# A Boundary Search based ACO Algorithm Coupled with Stochastic Ranking

Guillermo Leguizamón and Carlos A. Coello Coello

***Abstract*— The Ant Colony Optimization metaheuristic has dramatically evolved in the last years. The area of continuos optimization has recently received more attention from the research community working with the ACO metaheuristic. In this paper we present a boundary search based ACO algorithm for solving nonlinear constrained optimization problems. The aim of this work is twofold. Firstly, we present a modified search engine which implements a boundary search approach based on a recently proposed ACO metaheuristic for continuos problems. Secondly, we propose the incorporation of the stochastic ranking technique to deal with feasible and infeasible solutions during the search which focuses on the boundary region. In our experimental study we compare the overall performance of the proposed ACO algorithm by including two different complementary constraint-handling techniques: a penalty function and stochastic ranking. In addition, we include in our comparison of results the Stochastic Ranking algorithm, which was originally implemented using an Evolution Strategy as its search engine.**

## I. INTRODUCTION

The Ant Colony Optimization (ACO) metaheuristic has been extensively applied to solve plenty of combinatorial optimization problems. The ACO metaheuristic (Corne et al. [4], Dorigo and Stüztle [5]) includes a variety of algorithms derived from the behavior of colonies of real ants. 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 which is usually represented as a construction graph.

One of the first ACO extensions to operate on continuous spaces can be found in Bilchev et al. [2] 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. [17], Lei et al. [14], [15], Dreo et al. [6], Monmarché et al. [19], and Pourtakdoust et al. [20]. More recently, an extension of the ACO metaheuristic to continuous domains and applied to continuous and mixed discrete-continuous problems is presented by K. Socha [23]. This proposal follows the original conception of 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. In another recent work by Socha et al. [24], the former ideas proposed by Socha [23] regarding continuous domains are extensively presented and details concerning implementation issues are given through the ACO$_{\mathbb{R}}$ algorithm. The experimental study presented by the authors considers a test suite of several unconstrained continuous optimization problems. In addition, an analysis of the behavior of ACO$_{\mathbb{R}}$ is presented regarding the impact of its main parameters on the algorithm's performance: $q$ and $\xi$. In Leguizamón et al. [12] a new constraint-handling technique is implemented in an ACO algorithm for continuous problems based on the former work by Bilchev et al. [2]. Leguizamón et al.'s work introduces 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 respect to other state-of-the-art algorithm when dealing with nonlinear problems having active constraints. It is also worth noting that the boundary approach has been studied from the evolutionary computation perspective. For example, in Michalewicz et al. [18] the efficiency of this approach is shown by using two constrained optimization problems: Keane's function (also known as $G02$) [9] 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, Schoenauer et al. [22] propose some evolutionary operators capable of exploring a general surface of dimension $n-1$ ($n$ is the number of variables) for 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. [25] 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, Wu et al.'s work

---

Guillermo Leguizamón is with LIDIC (Research Group). Universidad Nacional de San Luis - Ej. de Los Andes 950 - (D5700HHW) San Luis, ARGENTINA. (email: legui@unsl.edu.ar). Carlos A. Coello Coello is with CINVESTAV-IPN (Evolutionary Computation Group), Departamento de Computación, Av. IPN No. 2508, Col. San Pedro Zacatenco, Mexico D.F. 07300, MEXICO. (email: ccoello@cs.cinvestav.mx).

does not involve any explicit boundary operator. Based on the proposal of Socha et al. [24], Leguizamón et al. [13] adopted the algorithm ACO$_{\mathbb{R}}$, one of the more recent ACO extensions for continuous search spaces and showed how the boundary approach could be included in a more advanced search engine based on the ACO metaheuristic. In that work, a new algorithm called ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ is compared against ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ (see Leguizamón et al. [11]), a boundary search based ACO algorithm designed according the former ideas by Bilchev et al. [2]. The new algorithm ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$ was found to be a suitable alternative when facing constrained optimization problems. However, both algorithms, ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ and ACO$^{(S)}_{\mathcal{B}\mathbb{R}}$, include a penalty function as their complementary technique adopted to handle the problem's constraints. Although penalty functions are a suitable approach, they usually need an extensive preliminary experimental study to tune the values of their penalty factors. Regarding this situation, we propose in this paper the use of the stochastic ranking approach [21] in order to avoid the use of penalty factors as well as to achieve an improved performance of the ACO algorithm when dealing with constrained problems.

The remainder of this paper is organized as follows. Section II 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 [12], [11]) is presented. The ACO algorithm ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$ (based on ACO$_{\mathbb{R}}$) which implements the boundary approach is presented in Section III. On the other hand, Section IV presents ACO$^{(SR)}_{\mathcal{B}\mathbb{R}}$, which is our proposed algorithm for boundary search incorporationg stochastic ranking as its complementary constraint-handling technique. In addition, the ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ is also presented. The test problems and experimental results are presented and analyzed in Section V. Finally, our conclusions and some possible paths for future research are provided in Section VI.

## II. THE BOUNDARY SEARCH APPROACH

We are interested in solving the general nonlinear programming problem whose aim is to find $\mathbf{x}$ so as to optimize:

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

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:

$$\begin{array}{ll} g_j \leq 0, \;\text{ for }\; j = 1, \ldots, q & \text{and} \\ h_j = 0, \;\text{ for }\; j = q+1, \ldots, m. & \end{array}$$

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 corresponding optimal solution will lie on the region defined by $h_j(\mathbf{x}) = 0$. Furthermore, for many problems, the best solutions may lie 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.

### *A. Approaching the boundary*

We describe here a general boundary approach (proposed in [12], [11]) 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 $(\mathbf{x}_4, \mathbf{y}_4)$.

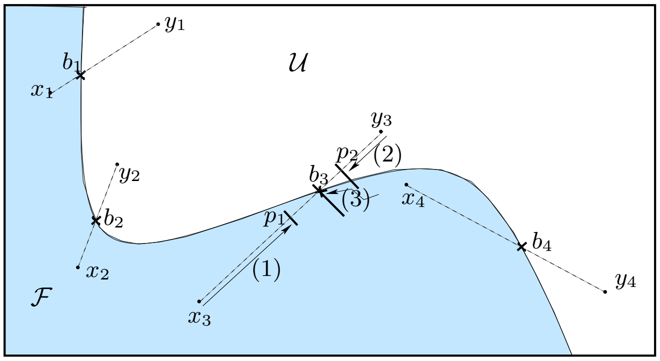


Fig. 1. Given one feasible and one infeasible point, the corresponding point lying on the boundary can be easily reached by using a simple binary search. In this way, each point on the boundary can be reached from at least a pair of points $(\mathbf{x}, \mathbf{y})$ with $\mathbf{x} \in \mathcal{F}$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

```
m: real vector;
repeat
    m = mid_point_between(x, y);
    if Is_on_Boundary(m) then
        return m; { m is a point lying on the boundary }
    end if
    if Feasible(m) then
        x = m;
    else
        y = m;
    end if
until (dist_to_boundary(m) ≤ δ) AND (Feasible(m));
return m; {The closest point to the boundary according to δ }
```

*B. 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 pairs of points $(\mathbf{x}, \mathbf{y})$ as described above. This space can be considered a *genotypic space* as known in 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 corresponding *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 $\phi$, which represents a measure of solutions quality and gives as a result an element from the set $\mathcal{E} = \{e \in \mathbb{R} | e = \phi(\mathbf{b})\}$.

From the above description, it is clear that the search engine must deal with the exploration of space $\mathcal{G}$. Figure 2 shows a set of hypothetical points in $\mathcal{G}$, a problem constraint and the corresponding points on the boundary. In the third pair of points (from left to right) we represent 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, we can create a population of individuals where each one of them represents an element of set $\mathcal{G}$. Therefore, suitable operators to be chosen could be any crossover and/or mutation operators appropriate 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 alternatives for the ACO metaheuristic in the following sections.

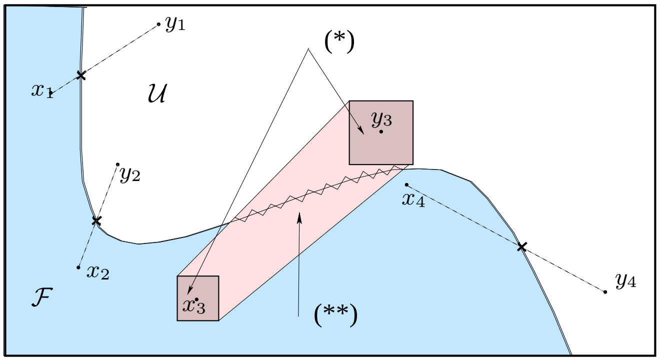


Fig. 2. A set of hypothetical points in $\mathcal{G}$, a problem constraint and the corresponding points on the boundary where (*) indicates the possible exploration regions for $\mathbf{x}_3$ and $\mathbf{y}_3$ and (**) indicates the corresponding points on the boundary region based on possible perturbations of $\mathbf{x}_3$ and $\mathbf{y}_3$.

*C. 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 $h(\mathbf{b})$, i.e., all solutions generated will be feasible without requiring any *ad-hoc* boundary operator.

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 remain focused on a particular constraint over the whole run or may move from one problem constraint to another depending on a particular condition. In our previous work [12], 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 conditions could be considered, for example, taking into account the population diversity 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 focus on a different problem constraint.

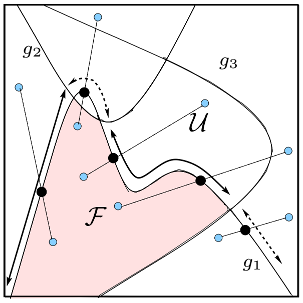


Fig. 3. Feasible search space defined by 3 inequality constraints. The search proceeds on the boundary of constraint $g_1$.

As an illustrative example, Figure 3 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 3). 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 3). 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 [21] to make the comparisons among the solutions generated [10] and thus avoid the inclusion and tuning of any penalty factor when evaluating a solution.

## III. THE PROPOSED ALGORITHM $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$ FOR BOUNDARY APPROACH

In this section we describe the design of the $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$ algorithm which implements the boundary search. The search engine involved in $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$ is based on the $\text{ACO}_{\mathbb{R}}$ algorithm presented in [24]. Before explaining the implementation of $\text{ACO}^{(S)}_{\mathcal{B}\mathbb{R}}$, we first describe briefly the main characteristics of $\text{ACO}_{\mathbb{R}}$ as it was proposed and tested in [24] on unconstrained continuous optimization benchmark problems.

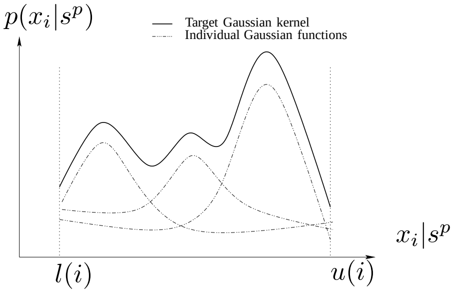


Fig. 4. (filled line): a continuous probability density function $p(x|s^p)$ where $x_i \in [l(i), u(i)]$, and $s^p$ is a partial solution under construction (see [24] for further details) and (dotted line): a possible set of three Gaussian functions to achieve by superposition a Gaussian Kernel which approximates the corresponding multimodal Gaussian function (fille line).

Taking into account that the ACO metaheuristic works by incrementally building the solutions according to a biased (by pheromone trail) probabilistic choice of solutions components, the $\text{ACO}_{\mathbb{R}}$ 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 4 (fille line) 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 4, Socha et al. [24] proposed a Gaussian Kernel which is defined as a weighted sum of several one-dimensional Gaussian function $g^i_l(x)$ as follows:

$$G^i(x) = \sum_{l=1}^{k} \omega_l g^i_l(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma^i_l \sqrt{2\pi}} e^{-\frac{(x-\mu^i_l)^2}{2(\sigma^i_l)^2}} \qquad (1)$$

where $i \in \{1, \ldots, n\}$ identifies the number of dimension, i.e., $\text{ACO}_{\mathbb{R}}$ 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 4 (dotted line) shows a superposition of three Gaussian functions which could approximate the hypothetical multimodal Gaussian function (filled line).

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. [8]. 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 corresponding values of each problem dimension, i.e., $x_l^1, \ldots, x_l^n$, and the value of the objective function $f(\mathbf{x}_l)$ which are stored satisfying that $f(\mathbf{x}_1) \leq \cdots \leq f(\mathbf{x}_l) \leq \ldots f(\mathbf{x}_k)$. On the other hand, the vector of weights $\boldsymbol{\omega}$ should satisfy that $\omega_1 \geq \cdots \geq \omega_l \geq \cdots \geq \omega_k$.

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 $\mu^i$, 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 solutions in $T$ are stored according to their rank, i.e., the highest the rank of the solution, the lowest the corresponding index in $T$. The weight $\omega_l$ associated to Gaussian function $g_l^i$ is obtained as:

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

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 solutions are preferred, otherwise, a large value for $q$ implies a more uniform probability. As mentioned in [24], 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:

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{|x_e^i - x_l^i|}{k-1} \tag{3}$$

where $l \in \{1, \ldots, k\}$ is the kernel number with respect to which the deviation is calculated and $\xi > 0$, which is the same for all dimensions, has an effect similar to that of the pheromone evaporation rate in ACO. Thus, the higher the value of $\xi$, 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 $\sigma^i$ since the sampling process of Gaussian kernel $G^i$ is accomplished as follows. Given the elements of vector $\boldsymbol{\omega}$ calculated as in Eq. 2, the sampling is done in two phases: 1) choose one of the $k$ Gaussian functions of $G^i$ according to the following probability:

$$p_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r}, \tag{4}$$

and, 2) after function $g_l^i$ has been chosen, a sampling is accomplished (perhaps using a random number generator based on a parameterized normal or an uniform distribution in conjunction with, for instance, the Box-Muller method [3]). Since at each step only one Gaussian function is used (let us say $g_l^i$), it is only needed $\sigma_l^i$ instead of the whole vector $\boldsymbol{\sigma}^i$. The pheromone update is achieved by considering a set $A$ of the newly generated solutions[1]. 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 solutions $A$ are ranked. In other words, the old solutions compete against the newly generated ones to conform the updated $T$ which maintain its cardinality ($k$) throughout the whole search process.

To adapt ACO$_{\mathbb{R}}$ to deal with constrained problems by implementing the boundary approach described above is rather straightforward. The proposed algorithm ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$, instead of maintaining one archive $T$, now maintains two archives for similar purposes, $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ which represent respectively the points on the feasible and infeasible parts 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 $\phi$. It is worth remarking that solutions in $T_{\mathcal{B}}$ are ranked according to the solution quality given by $\phi$. Taking into account this ranking, the solutions in $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ are then ranked accordingly.

As in the original ACO$_{\mathbb{R}}$ algorithm, vector $\boldsymbol{\omega}$ is intended for sampling the chosen Gaussian function, however, the situation is different in ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$ since there exist two independent archives $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ from which 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 $T_{\mathcal{U}}$. Table I represents a general outline of the archives $T_{\mathcal{F}}$, $T_{\mathcal{U}}$, $T_{\mathcal{B}}$, $\boldsymbol{\omega}$, and $\mathcal{E}$. 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 noticed 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 influences the generation of new and better quality solutions in the space $\mathcal{G}$.

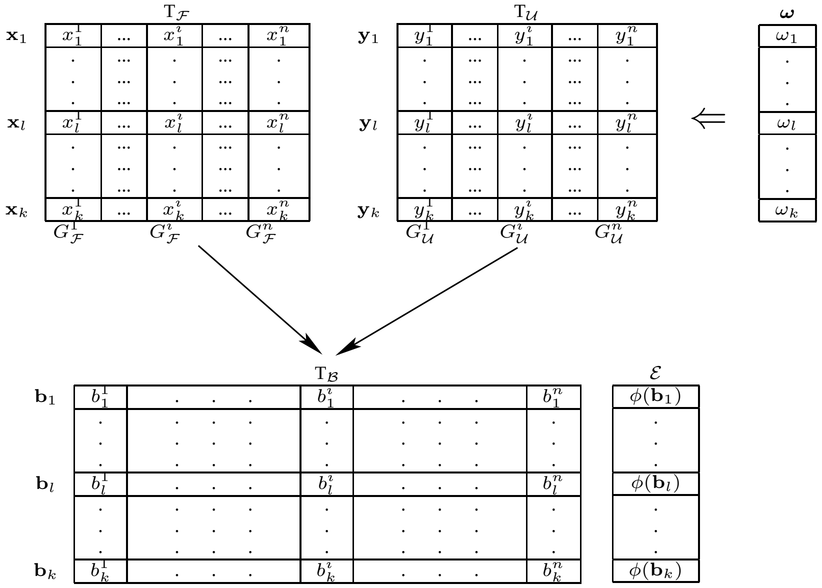
A general outline of ACO$_{\mathcal{B}\mathbb{R}}^{(S)}$ is presented in Algorithm 2 which displays its main components. In line 1, archives $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$ are initialized by randomly generated solutions in the feasible and infeasible search space regarding the problem constraint at hand. Similarly, vector $\boldsymbol{\omega}$ is initialized according to Eq. 2 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

---

[1] Set $A$ represents the set of ants according to Socha et al. [24].

TABLE I

REPRESENTATION OF THE $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$ SEARCH SPACE DIVIDED IN FEASIBLE AND INFEASIBLE POINTS



$T_{\mathcal{F}}$
$T_{\mathcal{U}}$
$\boldsymbol{\omega}$
$\mathbf{x}_1$
$\mathbf{x}_l$
$\mathbf{x}_k$
$\mathbf{y}_1$
$\mathbf{y}_l$
$\mathbf{y}_k$
$x_1^1$ ... $x_1^i$ ... $x_1^n$
$x_l^1$ ... $x_l^i$ ... $x_l^n$
$x_k^1$ ... $x_k^i$ ... $x_k^n$
$y_1^1$ ... $y_1^i$ ... $y_1^n$
$y_l^1$ ... $y_l^i$ ... $y_l^n$
$y_k^1$ ... $y_k^i$ ... $y_k^n$
$\omega_1$
$\omega_l$
$\omega_k$
$\Leftarrow$
$G_{\mathcal{F}}^1$ $G_{\mathcal{F}}^i$ $G_{\mathcal{F}}^n$
$G_{\mathcal{U}}^1$ $G_{\mathcal{U}}^i$ $G_{\mathcal{U}}^n$
$T_{\mathcal{B}}$
$\mathcal{E}$
$\mathbf{b}_1$
$\mathbf{b}_l$
$\mathbf{b}_k$
$b_1^1$ . . . $b_1^i$ . . . $b_1^n$
$b_l^1$ . . . $b_l^i$ . . . $b_l^n$
$b_k^1$ . . . $b_k^i$ . . . $b_k^n$
$\phi(\mathbf{b}_1)$
$\phi(\mathbf{b}_l)$
$\phi(\mathbf{b}_k)$

$T_{\mathcal{B}}$. Then, function "BuildSols" is in charge of generating new solutions through the Gaussian kernel obtained from the corresponding archives (lines 4 and 5). In order to further 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 to the solutions quality given by function $\phi$, and the best first $k$ solutions in the ranking will be now part of the archive $T_{\mathcal{B}}$ which is used as a reference to get the new $T_{\mathcal{F}}$ and $T_{\mathcal{U}}$. Let us say that the new set of points 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 what the function "Update" does.

**Algorithm 2** A general outline of the $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$ algorithm

```
1: init($T_{\mathcal{F}}$,$T_{\mathcal{U}}$, $\omega$);
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
```

## IV. THE $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(Pen)}$ AND $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(SR)}$ ALGORITHMS

Based on the above modifications for the original $\text{ACO}_{\mathbb{R}}$ we define here two algorithms, $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(Pen)}$ and $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(SR)}$. The name $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(Pen)}$ corresponds to the algorithm called $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(S)}$[2] which was proposed and studied in [13]. The second algorithm, which constitutes the main proposal ot this work, is called here $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(SR)}$ where the complementary constraint-handling technique is the stochastic ranking approach proposed by Runnarson et al. [21]. The main characteristic of $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(Pen)}$ is that the function $\phi$ is used to find the values in the structure $E$ (see Table I) must include a penalty factor in order to evaluate the solutions on the boundary (structure $T_{\mathcal{B}}$). However, it is well known that the main drawback of this technique is the problem to find the most suitable penalty function and/or the corresponding penalty factors involved. In Section V, we will show the penalty function used and the corresponding penalty factors which have been extensively studied in Leguizamón et al. [12], [11], and [13]. For the $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(SR)}$ algorithm, the mechanism is slightly different, nevertheless, its implementation is straightforward. First of all, it is necesaty to include another structure associated to archive $T_{\mathcal{B}}$ to keep the extent of violation of the problem constraints. Let us call this new structure $V = \{\nu(\mathbf{b}_1), \ldots, \nu(\mathbf{b}_l), \ldots, \nu(\mathbf{b}_k)\}$, where $\nu$ is a function that returns precisely the extent of violations of the problem constraints given by $\nu(\mathbf{b}) = \sum_{j=1}^{q} max\{0, g_j(\mathbf{b})\}^2 + \sum_{j=q+1}^{m} |h_j(\mathbf{b})|^2$ (similarly as defined in [21], however, any other suitable function can be applied) and $\phi = f$, i.e., the objective function. After that, function "Sort" (line 7, Algorithm 2) should be accordingly changed. Following the proposal of Runarsson et al., the former function "Sort" which implements any classical sorting algorithm, is modified now in the way that implements a sort-like procedure (see Algorithm 3) to proceed with the

---

[2] We change the name here since both algorithms are based on Socha et al.'s proposal where the difference is in the complementary constraint-handling technique.

stochastic ranking of the newly generated solutions. It should be noticed that the indexes $I_j$ and $I_{j+1}$ in function "swap" point to the corresponding structures (e.g., $T_{\mathcal{B}}$ or other) to produce the swaps when necessary. Runarsson et al. suggest the setting $N = \lambda$ for the number of solutions adjacent to be compared. In our case, $\lambda$ indicated the number of solutions in the corresponding structure to be sorted. Thus, if $|T_{\mathcal{B}}| = k$ and $|A_{\mathcal{B}}| = Na$, then $\lambda = k+N_a$ (see line 8 in Algorithm 3).

**Algorithm 3** A general outline of the stochastic ranking algorithm usign a bubble-sort like algorithm as defined in [21]. $P_f$ represents the probability of using only the objective function for comparisons in ranking in infeasible regions of the search space for which a value of $0.4 < P_f < 0.5$ was reported as the most appropiate. Parameters $N$ and $\lambda$ represents respectively the maximum number of sweeps and the number of solutions that are ranked by comparing adjacent solutions in at least $\lambda$ sweeps, and $rnd \in U(0,1)$.

```
1:  $I_j = j, \forall j\{1,\ldots,\lambda\}$
2:  for $i$ in $1 : N$ do
3:    for $j$ in $1 : \lambda - 1$ do
4:      if $((\nu(\mathbf{x}_{I_j}) == \nu(\mathbf{x}_{I_{j+1}}))||(rnd < P_f))$ then
5:        if $(\nu(\mathbf{x}_{I_j}) > \nu(\mathbf{x}_{I_{j+1}}))$ then
6:          swap($I_j, I_{j+1}$)
7:        end if
8:      else
9:        if $(\nu(\mathbf{x}_{I_j}) > \nu(\mathbf{x}_{I_{j+1}})$ then
10:         swap($I_j, I_{j+1}$)
11:       end if
12:     end if
13:   end for
14:   if no swap done then
15:     break
16:   end if
17: end for
```

## V. EXPERIMENTS AND RESULTS

The main objective of our experimental study is to analyze the quality of results as well as the performance of ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ and ACO$^{(SR)}_{\mathcal{B}\mathbb{R}}$ regarding the number of feasible solutions found. In addition, we make a comparison with one of these two algorithms and the original stochastic ranking approach, as described in [21] (using an evolution strategy as its search engine). Before presenting the results we will describe some common characteristics of ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ and ACO$^{(SR)}_{\mathcal{B}\mathbb{R}}$ regarding their application to the different test cases. Indeed, ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ and ACO$^{(SR)}_{\mathcal{B}\mathbb{R}}$ 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 policies to deal with the way of approaching to the boundary were extensively studied in Leguizamón et al. [12], [11] for the algorithm ACO$^{(B)}_{\mathcal{B}\mathbb{R}}$. From these earlier results, we adopt the so called S$_{act}$ policy, which showed the best performance in all the test cases studied. However, the other policies 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 is represented by the parameter $t_c$ as explained in Section II-C. In addition, for problems with more than one constraint, we incorporate a penalty function for algorithm ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ of the form:

$$\phi(x,\mu) = f(x)+\mu(\sum_{j=1}^{q}\max\{0,g_j(x)\}+\sum_{j=q+1}^{m}|h_j(x)|) \quad (5)$$

where $\mu$ is a fixed penalty factor. Also, it is worth remarking that each solution is always lying on the boundary of the feasible space corresponding to the constraint under consideration. This sort of penalty function was previously adopted in [12], [11] due to its simplicity, since our interest was to assess the advantages of the boundary approach proposed. However, other constraint-handling techniques are evidently possible as the stochastic ranking approach proposed in this article (algorithm ACO$^{(SR)}_{\mathcal{B}\mathbb{R}}$). The penalty factors $\mu$ used in ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ were experimentally determined for each particular problem (see [13]) and are shown later. All the algorithms considered in this experimental study (i.e., ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$, ACO$^{(SR)}_{\mathcal{B}\mathbb{R}}$, and SR) 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 $G07$, $G09$, $G10$, $G11$, $G13$, $G14$, $G15$, $G17$, $G21$, $G23$, $G24$ [16], and $G25$ [7]. At earlier experiments with ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ in [13], we initially chose similar parameter settings as those used in [24] where $N_a = 2$, $k = 50$, $\xi = 0.85$, and $q \in \{0.0001, 01\}$; where the higher value for parameter $q$ was chosen for multimodal functions. The preliminary results from ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ by using the above parameter setting was rather discouraging since the algorithm was not capable of achieving any feasible solution for all the test problems adopted. 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 $N_a = 50$ which was the setting for the number of ants used in ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ and ACO$^{(SR)}_{\mathcal{B}\mathbb{R}}$ in the experiments presented in this section. The penalty factors involved in function $\phi$ (Eq. 5) for each problem using ACO$^{(Pen)}_{\mathcal{B}\mathbb{R}}$ 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 $G24$ ($\mu = 10000$).

TABLE II

RESULTS FROM ACO$_{\mathcal{BR}}^{(Pen)}$ AND ACO$_{\mathcal{BR}}^{(SR)}$ ACCORDING TO THE PARAMETER SETTING $q = 0.0001$ AND $\xi = 0.85$ USED FOR SOME TEST CASES IN [24]. THE REMAINING PARAMETER VALUES USED IN THE EXPERIMENT ARE $k = 50$, $N_a = 50$, AND $T_{max} = 10000$. THE PARAMETER SETTING FOR ACO$_{\mathcal{BR}}^{(SR)}$ WAS $P_f = 0.45$ FOR ALL PROBLEMS, EXCEPT FOR PROBLEM $G23$ FOR WHICH $P_f = 0.2$.

| | ACO$_{\mathcal{BR}}^{(SR)}$ | | | | ACO$_{\mathcal{BR}}^{(Pen)}$ | | | |
|---|---|---|---|---|---|---|---|---|
| Prob. | BF | Mean | Worst | #Fea | BF | Mean | Worst | #Fea |
| G01 | -15.000 (*) | -15.000 | -15.000 | 30 | -15.000 (*) | -15.000 | -15.000 | 30 |
| G04 | -30665.539 (*) | -30665.539 | -30665.539 | 30 | -30665.539 (*) | -30665.539 | -30665.539 | 30 |
| G05 | **5126.49** (*) | 5127.8387 | 5178.7558 | 30 | 5126.5083 | 5143.6240 | 5159.6303 | 27 |
| G06 | - 6961.814 (*) | -6961.813 | -6961.8129 | 30 | -6961.814 (*) | -6961.8137 | -6961.813 | 30 |
| G07 | 24.306 (*) | 24.537 | 24.832 | 30 | 24.306 (*) | 24.530 | 24.985 | 25 |
| G09 | 680.630 (*) | 680.630 | 680.630 | 30 | 680.630 (*) | 680.630 | 680.630 | 30 |
| G10 | **7049.3261** (+) | 7155.9941 | 7368.4658 | 30 | 7058.3559 | 7208.0776 | 7506.7651 | 28 |
| G13 | 0.05394 (*) | 0.054003 | 0.054894 | 30 | 0.053951 | 0.054112 | 0.054637 | 23 |
| G14 | **-47.76489** (*) | -47.683451 | -47.451402 | 30 | -47.624847 | -45.268413 | -41.556510 | 28 |
| G15 | 961.7150 (*) | 961.7150 | 961.7150 | 30 | 961.71515 | 961.71496 | 961.71520 | 30 |
| G17 | **8854.3105** (+) | 8963.7792 | 8963.7792 | 16 | 8871.682 | 9029.559 | 9212.925 | 29 |
| G21 | **193.72828** (+) | 194.1571 | 194.6119 | 20 | 193.79061 | 193.83093 | 193.90968 | 6 |
| G23 | **-303.5474** | 22.5463 | 170.625 | 17 | -300.80877 | -49.064338 | 130.72998 | 4 |
| G24 | 5.50801 (*) | 5.50801 | 5.50801 | 30 | 5.50801 (*) | 5.50801 | 5.50801 | 30 |

All of these values were set based on our previous work [12] in which similar values were adopted for the so called ACO$_{\mathcal{BR}}^{(B)}$. On the other hand, for ACO$_{\mathcal{BR}}^{(SR)}$ we set $P_f = 0.45$ and $N = \lambda$. 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$_{\mathcal{BR}}^{(B)}$ algorithm was implemented in C Language running under Suse-Linux. It is important to remark that the test suite considered includes problems with only one constraint. For these problems ($G02$,[3] $G03$, $G11$, and $G25$), the application of either ACO$_{\mathcal{BR}}^{(Pen)}$ or ACO$_{\mathcal{BR}}^{(SR)}$ gives the same results. The reason is because when a problem has only one constraint, the boundary approach generates only feasible solutions, i.e., there is no need to use a complementary constraint-handling technique. Thus, both ACO approaches still have the same performance, because their search engine is exactly the same. However, we will further show the results for these problems when comparing the performance of ACO$_{\mathcal{BR}}^{(SR)}$ with SR.

Table II displays the results from algorithms ACO$_{\mathcal{BR}}^{(SR)}$ and ACO$_{\mathcal{BR}}^{(Pen)}$ for the test problems with more than one constraint. The columns show respectively the best found (BF), mean (Mean), and worst (Worst) values and the number of feasible solutions found out of 30 independent runs. The (*) in column BF means that the algorithm achieved the best known or optimal value whereas (+) means the the best found value is very close to the best known or optimal value. In column BF some values are in boldface indicating that the corresponding algorithm found the best value. It can be observed that for $G01$, $G04$, $G06$, $G07$, $G09$, $G13$, $G15$, and $G25$ the two algorithms behave very similarly regarding quality of solutions. However, the number of feasible solutions found is always 30 for ACO$_{\mathcal{BR}}^{(SR)}$. On the other hand, we can observe that ACO$_{\mathcal{BR}}^{(SR)}$ achieved a better performance for problems $G05$, $G10$, $G14$, $G17$, $G21$, and $G23$, considering both, the quality of results and number of feasible solutions found. It is important to remark that for problem $G23$, we used $P_f = 0.2$ in order to obtain feasible solutions, otherwise (with $P_f = 0.45$), ACO$_{\mathcal{BR}}^{(SR)}$ was not able to find any feasible solutions. Finally, Table III shows the results from ACO$_{\mathcal{BR}}^{(SR)}$ and SR for some problems. The remaining problems considered are not shown since both algorithms perform similarly, including those problems with only one constraint. However, for problems $G02$, $G10$, $G14$, $G17$, $G21$, and $G24$ these two algorithms behave differently. For $G02$, ACO$_{\mathcal{BR}}^{(SR)}$ found the best known value for this problem, however, for $G10$, SR found a slightly better value than ACO$_{\mathcal{BR}}^{(SR)}$. For problems $G14$, $G21$, and $G23$, the results shows a more clear difference between these two algorithms where ACO$_{\mathcal{BR}}^{(SR)}$ clearly outperforms SR (it must be noticed that we set for ACO$_{\mathcal{BR}}^{(SR)}$and SR, $P_f = 0.45$ except for $G23$ for which $P_f = 0.2$)

TABLE III

COMPARISON OF ACO$_{\mathcal{BR}}^{(SR)}$ WITH RESPECT TO STOCHASTIC RANKING. FOR ALL PROBLEMS AND BOTH ALGORITHMS COMPARED, WE SET $P_f = 0.45$ EXCEPT FOR $G23$ FOR WHICH $P_f = 0.2$. NUMBERS IN BOLDFACE INDICATE WHICH ALGORITHM FOUND THE BEST VALUE.

| | | ACO$_{\mathcal{BR}}^{(SR)}$ | | SR | |
|---|---|---|---|---|---|
| Prob. | Opt | BF | #Fea | BF | #Fea |
| G02 | 0.803619 | **0.803619** | 30 | 0.803515 | 30 |
| G10 | 7049.248 | 7049.361 | 30 | **7049.331** | 30 |
| G14 | -47.7648 | **-47.7648** | 30 | -42.5805 | 30 |
| G17 | 8853.5396 | **8854.3105** | 16 | 8856.1360 | 30 |
| G21 | 193.7245 | **193.72828** | 20 | NA | NA |
| G23 | -400.055 | **-303.5474** | 17 | -46.047 | 2 |

## VI. DISCUSSION

In this paper, we presented an alternative ACO algorithm (ACO$_{\mathcal{BR}}^{(SR)}$) with a a new search engine for implementing the boundary search approach. The search engine is an adaptation of a recent proposal for continuous problems (ACO$_{\mathbb{R}}$). The new algorithm, called ACO$_{\mathcal{BR}}^{(SR)}$ includes stochastic ranking as a complementary mechanism for problems with more than one constraint. For testing ACO$_{\mathcal{BR}}^{(SR)}$, we have

---

[3] We considered this problem as having one constraint.

also used a version with a penalty function ($\text{ACO}_{\mathcal{B}\mathbb{R}}^{(Pen)}$). The results showed a better performance of $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(SR)}$ with respect to $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(Pen)}$, specially regarding the number of feasible solutions. In addition, the overall performance of $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(SR)}$ was compared to SR, showing the potential of this method as an alternative or complementary approach for constrained optimization problems. Future works include the use of a hybrid version of $\text{ACO}_{\mathcal{B}\mathbb{R}}^{(SR)}$with local search, e.g., by doing the main exploration on $\mathcal{G}$ and a complementary exploration on $\mathcal{B}$. In addition, we are also interested in the design of a more general approach which includes the boundary approach as a component that can be triggered when certain conditions are met.

## ACKNOWLEDGMENT

The second author gratefully acknowledges support from CONACyT project no. 42435-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. C. Parmee. The ant colony metaphor for searching continuous design spaces. *Lecture Notes in Computer Science*, 993:25–39, 1995.

[3] 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.

[4] D. Corne, M. Dorigo, and F. Glover, editors. *New Ideas in Optimization*. McGraw-Hill International, 1999.

[5] M. Dorigo and T. Stützle. *Ant Colony Optimization*. Mit-Press, 2004.

[6] 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.

[7] 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.

[8] Michael Guntsch and Martin Middendorf. A population based approach for aco. In *EvoWorkshops*, pages 72–81, 2002.

[9] Andy Keane. Genetic algorithms digest, v8n16, 1994.

[10] G. Leguizamón and C. Coello Coello. A boundary ACO algorithm with stochastic ranking. (In preparation).

[11] G. Leguizamón and C. Coello Coello. Boundary search for constrained numerical optimization in ACO algorithms (an extended version of [12]). (Submitted).

[12] G. Leguizamón and C. Coello Coello. Boundary Search for Constrained Numerical Optimization Problems in ACO algorithms. In Marco Dorigo, Luca Maria Gambardella, Mauro Birattari, Alcherio Martinoli, Riccardo Poli, and Thomas Stützle, editors, *Ant Colony Optimization and Swarm Intelligence*, pages 108–119. Springer. Lecture Notes in Computer Science Vol. 4150, September 2006.

[13] G. Leguizamón and C. Coello Coello. An Advanced ACO Algorithm Implementing Boundary Search for Constrained Numerical Optimization Problems. Technical Report EVOCINV-01-2007, Evolutionary Computation Group (EVOCINV), Computer Science Department, CINVESTAV-IPN, http://delta.cs.cinvestav.mx/˜ccoello/2007.html, 2007.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, 2000.

[22] 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.

[23] 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.

[24] Krzysztof Socha and Marco Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 2007. In press.

[25] 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.