Boundary Search for Constrained Numerical Optimization Problems with an Algorithm Inspired on the Ant Colony Metaphor

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

[1]LIDIC - Universidad Nacional de San Luis
Ejército de los Andes 950
San Luis 5700, ARGENTINA
legui@unsl.edu.ar

[2]CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Computación
Av. IPN No. 2508. Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO
ccoello@cs.cinvestav.mx

Abstract

This paper presents a novel boundary approach which is included as a constraint-handling technique in an algorithm inspired on the ant colony metaphor. The necessity of approaching the boundary between the feasible and infeasible search space for many constrained optimization problems is a paramount challenge for every constraint-handling technique. Our proposed technique precisely focuses the search on the boundary region and can be either used alone or in combination with other constraint-handling techniques depending on the type and number of problem constraints. For validation purposes, an algorithm inspired on the ant colony metaphor is adopted as our search engine which works following one of the principles of the ant colony approach, i.e., a population of agents iteratively, cooperatively, and independently search for a solution. Each ant in the distributed algorithm applies a simple mutation-like operator which explores the neighborhood region of a particular point in the search space (*individual search level*). The operator is designed for exploring the boundary between the feasible and infeasible search space. In addition, each ant obtains global information from the colony in order to

exploit the most promising regions of the search space (*cooperation level*). We compare our proposed approach with respect to a well-known constraint-handling technique that is representative of the state-of-the-art in the area, using a set of standard test functions.

1 Introduction

Ant System (AS) [1] 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: AS with an *elitist strategy* for updating the pheromone trail levels, AS_{rank} (a rank-based version of Ant System) [2], $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [3], and the Ant Colony System (ACS) [4]; all of them originally designed to operate on combinatorial optimization problems. A detailed description of these versions can be found in [5].

On the other hand, it is interesting to note that several applications inspired by the ant colony metaphor, described in the following, have been developed to operate on continuous spaces. In addition, it is worth remarking that all of these applications fit well under the swarm intelligence framework, which includes the algorithms based on the ACO approach. More precisely, Bilchev and Parmee [6] proposed an algorithm for continuous spaces 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 algorithms inspired by the ant colony metaphor to continuous search spaces. However, all of these approaches only deal with unconstrained optimization problems. For example, Ling et al. [7] report a general proposal for 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 and Qidi [8, 9] also take some ideas from the ant colony metaphor to design optimization algorithms for continuous spaces by dividing the search space into n subregions —i.e., a discrete view of the search space, however, different from the discretization initially proposed by Bilchev. Lei and Qidi's approach was applied to a set of one-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, overlapping 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 interval in which an ant will deposit its pheromone is chosen according to a probability value which 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, Dréo and Siarry [10] proposed an alternative algorithm for continuous spaces inspired by the ant colony metaphor which introduces 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[1] 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. [13] presented an algorithm called API which implements a parallel search scheme in the solution space based on the definition of hunter sites (points in the search space), which are established based on 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. On the other hand, Pourtakdoust and Nobahari [14] propose another algorithm inspired by the ant colony metaphor 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 [13] and a genetic algorithm (GA).

Finally, a recent extension of the ACO metaheuristic to continuous domains and applied to continuous and mixed discrete-continuous problems is presented by Socha [15] and Socha and Dorigo [16] which can be considered the first proposal that 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 function (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 $a_i \leq x_i \leq b_i$. The experimental study involves a set of continuous unconstrained problems and the results are better than other ACO inspired/based algorithms and competitive with respect to some other non-ACO algorithms.

In this paper, we introduce a novel boundary approach for solving nonlinear constrained problems, which is also inspired by the ant colony metaphor. It is worth noting, however, that our proposal can be coupled to

[1] According to the authors, this concept is similar to that used with Particle Swarm Optimization in [11] and to path-relinking [12].

other metaheuristics (e.g., particle swarm optimization or an evolutionary algorithm), and it is expected to be highly competitive in problems with active constraints. Our approach is mainly based on the work of Bilchev and Parmee [6]. The reason for not adopting one of the more recent ACO inspired/based approaches for continuous search spaces is that our main aim was to emphasize the performace-related aspects of the boundary approach rather than focusing on a possibly more advanced search engine.

The remainder of this paper is organized as follows. Section 2 presents a brief description of the eariler works on boundary search using evolutionary algorithms. Section 3 describes our proposal. The ant colony inspired (ACI) algorithm, which is the search engine used to study the applicability of our proposed boundary approach is presented in Section 4. The test problems and experimental results are presented and analyzed in Section 5. Finally, our conclusions and some possible paths for future research are provided in Section 6.

2 Constraint-Handling and Boundary Search

Michalewicz et al. [17] wrote one of the first papers on boundary search through the use of evolutionary algorithms. The efficiency of this approach was shown by using two constrained optimization problems: Keane's function (also known as $G02$) [18] and another function with one equality constraint (also known as $G03$). For solving $G02$ the authors proposed two genetic operators: the *geometrical* crossover and a special mutation operator. Both operators generate offspring lying on the boundary between the feasible and infeasible search space. Similarly for $G03$, they proposed the *spherical* crossover which only generates points on the surface of the sphere given as the only constraint.

Schoenauer and Michalewicz [19] proposed several evolutionary operators capable of exploring a general surface of dimension $n - 1$ (n is the number of variables). The design of these operators depends on the surface representation: curves-based, plane-based, and parametric representation.

Wu and Simpson [20] 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.

The reduction of the search space is one of the most relevant characteristics of the boundary search approach since the exploration considers only the boundary of the feasible search space. However, many of the test cases considered so far by other researchers only include problems with one or two constraints (e.g., $G02$ and $G03$[2], respectively). In these cases, it is 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. This was precisely the motivation for the research reported in this paper.

[2]Keane's function can be considered as having one constraint since one of them is ignored. Therefore, the search proceeds focusing only on the active constraint.

3 An Alternative Boundary Search Approach

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

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

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):

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

where n is the number of decision variables, and $l(i)$ and $u(i)$ are, respectively, the lower and upper bounds of each decision variable x_i. The feasible set $\mathcal{F}$ is defined by the intersection of $\mathcal{S}$ and a set of additional $m \geq 0$ constraints:

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

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 $\mathcal{F}$.

In the following we first explain the main characteristics of the boundary operator designed to approach the boundary of a particular constraint. Afterwards, we describe in detail the proposed technique that takes advantage of the boundary operator to explore some specific regions of the boundary of the feasible search space.

3.1 The boundary operator

We propose here a general boundary operator 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 and infeasible (shadowed area) 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 $(\mathbf{x}_4, \mathbf{y}_4)$.

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($\mathbf{m}$)

$\leq \xi$) 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 ξ is given by $log_2(r)$ where $r = (dist(\mathbf{x}, \mathbf{y})))/\xi$ and function $dist$ represents the Euclidean distance between points $\mathbf{x}$ and $\mathbf{y}$. Thus, the closer to the boundary, the larger $log_2(r)$.

Algorithm 1 BS($\mathbf{x}$,$\mathbf{y}$: 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)≤ ξ) AND (Feasible(m));
13: return m; {The closest point to the boundary according to ξ }
```

Given one feasible and one infeasible point, function BS returns either a point on the boundary or one which is close enough to the boundary according to a parameter ξ. The **until** condition is applied as it when considering an inequality constraint, otherwise "AND (Feasible($\mathbf{m}$))' is dropped.

So far, we have shown how a point lying on the boundary can be represented through a pair of points. Now we need to consider the exploration of the search space. For example, from the perspective of evolutionary algorithms, the candidate operators are the classical crossover and mutation. However, for the ACI approach proposed in this work we suggest the application of any specialized real-coded mutation-like operator (the particular mutation operator proposed for our implementation will be further described in section 4). Independently of the selected mutation operator; it should behave as follows: given a pair of points $(\mathbf{x}, \mathbf{y})$, one point feasible and the other one infeasible, any or both of them undergo mutation. For example, we can consider the pair of points $(\mathbf{x}_4, \mathbf{y}_4)$ in Figure 1 (lower-right) which represents point $\mathbf{b}_4$ on the boundary. In this case, the feasible point $\mathbf{y}_4$ undergoes mutation, giving as a result a point $\mathbf{y}'_4$ in the feasible search space. After this process, the new point lying on the

boundary is obtained by decoding $(\mathbf{x}_4, \mathbf{y}'_4)$, which gives us $\mathbf{b}'_4$.

3.2 The proposed method

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. For the last case, 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 on the boundary of the space defined by any of the constraints.

For 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. In order to show the robustness of our method in the absence of information about the active constraints of a problem, we will show in our experimental study (see Section 5) a more general approach to apply the boundary operators. As an illustrative example, Figure 2 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 2). However, the application of the boundary operator with respect to constraint g_1 will eventually produce points violating constraints g_2 and g_3 (dotted line in Figure 2). 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.

4 Boundary Approach in an ACI algorithm

A possible design to apply some of the principles of the ant metaphor in continuous search problems is by discretizing the continuous search space in some way. In this work we use a discrete structure to represent a set of different points spread on the search space. These points are called *directions*, following Bilchev and Parmee'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 deteails see [6]. In our proposal, 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 3 shows (left) a discrete structure with $k = 4$ search directions and (right) the corresponding 4 points on a 2-dimensional search space. The 4 points are the result of the corresponding application of function BS on the 4 hypothetical directions.

A general outline of the ACI algorithm is shown in Algorithm 2. It is worth remarking that the original ACI proposal [6] for continuous domains is used to proceed with the local exploration after a genetic algorithm has finished with the global search. However, the algorithm proposed here, is in charge of performing the entire search process. More precisely, our ACI 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 ACI algorithm

```
t = 0
initialize A(t)
evaluate A(t)
while (stop condition not met) do
  t = t + 1
  update_dirs_trail
  allocate_ants A(t)
  evaluate A(t)
end while
```

General outline of the ACI algorithm for continuous problems. In the Appendix (see Algorithm 3) we show a more detailed version of the algorithm adopted in this paper including its more important components.

The ACI algorithm (see Algorihm 2) works as follows: `initialize A(t)` generates k random directions, sets the initial values for the trail structure, and "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 l generates a new solution through any valid 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_dirs_trail` is in charge of updating the k directions (according to the solutions found) and accumulating pheromone trail 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 l and $0 \leq \rho \leq 1$ is the pheromone trail evaporation rate; `allocate_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 on charge of searching in the neighborhood of the corresponding 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. Figure 4 shows a hypothetical situation with 9 ants and 3 search directions.

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

1. *individual search*: involves the strategy followed by each ant to search in its neighborhood. In our case, we have chosen for our implementation a mutation-like operator ψ[3], such that $\psi(\mathbf{x}, \mathbf{y}) = (\mathbf{x}', \mathbf{y}')$ 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 trail structure (τ) where τ_l represents the accumulation of pheromone trail on direction l. 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 $\Delta_R(t) = R(1 - r^{(1-t/T)})$ where r is a random number in the range $[0..1]$ and T 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.

Figure 5 represents the successive points (2-dimensional vectors) on direction l at iteration t where $\mathbf{p}_l^t = BS(d_l^t)$, i.e., a point obtained by the application of function BS on the pair of points represented by d_l at iteration t, in this example $t \in \{0, 1, 2, 3, 4\}$. Thus, a square represents the neighborhood for a particular point. Following the algorithm, at the first iteration a fixed number of ants are distributed on the k directions, i.e., the ants that were allocated to direction l at the iteration $t = 0$ will start the search from point $\mathbf{p}_l^0$. For example, in Figure 5, $\mathbf{p}_l^0$ is the starting point on direction l, $\mathbf{p}_l^1$ is the best point found by the ants allocated to direction l by using $\Delta_R(0)$[4]. As t increases, new regions of the search space are independently explored in each direction. For our example, the remaining successively generated points on direction l are $\mathbf{p}_l^2$, $\mathbf{p}_l^3$, $\mathbf{p}_l^4$, and so on. Thus, the ACI algorithm can be seen as a trajectory approach which simultaneously searches on different directions and exploits the past experience to guide the search towards the most promising regions according to the quality of the results. Furthermore, the accumulated pheromone trail will decrease on directions that produce low-quality solutions due to the effects of the

[3]However, other alternative mutation operators are also possible.

[4]It should be noticed that $\Delta_R(t)$ is not a monotonically decreasing value.

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.

5 Analysis of Results

The application of our approach, called ANT-$\mathcal{B}$oundary (ANT-$\mathcal{B}$ for short) requires 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_j where $j \in \{1, \ldots, m\}$). These three ways of exploring the search space are presented first in our experimental study in order to analyze the performance of the ANT-$\mathcal{B}$ on each of the considered problems. 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. In addition, for problems with more than one constraint, we incorporate a penalty function of the form:

$$\phi(x, \mu) = f(x) + \mu(t)(\sum_{j=1}^{q} \max\{0, g_j(x)\} + \sum_{j=q+1}^{m} |h_j(x)|) \tag{2}$$

where $\mu(t)$ is a dynamic penalty factor which could change as t, the elapsed iteration, increases with $\mu(0) \leq \mu(1) \leq \mu(2) \cdots \leq \mu(T)$. Alternatively, the penalty factor can be fixed throughout the run, i.e., $\mu(t) = \mu_0$ for all $1 \leq t \leq T$. 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 parameter setting used in this experimental study was empirically determined. More precisely, the parameter values are the following: $N_a = 50$ ants (population size), $k = 20$ directions (number of reference points), maximum number of iterations = 30000, the evaporation rate $\rho = 0.5$, $t_c > 0$ is the number of iterations that ANT-$\mathcal{B}$ focuses on one constraint in turn. When $t_c = 0$, ANT-$\mathcal{B}$ focuses on only one constraint throughout the whole run. We set $t_c = 200$ for the policy S_{act} and S_{all}. The penalty factor $\mu(t)$ was experimentally determined for each particular problem and is shown in the corresponding tables of results. ANT-$\mathcal{B}$ was 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 $G13$ [21]. In addition, we consider other problems recently

labelled as $G14$, $G15$, $G17$, $G21$, $G23$, $G24$ [22], and $G25$ [23]. 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 ANT-$\mathcal{B}$ algorithm was implemented in the C programming language running under Suse-Linux.

5.1 Study of the application of ANT-$\mathcal{B}$

We have divided the presentation of the results into two groups according to the following criteria: the first group, is displayed in Tables 1 and 2. Table 1 includes two special cases since they were the first problems on which the boundary approach was applied (problems $G02$ and $G03$). In addition, these problems have one and two constraints respectively. However, the second constraint of problem $G02$ is not considered since it is not active at the best known value. The columns in this table show the setting for the number of variables, the best value found (BF), Mean, Standard Deviation (Std), Worst, number of feasible solutions out of 30 runs (#Fea), and the mean number of evaluations of function $\phi(x, \mu)$ (Eq. 2) to get the best value found (Mean(#E)). On the other hand, Table 2 shows two problems both of which include one equality constraint (problems $G11$ and $G25$). Accordingly, no penalty values (μ) need to be applied for this first group of problems. In the remaining tables, the column "No. of variables" is replaced by "Cnst", indicating the criteria adopted to proceed with the boundary search, i.e., S_j ($j \in \{1, \ldots, m\}$), S_{act}, or S_{all}. In addition, the best known or global optimum value for each problem is shown in parenthesis.

We tested $G02$ setting the number of variables as $n = 20, 50,$ and 100. ANT-$\mathcal{B}$ succeeded in finding the best known value for $n = 20$ [24]. In addition, it was able to find a better quality result than the best objective reported in [19] where $n = 50$ and $f(\mathbf{x}^*) = 0.831937$. For $n = 100$, we found 0.8456841707 as the best value in our experimental study. Also, it is worth remarking that all the solutions found were feasible for all n and very similar among themselves as can be observed in the columns Mean, Std, and Worst. With respect to problem $G03$, we considered $n = 20$ and $n = 50$ variables. ANT-$\mathcal{B}$ found the optimum feasible solution for both cases in all runs. Figure 6 shows a convergence graph for problems $G02$ (left) and $G03$ (right) with $n = 20$ variables. For each of them are plotted the mean best found values out of 30 runs for each generation. It can be observed for problem $G03$ (right) that before iteration 500 the algorithm achieves a mean value close to the optimum, whereas for problem $G02$ it needs about 2000 iterations to approach the corresponding best known value. Similarly to $G02$ and $G03$, in the remaining problems of this group ($G11$ and $G25$), our approach reached the optimum in all cases.

The second group of test cases is conformed by some problems having more than one constraint which have been frequently used in the specialized literature: $G01$, $G04$, $G05$, $G06$, $G07$, $G09$, $G10$, and $G13$. We also include problem $G24$ [22] in this subgroup. Only for $G10$, we adopted a dynamic penalty ($\mu(t) = 1.05 \times \mu(t-1)$ for $t = 0, 1, \cdots, T$; with $\mu(0) = 200000$). The static penalty factors adopted for the remaining problems are (i.e., for $t = 0, 1, \cdots T$): $G01$, $\mu(t) = 1000$; $G04$, $\mu(t) = 800000$; $G05$, $\mu(t) = 10$; $G06$, $\mu(t) = 10000$; $G07$, $\mu(t) = 20000$; $G09$, $\mu(t) = 2000$, $G13$, $\mu(t) = 0.2$; and $G24$, $\mu(t) = 1000$. The results for this group of

problems are displayed in Tables 3 and 4. It must be noticed that these problems include different numbers and complexities of the equality and inequality constraints which are active at the best known or optimum solution. As indicated in column "Cnst.", each row shows the results when ANT-$\mathcal{B}$ was applied adopting one of the following criteria: search exclusively on constraint j (S_j, $j = 1, \ldots, m$), on all the active constraints in turn (S_{act}), and over all the constraints in turn (S_{all}). For example, problem $G01$ has 6 active constraints. Accordingly, ANT-$\mathcal{B}$ performs ideally when searching on those active constraints. Similarly, the algorithm succeeded in finding the optimal solution when using both S_{act} and S_{all}. However, its performance slightly decays when searching on the non active constraints as could be expected. This situation is more dramatic for problem $G04$ which has two active constraints. In this case, ANT-$\mathcal{B}$ only finds high quality feasible solutions when searching with S_1, S_6, S_{act}, and S_{all}. The last case deserves an additional explanation since for S_{all}, constraints 2 and 5 were not considered. In fact, for these constraints the approach was not capable of generating any pair of points $(\mathbf{x}, \mathbf{y})$ as needed to obtain a point $\mathbf{b}$ lying on the boundary (function BS in Algorithm 1). It seems that for these constraints all the solutions are feasible on the corresponding range of values for variables x_i, $i = 1, \ldots, 5$. Therefore, we do not have any boundary between the feasible and the infeasible search space for constraints 2 and 5. Regarding constraints 3 and 5, ANT-$\mathcal{B}$ found no feasible solution at all. However, this can be explained because feasible solutions for this problem are generally far from the boundary of these constraints. It is also important to remark that by using S_6, ANT-$\mathcal{B}$ reported the optimal value for $G04$ ($f(\mathbf{x}^*) = 30665.359$). However, for the search options S_1 and S_{act}, ANT-$\mathcal{B}$ found feasible solutions with an objective value of 30665.542 where all the constraints are feasible (in this case constraints $g_i(\mathbf{x}) \leq 0$, for $i = 1, , \ldots, 5$, and $g_6(\mathbf{x}) = y$ where $0 < y < 10^{-5}$).

A similar situation can be seen for problem $G05$ which has three equality constraints. Accordingly, ANT-$\mathcal{B}$ finds a high quality solution for this problem (very near to the optimal one) when searching on the corresponding equality constraints, S_{act} and S_{all}. On the other hand, problem $G06$ has two inequality constraints which are active at the optimum. ANT-$\mathcal{B}$ performs optimally for this problem by following any of the three applicable strategies: S_1, S_2, and S_{act}. The last problem in Table 3 has six active constraints and ANT-$\mathcal{B}$ performs similarly to $G01$ since the best results were obtained when searching on the active constraints or by using S_{act} or S_{all}.

Problem $G09$ has two active constraints for which ANT-$\mathcal{B}$ found the optimum value. However, searching on the non active constraints can give results far from the expected value (see S_2 and S_3). $G10$ constitutes one of the most difficult test cases not only for our approach, but also for most other constraint-handling techniques. ANT-$\mathcal{B}$ found feasible solutions with all the search strategies except for S_5 and S_6. Note the small number of feasible solutions found for this problem, as well as the large standard deviation value produced (with respect to the deviations of the other problems). Another interesting problem is $G13$ whose feasible search space is defined by three nonlinear equality constraints. For this problem ANT-$\mathcal{B}$ found the optimal solution following any of the four applicable search strategies. Finally, it can be seen that ANT-$\mathcal{B}$ performs optimally on problem $G24$ which has two active inequality constraints where the optimal solution was found for all strategies in each run (see #Fea).

Also, it is worth remarking that ANT-$\mathcal{B}$ needs an important number of evaluations to reach the highest quality

value for each problem as can be observed in the corresponding column Mean(#E). The lowest value for this variable is obtained in most cases when using S_{act} as the search option. In general, the high values for Mean(#E) can be explained due to the design of ANT-$\mathcal{B}$ which readily approaches the feasible region (particularly the boundary between the feasible and infeasible search space). However, the mutation-like operator used here needs several additional iterations to produce improved solutions in the promising search regions.

5.2 Comparison with an state-of-the-art algorithm

In this section we compare the best quality results from ANT-$\mathcal{B}$ (we use S_{act} as the most efficient search criteria) with respect to the results of one of the best constraint-handling technique known to date: Stochastic Ranking (SR) [21]. Table 5 shows for each problem considered, the optimum, and the corresponding Best value found (BF), average (Mean), and Worst values respectively from ANT-$\mathcal{B}$ and SR[5]. The parameter setting used for SR was as follows: $\mu = 30$, $\lambda = 200$, Gaussian Mutation, $\varphi = 1$, $P_f = 1$, $G_m = 1750$, and $\delta = 0.0001$ (see [21] for further details).

The performance of ANT-$\mathcal{B}$ is comparable in many ways with respect to SR. From the perspective of the best values found (BF) ANT-$\mathcal{B}$ reaches similar values as SR in all the problems considered. For $G02$, ANT-$\mathcal{B}$ reached the best known value reported in [17] by using an *ad hoc* boundary operator. On the opposite side, for $G10$, ANT-$\mathcal{B}$ did not obtain the optimal solution. However, the results achieved in all cases are highly competitive.

5.3 Additional testbed for constrained problems

In this section we show the experimental results from the application of ANT-$\mathcal{B}$ and SR to some additional test cases $G14$, $G15$, $G17$, $G20$, $G21$, and $G23$ which have been recently incorporated in [22]. Table 6 displays on its columns, the problem names and the corresponding best known value. On the rows we show, the optimum (or best known value), best value found (BF), Mean, Worst, and the number of feasible solutions found (out of 30 runs) for ANT-$\mathcal{B}$ and SR respectively. The symbol $*$ means that no feasible solutions were found. The penalty values used in ANT-$\mathcal{B}$ were as follows: $G14$, $\mu(0) = 150.8$; $G15$; $\mu = 9.5; G17$, $\mu(0) = 400$; $G21$, $\mu(0) = 1500$; and $G23$, $\mu(0) = 13500$. All of them, except for problem $G15$, used dynamic penalty values as indicated at the beginning of this section. The parameter setting for SR was as described above.

For problem $G15$, both algorithms performed very well, achieving a value that is slightly better than the best value previously known. SR had the best performance for this problem and all the solutions that it obtained were feasible. The following corresponds to the best solution found by the ANT-$\mathcal{B}$: $\mathbf{x}^* = (3.50883921, 0.21725025, 3.55539723)$. In addition, we show for this problem four plots (see Figure 7) corresponding to the 20 directions on the search interval at iterations 1, 10000, 20000, and 30000. It can be observed that at iteration 1 the search

[5]SR was run by the authors using Thomas Runarsson's code, which is available at:
`http://cerium.raunvis.hi.is/~tpr/software/sres/index.html`.

directions are fairly spread on the corresponding search interval. However, at iteration 10000 the ANT-$\mathcal{B}$ starts to converge towards a suboptimal region. This situation changes at iteration 20000 where the distribution of the directions is more spread on a different area. Finally, at iteration 30000, almost all search directions have converged to a tightly clustered region where the best value known so far is located.

With respect to the quality of results, something similar happened in $G17$ where ANT-$\mathcal{B}$ and SR achieved an objective value very close to the best known value recently reported in [22] (SR marginally outperformed our approach). The best solution found by ANT-$\mathcal{B}$ is: $\mathbf{x}^* = (203.25057161, 98.51080589, 383.16031685, 419.98475072, -11.21762319, 0.07194439)$. However, ANT-$\mathcal{B}$ only obtained 3 feasible solutions (in 30 runs) whereas SR obtained 30. Nevertheless, it is remarkable that the final violation of the constraints was very small for our approach as can be observed in Figure 8 which displays the mean value (in logarithmic scale) of constraints' violations (Mean(V)) out of 30 runs for the problems in Table 6 with respect to the best so far solution during the 30000 iterations—we have only plotted the first 3000 iterations since after that, the constraints violations are insignificant with respect to the earlier iterations.

On the other hand, the performance of SR for the other problems is not as good as for problems $G15$ and $G17$. Although SR obtained 30 feasible solutions for problem $G14$, its best value is far from the best known. However, ANT-$\mathcal{B}$ converged 15 times to a feasible solution (out of 30 runs), achieving a very good performance taking into account BF, Mean, and Worst values. For the remaining problems ($G21$ and $G23$), SR was not able to find any feasible solutions, whereas ANT-$\mathcal{B}$ performed well for these two problems. The best found values are very close to the best known. However, for $G23$ there is an important distance between the best and the worst values found. The best solution obtained by ANT-$\mathcal{B}$ for problem $G21$ is: $\mathbf{x}^* = (193.78298439, 0.00000000, 17.32778946, 100.01064200, 6.68460537, 5.99149380, 6.21459710)$ whereas the best solution for problem $G23$ is: $\mathbf{x}^* = (0.00000000, 99.99979243, 0.00000000, 99.99979275, 0.01530904, 0.01531166, 100.00000000, 199.99979277, 0.01000000)$.

Finally, we show in Table 7 a comparison on the number of solution evaluations regarding ANT-$\mathcal{B}$ and SR. Columns $\bar{e}_{\text{ANT-}\mathcal{B}}$ and $\bar{e}_{\text{SR}}$ represent, respectively, the average number of evaluations to obtain the best solution for ANT-$\mathcal{B}$ and SR. It can be observed that for problems $G01$, $G02$, $G03$, $G04$, $G06$, $G07$, $G09$, $G13$, $G24$, and $G25$; $\bar{e}_{\text{ANT-}\mathcal{B}}$ is less than $\bar{e}_{\text{SR}}$ where the difference between these two values is remarkable for some of them. On the other hand, for problems $G10$, $G11$, $G14$, $G15$, and $G17$; $\bar{e}_{\text{SR}}$ is less than $\bar{e}_{\text{ANT-}\mathcal{B}}$. However, for $G10$, ANT-$\mathcal{B}$ outperforms SR considering the quality of the solutions found. For the remaining problems ($G21$ and $G23$), it can be observed a large number of evaluations for ANT-$\mathcal{B}$. Nevertheless, ANT-$\mathcal{B}$ was the only algorithm able to find an acceptable number of good quality feasible solutions for these two problems. It is also worth remarking that SR was run with a larger number of generations on problems $G21$ and $G23$, more precisely $G_m = 3500$ (i.e., 700000 evaluations). In spite of that, SR was not capable of finding any feasible solution for these two problems, despite performing a much larger number of evaluations than before. From the above situation, it is not possible to claim that ANT-$\mathcal{B}$ outperforms SR. As a matter of fact, on average, SR outperforms ANT-$\mathcal{B}$ over the set of test problems

(taken as a whole), if we take into account the number of function evaluations (from Table 7, we can obtain 187428 and 107450, respectively, as the average number of function evaluations for ANT-$\mathcal{B}$ and SR on the whole test set, excluding $G21$ and $G23$). However, if the results are analyzed per test problem, ANT-$\mathcal{B}$ is more consistent in terms of performing less evaluations, since ANT-$\mathcal{B}$ requires less evaluations in 9 problems, and SR requires less evaluations only in 6 problems, and is not able to reach a feasible solution in two more problems).

6 Conclusions and Future Work

In this paper we presented an alternative approach to reach the boundary between the feasible and infeasible search space which could be useful when facing problems with active constraints. For the initial testing of this method we have used an ACI algorithm as a search engine (ANT-$\mathcal{B}$) and a penalty function as a complementary mechanism for problems with more than one constraint. The overall performance of ANT-$\mathcal{B}$ was satisfactory for all of the problems considered. The comparison with a state-of-the-art algorithm shows the potential of this method as an alternative or complementary approach for constrained optimization problems. In fact, for some problems, ANT-$\mathcal{B}$ was able to improve the corresponding best known solutions (e.g., $G02$ (with $n = 50$ variables), $G15$, and $G17$). It is clear that further improvements should be considered. First, it is necessary to incorporate some mechanism aiming at reducing the number of evaluations. In addition, it would be desirable to implement self-adaptation mechanisms and alternative exploration operators. Also, trying a different search engine could be more suitable for the boundary approach. For example, the ACO algorithm proposed by Socha [15] could act as a more appropriate ACO-based search engine. Furthermore, differential evolution [25] and evolution strategies [26] are also highly recommended candidates to be tried as search engines, due to the good performance that they have shown in numerical optimization problems.

Acknowledgments

The authors thank the anonymous reviewers for their valuable comments which greatly helped to improve the contents of this paper.

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] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Trans. on Systems, Man, and Cybernetics–Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[2] B. Bullnheimer, G. Kotsis, and C. Strauss, *Kluwer Series on Applied Optmization*, 1997, ch. Parallelization Strategies for the Ant System, pp. 87–100.

[3] T. Sttzle and H. Hoos, "MAX-MIN Ant System," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[4] M. Dorigo and L. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

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

[6] G. Bilchev and I. Parmee, "The Ant Colony Metaphor for Searching Continuous Design Spaces," in *Evolutionary Computing. AISB Workshop*, T. C. Fogarty, Ed. Sheffield, UK: Springer, April 1995, pp. 25–39.

[7] C. Ling, S. Jie, Q. Ling, and C. Hongjian, "A Method for Solving Optimization Problems in Continuous Space Using Ant Colony Algorithm," in *Proceedings of the Third International Workshop, (ANTS'2002)*, M. Dorigo, G. D. Caro, and M. Sampels, Eds. Brussels, Belgium: Springer Verlag. Lecture Notes in Computer Science Vol. 2463, 2002, pp. 288–289.

[8] W. Lei and W. Qidi, "Ant System Algorithm for Optimization in Continuous Space," in *Proceedings of the 2001 IEEE International Conference on Control Applications*, Mexico City, Mexico, September 2001, pp. 395–400.

[9] ——, "Further Example Study on Ant System Algorithm based Continuous Space Optimization," in *Proceedings of the 4th Congress on Intelligent and Automation*, Shangai, P.R. China, June 2002, pp. 2541–2545.

[10] J. Dréo and P. Siarry, "A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions," in *Proceedings of the Third international Workshop on Ant Algorithms - ANTS 2002*, M. Dorigo, G. Di Caro, and M. Sampels, Eds. Brussels, Belgium: Springer-Verlag. Lecture Notes in Computer Science Vol. 2463, September 2002, pp. 216–221.

[11] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. IV. Perth, Australia: IEEE Service Center, 1995, pp. 1942–1948.

[12] F. Glover and M. Laguna, *Tabu Search*. Boston/Dordrecht/London: Kluwer Academic Publishers, 1997.

[13] N. Monmarché, G. Venturini, and M. Slimane, "On how pachycondyla apicalis ants suggest a new search algoritm," *Future Generation Computer Systems*, vol. 16, pp. 937–946, 2000.

[14] S. Pourtakdoust and H. Nobahari, "An Extension of Ant Colony Systems to Continuos Optimization Problems," in *Proceedings of Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS Workshop 2004*, M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, Eds. Brussels, Belgium: Springer-Verlag, 2004, pp. 294–301. Lecture Notes in Computer Science Vol. 3172.

[15] K. Socha, "ACO for Continuos and Mixed-Variable Optimization," in *Proceedings of Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS Workshop 2004*, M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, Eds. Brussels, Belgium: Springer-Verlag. Lecture Notes in Computer Science Vol. 3172, 2004, pp. 25–36.

[16] K. Socha and M. Dorigo, "Ant Colony Optimization for Continuous Domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1115–1173, March 2008.

[17] Z. Michalewicz, G. Nazhiyath, and M. Michalewicz, "A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems," in *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, L. J. Fogel, P. J. Angeline, and T. Bäck, Eds. Cambridge, MA: The MIT Press, 1996, pp. 305–311.

[18] A. J. Keane, "Experiences with optimizers in structural design," in *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control*, I. C. Parmee, Ed. Plymouth, UK: University of Plymouth, 1994, pp. 14–27.

[19] M. Schoenauer and Z. Michalewicz, "Evolutionary Computation at the Edge of Feasibility," in *Parallel Problem Solving from Nature – PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin: Springer, 1996, pp. 245–254.

[20] Z. Wu and A. Simpson, "A self-adaptive boundary search genetic algorithm and its application to water distribution systems," *Journal of Hydraulic Research*, vol. 40, no. 2, pp. 191–203, 2002.

[21] T. P. Runarsson and X. Yao, "Stochastic Ranking for Constrained Evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[22] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. C. Coello, and K. Deb. (2006). Problem Definitions and Evaluation Criteria for the CEC Special Session on Constrained Real-Parameter Optimization, Nanyang Technological University, Singapore. [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-06/cec2006.zip

[23] C. A. Floudas and P. M. Pardalos, *A collection of test problems for constrained global optimization algorithms*. Berlin: Springer-Verlag. Lecture Notes in Computer Science Vol. 453, June 1990.

[24] S. Hamida and M. Schoenauer, "ASCHEA: New Results Using Adaptive Segregational Constraint Handling," in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC'2002)*, vol. 1. Piscataway, New Jersey: IEEE Service Center, May 2002, pp. 884–889.

[25] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution. A Practical Approach to Global Optimization*. Berlin: Springer, 2005.

[26] H. Schwefel, *Evolution and Optimum Seeking*. New York: John Wiley & Sons, 1995.

[27] D. Himmelblau, *Applied Nonlinear Programming.* New York: McGraw-Hill, 1972.

[28] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes.* Berlin, Germany: Springer-Verlag. Lecture Notes in Economics and Mathematical Systems Vol. 187, 1981.

[29] S. Koziel and Z. Michalewicz, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[30] T. G. W. Epperly and R. Swaney. (1996). Global optimization test problems with solutions. [Online]. Available: http://citeseer.ist.psu.edu/147308.html

[31] Q. Xia. (1996). Global optimization test problems. [Online]. Available: http://www.mat.univie.ac.at/˜neum/glopt/xia.txt

Appendix: Problems

- **G01** [23]

  Minimize:

$$f(\mathbf{x}) = 5 \cdot \sum_{i=1}^{4} x_i - 5 \cdot \sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

  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 $0 \leq x_i \leq 1$ $(i = 1, ..., 9)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$ and $0 \leq x_{13} \leq 1$. 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** [18]

  Maximize:

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

  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.

- **G03** [17]

  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** [27]

  Minimize:

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

  subject to:

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

  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** [28]

  Minimize:

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

  subject to:

$$g_1(\mathbf{x}) = -x_4 + x_3 - 0.55 \leq 0$$
$$g_2(\mathbf{x}) = -x_3 + x_4 - 0.55 \leq 0$$
$$h_3(\mathbf{x}) = 1000sin(x_3 - 0.25) + 1000sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$
$$h_4(\mathbf{x}) = 1000sin(x_3 - 0.25) + 1000sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$
$$h_5(\mathbf{x}) = 1000sin(x_4 - 0.25) + 1000sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

where $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$, and $-0.55 \leq x_4 \leq 0.55$. The best known solution [29] is $\mathbf{x}^* = (679.94453, 1026.067, 0.1188764, -0.3962336)$ where $f(\mathbf{x}) = 5126.4981$.

- **G06** [23]

  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 10).

- **G07** [28]

  Minimize:

$$\begin{aligned} f(\mathbf{x}) \quad = \quad & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ & +29x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_110 - 7)^2 + 45 \end{aligned}$$

  subject to:

$$g_1(\mathbf{x}) = -105 + 4x_1 + 5x_2 - 3x_7 - 9x_8 \leq 0$$
$$g_2(\mathbf{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$
$$g_3(\mathbf{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$
$$g_4(\mathbf{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$
$$g_5(\mathbf{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$
$$g_6(\mathbf{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - x_6 \leq 0$$
$$g_7(\mathbf{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$
$$g_8(\mathbf{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

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** [28]

  Minimize:

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

  subject to:

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

  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** [28]

  Minimize:

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

  subject to:

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

  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 best known solution is $\mathbf{x}^* = (584.3282028010, 1354.1644876700, 5110.7156493300, 182.4326280510, 295.5675740820, 217.5673719490, 286.8650539690, 395.5675740820)$, where $f(\mathbf{x}^*) = 7049.2083398$[6].

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

- **G11** [29]

  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** [28]

  Minimize:

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

  subject to:

$$\begin{aligned} h_1(\mathbf{x}) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ h_2(\mathbf{x}) &= x_2 x_3 - 5 x_4 x_5 = 0 \\ h_3(\mathbf{x}) &= x_1^2 + x_2^3 + 1 = 0 \end{aligned}$$

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

- **G14** [27]

  Minimize:

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

  subject to:

$$\begin{aligned} h_1(\mathbf{x}) &= x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0 \\ h_2(\mathbf{x}) &= x_4 + 2x_5 + x_6 + x_7 - 1 = 0 \\ h_3(\mathbf{x}) &= x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0 \end{aligned}$$

  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** [27]:

  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 $x^* = (3.51211626026935, 0.216988345475683, 3.55217615445509)$ where $f(x^*) = 961.715172$.

- **G17** [27]

  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:

$$h_1(\mathbf{x}) = -x_1 + 300 - \frac{x_3x_4}{131.078}cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}cos(1.47588)$$

$$h_2(\mathbf{x}) = -x_2 - \frac{x_3x_4}{131.078}cos(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}cos(1.47588)$$

$$h_3(\mathbf{x}) = -x_5 - \frac{x_3x_4}{131.078}sin(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}sin(1.47588)$$

$$h_4(\mathbf{x}) = 200 - \frac{x_3x_4}{131.078}sin(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}sin(1.47588)$$

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 $\mathbf{x}^* = (201.784467214523659, 99.9999999999999005, 383.071034852773266, 420, -10.9076584514292652, 0.0731482312084287128)$ where $f(x^*) = 8853.53967480648$.

- **G21** [30]

  Minimize:

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

  subject to:

$$\begin{aligned}
g_1(\mathbf{x}) &= -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0 \\
h_1(\mathbf{x}) &= -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0 \\
h_2(\mathbf{x}) &= 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0 \\
h_3(\mathbf{x}) &= -x_5 + ln(-x_4 + 900) = 0 \\
h_4(\mathbf{x}) &= -x_6 + ln(x_4 + 300) = 0 \\
h_5(\mathbf{x}) &= -x_7 + ln(-2x_4 + 700) = 0
\end{aligned}$$

  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 $x^* = (193.783493, 0, 17.3272116, 100.0156586, 6.684592154, 5.991503693, 6.214545462)$ where $f(x^*) = 193.7783493$.

- **G23** [31]

  Minimize:

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

  subject to:

$$\begin{aligned}
g_1(\mathbf{x}) &= x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0 \\
g_2(\mathbf{x}) &= x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0 \\
h_1(\mathbf{x}) &= x_1 + x_2 - x_3 - x_4 = 0 \\
h_2(\mathbf{x}) &= 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0 \\
h_3(\mathbf{x}) &= x_3 + x_6 - x_5 = 0 \\
h_4(\mathbf{x}) &= x_4 + x_7 - x_8 = 0
\end{aligned}$$

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** [23]:

  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 11 shows the feasible search space determined by the two inequelity constraints and the approximate position of x^* which lies on the boundary.

- **G25** [23]:

  Minimize:

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

  subject to:

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

  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 12 shows point $\mathbf{x}^*$ which lies on the boundary (in this case the boundary is equivalent to $\mathcal{F}$).

Algorithm 3 A pseudo-code for the ANT-$\mathcal{B}$ algorithm

// update_trail: lays a pheromone trail on the respective directions

update_trail(Trail: real vector)

for l in $1:k$ **do**

  Trail$[l] = (1-\rho)*$Trail$[l] + \Delta$Trail$[l]$ // see Section 4

end for

// update_directions: the new reference points on each direction

update_directions(Dirs: directions)

for l in $1:k$ **do**

  Dirs$[l]$ =best_pair_of_points_found_on_direction(l)

end for

// allocate_ants: sends the ants to search on different directions

allocate_ants(A: colony, Trail:real vector, Dirs: directions, ctr: integer)

for i in $1:N_a$ **do**

  d = choose_dir(Trail) // a direction probabilistically chosen according to the pheromone trail (Eq. 1)

  ant i applies mutation (regarding 'ctr') on the pair of n-dimensional points represented by Dirs$[d] = (\mathbf{x}_d, \mathbf{y}_d)$

  ant i saves the new pairs of points as $(A.\mathbf{x}_i, A.\mathbf{y}_i) = (\mathbf{x}'_d, \mathbf{y}'_d)$

end for

// evaluate: obtains the objective value for the solutions found and the set the best solution on each direction

evaluate(A: colony)

for i in $1:N_a$ **do**

  $A.\mathbf{b}_i = BS(A.\mathbf{x}_i, A.\mathbf{y}_i)$ // obtains the respective point on the boundary (see Algorithm 1)

  A.eval$_i = F(A.\mathbf{b}_i)$ // evaluation of point $\mathbf{b}_i$

end for

// main program

main()

$t = 0$

ctr= initial_constraint // 'ctr' represents the problem constraint under consideration

init_d(Dirs,ctr) // generates k random n-dimensional pair of points regarding 'ctr'

init_t(Trail) // set the k initial values for the pheromonte trail structure

allocate_ants($A(t)$, Trail, Dirs, ctr);

evaluate($A(t)$)

while (stop condition not met) **do**

  $t = t + 1$

  if (change constraint) **then**

    ctr= get_next_ctr(ctr) // The search continues considering another problem constraint following a

                // *Rendez-Vous* policy using either S$_{all}$, S$_{act}$, or S$_j$ (see at the beginning of section 5).

    new_directions(Dirs,ctr) // similar to 'update_directions' except that it could be necessary the generation

                // of new points according to the new 'ctr'

  else

    update_directions(Dirs,ctr)

  end if

  update_trail(Trail)

  allocate_ants($A(t)$,Trail,Dirs,ctr);

  evaluate($A(t)$)

end while

Figure Captions

Figure 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. On the right side it is shown the application of a hypothetical mutation-like operator on points $(\mathbf{x}_4, \mathbf{y}_4)$.

Figure 2: Feasible search space defined by 3 inequality constraints. The search proceeds on the boundary of constraint g_1. It can be observed that the second and fifth points on the boundary of g_1 (from left to right) are infeasible.

Figure 3: A discrete structure with $k = 4$ search directions (left) and the corresponding 4 points on a 2-dimensional search space (right) obtained through the application of function BS from the 4 hypothetical search directions.

Figure 4: Nine ants are distributed on three search directions; 2, 4, and 3 ants respectively allocated on directions 1, 2, and 3. CR, NP, and NR stand respectively for "current reference point", "new points obtained through a valid mutation operator", and "new reference point (the best of the newest generated points on a particular direction)".

Figure 5: Sequence of points generated in the search space and limited to the extent of values in each dimension. Each point $\mathbf{p}_l^t$ is obtained by the application of function BS on the pair of points represented by d_l at iteration t.

Figure 6: Convergence of the best average values in each iteration for problem $G02$ (left) and $G03$ (right). It can be obsereved a fast convergence to the best known ($G02$) and optimal ($G03$) values before iteration 500.

Figure 7: 20 search directions at different stage of the ANT-$\mathcal{B}$ running for problem $G15$: (a) fairly sparse at the beginning, (b) search concentrated on a suboptimal region, (c) moving to another the region, and (d) converging to the best knwon solution.

Figure 8: Mean(V) values (in log scale) of out 30 run for problems $G14$, $G15$, $G17$, $G21$, and $G23$ during the first 3000 interations. It can be observed a rapidly decreasing value of constraint violation for all these problems.

Figure 9: Keane's function with $n = 2$.

Figure 10: Problem G6. Floudas-Pardalos' function.

Figure 11: Approximate position of the best known value on the boundary of the feasible search space regarding constraints g_1 and g_2.

Figure 12: Best known solution and the feasible search space determined by equality constraint h.

Figures on Individual Pages

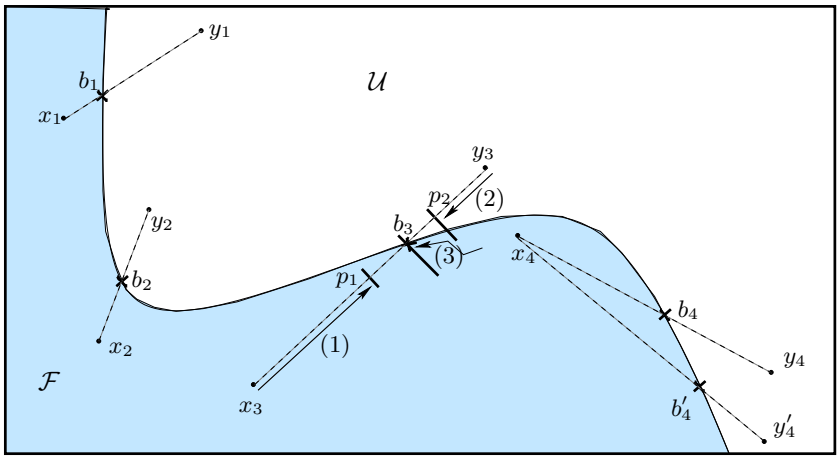

y_1
b_1
x_1
$\mathcal{U}$
y_3
p_2
(2)
b_3
y_2
(3)
x_4
b_2
p_1
b_4
y_4
x_2
(1)
$\mathcal{F}$
x_3
b_4'
y_4'

Figure 1:

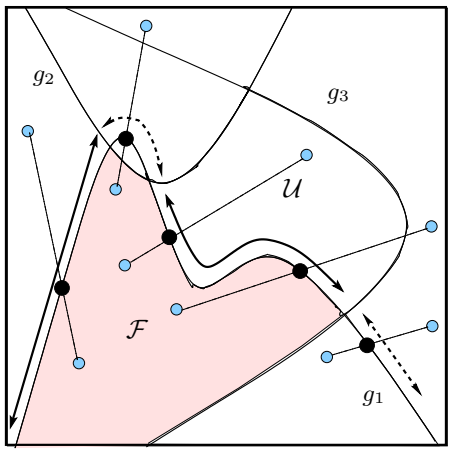


Figure 2:

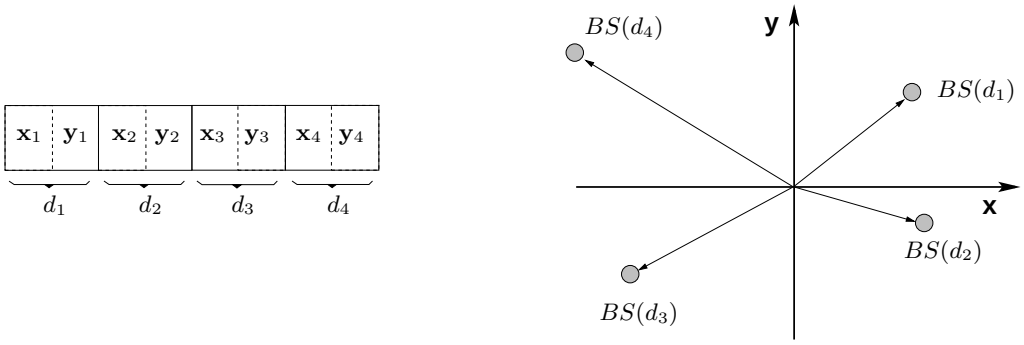

$\mathbf{x}_1$ $\mathbf{y}_1$ $\mathbf{x}_2$ $\mathbf{y}_2$ $\mathbf{x}_3$ $\mathbf{y}_3$ $\mathbf{x}_4$ $\mathbf{y}_4$
d_1 d_2 d_3 d_4
$BS(d_4)$
y
$BS(d_1)$
x
$BS(d_2)$
$BS(d_3)$

Figure 3:

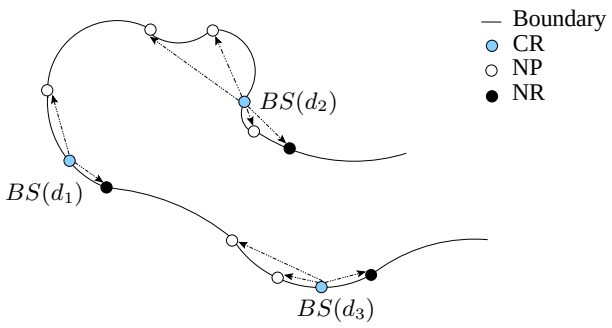


Figure 4:

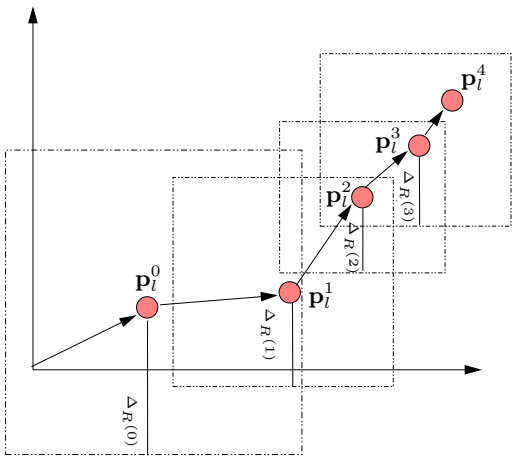

$\mathbf{p}_l^4$
$\mathbf{p}_l^3$
$\mathbf{p}_l^2$
$\mathbf{p}_l^1$
$\mathbf{p}_l^0$
$\Delta_R(3)$
$\Delta_R(2)$
$\Delta_R(1)$
$\Delta_R(0)$

Figure 5:

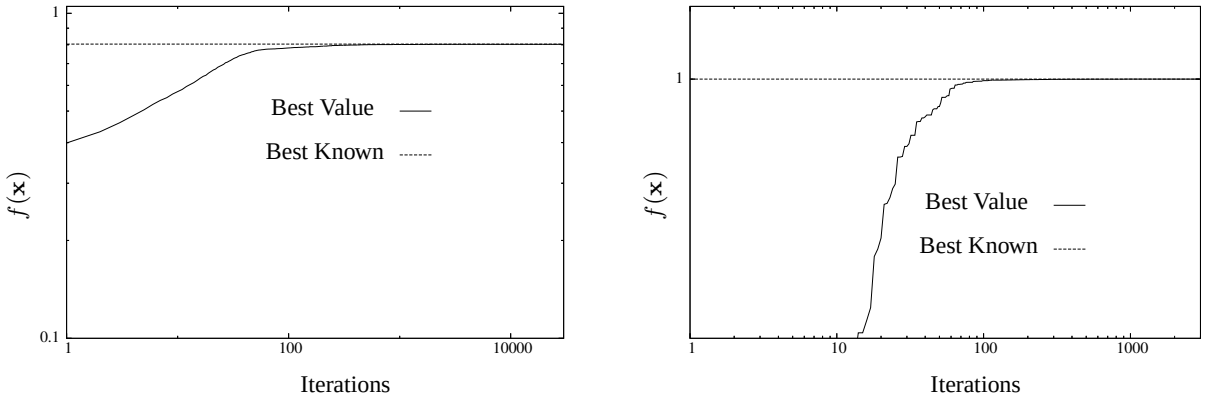
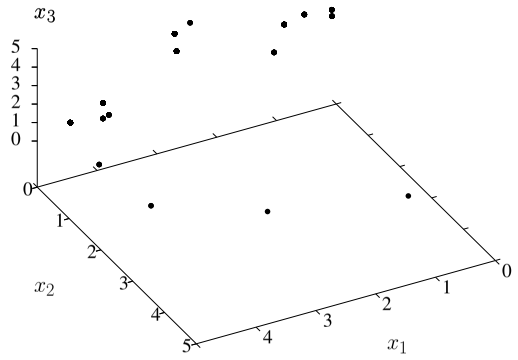
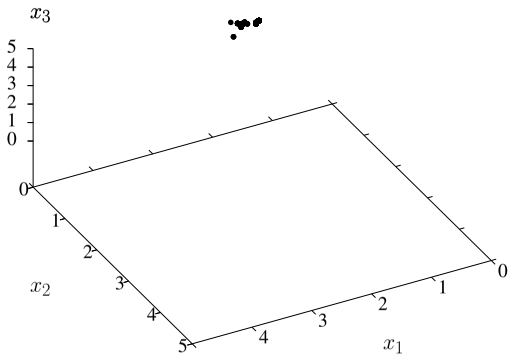

1
0.1
$f(\mathbf{x})$
Best Value
Best Known
1
100
10000
Iterations
1
$f(\mathbf{x})$
Best Value
Best Known
1
10
100
1000
Iterations

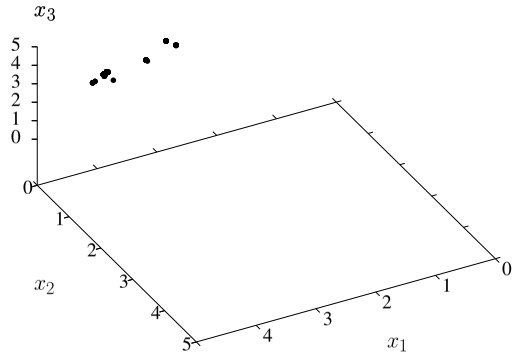
Figure 6:



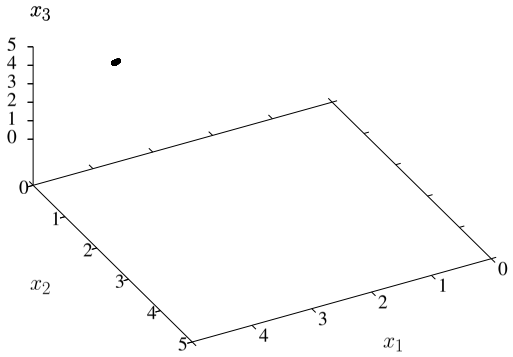
(a) At iteration 1



(b) At iteration 10000



(c) At iteration 20000



(d) At iteration 30000

Figure 7:

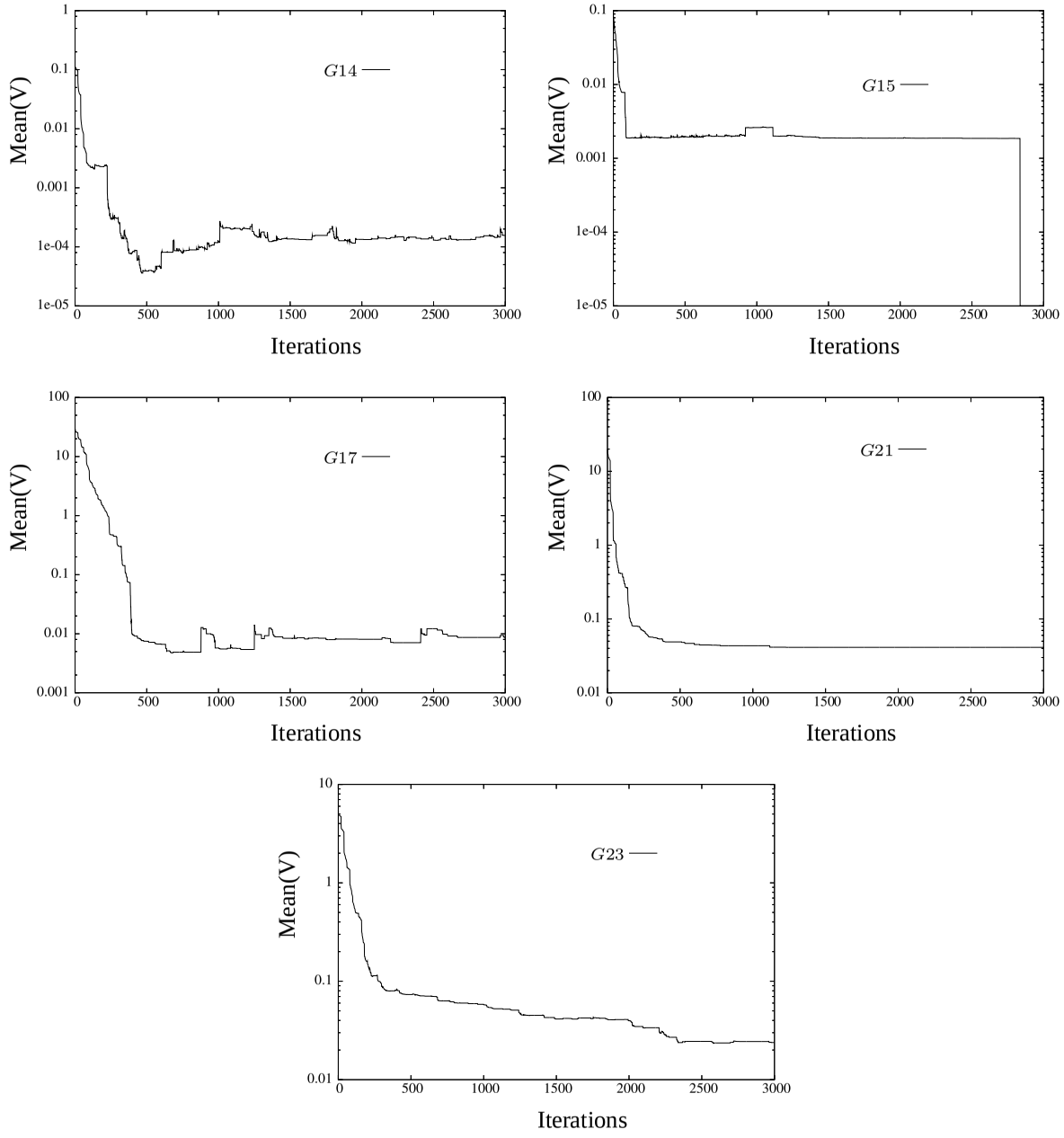


Figure 8:

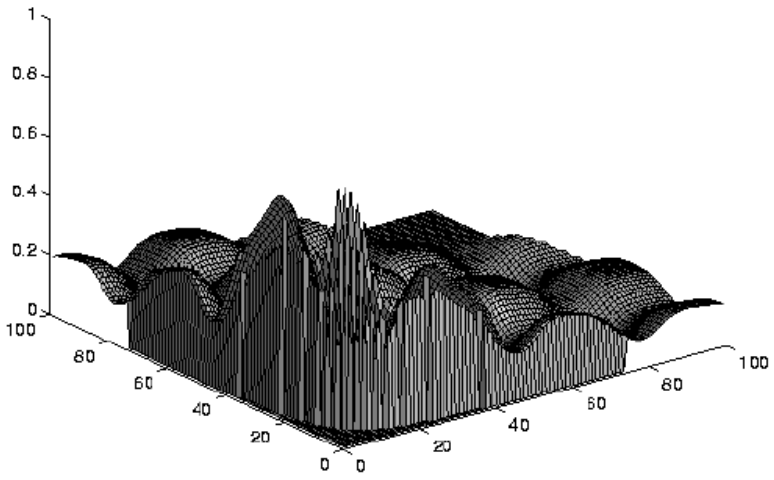


Figure 9:

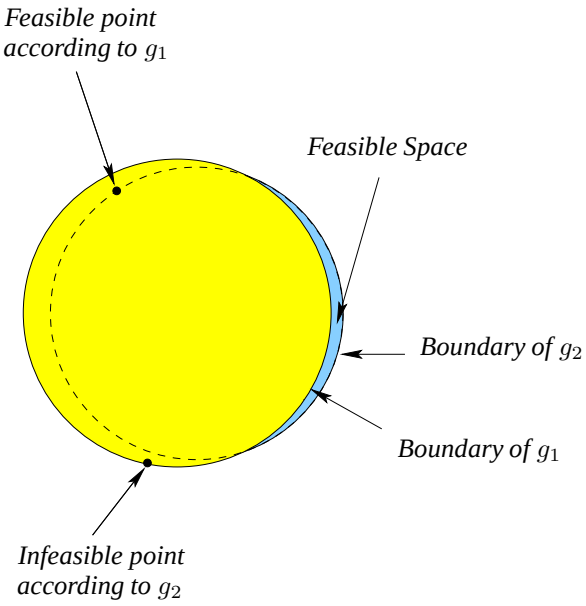

Feasible point
according to g_1
Feasible Space
Boundary of g_2
Boundary of g_1
Infeasible point
according to g_2

Figure 10:

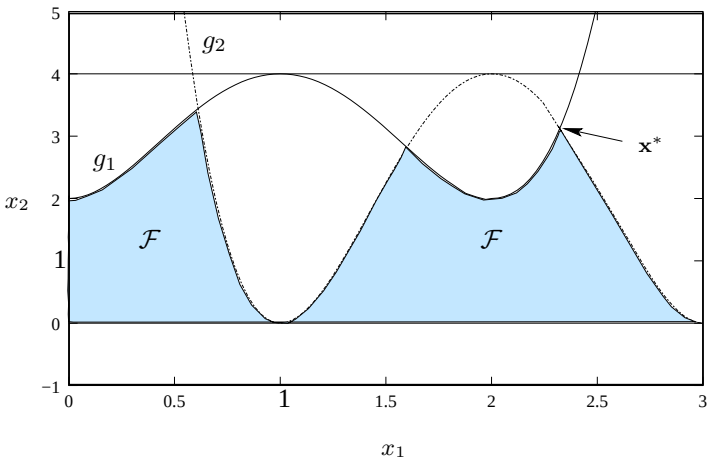


Figure 11:

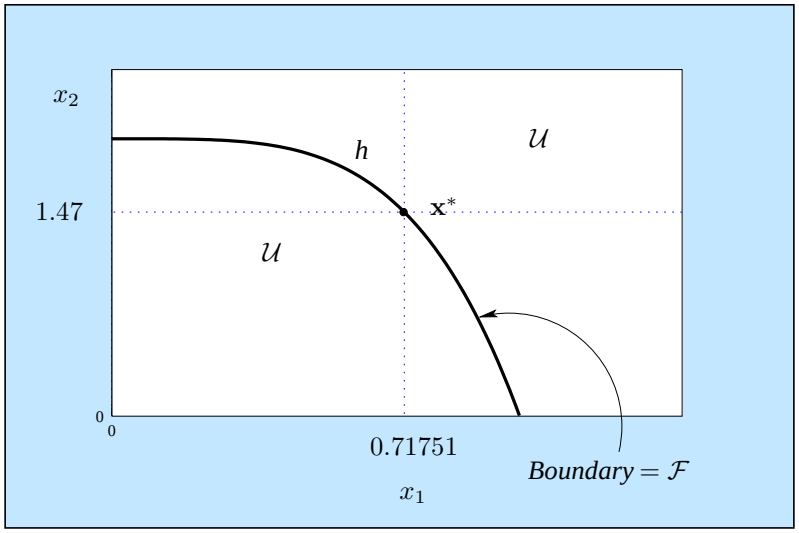

x_2
h
$\mathcal{U}$
1.47
$\mathbf{x}^*$
$\mathcal{U}$
0
0
0.71751
x_1
Boundary $= \mathcal{F}$

Figure 12:

Table Captions

Table 1: The results for problems $G02$ (Keane's function) and $G03$ show a robust behavior of ANT-$\mathcal{B}$ (see Std values) for all the instances tested of these problems.

Table 2: For problems G11 and G25 it is unnecessary to use a penalty factor.

Table 3: Results for problems $G01$, $G04$, $G05$, $G06$, and $G07$. For each problem, we show the different alternatives of using ANT-$\mathcal{B}$, i.e., S_j (for $j = 1 \ldots m$), S_{act}, and S_{all}. For some of these problems, ANT-$\mathcal{B}$ fails to find any feasible solution: $G4$ with options S_2 and S_4, and $G05$ with options S_1 and S_2.

Table 4: Results for problems $G09$, $G10$, $G13$, and $G24$. For each problem, we show the different alternatives of using ANT-$\mathcal{B}$, i.e., S_j (for $j = 1 \ldots m$), S_{act}, and S_{all}. ANT-$\mathcal{B}$ fails to find any feasible solution for problem $G10$ with options S_5 and S_6. Also for $G10$, one of the hardest problems considered, we can observe a large standard deviation values.

Table 5: Comparison of ANT-$\mathcal{B}$ with respect to one of the best constraint-handling technique known to date: stochastic ranking (SR). Both algorithms perform similarly on all the problems with respect to the BF value. However, ANT-$\mathcal{B}$ is more robust that SR when considering the Mean and Worst values. **Boldface** is used to indicate cases in which an algorithm was able to reach the optimum (or best known value) for a problem.

Table 6: Comparison of ANT-$\mathcal{B}$ with SR for a set of additional test cases. With respect to the quality of the results, ANT-$\mathcal{B}$ performs better than SR on problem $G14$, similarly on $G17$, and almost the same on $G15$. However, SR (by doubling the value of parameter G_m) was not capable of achieving any feasible solution for problems $G21$ and $G23$. On the contrary, SR obtained a larger number of feasible solutions on problems $G14$ and $G17$.

Table 7: Average number of evaluations to obtain the best solution for ANT-$\mathcal{B}$ and SR on the test problems considered (* means 'not available'). Clearly, there is no clear trend on the performace of the two algorithms with respect to the number of evaluations as can be seen that $\bar{e}_{\text{ANT-}\mathcal{B}} < \bar{e}_{\text{SR}}$ for about half of the problems considered and $\bar{e}_{\text{ANT-}\mathcal{B}} > \bar{e}_{\text{SR}}$ for the remaining half of the set.

Tables on Individual Pages

No. of Variables (n)	BF	Mean	Std	Worst	# Fea	Mean(#E)
	Problem G02					
20	0.8036190867	0.8025656939	0.0032	0.7930839658	30	29500
50	0.8352618814	0.8339309692	0.0021	0.8259508014	30	35900
100	0.8456841707	0.8446936011	0.0007	0.8423509002	30	46700
	Problem G03					
20	1.0	1.0	0.0	1.0	30	140000
50	1.0	1.0	0.0	1.0	30	389500

Table 1:

Cnst.	BF	Mean	Std	Worst	# Fea.	Mean(#E)
	Problem G11 (0.75)					
S_1	0.75	0.75	0.0	0.75	30	70400
	Problem G25 (16.73889)					
S_1	-16.73889	-16.73889	0.0	-16.73889	30	10600

Table 2:

Cnst.	BF	Mean	Std	Worst	# Fea.	Mean(#E)
	Problem G01 (-15.00)					
S_1	-15.00	-14.99	0.001	-14.996	30	274800
S_2	-15.00	-14.96	0.012	-14.995	30	159720
S_3	-15.00	-14.99	0.001	-14.965	30	381800
S_4	-14.27	-13.54	.38	-13.18	29	544400
S_5	-13.84	-13.48	0.32	-13.04	25	433800
S_6	-14.22	-13.39	0.47	-13.00	26	407200
S_7	-15.00	-14.78	0.2	-14.65	26	213400
S_8	-15.00	-14.74	0.49	-14.46	27	723400
S_9	-15.00	-14.67	0.76	-13.08	30	454800
S_{act}	-15.00	-15.00	0	-15.00	30	81400
S_{all}	-15.00	-15.00	0	-15.00	30	104000
	Problem G04 (-30655.539)					
S_1	-30665.542	-30665.357	0.04	-30665.279	30	20433
S_2	-	-	-	-	-	-
S_3	-23131.630	-23131.630	0.0	-23131.630	0	23204
S_4	-	-	-	-	-	-
S_5	-26469.496	-26469.496	0.0	-26469.496	0	20608
S_6	-30665.539	-30665.523	0.014	-30665.087	30	22985
S_{act}	-30655.542	-30665.542	0.0	-30655.542	30	21457
S_{all}	-30665.119	-30661.330	2.7847685814	-30654.114	30	22139
	Problem G05 (5126.49)					
S_1	-	-	-	-	-	-
S_2	-	-	-	-	-	-
S_3	5126.50	5133.29	9.284	5147.81	6	100800
S_4	5126.51	5134.70	11.219	5164.91	11	340000
S_5	5126.68	5130.55	3.656	5136.08	11	180000
S_{act}	5126.50	5138.37	8.20	5132.14	6	94000
S_{all}	5126.50	5143.77	10.60	5163.56	5	135800
	Problem G06 (-6961.81)					
S_1	-6961.79	-6961.71	0.075	-6169.54	11	122600
S_2	-6961.81	-6961.72	0.097	-6961.34	25	103000
S_{act}	-6961.81	-6961.74	0.070	-6961.71	25	80000

Table 3:

Cnst.	BF	Mean	Std	Worst	# Fea.	Mean(#E)
	Problem G07 (24.306)					
S_1	24.37	29.59	4.83	42.97	30	70000
S_2	24.51	35.10	23.02	121.56	30	133600
S_3	24.56	28.31	5.54	50.83	30	83600
S_4	24.79	54.17	70.46	380.03	30	83600
S_5	24.52	34.52	16.39	77.19	30	85000
S_6	24.79	31.12	6.46	48.40	30	720800
S_7	33.08	38.86	4.01	46.53	30	71200
S_8	41.03	46.86	20.92	127.06	30	260200
S_{act}	24.37	24.64	0.15	24.92	30	35600
S_{all}	24.38	24.76	0.16	25.22	30	56000
	Problem G09 (680.63)					
S_1	680.63	680.66	0.10	681,29	30	80400
S_2	1664.00	1890.01	119.92	1982.72	5	108000
S_3	840.00	880.82	15.06	890.56	29	22200
S_4	680.63	680.96	0.96	681.95	29	43000
S_{act}	680.63	680.67	0.026	680.72	30	7400
S_{all}	680.65	680.75	0.056	680.89	30	19400
	Problem G10 (7049.2083)					
S_1	7101.50	7346.61	202.15	7682.20	9	147700
S_2	7063.02	8169.68	1866.32	10325.00	3	131600
S_3	7057.27	7406.51	148.60	7518.91	9	148600
S_4	7095.27	7349.83	360.00	7604.39	2	128200
S_5	-	-	-	-	-	-
S_6	-	-	-	-	-	-
S_{act}	7052.30	7199.01	175.01	7943.15	30	42800
S_{all}	7068.04	7141.87	52.27	7239.54	30	9800
	Problem G13 (0.053950)					
S_1	0.053950	0.054908	0.00054	0.055386	6	29800
S_2	0.053950	0.054372	0.00044	0.054968	4	7400
S_3	0.053950	0.054637	0.00017	0.054394	6	7200
S_{act}	0.053950	0.054736	0.001	0.058462	15	19800
	Problem G24 (-5.508013)					
S_1	-5.508013	-5.508013	0.0	-5.508013	30	5800
S_2	-5.508013	-5.508013	0.0	-5.508013	30	24000
S_{act}	-5.508013	-5.508013	0.0	-5.508013	30	21400

Table 4:

		BF		Mean		Worst	
Prob.	Opt[7]	ANT-$\mathcal{B}$	SR	ANT-$\mathcal{B}$	SR	ANT-$\mathcal{B}$	SR
G01	-15.000	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**
G02	0.803619	**0.803619**	0.803515	0.802656	0.781975	0.793083	0.726288
G03	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
G04	-30665.539	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30666.539**	**−30665.539**
G05	5126.498	5126.50	**5126.497**	5138.37	5128.881	5132.14	5142.472
G06	-6961.814	**−6961.81**	**−6981.814**	-6961.74	-6875.940	-6961.71	-6350.262
G07	24.306	24.37	24.307	24.64	24.374	24.92	24.642
G09	680.630	**680.63**	**680.63**	680.67	680.56	680.72	680.763
G10	7049.2083	7052.30	7054.316	7199.01	7559.192	7943.15	8835.655
G11	0.75	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**
G13	0.053950	**0.053950**	0.053957	0.054908	0.057006	0.055386	0.216915
G24	-5.508013	**−5.508013**	**−5.508013**	**−5.508013**	**−5.508013**	**−5.508013**	**−5.508013**
G25	-16.73819	**−16.73819**	**−16.73819**	**−16.73819**	**−16.73819**	**−16.73819**	**−16.73819**

Table 5:

		$G14$ (-47.764411)	$G15$ (961.715172)	$G17$ (8853.539674)	$G21$ (193.778349)	$G23$ (-400.002500)
BF	ANT-$\mathcal{B}$	-47.760268	961.715099	8855.819335	193.782989	-399.984877
	SR	-39.412791	961.715022	8856.136000	$*$	$*$
Mean	ANT-$\mathcal{B}$	-47.651929	961.715636	8937.446289	194.345108	-249.007506
	SR	-36.526091	961.715496	8893.396000	$*$	$*$
Worst	ANT-$\mathcal{B}$	-46.723707	961.717224	8952.621093	202.067779	-28.448352
	SR	-33.003904	961.725354	8951.007000	$*$	$*$
#Fea	ANT-$\mathcal{B}$	15	30	3	19	16
	SR	30	30	30	$*$	$*$

Table 6:

Problem	$\bar{e}_{\text{ANT-}\mathcal{B}}$	$\bar{e}_{\text{SR}}$
$G01$	81400	149600
$G02$	29500	233400
$G03$	140000	212000
$G04$	21457	77600
$G05$	94000	52400
$G06$	80000	111600
$G07$	35600	141400
$G09$	7400	111000
$G10$	42800	17200
$G11$	70400	10400
$G13$	7200	67200
$G14$	1250000	349600
$G15$	695600	73200
$G17$	411500	74000
$G21$	760000	*
$G23$	763100	*
$G24$	21400	23400
$G25$	10600	15200

Table 7:

Authors' Biographies

Guillermo Leguizamón is an Assistant Professor of Computer Science at the National University of San Luis, Argentina. He received a PhD (2004) in metaheuristics for constrained optimization problems. His current research interests involve the design and application of ant colony optimization, evolutionary algorithms, differential evolution and other metaheuristics to continuous and combinatorial problems. He has published book chapters, journal papers, and around 55 conference papers. He is part of the *Research and Development Lab on Computational Intelligence* as responsible of the "Metaheuristics Research Group". Currently, he is holding collaborations (joint publications, visits, and exchanges) with some national and international universities.

Carlos A. Coello Coello received the B.Sc. degree in civil engineering from the Universidad Autónoma de Chiapas, México, and the M.Sc. and the PhD degrees in computer science from Tulane University, USA, in 1991, 1993, and 1996, respectively.

He is currently professor (CINVESTAV-3D Researcher) at the computer science department of CINVESTAV-IPN, in Mexico City, México. Dr. Coello has authored and co-authored over 180 technical papers and several book chapters. He has also co-authored the book *Evolutionary Algorithms for Solving Multi-Objective Problems* (Second Edition, Springer, 2007). Currently, Dr. Coello is associate editor of the *IEEE Transactions on Evolutionary Computation* and serves in the editorial board of 7 other international journals. He also chairs the *IEEE Computationally Intelligent Society Task Force on Multi-Objective Evolutionary Algorithms*. He received the *2007 National Research Award* from the Mexican Academy of Sciences in the area of *Exact Sciences*. He is a member of the IEEE, the ACM, Sigma Xi, and the Mexican Academy of Sciences.

His major research interests are: evolutionary multi-objective optimization and constraint-handling techniques for evolutionary algorithms.