypothetical PDF that could be eventually found during the search. It can be observed a multimodal PDF used to obtain a value for the variable on dimension $i \in \{1, \ldots, n\}$. To approximate a multimodal PDF that looks like the one in Figure 6, Socha et al. [29] proposed a Gaussian Kernel which is defined as a weighted sum of several one-dimensional Gaussian function $g_l^i(x)$ as follows:

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(x-\mu_l^i)^2}{2(\sigma_l^i)^2}} \tag{2}$$

where $i \in \{1, \ldots, n\}$ identifies the number of dimension, i.e., ACO$_{\mathbb{R}}$ uses as many Gaussian kernel PDFs as the number of dimensions of the problem. In addition, G^i is parameterized with three vectors: $\boldsymbol{\omega}$, the vector of weights associated with the individual Gaussian functions; $\boldsymbol{\mu}^i$, the vector of means; and $\boldsymbol{\sigma}^i$, the vector of standard deviations. All these vectors have cardinality k, which constitutes the number of Gaussian functions involved. Figure 7 shows a superposition of three Gaussian function which could approximate the hypothetical multimodal Gaussian function displayed in Figure 6.

In ACO$_{\mathbb{R}}$, a solution archive called T is used to keep track of a number of solutions similarly to the Population Based ACO (PB_ACO) proposed by Guntsch et al. [11]. The cardinality of archive T is k, that is, the number of kernels that conform the Gaussian kernel. For each solution $\mathbf{x}_l \in \mathbb{R}^n$, ACO$_{\mathbb{R}}$ maintains the respective values of each problem dimension, i.e., $x_l^1, \ldots, x_l^n$, and the value of the objective function $f(\mathbf{x}_l)$

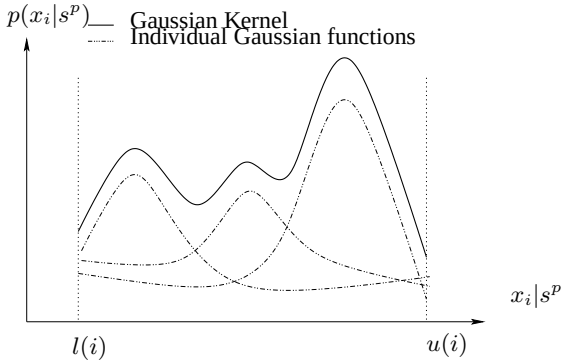


Fig. 7 A possible set of three Gaussian functions to achieve by superposition a Gaussian Kernel which approximate the multimodal Gaussian function as presented in figure 6

which are stored satisfying that $f(\mathbf{x}_1) \leq \ldots \leq f(\mathbf{x}_l) \leq \ldots f(\mathbf{x}_k)$. On the other hand, the vector of weights $\boldsymbol{\omega}$ should satisfy that $\omega_1 \geq \ldots \geq \omega_l \geq \ldots \geq \omega_k$.

The solutions in T are therefore used to dynamically generate probability density functions involved in the Gaussian kernels. More specifically, for obtaining the Gaussian kernel G^i, the three parameters $\boldsymbol{\omega}$, $\boldsymbol{\mu}^i$, and $\boldsymbol{\sigma}^i$ need to be calculated. Thus, for each G^i, the values of the i-th variable of the k solutions in T become part of the elements of vector μ^i, that is, $\boldsymbol{\mu}^i = \{\mu_1^i, \ldots, \mu_n^i\} = \{x_1^i, \ldots, x_n^i\}$. Vector $\boldsymbol{\mu}$ is generated as follows: each solution that is added to the archive T is evaluated and ranked (ties are broken randomly). The solution in T are stored according to their rank, i.e., the highest the rank of the solution, the lowest the respective index in T. The weight ω_l associated to Gaussian function g_l^i is obtained as:

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{3}$$

with mean 1.0 and standard deviation qk, where q is a parameter of ACO$_{\mathbb{R}}$ which controls the preference of the ranked solutions. Thus, when q is small, the best-ranked solution are preferred, otherwise, a large value for q implies a more uniform probability. As mentioned in [29], the influence of this parameter on ACO$_{\mathbb{R}}$ is similar to adjusting the balance between the iteration-best and the best-so-far pheromone updates used in traditional ACO algorithms. On the other hand, each component of the deviation vector $\boldsymbol{\sigma}^i = \{\sigma_1^i, \ldots, \sigma_k^i\}$ is obtained as:

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{|x_e^i - x_l^i|}{k-1} \tag{4}$$

where $l \in \{1, \ldots, k\}$ is the kernel number with respect deviation is calculated and $\xi > 0$ which is he same for all dimensions, has an effect similar to that of the pheromone evaporation rata in ACO. Thus, the higher the value of ξ, the lower the convergence speed of the algorithm.

For obtaining a solution component at step i (in the construction solution process) it is only necessary to calculate the l-th component of σ^i since the sampling process of Gaussian kernel G^i is accomplished as follows. Given the elements of vector $\boldsymbol{\omega}$ calculated as in Eq. 3, the sampling is done in two phases: 1) choose one of the k Gaussian functions of G^i according to the following probability:

$$p_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r}, \tag{5}$$

and, 2) after function g_l^i has been chosen, a sampling is accomplished may be using a random number generator capable of generating random numbers according to a parameterized normal distribution or by using a uniform random generator in conjunction with, for instance, the Box-Muller method [4]. Since at each step only one Gaussian function is used (let us say g_l^i), it is only needed σ_l^i instead the whole vector $\boldsymbol{\sigma}^i$. The pheromone update is achieved by considering a set A of the newly generated solutions[3]. The new T (in the next algorithm iteration) is obtained as $T = rank(T \oplus A)$, i.e., the old solutions in the archive T plus the set of newly created solution A are ranked. In other words, the old solutions compete against the newly generated ones to conform the updated T which maintain its cardinality (k) trough the whole search process.

To adapt $\text{ACO}_{\mathbb{R}}$ to deal with constrained problems by implementing the boundary approach described above is rather straightforward. The proposed algorithm $\text{ACO}_{\mathcal{BR}}^{(S)}$, instead of maintaining one archive T, it maintains two archives for similar purposes, $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ which represent respectively the points on the feasible and infeasible part of space $\mathcal{G}$. A third archive, $T_{\mathcal{B}}$, is also considered which is obtained by applying function BS the each point from $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$. More precisely, $T_{\mathcal{B}} = \{\mathbf{b}_e | \mathbf{b}_e = BS(\mathbf{x}_e, \mathbf{y}_e), e = 1, \ldots, k\}$. Solutions in $T_{\mathcal{B}}$ are evaluated by means of function ϕ. It is worth remarking that solution in $T_{\mathcal{B}}$ are ranked according the solution quality given by ϕ. Taking into account this ranking, the solutions in $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ are then ranked accordingly.

As in the original $\text{ACO}_{\mathbb{R}}$ algorithm, vector $\boldsymbol{\omega}$ is intended for sampling the chosen Gaussian function, however, the situation is different in $\text{ACO}_{\mathcal{BR}}^{(S)}$ since there exist two independent archives $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ from which

[3] Set A represents the set of ants according to Socha et al. [29].

the Gaussian Kernels are built, i.e., to explore the search space $\mathcal{G}$, it is necessary to process both archives from which the solutions on the boundary are obtained. In addition, we define two additional structures $A_{\mathcal{F}}$ and $A_{\mathcal{U}}$ associated respectively to archives $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$. These two structures, similarly as in the original ACO$_{\mathbb{R}}$, represent the newly solutions found according to the Gaussian kernels from $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$. Figure 3.2 represents a general outline of the archives $T_{\mathcal{F}}$, $T_{\mathcal{U}}$, $T_{\mathcal{B}}$, $\boldsymbol{\omega}$, and $\mathbb{E}$. The last one is associated to $T_{\mathcal{B}}$ and maintains the value corresponding to the evaluation quality of solution in $T_{\mathcal{B}}$. It should be notice that $T_{\mathcal{B}}$ is not used to build any Gaussian Kernel, however, the ranking of the solution in it will influence the ranking of solutions in $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$, which clearly influence the generations of new and better quality solutions in the space $\mathcal{G}$.

A general outline of ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ is presented in Algorithm 3 which displays its main components. In line 1, archives $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ are initialized by randomly generating solutions in the feasible and infeasible search space regarding the problem constraint at hand. Similarly, vector $\boldsymbol{\omega}$ is initialized according to Eq. 3 which includes the parameters q and k as explained above. The main loop includes a call to function "Boundary", which is in charge of applying function BS to each pair of points respectively from $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ and returns the archive $T_{\mathcal{B}}$. Then, function "BuildSols" is in charge of generate new solutions through the Gaussian kernel obtained from the respective archives (lines 4 and 5). In order to furtherly obtain $A_{\mathcal{B}}$, i.e., the newly generated solutions on the boundary, function "Boundary" is then applied to $A_{\mathcal{F}}$ and $A_{\mathcal{U}}$. After that, $T_{\mathcal{B}}$ plus $A_{\mathcal{B}}$ are ranked according the solutions quality given by function ϕ, and the best first k solutions in the ranking will be part now contents of archive $T_{\mathcal{B}}$ which is used as reference to get the new $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$. Let say that the new set of point on the boundary is $T_{\mathcal{B}} = \{\boldsymbol{b}_{i_1}, \ldots, \boldsymbol{b}_{i_k}\}$ where $\boldsymbol{b}_{ir}$ comes either from $T_{\mathcal{B}}$ or $A_{\mathcal{B}}$, therefore the new $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ are obtained respectively from $T_{\mathcal{F}} \oplus A_{\mathcal{F}}$ and $T_{\mathcal{U}} \oplus A_{\mathcal{U}}$ taking into account the ranked solutions in the new $T_{\mathcal{B}}$. This is precisely that function "Update" does.

To make the things clearer, let us give a hypothetical example with $k = 4$, and number of ants $N_a = 6$ ($Na = |A_{\mathcal{F}}| = |A_{\mathcal{U}}|$), where

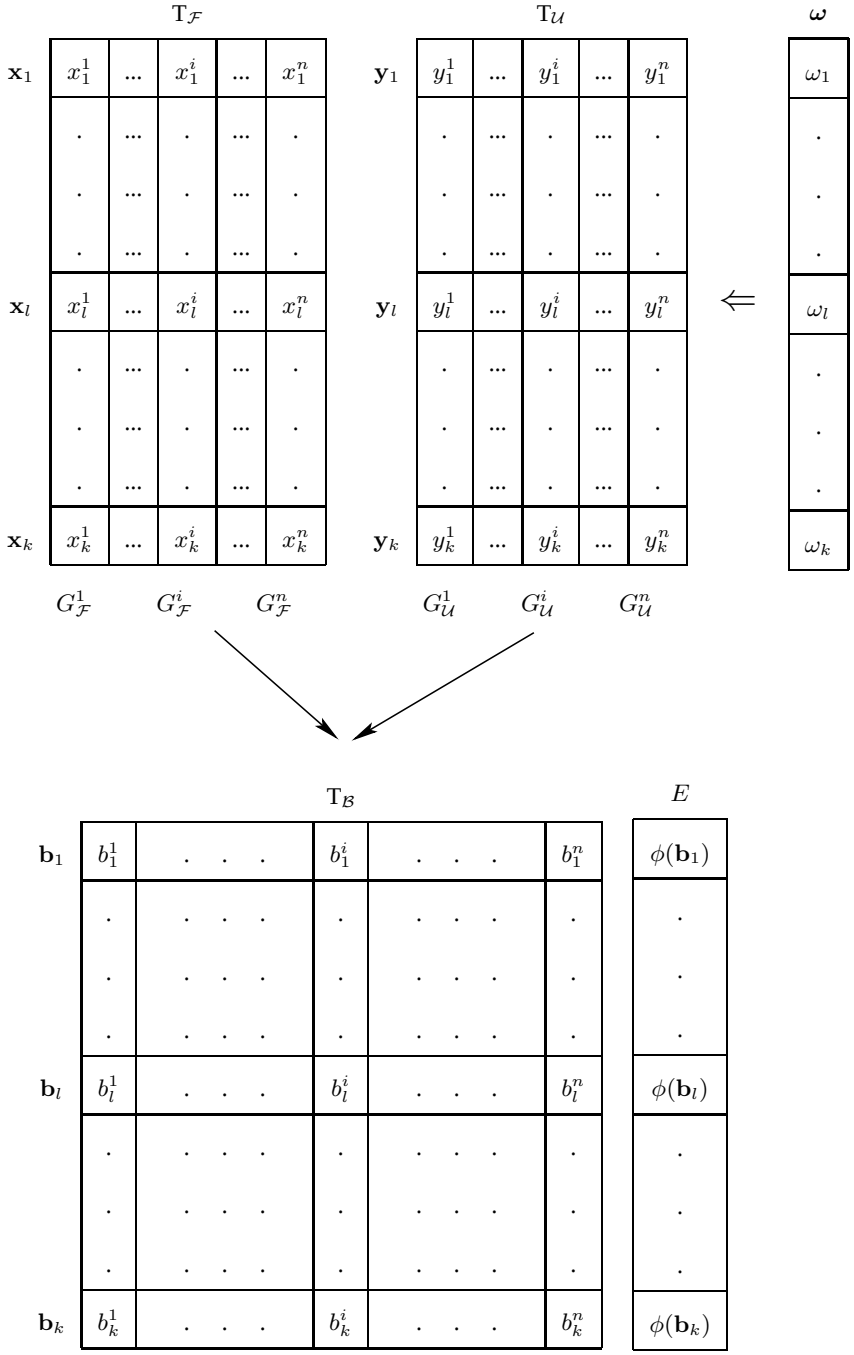
$$T_{\mathcal{F}} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}, T_{\mathcal{U}} = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4\}, \text{ and}$$

$$T_{\mathcal{B}} = \{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4\}, \text{ with } \mathbf{b}_e = BS(\mathbf{x}_e, \mathbf{y}_e), e \in \{1 \ldots k\}$$

are the current archives and respective solutions generated on the boundary, and

$$A_{\mathcal{F}} = \{\mathbf{x}^a_1, \mathbf{x}^a_2, \mathbf{x}^a_3, \mathbf{x}^a_4, \mathbf{x}^a_5\, \mathbf{x}^a_6\}, A_{\mathcal{U}} = \{\mathbf{y}^a_1, \mathbf{y}^a_2, \mathbf{y}^a_3, \mathbf{y}^a_4, \mathbf{y}^a_5\, \mathbf{y}^a_6\}, \text{ and}$$

Table 1 Representation of the $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$ search space divided in feasible and infeasible points



$$A_{\mathcal{B}} = \{\mathbf{b}_1^a, \mathbf{b}_2^a, \mathbf{b}_3^a, \mathbf{b}_4^a, \mathbf{b}_5^a, \mathbf{b}_6^a\}, \text{ with } \mathbf{b}_e^a = BS(\mathbf{x}_e^a, \mathbf{y}_e^a), e \in \{1 \dots N_a\}$$

are the newly solutions found belonging to $\mathcal{F}$ and $\mathcal{U}$ by using the Gaussian kernels obtained from $T_{\mathcal{F}}$ and $T_{\mathcal{F}}$ respectively. In addition, the decoding process applied to those solutions gives the respective set $A_{\mathcal{B}}$ of solutions on on the boundary. It is assumed that $T_{\mathcal{B}}$ and $A_{\mathcal{B}}$ are ranked satisfying that $\phi(\mathbf{b}_1) \leq \phi(\mathbf{b}_2) \leq \phi(\mathbf{b}_3) \leq \phi(\mathbf{b}_4)\}$ and $\phi(\mathbf{b}_1^a) \leq \phi(\mathbf{b}_2^a) \leq \phi(\mathbf{b}_3^a) \leq \phi(\mathbf{b}_4^a) \leq \phi(\mathbf{b}_5^a) \leq \phi(\mathbf{b}_6^a)$. Let us assume in addition

Algorithm 3 A general outline of $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$ algorithm

1: init($T_{\mathcal{F}}$,$T_{\mathcal{U}}$, ω);
2: **for** t in 1 : T_{max} **do**
3: $T_{\mathcal{B}}$ =Boundary($T_{\mathcal{F}}, T_{\mathcal{U}}$)
4: $A_{\mathcal{F}}$ =BuildSols($T_{\mathcal{F}}$);
5: $A_{\mathcal{U}}$ =BuildSols($T_{\mathcal{U}}$);
6: $A_{\mathcal{B}}$ =Boundary($A_{\mathcal{F}}, A_{\mathcal{U}}$)
7: $T_{\mathcal{B}}$ =First$_k$(Sort($T_{\mathcal{B}} \oplus A_{\mathcal{B}}$))
8: Update($T_{\mathcal{F}}$,$T_{\mathcal{U}}$,$\mathbb{E}$); { According to the new $T_{\mathcal{B}}$}
9: **end for**

that $\{\mathbf{b}^a_1, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}^a_2, \mathbf{b}^a_3, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}^a_4, \mathbf{b}^a_5, \mathbf{b}^a_6\}$[4] is the ranked set from $T_{\mathcal{B}} \oplus A_{\mathcal{B}}$. Thus, the first $k = 4$ elements of that set will conform new archive $T_{\mathcal{B}} = \{\mathbf{b}^a_1, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}^a_2\}$. Taking into account the index of the elements in $T_{\mathcal{B}}$, the respective archives that will contain the new points in $\mathcal{F}$ and $\mathcal{U}$ are $T_{\mathcal{F}} = \{\mathbf{x}^a_1, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}^a_2\}$ and $T_{\mathcal{U}} = \{\mathbf{y}^a_1, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}^a_2\}$.

4 Experimental Results

In the first part of this section a preliminary study on the behavior of $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$ is accomplished. The main objective is to analize to influence of an important parameter on its performance, more specifically the parameter q which controls the preference of the ranked solutions. The second part aims at comparing the results of $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$ against $\text{ACO}^{(B)}_{\mathcal{B}\mathbb{R}}$ and SR algorithms which performance was compared in [18,17].

Before presenting the results we will describe some common characteristics of $\text{ACO}^{(B)}_{\mathcal{B}\mathbb{R}}$ and $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$ regarding their application to the different test cases. Indeed, $\text{ACO}^{(B)}_{\mathcal{B}\mathbb{R}}$ and $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$ require minimum changes when applied to the different test cases considered: the objective function, number of variables, range of each variable, and constraints. However, the policy to determine on which constraint the search should focus needs to be considered when a problem has more than one constraint: a) we can focus the search on all the constraints, but considering one constraint in turn by controlling the change through a particular condition (S_{all}), b) similar to the previous alternative but considering only the active constraints (S_{act}), or c) just considering one constraint during the whole run (S_c where $c \in \{1, \ldots, m\}$). These three policy to deal with the way of

[4] Obtained by merging $T_{\mathcal{B}}$ and $A_{\mathcal{B}}$

approaching to the boundary were extensively study in Leguizamón et al. [18, 17] for algorithm ACO$_{\mathcal{B}\mathbb{R}}^{(B)}$. From these earlier results, we adopt the so called policy S$_{act}$, which showed the best performance through all the test cases studied. However, the other policy are also a valuable and efficient alternative when no information is available with respect to the possible active constraints.

In our experiments, the condition to produce a change on the search from one constraint to another is given by an elapsed number of iterations and it is represented by the parameter t_c as explained in section 2.3. In addition, for problems with more than one constraint, we incorporate a penalty function of the form:

$$\phi(x,\mu) = f(x) + \mu(t)(\sum_{j=1}^{q} \max\{0, g_j(x)\} + \sum_{j=q+1}^{m} |h_j(x)|) \tag{6}$$

where $\mu(t)$ is a dynamic penalty factor which could change as t, the elapsed iteration, increases with $\mu(0) \leq \mu(1) \leq \mu(2) \ldots \leq \mu(T_{max})$. Alternatively, the penalty factor can be fixed throughout the run, i.e., $\mu(t) = \mu_0$ for all $1 \leq t \leq T_{max}$. Regardless of the penalty function adopted, it is worth remarking that each solution is always lying on the boundary of the feasible space corresponding to the constraint under consideration. Note that a penalty function was adopted due to its simplicity, since our interest was to assess the advantages of our proposed approach. However, other constraint-handling techniques are evidently possible. The penalty factors $\mu(t)$ were experimentally determined for each particular problem and are showed later for ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$ and ACO$_{\mathbb{R}}$ respectively.

All the algorithms considered in this experimental study (i.e., ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$, ACO$_{\mathcal{B}\mathbb{R}}^{(B)}$, and SR[5]) were executed 30 times with different seeds for each parameter combination. The problems studied include a set of well-known test cases traditionally adopted in the specialized literature: $G01$ to $G07$, $G09$, $G10$, $G11$, $G13$, $G14$, $G15$, $G17$, $G21$, $G23$, $G24$[21], and $G25$ [9].

The whole experimental study was performed on a Laptop with an Intel® Pentium® M Processor 725, running at 1.6 Ghz, and with 512 Mbytes of RAM. The ACO$_{\mathcal{B}\mathbb{R}}^{(B)}$ algorithm was implemented in C Language running under Suse-Linux.

[5] The parameter setting for SR is showed in section 4.2

4.1 Tunning of parameter q for ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$

In the earlier experiments with ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ we initially chose a similar parameter setting as used in [29] where $N_a = 2$, $k = 50$, $\xi = 0.85$, and $q \in \{0.0001, 01\}$. The higher value for parameter q was chosen for multimodal functions. The preliminary results from ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ by using the above parameter setting was rather discouraging since the algorithm was not capable of achieving any feasible solution for all tested problems. After that we considered a larger number of ants (i.e., $N_a \gg 2$) for generating a larger sampling of solutions according to the $k = 50$ Gaussian kernels. More specifically we set $N_a = 50$ as was set for ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ (see section 4.2). In addition we considered an intermediate value for parameter q, thus, this parameter was set to $q \in \{0.0001, 0.01, 0.1\}$. The penalty factor involved in function ϕ (Eq. 6) for each problem were as follows: $G01$ ($\mu = 1000$), $G04$ ($\mu = 5000000$), $G05$ ($\mu = 10$), $G06$ ($\mu = 10^{11}$), $G07$ ($\mu = 20000$), $G09$ ($\mu = 200000$), $G10$ ($\mu = 20000000$, $G13$ ($\mu = 0.1$), $G14$ ($\mu = 150$), $G15$ ($\mu = 10$), $G17$ ($\mu = 1000$), $G21$ ($\mu = 3000$), $G23$ ($\mu = 1000$), and $G24$ ($\mu = 10000$). All of these values[6] were set regarding the previous work [18] in which similar values were used for ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ (in section 4.2 we describe these values when comparing ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ against ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ and SR.)

We have divided de presentation of the results in two groups. The first group (Table 2) includes the test problems for which ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ achieved good quality results by using $T_{max} = 5000$, i.e., at most 250000 function evaluations (in our case, corresponds to function ϕ in Eq. 6). The second one in Table 3 shows the results for those problems that needed a larger umber of iterations to achieve better results. Both tables show in the respective columns: the problem name (Prob.), the best known or optimal value (Opt), the best found value (BF), mean value (Mean), worst value (Worst), mean number of function evaluations to achieve the best found (Mean(#E)), and the number of feasible solutions out of 30 runs.

It must be observed that neither Table 2 or 3 show the results for problem $G05$ for which any feasible solution was found, however the solutions found were slightly infeasible. For this problem we considered a different setting and the respective results will be showed later.

Table 2 shows the results obtained from ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ according to the parameter setting $q = 0.0001, 0.01$, and 0.1 (up to down on the respective row for for each problem) and $\xi = 0.85$. First of all, it can be observed that for all values of parameter q, ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ was able to to find high quality results. Also, it worth remarking that

[6] Note that all penalty factors are fixed, however, we used some dynamic penalty factor with the algorithm ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ in section 4.2

Table 2 Results from $\text{ACO}_{\mathcal{BR}}^{(S)}$ according to the parameter setting $q = 0.0001, 0.01$, and 0.1 (up to down on the respective row for for each problem) and $\xi = 0.85$ used for some test cases in [29]. The remaining values parameter used in the experiment are $k = 50$, $N_a = 50$, and $T_{max} = 5000$. The (*) symbol in column #Fea indicates that the average on the violation constraints are in between 0.0001 and 0.0002. Since our policy, solutions which violates constraints for more than 0.0001 are considered infeasible.

		$\text{ACO}_{\mathcal{BR}}^{(S)}$				
Prob.	Opt[7]	BF	Mean	Worst	Mean(#E)	#Fea
G01	-15.000	-15.000	-14.9609	-14.7281	66750	30
		-15.000	-15.000	-15.000	92100	30
		-15.000	-15.000	-15.000	161800	30
G03	1.000	1.000	1.000	1.000	104000	30
		1.000	1.000	1.000	129600	30
		1.000	1.000	1.000	157100	30
G04	-30665.539	-30665.539	-30665.539	-30665.539	68900	30
		-30665.5722	-30665.5722	-30665.5722	126100	28
		-30665.539	-30665.539	-30665.539	166950	30
G06	-6961.814	-6961.814	-6961.8137	-6961.813	52000	30
		-6961.814	-6961.813	-6961.8129	89800	30
		-6961.814	-6961.813	-6961.8125	116550	30
G07	24.306	24.306	24.530	24.985	30150	25(*)
		24.306	24.470	24.815	55100	27
		24.306	24.3293	24.392	88000	26
G09	680.630	680.630	680.630	680.630	49000	30
		680.630	680.630	680.630	89200	30
		680.630	680.630	680.630	116900	30
G10	7049.2480	7058.3559	7208.0776	7506.7651	85000	28
		7049.3369	7160.5849	8127.4853	120340	28
		7058.5097	7100.442	7162.373	58050	22
G11	0.75	0.75	0.75	0.75	17110	30
		0.75	0.75	0.75	17305	30
		0.75	0.75	0.75	18005	30
G13	0.053950	0.053951	0.054112	0.054637	10050	23 (*)
		0.053950	0.054033	0.054596	11680	18
		0.053980	0.053980	0.053980	17030	1 (*)
G14	-47.76441	-47.624847	-45.268413	-41.556510	240800	28
		-47.71844	-47.11483	-44.32239	242900	29
		-47.74250	-47.67783	-47.61127	243390	16 (*)
G15	961.715022	961.715148	961.714965	961.715209	165050	30
		961.715148	961.71496	961.715209	178350	30
		961.715200	961.715321	961.715390	199900	30
G21	193.7783	193.79061	193.83093	193.90968	102300	6
		193.78950	193.83940	193.98170	145100	10
		193.79360	193.83450	193.91410	180250	20
G24	-5.508013	-5.508013	-5.508013	-5.508013	19800	30
		-5.508013	-5.508013	-5.508013	22950	30
		-5.508026	-5.508026	-5.508026	23950	30
G25	-16.73889	-16.73893	-16.73893	-16.73893	9400	30
		-16.73819	-16.73819	-16.73819	10850	30
		-16.73893	-16.73893	-16.73893	12200	30

the best found values (columns BF) are very similar for the different q values, however, it is remarkable the increasing in the number of evaluations (column #Fea) when increasing the q value. This situation is easily explained because for smaller q values, more importance is given to the best-so-far solution which increases the convergence of $\text{ACO}_{\mathcal{BR}}^{(S)}$. Although for all problems from this group the different values of parameter q does not affect substantially the quality of results, it can be noticed some differences regarding #Fea. For example, for problem $G10$ $\text{ACO}_{\mathcal{BR}}^{(S)}$ achieved the larger number of feasible solutions when setting $q = 0.1$. A similar situation is observed for problem $G21$, where #Fea increases as the value of q increases. Clearly, for these problems a more explorative strategy improves the $\text{ACO}_{\mathcal{BR}}^{(S)}$ performance. In the case of problem $G14$, the situation described before is different concerning the increasing in #Fea, however, the quality of results for this problem are still better for $q = 0.1$. Finally, it is noticeable that for the reported results in Table 2

of some problems (indicated with (*)) and q values, the average on the violation constraints are very low, just in between 0.0001 and 0.0002 (according to our policy, solutions which violate constraints for more than 0.0001 are considered infeasible). As a conclusion, it can be said that lower values of q increase the velocity of convergence of the algorithm which could be useful for some type of problems. On the other hand, larger values of q make ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$ a more explorative algorithm, however, with an increased number of function evaluation. This is certainly the way in which ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$ achieved better results for some test problems.

Table 3 Results from ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$ according to the parameter setting $q = 0.0001, 0.01$, and 0.1 (up to down on the respective row for for each problem) and $\xi = 0.85$ used for some test cases in [29]. The remaining values parameter used in the experiment are $k = 50$, $N_a = 50$, and $T_{max} = 10000$. The (*) symbol in column #Fea indicates that the average on the violation constraints are in between 0.0001 and 0.0002. Since our policy, solutions which violates constraints for more than 0.0001 are considered infeasible.

		ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$				
Prob.	Opt[8]	BF	Mean	Worst	Mean(#E)	#Fea
G02	0.803619	0.753613	0.640071	0.531255	23250	30
		0.803619	0.697796	0.545772	30100	30
		0.803619	0.777522	0.683394	31550	30
G17	8853.5397	8871.682	9029.559	9212.925	345200	29
		8866.86523	9002.2568	9197.17871	359800	29
		9017.47851	9017.47851	9017.47851	494700	1
G23	-400.0025	-300.80877	-49.064338	130.72998	295000	4
		23.453075	73.200012	122.94695	325000	2
		188.41090	188.41090	188.41090	360500	1

With respect to the remaining problems considered in this work, that is, $G02$, $G17$, and $G23$; it was necessary to increase the number of iterations (consequently the number of function evaluations) to reach competitive results. In Table 3 are showed the obtained results from ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$ with $T_{max} = 10000$. The influence of q in the number of evaluation is similar as observed in Table 2, however, we can find some difference on column #Fea. For problem $G02$, all solutions were feasible for the different values of q, but the quality of the results improved for $q = 0.01,$ and 0.1. The situation is different in some way for $G17$ since the best quality results and the larger number of feasible solutions were obtained for $q = 0.0001$ and 0.01. Certainly, the worst performance of ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$ was on problem $G23$ for which only a few feasible solutions were found and the best setting was for $q = 0.0001$.

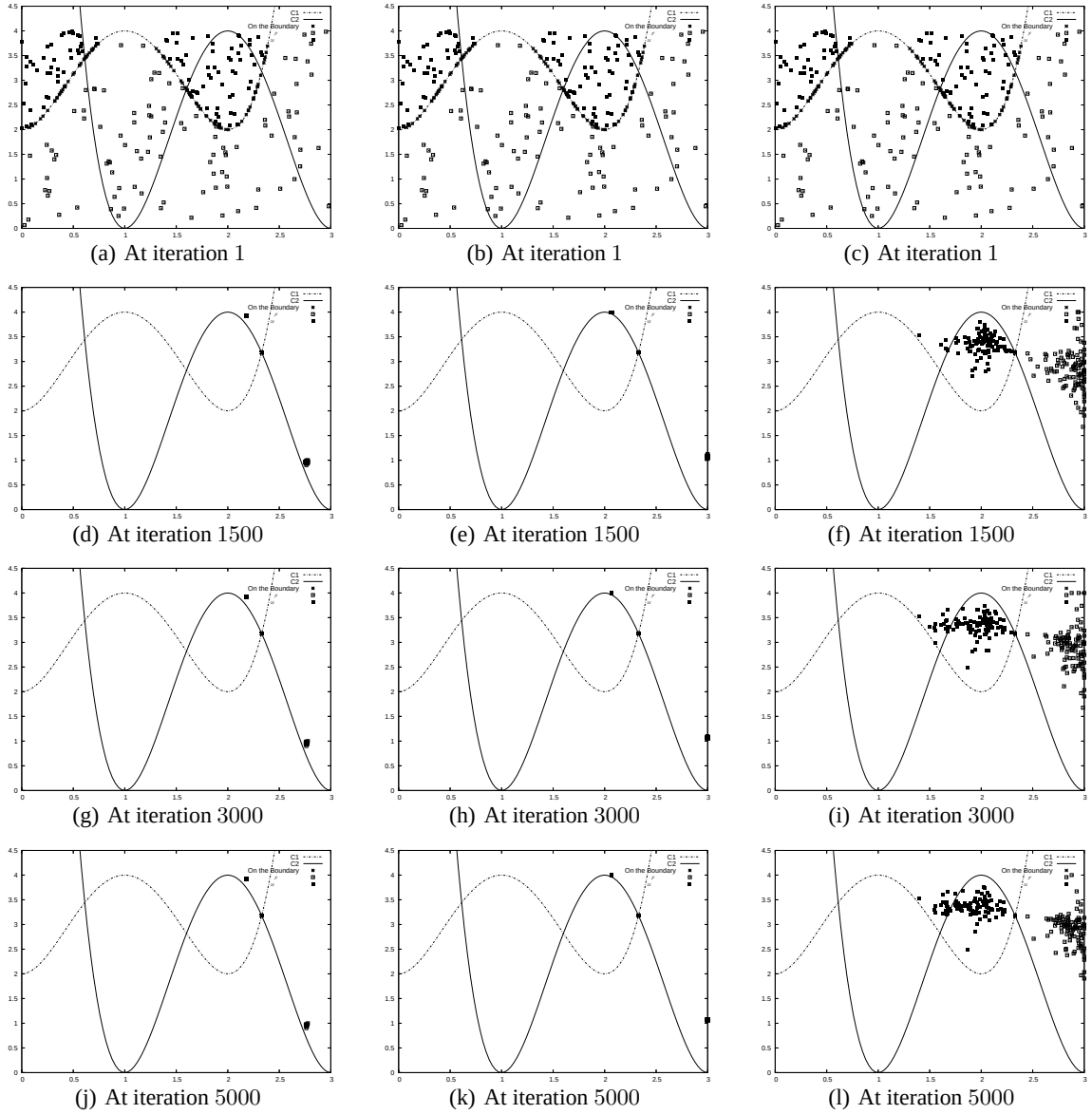
It is also interesting to visually the distribution of points in the search space $\mathcal{G}$ during the exploration according to different values of q. For a visualization purposes, we show the mentioned distribution of points for two problems $G24$ and $G25$ which have 2 inequality and 1 equality constraints respectively (see Appendix A). Figures 8 and 9 show for the three values of q considered (from left to right) the distribution of points in the search space $\mathcal{G}$ and the respective represented (decoded) points on the boundary $\mathcal{B}$. Problem $G25$ has only one

constraint, thus ACO$_{\mathcal{BR}}^{(S)}$ always focuses on that constraint whereas $G25$ has two constraints and the two of then are active at the optimum. In this last case, for reasons of clarity, we obtain the points distribution by running ACO$_{\mathcal{BR}}^{(S)}$ focusing exclusively on constraint 1. For both problems, the respective initial points in space $\mathcal{G}$ are the same, therefore, in the sequence of subfigures the first row are the same. Also, it is important to remark the distribution of points on the boundary for $G24$ and $G25$ (on constraint 1) in each subfigure. As the run progresses, the set of points (remember that the points in $\mathcal{G}$ are the basis for building the successive kernels through the run) tends to get clustered in a particular area. For both problems, that area correspond to a set of points $(\mathbf{x}, \mathbf{y}) \in \mathcal{G}$ for which $BS(\mathbf{x}, \mathbf{y})$ is close to the respective optimum solution. When $q = 0.0001$, it can be observed a rapid convergence of ACO$_{\mathcal{BR}}^{(S)}$. Similar situation is observed for $q = 0.01$, however, for $G25$ the clusters are less tight than the clusters for $G24$. For $q = 0.1$, the behavior of ACO$_{\mathcal{BR}}^{(S)}$ is clearly more explorative, nevertheless, for these two problems, all this spread points decode in a very tight area of the boundary around the optimal one.

Finally, we made an additional experiment in order to solve problem $G05$ for which ACO$_{\mathcal{BR}}^{(S)}$ found no feasible solution in the experiments reported before. In this case, we modify the value of ξ based on its proposal. More precisely, parameter ξ controls the influence the variability of the PDFs. Therefore, the lower ξ, the more slight are the perturbations on the points of space $\mathcal{G}$ which is precisely the difficulty with problem $G05$, that is, the solutions found were slightly infeasible and very close to the optimal one. For that reason we run ACO$_{\mathcal{BR}}^{(S)}$ for $G05$ with $\xi = 0.3$ and $q = 0.1$. In this case ACO$_{\mathcal{BR}}^{(S)}$ found 27 feasible solutions where the best found, mean, and worst values were respectively 5126.5083, 5143.6240, and 5159.6303. Figure 10 shows the behavior of ACO$_{\mathcal{BR}}^{(S)}$ for problem $G24$ (as an example) by setting $\xi = 0.3$ and $q = 0.1$. it can be seen a few number of no well defined clusters after at iteration 1500. When ACO$_{\mathcal{BR}}^{(S)}$ reaches iteration 3000, arises a set of more defined clusters. At the end, the algorithm converged to two clusters (one the feasible region and the other on the infeasible one). These behavior occurs when avoiding selective pressure on the best-so-far solutions (a larger q) and making small perturbation on the solutions to explore the search space (a smaller ξ).

4.2 Performance Comparison of ACO$_{\mathcal{BR}}^{(S)}$, ACO$_{\mathcal{BR}}^{(B)}$, and SR

In this section we compare the best quality results from ACO$_{\mathcal{BR}}^{(S)}$ (the adapted ACO$_{\mathbb{R}}$ for boundary search), ACO$_{\mathcal{BR}}^{(B)}$ (the ant algorithm proposed in Leguizamón et al. [18]), and Stochastic Ranking (SR) [26] (a well



(a) At iteration 1 (b) At iteration 1 (c) At iteration 1

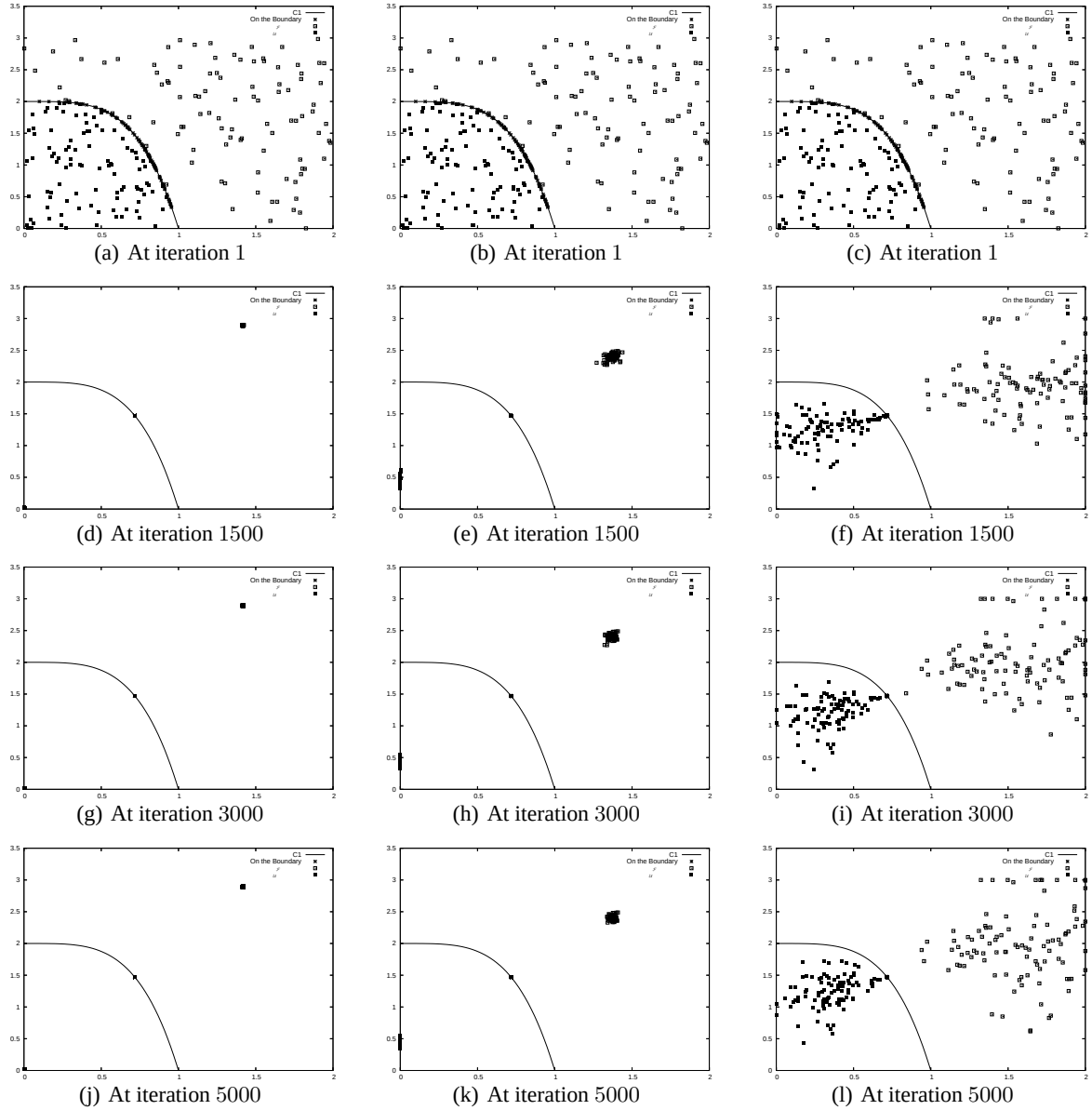
(d) At iteration 1500 (e) At iteration 1500 (f) At iteration 1500

(g) At iteration 3000 (h) At iteration 3000 (i) At iteration 3000

(j) At iteration 5000 (k) At iteration 5000 (l) At iteration 5000

Fig. 8 Test case $G24$: distribution of feasible and infeasible points on the search space $\mathcal{G}$ and the respective points on the boundary after the application of function BS. From left to right $q = 0.0001, 0.01$, and 0.1 (ξ was fixed to 0.85).

known constraint-handling technique). As explained before, we use S_{act} as the most efficient search criteria for the ACO algorithms, i.e., for ACO$^{(S)}_{\mathcal{BR}}$ and ACO$^{(B)}_{\mathcal{BR}}$. The parameter setting for ACO$^{(B)}_{\mathcal{BR}}$ was taken from [18, 17]: $N_a = 50$ ants (population size), $k = 20$ directions (number of reference points), maximum number of iterations $T_{max} = 30000$, evaporation rate $\rho = 0.5$, and $t_c = 200$ for the used policy S_{act}.



(a) At iteration 1 (b) At iteration 1 (c) At iteration 1

(d) At iteration 1500 (e) At iteration 1500 (f) At iteration 1500

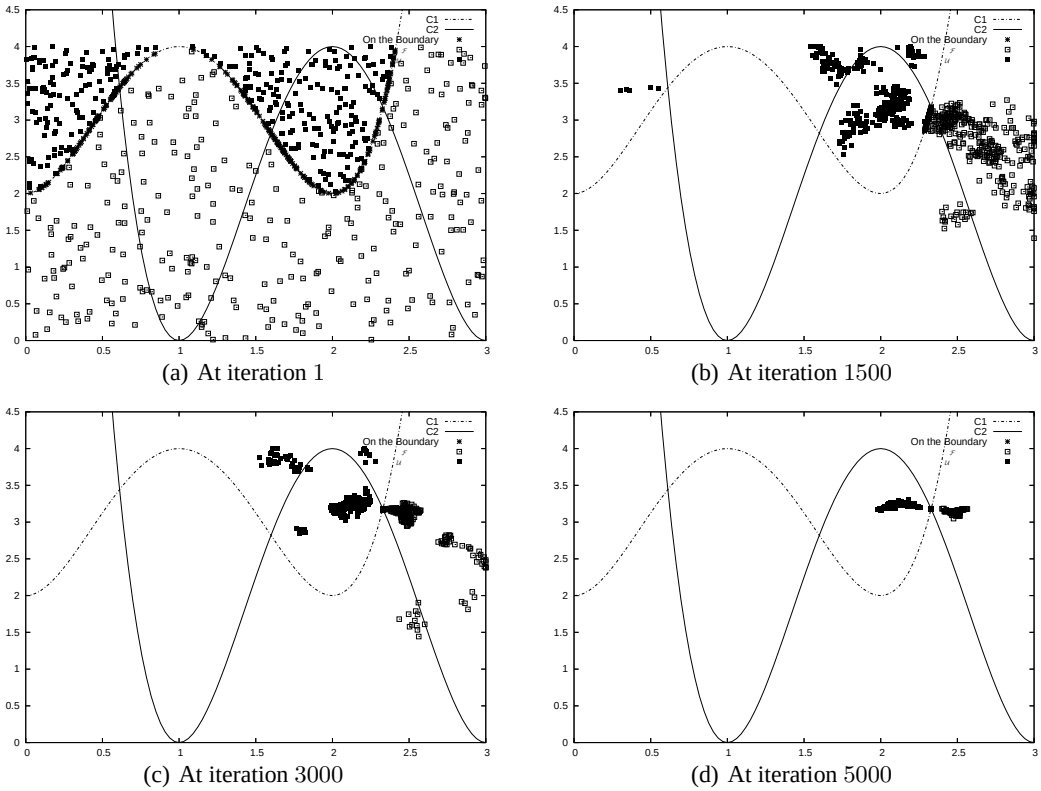
(g) At iteration 3000 (h) At iteration 3000 (i) At iteration 3000

(j) At iteration 5000 (k) At iteration 5000 (l) At iteration 5000

Fig. 9 Test case $G25$: distribution of feasible and infeasible points on the search space $\mathcal{G}$ and the respective points on the boundary after the application of function BS. From left to right $q = 0.0001, 0.01$, and 0.1 (ξ was fixed to 0.85). Note that some points have been clustered on the lower-left corner of the figures (see subfigures (d), (e), (g), (h), (j), and (k))

We adopted a dynamic penalty[9] ($\mu(t) = 1.05 \times \mu(t-1)$ for $t = 0, 1, \cdots, T_{max}$) for problems $G10$, $\mu(0) = 200000$; $G14$, $\mu(0) = 150.8$; $G17$, $\mu(0) = 400$; $G21$, $\mu(0) = 1500$; and $G23$, $\mu(0) = 13500$. The static penalty factors adopted for the remaining problems are (i.e., $\mu(t) = \mu_o$ for $t = 0, 1, \cdots T_{max}$):

[9] It is important to remark that the proposed algorithm in this work (ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$) only used a fixed penalty factor for all tested problems.



(a) At iteration 1 (b) At iteration 1500

(c) At iteration 3000 (d) At iteration 5000

Fig. 10 Test case $G24$: distribution of feasible and infeasible points on the search space $\mathcal{G}$ and the respective points on the boundary after the application of function BS by setting $q = 0.1$ and $\xi = 0.3$. By this setting, ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ accomplishes highly spread search however, making small perturbations on the points in $\mathcal{G}$ ($k = 200$ kernels were used for better visualization purposes.)

$G01$, $\mu(t) = 1000$; $G04$, $\mu(t) = 800000$; $G05$, $\mu(t) = 10$; $G06$, $\mu(t) = 10000$; $G07$, $\mu(t) = 20000$; $G09$, $\mu(t) = 2000$; $G13$, $\mu(t) = 0.2$, $G15$; and $G24$, $\mu(t) = 1000$.

The parameter setting for SR was as follows: $\mu = 30$, $\lambda = 200$, Gaussian Mutation, $\varphi = 1$, $P_f = 1$, $G_m = 1750$, and $\delta = 0.0001$ (see [26] for further details). With respect to ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ we report here the best results found in the preliminary experiments reported in the section before.

For test suite considered, the three algorithms perform almost identically for many of them: $G01$, $G03$, $G04$, $G06$, $G09$, $G11$, $G15$, $G24$, and $G25$, except for the number of evaluation functions (see a comparison later in this section). Due to that fact, we do not display any results for the above mentioned problem. For the remaining problems, i.e., $G02$, $G05$, $G07$, $G13$, $G14$, $G17$, $G21$, and $G23$; we show the respective results in Table 4 which shows for each problem considered: the average of the best found out of 30 runs (Mean) and

Worst values respectively from ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$, ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$, and SR. The (*) indicates that the respective optimum (or best known) was found and (+) indicates that the best found solution was very close to the optimum.

Table 4 Comparison of ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ with respect to ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ [18,17] and SR [26]. (*) indicates that the respective optimum (or best known) was found whereas (+) indicates that the best found solution was very close to the optimum.

	Mean			Worst		
Prob.	ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$	ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$	SR	ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$	ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$	SR
G02	0.77522 (*)	**0.802656** (*)	0.781875 (+)	0.683394	**0.793083**	0.726288
G05	5143.624 (*)	5138.37(*)	**5128.881** (*)	5159.63	**5132.14**	5142.472
G07	24.530 (*)	24.640 (*)	**24.374** (*)	**24.620**	24.920	24.642
G10	**7160.584** (+)	7199.01 (+)	7559.192 (+)	**7377.647**	7943.15	8835.655
G13	**0.054112** (*)	0.054908 (*)	0.057006 (*)	**0.054637**	0.055386	0.216915
G17	9029.559 (+)	8937.446289 (+)	**8893.396000** (+)	9212.925	8952.621093	**8951.00700**
G21	**193.83093** (+)	194.345108 (+)	NA	**193.90968**	202.067779	NA
G23	-49.064338	**-249.007506**	NA	130.7272998	**-28.448352**x	NA

First of all, it can be observed SR was not able to find any feasible solution for problems $G21$ and $G23$. In addition we run SR by setting $G_m = 3500$ in order to increase the number of evaluations for these two problems, nevertheless, SR could not ontain any feasible solutions for $G21$ and $G23$. For these two problems, ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ and ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ beahave similarly on problem $G23$, where ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ slightly outperforms ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ regarding mean and worst values. For problem $G24$, neither ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ or ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ found good quality results, however, ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ found better quality results. For the remaining problems in Table 4, we can observe the following: a) the tree considered algorithms perform similarly when considering the best found result (indicated by (*) or (+)), except for problem $G02$ for which SR was no capable of finding the best knwon value 0.803619. However, a very close valur to the best knwon was found as indicated by (+), b) the best mean values were fairly distributed on the three tested algorithms, c) for the best worst values, we can rank the three algorithms as follows: ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$, ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$, and SR (this, of course, does not imply atht ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ is better than ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ and this one is better than SR.

As the final report of our experimental study we show in Table 5 a comparison on the number of solution evaluations regarding $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$, $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(B)}$, and SR. Columns $\bar{e}_{ACO_{\mathcal{B}}^{(S)}}$, $\bar{e}_{ACO_{\mathcal{B}}^{(B)}}$, and $\bar{e}_{\text{SR}}$ represent, respectively, the average number of evaluations to obtain the best solution for $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$, $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(B)}$, and SR. It can be observed that for problems $G01$, $G03$, $G05$, $G06$, $G07$, $G11$, $G14$, $G15$, $G17$, $G21$, $G23$, $G24$, and $G25$; $\bar{e}_{ACO_{\mathcal{B}}^{(S)}}$ is less than $\bar{e}_{ACO_{\mathcal{B}}^{(B)}}$ and $\bar{e}_{\text{SR}}$, where the difference between these two values is remarkable for some of them.

Table 5 Average number of evaluations to obtain the best solution for $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(B)}$, $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$, and SR on the test problems considered (* means 'not available'). Clearly, there is no clear trend on the performace of the three algorithms with respect to the number of evaluations. However, an importatn reduction in the mean number of evaluations was achieved by $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$ with respect to $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(B)}$ (indicated with * in the respective rows).

	Problem	$\bar{e}_{ACO_{\mathcal{B}}^{(S)}}$	$\bar{e}_{ACO_{\mathcal{B}}^{(B)}}$	$\bar{e}_{\text{SR}}$
*	$G01$	**66750**	81400	149600
	$G02$	54050	**29500**	233400
*	$G03$	**104000**	140000	212000
	$G04$	68900	**21457**	77600
*	$G05$	59850	94000	**52400**
*	$G06$	**52000**	80000	111600
*	$G07$	**30150**	35600	141400
	$G09$	49000	**7400**	111000
	$G10$	120340	42800	**17200**
*	$G11$	17110	70400	**10400**
	$G13$	10050	**7200**	67200
*	$G14$	**240800**	1250000	349600
*	$G15$	165050	695600	**73200**
*	$G17$	345200	411500	**74000**
*	$G21$	**102300**	760000	*
*	$G23$	**295000**	763100	*
*	$G24$	**19800**	21400	23400
*	$G25$	**9400**	10600	15200

5 Conclusions and Future Work

In this paper we presented an alterative ACO algorithm ($ACO_{\mathcal{B}\mathbb{R}}^{(S)}$) including a a new search engine for implementing the boundary search approach. The search engine is an adaptation of one recently proposed for continuos problems ($ACO_{\mathbb{R}}$). For testing the proposed algorithm we we have used the penalty function as a complementary mechanism for problems with more than one constraint as was done with $ACO_{\mathcal{B}\mathbb{R}}^{(B)}$, the first proposed ACO algorithm implementing the boudnary search. The overall performance of $ACO_{\mathcal{B}\mathbb{R}}^{(S)}$ was compared with $ACO_{\mathcal{B}\mathbb{R}}^{(B)}$ and SR, showing the potential of this method as an alternative or complementary approach for constrained optimization problems. It is also worth noticing that $ACO_{\mathcal{B}\mathbb{R}}^{(S)}$ was able to reduce the number of function evaluation for several of the tested problems in comparison with the respective results from $ACO_{\mathcal{B}\mathbb{R}}^{(B)}$.

Future works include the use of more advanced complementary constraint-handling technique to be used with the boundary approach, e.g., a combination with stochastic ranking (under devolpment [16]). Also, a hybrid version of $ACO_{\mathcal{B}\mathbb{R}}^{(S)}$ with local search is an interesting possibility for improving the algorithm performance for harder test problems by perturbating the points on the boundary, e.g., by doing the main exploration on space $\mathcal{G}$ and a complementary exploration in space $\mathcal{B}$. In addition, the authors suggest as research topic in this area the design of a more general approach which includes the boundary search as a component that can be triggered to proceed with the boundary search when some condition are met during the exploration of the search space.

Acknowledgments

The first author acknowledges support from Universidad Nacional de San Luis and the ANPCYT (National Agency for Promotion of Science and Technology). The second author acknowledges support from CONACyT project no. 45683-Y.

References

1. S. Baluja. An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, School of Computer Science, Carnegie Mellon University, 1995.
2. G. Bilchev and I. Parmee. Ant colony search vs. genetic algorithms. Technical report, Plymouth Engineering Design Centre, University of Plymouth, 1995.
3. G. Bilchev and I. C. Parmee. The ant colony metaphor for searching continuous design spaces. *Lecture Notes in Computer Science*, 993:25–39, 1995.
4. G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, pages 610–611, 1958.
5. D. Corne, M. Dorigo, and F. Glover, editors. *New Ideas in Optimization*. McGraw-Hill International, 1999.
6. M. Dorigo and T. Stützle. *Ant Colony Optimization*. Mit-Press, 2004.
7. J. Dréo and P. Siarry. A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions. In M. Dorigo et al., editor, *ANTS*, pages 216–221, Heidelberg, 2002. Springer-Verlag Berlin.
8. T. Epperly. Global optimization test problems with solutions. Available at http://citeseer.nj.nec.com/147308.html.
9. Christodoulos A. Floudas and P. M. Pardalos. *A collection of test problems for constrained global optimization algorithms*, volume 455. Springer-Verlag Inc., New York, NY, USA, 1990.
10. F.W. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
11. Michael Guntsch and Martin Middendorf. A population based approach for aco. In *EvoWorkshops*, pages 72–81, 2002.
12. D. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, New York, 1972.
13. Andy Keane. Genetic algorithms digest, v8n16, 1994.
14. J. Kennedy and R Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Perth, Australia. IEEE Service Center.
15. Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.
16. G. Leguizamón and C. Coello Coello. A boundary aco algorithm with stochastic ranking. In preparation.
17. G. Leguizamón and C. Coello Coello. Boundary search for constrained numerical optimization problems in ACO algorithms (an extended version of [18]). In preparation.
18. G. Leguizamón and C. Coello Coello. Boundary search for constrained numerical optimization problems in ACO algorithms. In *ANTS Workshop*, pages 108–119, 2006.
19. W. Lei and W. Qidi. Ant system algorithm for optimization in continuous space. In *Proceedings of the 2001 IEEE International Conference on Control Applications*, pages 395–400, Mexico City, Mexico, September 2001.

20. W. Lei and W. Qidi. Further example study on ant system algorithm based continuous space optimization. In *Proceedings of the 4^{th} Congress on Intelligent and Automation*, pages 2541–2545, Shangai, P.R. China, 10-14 June 2002.
21. J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. Coello Coello, and K. Deb. Problem Definitions and Evaluation Criteria for the CEC. Technical report, Special Session on Constrained Real-Parameter Optimization, School of Electrical and Electronic Engineering Nanyang Technological University, available at http://www.ntu.edu.sg/home5/lian0012/cec2006/technical_report.pdf, Singapore, 2006.
22. Chen Ling, Sheng Jie, Qin Ling, and Chen Hongjian. A method for solving optimization problems in continuous space using ant colony algorithm. In Marco Dorigo, Gianni Di Caro, and Michael Sampels, editors, *Proceedings of the Third International Workshop, (ANTS'2002)*, volume 2463 of *Lecture Notes in Computer Science*, pages 288–289. Springer Verlag, Brussels, Belgium.
23. Zbigniew Michalewicz, Girish Nazhiyath, and Maciej Michalewicz. A note on usefulness of geometrical crossover for numerical optimization problems. In Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck, editors, *Evolutionary Programming V: Proc. of the Fifth Annual Conf. on Evolutionary Programming*, pages 305–311, Cambridge, MA, 1996. MIT Press.
24. N. Monmarché, G. Venturini, and M. Slimane. On how pachycondyla apicalis ants suggest a new search algoritm. *Future Generation Computer Systems*, 16:937–946, 2000.
25. Seid H. Pourtakdoust and Hadi Nobahari. An extension of ant colony systems to continuos optimization problems. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004*, pages 294–301. Springer-Verlag.
26. Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, 2000.
27. M. Schoenauer and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 245–254, Berlin, 1996. Springer.
28. Krzysztof Socha. ACO for continuos and mixed-variable optimization. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004*, pages 25–36. Springer-Verlag.
29. Krzysztof Socha and Marco Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 2007. In press.
30. K. Schittkowski W. Hock. *Test Examples for Nonlinear Programming Codes*. Lecture Notes in Econ. and Math. and Syst. Springer-Verlag, Berlin, Germany, 1981.

31. Z.Y. Wu and A.R. Simpson. A self-adaptive boundary search genetic algorithm and its application to water distribution systems. *Journal of Hidraulic Research*, 40(2):191–203, 2002.

32. Q. Xia. Global optimization test problems. Available at http://www.mat.univie.ac.at/ neum/glopt/xia.txt.

A Problems

– **G01** [9]

Minimize:

$$f(\mathbf{x}) = 5 \cdot \sum_{i=1}^{4} x_i - 5 \cdot \sum_{i=1}^{4} x_i^2 - \sum i = 513 x_i$$

subject to:

$$g_1(\mathbf{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_2(\mathbf{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$

$$g_3(\mathbf{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(\mathbf{x}) = -8x_1 + x_{10} \leq 0$$

$$g_5(\mathbf{x}) = -8x2 + x_{11} \leq 0$$

$$g_6(\mathbf{x}) = -8x_3 + x_{12} \leq 0$$

$$g_7(\mathbf{x}) = -2x_4 - x_5 + x_{10} \leq 0$$

$$g_8(\mathbf{x}) = -2x_6 - x_7 + x_{11} \leq 0$$

$$g_9(\mathbf{x}) = -2x_8 - x_9 + x_{12} \leq 0$$

where the bounds are $0 \leq x_i \leq 1 (i = 1, ..., 9)$, $0 \leq x_i \leq 100 (i = 10, 11, 12)$ y $0 \leq x_{13} \leq 1$. The global minimum is at $\mathbf{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, where six constraints are active (g_1, g_2, g_3, g_7, g_8, and g_9) and $f(\mathbf{x}^*) = -15$.

– **G02** [13]

Maximize:

$$f(\mathbf{x}) = |\frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}|$$

subject to:

$$g_1(\mathbf{x}) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$

$$g_2(\mathbf{x}) = \sum_{i=1}^{n} -7.5n \leq 0$$

where $n = 20$ and $0 \leq x_i \leq 10$. The best known solution is at

$\mathbf{x}^* = (3.16237443645701, 3.12819975856112, 3.09481384891456, 3.06140284777302,$

$3.02793443337239, 2.99385691314995, 2.95870651588255, 2.92182183591092,$

$0.49455118612682, 0.48849305858571, 0.48250798063845, 0.47695629293225,$

$0.47108462715587, 0.46594074852233, 0.46157984137635, 0.45721400967989,$

$0.45237696886802, 0.44805875597713, 0.44435772435707, 0.44019839654132)$ where $f(\mathbf{x}^*) = 0.80619$ and constraint g_1 is close to being active.

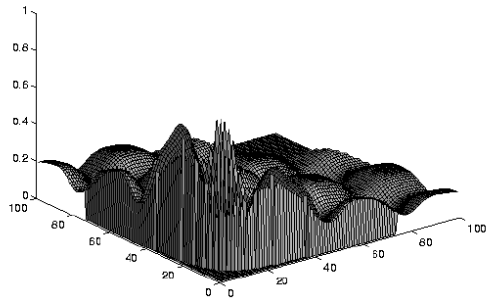


Fig. 11 Keane's function with $n = 2$.

– **G03** [23]

Maximize:

$$f(\mathbf{x}) = (\sqrt{n})^n \cdot \prod_{i=1}^{n} x_i$$

subject to:

$$\sum_{i=1}^{n} x_i^2 = 1$$

where $0 \leq x_i \leq 1 (i = 1, ..., n)$. The global optimum where $n = 10$ is at $\mathbf{x}^* = (1/\sqrt{n}, ..., 1/\sqrt{n})$ where $f(\mathbf{x}^*) = 1$.

– **G04** [12]

Minimize:

$$f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

subject to:

$$g_1(\mathbf{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 -$$
$$0.0022053x_3x_5 - 92 \leq 0$$
$$g_2(\mathbf{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 +$$
$$0.0022053x_3x_5 \leq 0$$
$$g_3(\mathbf{x}) = 80.51249 + 0.0071317x_2x_5 - 0.0029955x_1x_2 +$$
$$0.0021813x_3^2 - 100 \leq 0$$
$$g_4(\mathbf{x}) = -80.51249 - 0.0071317x_2x_5 + 0.0029955x_1x_2 -$$
$$0.0021813x_3^2 \leq 0$$
$$g_5(\mathbf{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 +$$
$$0.0019085x_3x_4 - 25 \leq 0$$
$$g_6(\mathbf{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 -$$
$$0.0019085x_3x_4 \leq 0$$

where $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ and $27 \leq x_i \leq 45$ $(i = 3, 4, 5)$. The optimum solution is at $\mathbf{x}^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ where $f(\mathbf{x}^*) = -30665.539$. Two constraints are active (g_1 and g_6).

– **G05** [30]

Minimize:

$$f(\mathbf{x}) = 3x_1 + 0.000001x_1^2 + 2x_2 + (0.000002/3)x_2^3$$

subject to:

$$g_1(\mathbf{x}) = -x_4 + x_3 - 0.55 \leq 0$$
$$g_2(\mathbf{x}) = -x_3 + x_4 - 0.55 \leq 0$$
$$h_3(\mathbf{x}) = 1000sen(x_3 - 0.25) + 1000sen(-x_4 - 0.25) + 894.8 - x_1 = 0$$
$$h_4(\mathbf{x}) = 1000sen(x_3 - 0.25) + 1000sen(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$
$$h_5(\mathbf{x}) = 1000sen(x_4 - 0.25) + 1000sen(x_4 - x_3 - 0.25) +_1 294.8 = 0$$

where $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-.55 \leq x_3 \leq 0.55$, and $-0.55 \leq x_4 \leq 0.55$. The best known solution [15] is $\mathbf{x}^* = (679.94453, 1026.067, 0.1188764, -0.3962336)$ where $f(\mathbf{x}) = 5126.4981$.

– **G06** [9]

Maximize:

$$f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to:

$$g_1(\mathbf{x}) = (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0,$$

$$g_2(\mathbf{x}) = -(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geq 0,$$

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The optimum solution is $\mathbf{x}^* = (14.095, 0.84296)$, $f(\mathbf{x}^*) = -6961.81381$. Both constraints are active at $\mathbf{x}^*$ (see Figure 12).

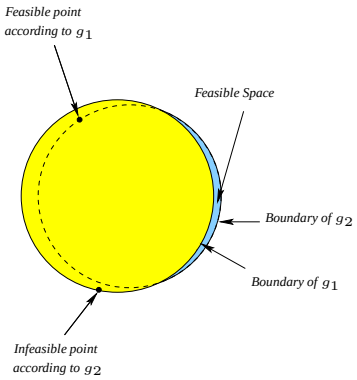


Fig. 12 Problem G6. Floudas-Pardalos' function.

– **G07** [30]

Minimize:

$$f(\mathbf{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2$$
$$+29x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_1 10 - 7)^2 + 45$$

subject to:

$$g_1(\mathbf{x}) = -105 + 4x_1 + 5x_2 - 3x_7 - 9x_8 \leq 0$$

$$g_2(\mathbf{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$

$$g_3(\mathbf{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$

$$g_4(\mathbf{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$$g_5(\mathbf{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$

$$g_6(\mathbf{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1 x_2 + 14x_5 - x_6 \leq 0$$

$$g_7(\mathbf{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$

$$g_8(\mathbf{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

where $-10 \leq x_i \leq 10 \quad (i = 1, ..., 10)$. The optimum solution is $\mathbf{x}^*$=(2.171996, 2.363683, 8.773926,5.095984, 0.9906548,1.430574, 1.3221644, 9.828726, 8.280092, 8.375927) where $f(\mathbf{x}^*)$ = 24.306209. Six constraints are active at $\mathbf{x}^*$: g_1, g_2, g_3, g_4, g_5, and g_6.

– **G09** [30]

Minimize:

$$f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2$$
$$+10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

subject to:

$$g_1(\mathbf{x}) = -127 +_2 x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$
$$g_2(\mathbf{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$
$$g_3(\mathbf{x}) = -196 + 23x_1 + x_2^2 + x_6^2 - 8x_7 \leq 0$$
$$g_4(\mathbf{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

where $-10 \leq x_i \leq 10 \quad (i = 1, ..., 7)$. The optimum solution is $\mathbf{x}^*$= (2.330499, 1.951372, -0.47775414, 4.365726, -0.6244870, 1.038131, 1.594227) where $f(\mathbf{x}^*)$ = 680.6300573. The active constraints at this point are: g_1 and g_4.

– **G10** [30]

Minimize:

$$f(\mathbf{x}) = x_1 + x_2 + x_3$$

subject to:

$$g_1(\mathbf{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$$
$$g_2(\mathbf{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$
$$g_3(\mathbf{x}) = -1 + 0.01(x_8 - x_5) \leq 0$$
$$g_4(\mathbf{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$
$$g_5(\mathbf{x}) = -x_2x_7 + 1250x_5 + x_3x_4 - 1250x_4 \leq 0$$
$$g_6(\mathbf{x}) = -x_3x_8 + 1250000 + x_2x_5 - 2500x_5 \leq 0$$

where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000 (i + 2, 3)$ and $10 \leq x_i \leq 1000 (i = 4, ..., 8)$. The optimum solutions is $\mathbf{x}^* = (584.3282028010, 1354.1644876700, 5110.7156493300, 182.4326280510, 295.5675740820, 217.5673719490, 286.8650539690, 395.5675740820)$, where $f(\mathbf{x}^*) = 7049.2083398$[10].

– **G11** [15]

Minimize:

$$f(\mathbf{x}) = x_1^2 + (x_2 - 1)^2$$

subject to:

$$h(\mathbf{x}) = x_2 - x_1^2 = 0$$

where $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$. The optimum solution is $\mathbf{x}^* = (+-1/\sqrt{2}, 1/2)$ and $f(\mathbf{x}) = 0.75$.

– **G13** [30]

Minimize:

$$f(\mathbf{x}) = e^{x_1 x_2 x_3 x_4 x_5}$$

subject to:

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$

$$h_2(\mathbf{x}) = x_2 x_3 - 5 x_4 x_5 = 0$$

$$h_3(\mathbf{x}) = x_1^2 + x_2^3 + 1 = 0$$

where $-2.3 \leq x_i \leq 2.3 \quad (i = 1, 2)$ y $3.2 \leq x_i \leq 3.2 \quad (i = 3, 4, 5)$. The optimumn solution is $\mathbf{x} = (-1.777143, 1.595709, 1.827247, 0.7636413, -0.763645)$ and $f(\mathbf{x}^*) = 0.0539498$.

– **G14** [12]

Minimize:

$$f(\mathbf{x}) = \sum_{i=1}^{10} x_i (ci + ln \frac{x_i}{\sum_{j=1}^{10} xj})$$

subject to:

$$h_1(\mathbf{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$

$$h_2(\mathbf{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$h_3(\mathbf{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

[10] See http://www.mat.univie.ac.at/ neum/glopt/coconut/Benchmark/Library2_new_v1.html, where problem $G10$ can be found as $h106$.

where the bounds are $0 < x_i \leq 10$ ($i = 1, \ldots, 10$), and $c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.1$, $c_8 = -10.708$, $c_9 = -26.662$, $c_{10} = -22.179$. The best known solution is at $x^* = (0.036002, 0.151412, 0.783686, 0.001725, 0.484752, 0.000695, 0.028175, 0.017604, 0.038714, 0.093207)$ where $f(x^*) = -47.764411$.

– **G15** [12]:

Minimize:

$$f(\mathbf{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

subject to:

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$
$$h_2(\mathbf{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

where the bounds are $0 \leq x_i \leq 10$ ($i = 1, 2, 3$). The best known solution is at $x^* = (3.51212812611795133, 0.21699875104295$ where $f(x^*) = 961.7150222$.

– **G17** [12]

Minimize:

$$f(\mathbf{x}) = f(x_1) + f(x_2)$$

where

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$

subject to:

$$h_1(\mathbf{x}) = -x_1 + 300 - \frac{x_3 x_4}{131.078} cos(1.48477 - x_6) + \frac{0.90798 x_3^2}{131.078} cos(1.47588)$$

$$h_2(\mathbf{x}) = -x_2 - \frac{x_3 x_4}{131.078} cos(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} cos(1.47588)$$

$$h_3(\mathbf{x}) = -x_5 - \frac{x_3 x_4}{131.078} sin(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} sin(1.47588)$$

$$h_4(\mathbf{x}) = 200 - \frac{x_3 x_4}{131.078} sin(1.48477 - x_6) + \frac{0.90798 x_3^2}{131.078} sin(1.47588)$$

where the bounds are $0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ and $0 \leq x_6 \leq 0.5236$. The best known solution is at $x^* = (212.684440144685, 89.1588384165537, 368.447892659317$ $4.16436988876356, 0.0680394595246655)$ where $f(x^*) = 8876.980680$.

– **G21** [8]

Minimize:

$$f(\mathbf{x}) = x_1$$

subject to:

$$g_1(\mathbf{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$$

$$h_1(\mathbf{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$

$$h_2(\mathbf{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$

$$h_3(\mathbf{x}) = -x_5 + ln(-x_4 + 900) = 0$$

$$h_4(\mathbf{x}) = -x_6 + ln(x_4 + 300) = 0$$

$$h_5(\mathbf{x}) = -x_7 + ln(-2x_4 + 700) = 0$$

where the bounds are $0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ and $4.5 \leq x_7 \leq 6.25$. The best known solution is at $x^* = (193.783493, 0, 17.3272116, 100.0156586, 6.684592154, 5.991503$ where $f(x^*) = 193.7783493$.

– **G23** [32]

Minimize:

$$f(\mathbf{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$$

subject to:

$$g_1(\mathbf{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$$

$$g_2(\mathbf{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$$

$$h_1(\mathbf{x}) = x_1 + x_2 - x_3 - x_4 = 0$$

$$h_2(\mathbf{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$$

$$h_3(\mathbf{x}) = x_3 + x_6 - x_5 = 0$$

$$h_4(\mathbf{x}) = x_4 + x_7 - x_8 = 0$$

where the bounds are $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ and $0.01 \leq x_9 \leq 0.03$. The best known solution is at $x^* = (0, 99.9999000001,$

$5.58738477217701e-026, 100, 0.000099999999, 0, 100, 200, 0.01)$ where

$f(x^*) = -400.002500$.

– **G24** [9]:

Maximize:

$$f(\mathbf{x}) = -x_1 - x_2$$

subject to:

$$x_2 \leq 2x_1^4 - 8x_1^3 + 8x_1^2 + 2$$

$$x_2 \leq 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1 + 36,$$

where the bounds are, $0 \leq x_1 \leq 3$ and $0 \leq x_2 \leq 4$. The best known solution is at $\mathbf{x}^* = (2.3295, 3.1783)$ where $f(\mathbf{x}^*) = -5.5079$. Figure 13 shows the feasible search space determined by the two inequelity constraints and the approximate position of x^* which lies on the boundary.

– **G25** [9]:

Minimize:

$$f(\mathbf{x}) = -12x_1 - 7x_2 + x_2^2$$

subject to:

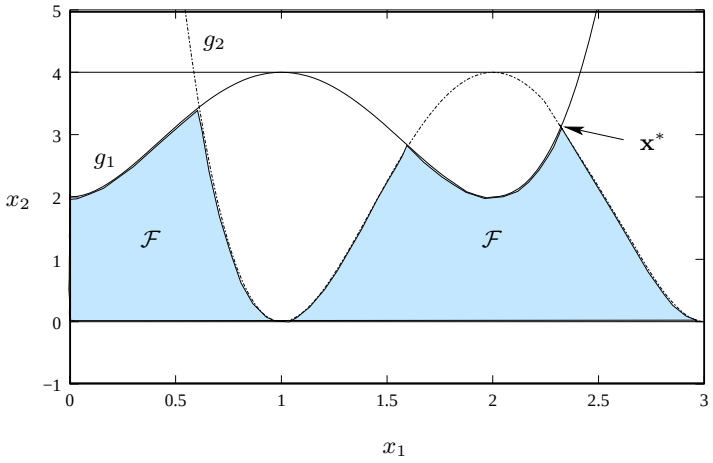


Fig. 13 Approximate position of the best known value on the boundary of the feasible search space regarding constraints g_1 and g_2.

$$-2x_1^4 + 2 - x_2 = 0$$

where the bounds are $0 \leq x_1 \leq 2$ and $0 \leq x_2 \leq 3$. The best known solution is at $\mathbf{x}^* = (0.71751, 1.470)$ where $f(\mathbf{x}^*) = -16.73889$. Figure 14 shows point $\mathbf{x}^*$ which lies on the boundary (in this case the boundary is equivalent to $\mathcal{F}$).

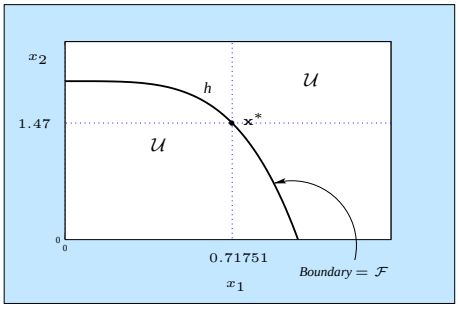


Fig. 14 Best known solution and the feasible search space determined by equality constraint h.