A Study of the Scalability of ACO$_{\mathbb{R}}$ for Continuous Optimization Problems

Guillermo Leguizamón*
UMI LAFMIA 3175 CNRS
CINVESTAV-IPN
Departmento de Computación
Av. IPN No. 2508. Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO
legui@unsl.edu.ar

Carlos A. Coello Coello†
CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
Av. IPN No. 2508. Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO
ccoello@cs.cinvestav.mx

Abstract

In the field of metaheuristics, evolutionary computation (EC) embodies some of the most relevant optimization algorithms for dealing with continuous problems, namely, Evolution Strategies (ESs), Differential Evolution (DE) and Particle Swarm Optimization (PSO). Many of these algorithms have been traditionally applied to different test suites to assess their respective performance. However, many of these test suites include (or are used with) only low dimensional problems which many state-of-the-art algorithms are capable of solving. For that reason, more complex and higher dimensional test suites have been recently proposed (e.g., the test suite proposed for the Special Session on Large Scale Global Optimization at the 2008 IEEE Congress on Evolutionary Computation). Ant Colony Optimization (ACO) algorithms have a long tradition as effective solvers of combinatorial optimization problems. However, several proposals of ACO algorithms for continuous problems have also

*On leave of absence from LIDIC - Universidad Nacional de San Luis, San Luis, Argentina.
†The second author is also affiliated to the UMI LAFMIA 3175 CNRS at CINVESTAV-IPN.

been successfully applied to both academic and real-world problems in the field of constrained and unconstrained continuous optimization. This paper aims at showing the behavior of a well-known ACO algorithm for continuous problems (the so-called $ACO_{\mathbb{R}}$) for large scale unconstrained continuous problems by considering a recently proposed test suite. In addition, we propose and study a simple mechanism to escape from local optima. Our results are compared with respect to those obtained by highly competitive algorithms that have been assessed with the previously mentioned test suite.

1 Introduction

The Ant Colony Optimization (ACO) metaheuristic [4, 5] embodies a class of algorithms derived from the main concepts involved in the behavior of real ant colonies. These algorithms implement a colony of artificial ants aimed at finding good solutions to a problem. The ants in the colony cooperate among them by indirect communication mediated by the environment (*stigmergy*). The most representative instantiations of the ACO metaheuristic are: the Ant System (AS), AS with an *elitist strategy* for updating the pheromone trail levels, AS_{rank} (a rank-based version of Ant System), $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS), and the Ant Colony System (ACS) [5]. All of them were originally designed to operate on combinatorial optimization problems (including some dynamic versions). There exist several extensions of the ACO metaheuristic for solving continuous problems: the first ACO extension to operate on continuous spaces can be found in Bilchev et al. [2]. Since then, a number of other proposals have been introduced, such as the algorithms of Monmarché et al. [17], Ling et al. [14], Lei et al. [13], Dreo et al. [6, 7], Qin Lin Chen et al. [15], Pourtakdoust et al. [18], Kong [11], Hu et al. [10], Socha [19] and Socha & Dorigo [21]. The algorithm adopted here is an extension of the ACO metaheuristic

for continuous domains introduced in [21].

It is also worth remarking some successful and recent applications of the ACO metaheuristic for continuous problems: Leguizamón & Coello [12] proposed an extension of the ACO metaheuristic for constrained continuous optimization problems; Afshar and Madadgar [1] proposed an application to solve reservoir operation problems; and Socha and Blum [20] proposed an approach for training of neural networks.

In this work, we study the scalability of $ACO_{\mathbb{R}}$ [21] when facing unconstrained continuous optimization problems of large dimensionality. The experimental study includes a recently proposed test suite of continuous problems which can help to assess the capacity of optimization algorithms for dealing with large dimensional problems. Additionally, a simple mechanism to escape from local optima is proposed. This pretends to be the first step towards improving the $ACO_{\mathbb{R}}$ in order to achieve a design of a more advanced and competitive ACO algorithm with respect to other state-of-the-art metaheuristic algorithms for continuous problems.

The remainder of this paper is organized as follows. Section 2 briefly describes the original version of the $ACO_{\mathbb{R}}$ algorithm and Section 3 presents the test problems adopted for our experimental study. The section about our experimental study (Section 4) involves three important subsections: Section 4.1 presents the results of a preliminary study of $ACO_{\mathbb{R}}$ in the test suite (this section is aimed to fine-tune the parameters of $ACO_{\mathbb{R}}$ and observe its resulting behavior). Section 4.2 describes a simple *diversity* mechanism to improve the performance of $ACO_{\mathbb{R}}$ (Algorithm $ACO_{\mathbb{R}}$-Div) and shows the corresponding results which are compared with those found at a previous stage of the experimental study. In Section 4.3, our improved results for the test suite considered are compared with those found by the set of metaheuristic algorithms presented

at the CEC'08 competition. Finally, in Section 5 we present our conclusions based on the results obtained by both $\text{ACO}_{\mathbb{R}}$ and $\text{ACO}_{\mathbb{R}}$-Div. In addition, some possible lines of future research are also outlined.

2 The $\text{ACO}_{\mathbb{R}}$ algorithm

Taking into account that the ACO metaheuristic works by incrementally building the solutions according to a biased (by pheromone trail) probabilistic choice of solution components, the $\text{ACO}_{\mathbb{R}}$ algorithm was designed with the aim of obtaining a set of *probability density functions* (PDFs). Each PDF is obtained from the search experience and is used to incrementally build a solution $\mathbf{x} \in \mathbb{R}^D$ considering in turn each component x_i ($\forall i = 1 \ldots D$).

To approximate a multimodal PDF, Socha & Dorigo [21] proposed a Gaussian kernel which is defined as a weighted sum of several one-dimensional Gaussian functions $g_l^i(x)$ as follows:

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

where $i \in \{1, \ldots, D\}$ 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 1 shows a superposition of $k = 5$ Gaussian functions which are intended to approximate a hypothetical multimodal Gaussian function with $k = 5$ kernels.

In $\text{ACO}_{\mathbb{R}}$, a solution archive called T is used to keep track of a number of solutions anal-

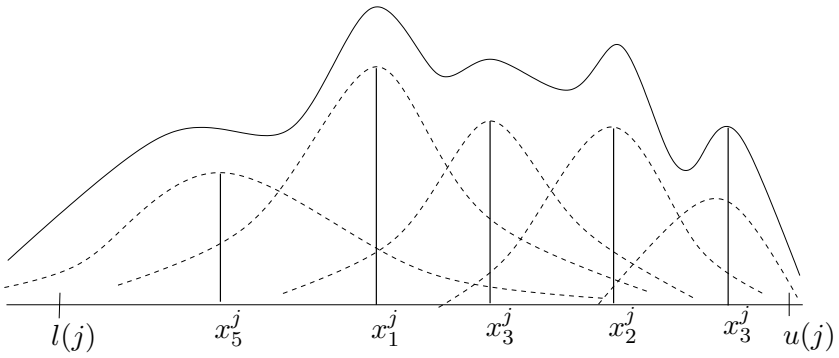


Figure 1: A possible set of $k = 5$ Gaussian functions (dashed line) to achieve, by superposition, a Gaussian kernel (solid line) which approximates a hypothetical multimodal Gaussian function. $x_1^j, x_2^j, x_3^j, x_4^j$, and x_5^j are respectively the values of the solutions at position j (i.e., the mean values for each Gaussian function). $l(j)$ and $u(j)$ are the respective lower and upper limits for dimension j.

ogously 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 $\mathbf{x}_l \in \mathbb{R}^D$, ACO$_\mathbb{R}$ maintains the corresponding values of each problem dimension, i.e., $x_l^1, \ldots, x_l^D$, and the value of the objective function $f(\mathbf{x}_l)$ which are stored satisfying that $f(\mathbf{x}_1) \leq \ldots \leq f(\mathbf{x}_l) \leq \ldots f(\mathbf{x}_k)$. For example, Figure 1 could represent $k = 5$ ranked solutions in structure T, where x_1^j represents the mean value corresponding to the Gaussian distribution with the highest probability of being chosen in the next step of solution construction. On the other hand, the vector of weights $\boldsymbol{\omega}$ should satisfy that $\omega_1 \geq \ldots \geq \omega_l \geq \ldots \geq \omega_k$.

The solutions in T are, therefore, used to dynamically generate probability density functions involved in the Gaussian kernels. More specifically, for obtaining the Gaussian kernel G^i, the three parameters $\boldsymbol{\omega}$, $\boldsymbol{\mu}^i$, and $\boldsymbol{\sigma}^i$ need to be calculated. Thus, for each G^i, the values of the i-th variable of the k solutions in T become part of the elements of vector μ^i, that is, $\boldsymbol{\mu}^i = \{\mu_1^i, \ldots, \mu_D^i\} = \{x_1^i, \ldots, x_D^i\}$. On the other hand, each component of the deviation vector

Algorithm 1 Outline of $ACO_{\mathbb{R}}$ algorithm

1: Init(T,ω);

2: Get(σ);

3: **for** $t \in 1 : t_{max}$ **do**

4: A = BuildSols(T, σ);

5: T = First$_k$(rank($T \oplus A$));

6: Update(σ);

7: **end for**

$\boldsymbol{\sigma}^i = \{\sigma_1^i, \ldots, \sigma_k^i\}$ is obtained as:

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{|x_e^i - x_l^i|}{k-1} \tag{2}$$

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 ξ, the lower the convergence speed of the algorithm.

The pheromone update is achieved by considering a set A[1] of size N_a which maintains the newly generated solutions regarding Eq. 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 maintains its cardinality (k) through the whole search process. Algorithm 2 describes a general outline of the $ACO_{\mathbb{R}}$ algorithm, where Init() is in charge of obtaining the initial set of kernels and sets the corresponding weights vector according

[1]The set A represents the set of ants according to Socha and Dorigo [21].

to parameter q; Get() calculates the σ values according to the initial values of T; BuildSol() obtains the new set of solutions; First$_k$ gives the best k solutions from the old set of solutions (T) plus the new one (A); and Update() obtains a new σ that will bias the sampling of the new set of solutions for the next iteration.

3 Benchmark Problems

To conduct the experimental study (next section) we have selected 6 problems from the benchmark functions prepared for the "Special Session and Competition on Large Scale Global Optimization" at the *2008 IEEE Congress on Evolutionary Computation (CEC'08)* [22]. The problems (see Table 3) represent a set of scalable functions for high-dimensional optimization. Particularly, the objective of this special session was to bring to the research community newer and challenging problems to assess current nature-inspired optimization algorithms as well as other, novel optimization algorithms. In this way, their respective capacities to deal with complex and high-dimensional problems can be better assessed and compared amongst them.

4 Experimental Study

This section presents the experimental study conducted to assess the performance of the ACO$_{\mathbb{R}}$ on high dimensional problems. We first present some preliminary results obtained from experiments towards observing the behavior of ACO$_{\mathbb{R}}$ under different parameters settings. This preliminary study was designed to help us detecting the main weaknesses of this algorithm when facing large scale problems. After that, we present the implemented proposal for dealing with the loss of diversity and the corresponding results. The last part includes a comparison with recently

Benchmark Problems	Search Range	$f(\mathbf{x}^*)$
$f_1(\mathbf{x}) = \sum_{i=1}^{D} z_i + f\_bias_1, \mathbf{z} = \mathbf{x} - \mathbf{o}$ $\mathbf{o} = (o_1, o_2, \ldots, o_D)$; the shifted global optimum	[-100,100]	-450
$f_2(\mathbf{x}) = max_i\{\|z_i\|, 1 \leq i \leq D\} + f\_bias_2, \mathbf{z} = \mathbf{x} - \mathbf{o}$ $\mathbf{o} = (o_1, o_2, \ldots, o_D)$; the shifted global optimum	[-100,100]	-450
$f_3(\mathbf{x}) = \sum_{i=1}^{D-1}(100 \cdot (z_i^2 - z_i)^2 + (z_i - 1)^2) + f\_bias_3$, $\mathbf{z} = \mathbf{x} - \mathbf{o} + \mathbf{1}$ $\mathbf{o} = (o_1, o_2, \ldots, o_D)$; the shifted global optimum	[-100,100]	390
$f_4(\mathbf{x}) = \sum_{i=1}^{D}(z_i^2 - 10 \cdot \cos(2\pi z_i) + 10) + f\_bias_4, \mathbf{z} = \mathbf{x} - \mathbf{o}$ $\mathbf{o} = (o_1, o_2, \ldots, o_D)$; the shifted global optimum	[-100,100]	-330
$f_5(\mathbf{x}) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + f\_bias_5, \mathbf{z} = \mathbf{x} - \mathbf{o}$ $\mathbf{o} = (o_1, o_2, \ldots, o_D)$; the shifted global optimum	[-100,100]	-330
$f_6(\mathbf{x}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2})$ $-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + f\_bias_6)$, $\mathbf{z} = \mathbf{x} - \mathbf{o}$; $\mathbf{o} = (o_1, o_2, \ldots, o_D)$; the shifted global optimum	[-100,100]	-330

proposed *ad hoc* algorithms for solving large dimensional problems. All the experiments were run on an Intel Pentium (R) 4, CPU 3.00Gz, and 1Gb RAM; OS Linux version 2.6.23.17-88.fc7 (Red Hat 4.1.2-27). The $\text{ACO}_{\mathbb{R}}$ algorithm was implemented in the C programming language.

4.1 Preliminary Results on the Test Suite Problems

The experimental design for the $\text{ACO}_{\mathbb{R}}$ included a variation of the main following parameters: $0 \leq q \leq 1$ and $0 \leq \xi \leq 1$. The remaining parameters were kept fixed: $k = 50$ and $N_a = 50$ which represent, respectively, the number of kernels and the number of ants, or, similarly, the

sizes of the structures T and A described in Algorithm 2.

The initial experimentation was aimed at detecting the regions of parameters values that produced the best performance. In order to do that, we used Latin Hypercube Sampling (LHS) to obtain a number of design points in a space filling way and considering problems of dimension $D = 100$. From this study, we detected that typical values found in the literature for parameters q and ξ were the best settings for the problems considered. Thus, we chose $\xi = 0.85$ and small values for q. In this case, we specifically chose the settings $q = 0.001$ and $q = 0.01$. Both values produced an algorithm that intensively exploits the information around the Gaussian kernel on the top of the ranking of solutions, i.e, located at the first position of the structure T.

In the following, we present the results of ACO$_{\mathbb{R}}$ in the set of functions f_1 to f_6 for dimensions $D = 100, 500,$ and 1000, respectively in Tables 3, 4, and 5 (see Appendix). The style adopted for the presentation of the results is similar to that followed in the Special Session at CEC'08. For example, our tables include three different parts according to the number of function evaluations (FES) which varies depending on the problem's dimension. The variation of FES will allow us to determine if the algorithm is capable of continuing the exploration of the search space as more evaluations are allowed. All the numerical results displayed in the tables represent the Error Values (i.e., $f(\mathbf{x}) - f(\mathbf{x}^*)$, where $\mathbf{x}$ and $\mathbf{x}^*$ are, respectively, the solution found and the optimal solution). The two possible settings for the parameter q are also included (second column). Additionally, at the end of each group determined by FES, the respective p-values are displayed. These values are obtained by applying the non-parametric Mann-Whitney-Wilcoxon test to assess the significance on the differences on the respective medians. When the differences between the results obtained by ACO$_{\mathbb{R}}$ with $q = 0.001$ and $q = 0.01$ are significant

at a level of 5% of confidence, the corresponding p-values are displayed in boldface.

Table 3 (dimension $D = 100$) shows that ACO$_{\mathbb{R}}$ performs fairly well on the different functions. For example, the best values are close to the optimal one for FES = $5.00e + 4$ and FES = $5.00e + 5$. However, for FES = $5.00e + 3$, ACO$_{\mathbb{R}}$ shows a poor performance (the number of function evaluations is, certainly, insufficient). For this dimension, clearly the best performance is achieved with $q = 0.01$ (2nd and 3rd groups). In the first group (FES = $5.00e + 3$), there is only one case in which $q = 0.001$ outperforms $q = 0.01$. In this group, for functions f_2,f_3, and f_4, no significant differences were detected. Particularly for function f_6, in all groups, no significant differences were found for this parameters setting. The situation is similar for $D = 500$ and $D = 1000$ (Tables 4 and 5) with respect to the performance of ACO$_{\mathbb{R}}$ as the number of evaluation is increased, i.e., the solutions quality is improved but not as much as when $D = 100$. Also, it can observed here that ACO$_{\mathbb{R}}$ does not show a consistent behavior running under the two selected values for parameter q. For the largest number of dimensions considered ($D = 1000$, Table5) the quality of the results are farther away from the optimal values.

It is clear, from this set of results, that ACO$_{\mathbb{R}}$ needs an extra mechanism to better explore high-dimensional search spaces. At this point, it is important to highlight the influence of parameter q in the behavior of ACO$_{\mathbb{R}}$. It is well known that small values for this parameter intensify the exploitation around the best solution found at each iteration. The setting $q = 0.001$ induces a high level of exploitation, whereas $q = 0.01$, although still small, allows the algorithm to be more exploratory (it must be recalled that ACO$_{\mathbb{R}}$ showed the best behavior when studying the whole space of the design points). Therefore, with $q = 0.001$ the algorithm could suffer of premature convergence and $q = 0.01$ will better avoid getting trapped in local optima, but it will

lose the possibility of better exploiting good quality solutions.

To finish this section, we show in Table 1 a summary of the influence of the parameter q on the behavior of $ACO_{\mathbb{R}}$ for all problems, dimensions, and FES studied. The meaning of the symbols are the following: $>$, $<$, and $=$; which indicate whether or not there exist significant statistical differences in the results achieved by the setting $q = 0.01$ with respect to $q = 0.001$.

Table 1: Summary of the influence of parameter q, where $> <$, and $=$ establish a ranking relation between $q = 0.01$ with respect to $q = 0.001$ based on the statistical test.

	FES = $5.00e+3$						FES = $5.00e+4$						FES = $5.00e+5$					
	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
$D = 100$	$>$	$=$	$=$	$<$	$>$	$=$	$>$	$>$	$>$	$>$	$>$	$=$	$>$	$>$	$>$	$>$	$>$	$=$
	FES = $2.50e+4$						FES = $2.50e+5$						FES = $2.50e+6$					
	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
$D = 500$	$=$	$=$	$<$	$=$	$<$	$=$	$>$	$>$	$=$	$>$	$=$	$<$	$>$	$<$	$=$	$>$	$=$	$<$
	FES = $5.00e+4$						FES = $5.00e+5$						FES = $5.00e+6$					
	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
$D = 1000$	$<$	$=$	$<$	$<$	$<$	$=$	$=$	$=$	$=$	$>$	$=$	$=$	$=$	$=$	$=$	$>$	$=$	$<$

4.2 Proposed Mechanism to Deal with the Loss of Diversity

In $ACO_{\mathbb{R}}$, the loss of diversity will produce a multi-modal Gaussian distribution where all the mean values tend to be the same or very close to each other, i.e., will be almost equivalent to a unimodal Gaussian distribution. When this scenario is observed, $ACO_{\mathbb{R}}$ will produce once and again the same sampling in the search space since the values of the vector σ are closer and closer

to 0. One possible mechanism to detect and maintain the kernels' diversity (i.e., the diversity of the solutions in structure T) is to first measure the degree of diversity and then control the evolution of the kernel's population. We adopted the following function [24] to measure the diversity of T:

$$div(T) = \frac{1}{N_{diag} \cdot k} \sum_{i=1}^{k} \sqrt{\sum_{j=1}^{D} (x_j^i - \bar{\mathbf{x}}_j)^2} \tag{3}$$

where N_{diag} is the length of the diagonal of the search space determined by the corresponding upper and lower limits for each decision variable, k is the number of kernels in T, x_j^i is the value at dimension j of the solution at position i in T, and $\bar{\mathbf{x}}^j$ is the average of all the values in dimension j. Function div (Eq. 3) returns a value in the range $[0.0, 0.5]$. Therefore, the higher the values returned, the more diversity is detected in the set of kernels. Based on this measurement of diversity, our mechanism keeps control of the degree of diversity at each algorithm's iteration. When a certain threshold (div_{min}) is reached, it means that the populations of kernels will not produce any further exploratory sampling. In Figure 2 (upper left handside) we show a hypothetical population of kernels clustered around $\mathbf{x}_1$, the best point in the current set of kernels. The multimodal Gaussian kernel will have a similar shape to that displayed in Figure 2 (lower left handside). To deal with the clustered population, a percentage p_{sol} of solutions in T (the worst ones) are replaced by a new set of solutions which are randomly generated (Figure 2, upper right handside). This new set of points will increase the values of vector σ and, consequently, will produce a more exploratory sampling in the following iterations. It must be noticed that this mechanism keeps the best current point $(\mathbf{x}_1)$ as the main attractor. However, a much more extended region around this point can be further explored since the Gaussian multimodal

distribution will now look like the one displayed in Figure 2 (lower right handside). After the replacement of the new randomly generated solutions, we additionally considered a time window (t_{div}) that allows ACO$_{\mathbb{R}}$ to proceed with the exploration of the search space without applying the diversity mechanism. This time window is included because the current population is not completely replaced and a low degree of diversity could rapidly be achieved.

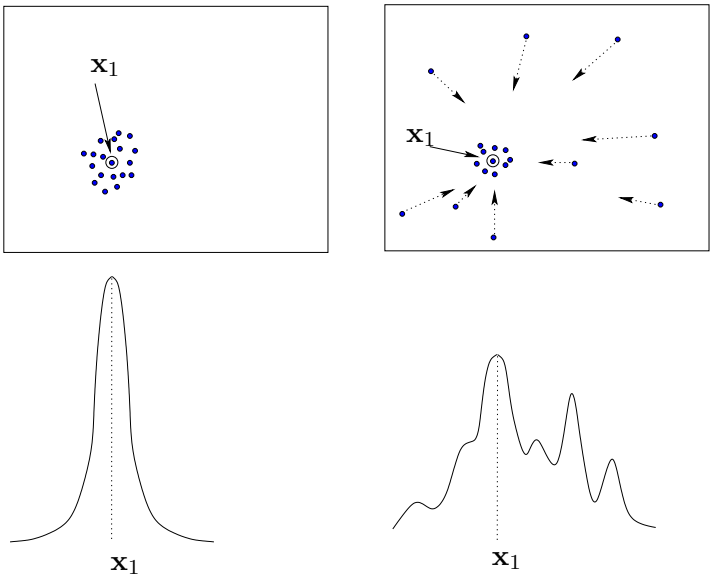
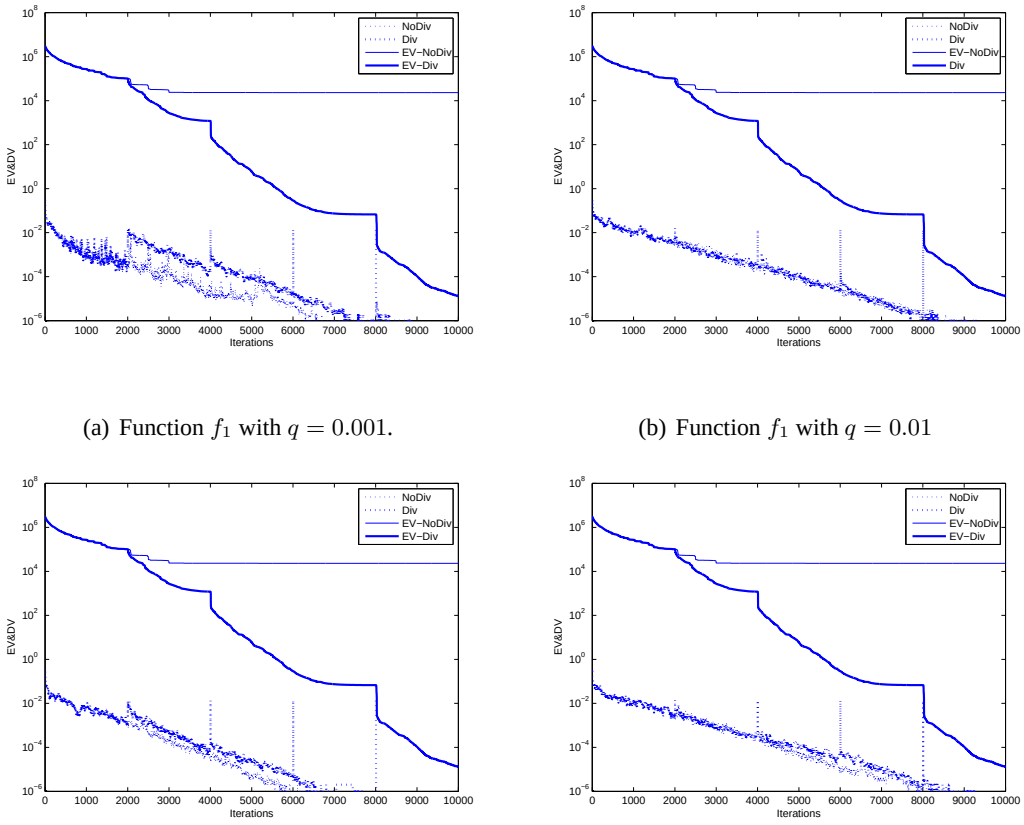


Figure 2: When the algorithm converges (i.e., the degree of diversity is low) the kernel structure is partially re-initialized. However, the best current solution ($\mathbf{x}_1$) still behaves as the main attractor for the following iterations, at least until a new best solution is found in the following samplings.

Figure 4.2 shows the evolution for the Error Values (EV) and Diversity Values (DV) when running respectively ACO$_{\mathbb{R}}$ and ACO$_{\mathbb{R}}$-Div (i.e., ACO$_{\mathbb{R}}$ incorporating the diversity mechanism) with values $q = 0.0001$ and $q = 0.01$. In order to do that, we have chosen functions f_1 and $f5$. Let us first analyze the influence of the diversity mechanism on the error values for both values of q. It can be observed a direct influence on the error values at the moment in which



(a) Function f_1 with $q = 0.001$.

(b) Function f_1 with $q = 0.01$

(c) Function f_5 with $q = 0.001$.

(d) Function f_5 with $q = 0.01$.

Figure 3: Evolution of the Error Values (EV) and Diversity Values (DV) for functions f_1 and f_5 when running ACO$_{\mathbb{R}}$ and ACO$_{\mathbb{R}}$-Div with $q = 0.001$ (left handside) and $q0.01$ (right handside).

the diversity mechanism takes place. For both functions, ACO$_{\mathbb{R}}$ gets stuck in a sub-optimal solution at around iteration 2500. At around the same time, the diversity mechanism is triggered in ACO$_{\mathbb{R}}$-Div since the threshold on the lowest allowable diversity value for this algorithm has been triggered. It is also interesting to compare the shapes of the plots displaying the evolution of the diversity values.

For $q = 0.001$ (both functions), the diversity values increase immediately after the respective mechanism incorporates the new set of randomly generated solutions. Particularly, a more clear

effect of this mechanism is observed when applied the first time. However, as the number of iterations increases, the effect of the diversity mechanism decreases. It must be recalled that when applied, the diversity mechanism maintains a percentage of the current solutions in T (the best ones) which makes the algorithm less exploratory at this stage. However, several improvements in subsequent iterations are still possible as shown in the respective EV plots for $ACO_{\mathbb{R}}$-Div. On the other hand, when comparing the diversity values of $ACO_{\mathbb{R}}$ and $ACO_{\mathbb{R}}$-Div with $q = 0.01$, it can be seen that the diversity plots for both algorithms are similar (mainly for function f_1). This explains the exploratory capacity of the algorithms ($ACO_{\mathbb{R}}$ and $ACO_{\mathbb{R}}$-Div) when increasing the values of parameter q. Nevertheless, better improvements are achieved through the diversity mechanism when considering the quality of the best solutions found.

Since the preliminary results shown in Section 4.1 were not conclusive with respect to the two studied values of the parameter q, we conducted experiments considering both values of this parameter (i.e., $q \in \{0.001, 0.01\}$) to study the effect of the diversity mechanism implemented in $ACO_{\mathbb{R}}$ (as metioned before, we call this new version, $ACO_{\mathbb{R}}$-Div). The added parameters for $ACO_{\mathbb{R}}$-Div were set as: $div_{min} = 0.01$, $p_{sol} = 20\%$, and $t_{div} = 0.2t_{max}$. The corresponding experimental study is presented next.

Tables 6, 7, and 8 (see Appendix) display the results obtained from $ACO_{\mathbb{R}}$ and $ACO_{\mathbb{R}}$-Div (in each table, the results are grouped according to FES and the values of the parameter q). In this case, we have reduced the data shown in the corresponding tables. More precisely, we show the best values found from the 25 trials performed by each algorithm. As for the former experimental study, we also compare the performance of the algorithms based on the statistical differences in their median values through the non-parametric Mann-Whitney-Wilcoxon test.

Thus, when there exist statistical differences in the results from one algorithm with respect to another, the respective best value is displayed in boldface. Let us proceed with the analysis of the results from a global perspective, i.e., considering dimensions $D = 100, 500$, and 1000. It can be observed that: *i)* when there exist significant differences between ACO$_{\mathbb{R}}$ and ACO$_{\mathbb{R}}$-Div (either for $q = 0.001$ or $q = 0.01$), the statistical test indicates that ACO$_{\mathbb{R}}$-Div outperforms ACO$_{\mathbb{R}}$; *ii)* when no statistical difference is reported, the best values found are the same or ACO$_{\mathbb{R}}$-Div achieved the best ones; *iii)* in general, for functions f_1, f_3, and f_5, ACO$_{\mathbb{R}}$-Div outperforms ACO$_{\mathbb{R}}$ for both values of parameter q. Also, this can be observed in some groups (with respect to FES) for function f_2, however, only for dimensions $D = 500$ and $D = 1000$; *iv)* for function f_6 no real differences were observed at all for ACO$_{\mathbb{R}}$ and ACO$_{\mathbb{R}}$-Div using any of the two values considered for $q = 0.001$ and $q = 0.01$; and *v)* when comparing ACO$_{\mathbb{R}}$-Div under $q = 0.001$ and $q = 0.01$, the best results are achieved by ACO$_{\mathbb{R}}$-Div with $q = 0.001$.

4.3 Comparison with some relevant algorithms

In this section we make an indirect comparison of the results achieved by ACO$_{\mathbb{R}}$-Div with the results reported by the eight ranked algorithms[2] that participated in the competition at CEC'08: 1) MTS[23], 2) LSDEA-gl [25], 3) jDEdynNP-F [3], 4) MLCC [26], 5) DMS-PSO [28], 6) DEwSAcc [27], 7) UEP [16], and 8) EPUS-PSO [9]. Table 4.3 shows the best reported values for the eight ranked algorithms and also, the best found values from the ACO$_{\mathbb{R}}$-Div algorithm (in boldface). In this case, the results correspond to dimension $D = 1000$ and $FES = 5.00e+6$ (i.e., the maximum number of dimensions and function evaluations regarding the respective protocol given in [22]). The values in parentheses indicate that ACO$_{\mathbb{R}}$-Div outperforms (from

[2]The ranking can be downloaded from `http://nical.ustc.edu.cn/papers/CEC2008_SUMMARY.pdf`

Table 2: Results achieved by the eight ranked algorithms that participated in the competition at CEC'08 and the results obtained by $ACO_{\mathbb{R}}$-Div with $q = 0.001$.

	Test Functions					
Algorithm	1	2	3	4	5	6
1) MTS	0.00e+00	4.72e-02	3.41e-04	0.00e+00	00.0e+00	1.24e-11
2) LSDEA-gl	(3.22e-13)	1.04e-05	(1.73e+03)	5.45e+02	1.71e-13	4.26e-13
3) jDEdynNP-F	(1.14e-13)	1.95+01	(1.31e+03)	2.17e-04	3.98e-14	1.47e-11
4) MLCC	(8.46e-13)	1.09e+02	(1.80e+03)	1.37e-10	4.18e-13	1.06e-12
5) DMS-PSO	0.00e+00	9.15e+01	(8.98e+09)	3.84e+03	00.0e+00	7.76e+00
6) DEwSAcc	(8.79e-03)	9.61e+01	(9.15e+03)	1.82e+03	(3.58e-03)	2.30e+00
7) UEP	(5.37e-12)	1.05e+02	(1.96e+03)	1.03e+04	(8.87e-04)	(1.99e+01)
8) EPUS-PSO	(5.53e+02)	4.66e+01	(8.37e+05)	7.58e+03	(5.89e+00)	1.89e+01
$ACO_{\mathbb{R}}$-Div ($q = 0.001$)	**00.0e+00**	**1.50e+02**	**1.29e+03**	**1.23e+04**	**8.00e-09**	**1.89e+01**

the perspective of the best found values) the respective algorithm in the table. It can be seen that $ACO_{\mathbb{R}}$-Div outperforms some of the compared algorithms in functions: f_1 [outperforms 6 algorithms], f_3 [outperforms 7 algorithms], f_5 [outperforms 2 algorithms], and f_6 [outperforms 1 algorithm]. These results indicate that $ACO_{\mathbb{R}}$-Div is competitive with respect to other *ad hoc* algorithms (some of which are quite elaborate) developed for solving continuous optimization problems. However, an enhanced version of $ACO_{\mathbb{R}}$-Div (e.g., with a better diversity preservation mechanism) and a direct comparison with state-of-the-art algorithms are still required to better estimate the actual position of $ACO_{\mathbb{R}}$-Div in a possible ranking of optimization algorithms used for solving large scale continuous problems.

5 Conclusions and Future Work

In this paper we have presented a study of scalability of $ACO_{\mathbb{R}}$ for large continuous optimization problems. An early experimental study allowed us to detect some weaknesses of $ACO_{\mathbb{R}}$ when dealing with large dimensionality problems. Based on these results, we proposed a simple mechanism that introduces diversity in the kernels' population when it reaches a diversity value under a certain minimum threshold. This new algorithm was called $ACO_{\mathbb{R}}$-Div and was capable of significantly improving the performance of $ACO_{\mathbb{R}}$.

We have also shown that our proposed $ACO_{\mathbb{R}}$-Div approach is competitive with respect to other bio-inspired metaheuristics that have been specifically designed to solve continuous optimization problems.

As part of our future work, we intend to study more elaborate mechanisms to detect and/or deal with the loss of diversity in $ACO_{\mathbb{R}}$. We also wish to perform a more comprehensive comparison of our approach with respect to state-of-the-art metaheuristics used for continuous optimization. Finally, we are also interested in applying *ad-hoc* and automatic tools for calibrating the parameter values of our approach.

6 Acknowledgments

The first author acknowledges the support from the UMI-LAFMIA 3175 CNRS at CINVESTAV-IPN and from the Universidad Nacional de San Luis, Argentina. The second author gratefully acknowledges support from CONACyT project no 103570.

References

[1] A. Afshar and S. Madadgar. Ant colony optimization for continuous domains: Application to reservoir operation problems. In *HIS '08: Proceedings of the 2008 8th International Conference on Hybrid Intelligent Systems*, pages 13–18, Washington, DC, USA, 2008. IEEE Computer Society.

[2] G. Bilchev and I.C. Parmee. The Ant Colony Metaphor for Searching Continuous Design Spaces. In Terence C. Fogarty, editor, *Evolutionary Computing. AISB Workshop*, pages 25–39. Springer, Sheffield, UK, April 1995.

[3] J. Brest, A. Zamuda, B. Boskovic, M. Sepesy Maucec, and V. Zumer. High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In *IEEE Congress on Evolutionary Computation*, pages 2032–2039, 2008.

[4] D. Corne, M. Dorigo, and F. Glover, editors. *New Ideas in Optimization*. McGraw-Hill International, London, UK, 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 Marco Dorigo, Gianni Di Caro, and Michael Sampels, editors, *Proceedings of the Third international Workshop on Ant Algorithms - ANTS 2002*, pages 216–221. Springer-Verlag. Lecture Notes in Computer Science Vol. 2463, Brussels, Belgium, September 2002.

[7] J. Dréo and P. Siarry. Continuous interacting ant colony algorithm based on dense heterarchy. *Future Generation Comp. Syst.*, 20(5):841–856, 2004.

[8] M. Guntsch and M. Middendorf. A population based approach for ACO. In Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and Günther R. Raidl, editors, *Applications of Evolutionary Computation. EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN*, pages 72–81, Kinsale, Ireland, April 2002. Springer. Lecture Notes in Computer Science Vol. 2279.

[9] S. Hsieh, T. Sun, C. Liu, and S. Tsai. Solving large scale global optimization using improved particle swarm optimizer. In *IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2008.

[10] X. Hu, J. Zhang, and Y. Li. Orthogonal methods based ant colony search for solving continuous optimization problems. *J. Comput. Sci. Technol.*, 23(1):2–18, 2008.

[11] M. Kong and P. Tian. A direct application of ant colony optimization to function optimization problem in continuous domain. In *ANTS Workshop*, pages 324–331, 2006.

[12] G. Leguizamón and C.A. Coello Coello. Boundary Search for Constrained Numerical Optimization Problems with an Algorithm Inspired on the Ant Colony Metaphor. *IEEE Transactions on Evolutionary Computation*, 13(2):350–368, April 2009.

[13] J. J. Liang and P. N. Suganthan. Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constrain-Handling Mechanism. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 316–323, Vancouver, BC, Canada, July 2006. IEEE.

[14] C. Ling, S. Jie, Q. Ling, and C. Hongjian. A method for solving optimization problems in continuous space using ant colony algorithm. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of the Third International Workshop, (ANTS'2002)*, pages 288–289, Brussels, Belgium, 2002. Springer Verlag. Lecture Notes in Computer Science Vol. 2463.

[15] L. Qin Ling Chen, J. Shen and H. Chen. An improved ant colony algorithm in continuous optimization. *Journal of Systems Science and Systems Engineering*, 12(2):224–235, 2003.

[16] C. MacNish and X. Yao. Direction matters in high-dimensional optimisation. In *IEEE Congress on Evolutionary Computation*, pages 2372–2379, 2008.

[17] 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.

[18] S.H. Pourtakdoust and H. 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, *Proceedings of Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS Workshop 2004*, pages 294–301. Lecture Notes in Computer Science Vol. 3172, Brussels, Belgium, 2004. Springer-Verlag.

[19] K. Socha. ACO for continuos and mixed-variable optimization. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Proceedings of Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS Workshop 2004*, pages 25–36, Brussels, Belgium, 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3172.

[20] K. Socha and C. Blum. An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. *Neural Comput. Appl.*, 16(3):235–247, 2007.

[21] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173, 2008.

[22] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. Benchmark functions for the cec'2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.

[23] L. Tseng and C. Chen. Multiple trajectory search for large scale global optimization. In *IEEE Congress on Evolutionary Computation*, pages 3052–3059, 2008.

[24] R. K. Ursem. Diversity-guided evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature VII (PPSN-2002)*, pages 462–471. Springer Verlag, 2002.

[25] Y. Wang and B. Li. A restart univariate estimation of distribution algorithm: sampling under mixed gaussian and lévy probability distribution. In *IEEE Congress on Evolutionary Computation*, pages 3917–3924, 2008.

[26] Z. Yang, K. Tang, and X. Yao. Multilevel cooperative coevolution for large scale optimization. In *IEEE Congress on Evolutionary Computation*, pages 1663–1670, 2008.

[27] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In *IEEE Congress on Evolutionary Computation*, pages 3718–3725, 2008.

[28] S. Zhao, J.J. Liang, P.N. Suganthan, and M.F. Tasgetiren. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In *IEEE Congress on Evolutionary Computation*, pages 3845–3852, 2008.

Appendix

Table 3: Errors values for problems 1-6 ($D = 100$) with $q \in \{0.001, 0.01\}$ up to $5.00e + 5$ FES

FES	q	Stats	1	2	3	4	5	6
$5.00e+3$	0.001	1^{st} (Best)	1.9326e+4	9.8291e+1	0.2217e+10	**8.0361e+2**	1.6193e+2	2.0307e+1
		13^{th} (Median)	3.2087e+4	1.1651e+2	0.5655e+10	1.1896e+3	2.4576e+2	2.1173e+1
		25^{th} (Worst)	4.8736e+4	1.3220e+2	1.2903+e10	1.7577e+3	4.6825e+2	2.1312e+1
	0.01	1^{st} (Best)	**1.3583e+4**	9.5175e+1	3.3488e+9	8.6345e+2	**1.5270e+2**	1.9405e+1
		13^{th} (Median)	2.4099e+4	1.1402e+2	6.1663e+9	1.0506e+3	2.1619e+2	2.1178e+1
		25^{th} (Worst)	4.7441e+4	1.2397e+2	9.5319e+9	1.5163e+3	5.6468e+	2.1337e+1
		p-values	**0.0179**	0.4492	0.6554	**0.0209**	**0.0209**	0.9613
$5.00e+4$	0.001	1^{st} (Best)	**6.0000e-6**	7.3209e+1	4.5120e+2	6.6045e+2	1.7520e-1	0.1888e-1
		13^{th} (Median)	8.9597e+1	8.4065e-1	1.5690e+5	8.6218e+2	0.1623e-1	2.1065e+1
		25^{th} (Worst)	1.7527e+3	1.0279e+2	5.4189e+8	1.0859e+3	3.6405e+1	2.1269e+1
	0.01	1^{st} (Best)	**6.0000e-6**	**5.4617e+1**	**2.5061e+2**	**5.5227e+2**	**1.0000e-5**	1.8871e+1
		13^{th} (Median)	8.0000e-5	7.2852e+1	4.2594e+2	7.0988e+2	6.6830e-1	2.1065e+1
		25^{th} (Worst)	0.7562e+1	9.1319e+1	1.8281e+3	1.0275e+3	0.4357e+1	2.1269e+1
		p-values	**5.8405e-8**	**4.6094e-5**	**1.9973e-6**	**3.0245e-5**	**1.8141e-6**	0.9923
$5.00e+5$	0.001	1^{st} (Best)	1.0000e-6	2.0334e+1	1.7257e+2	6.6045e+2	1.7460e-1	1.8884e+1
		13^{th} (Median)	4.0630e+1	2.6511e+1	1.5624e+5	8.6217e+2	0.1623e+1	1.9858e+1
		25^{th} (Worst)	1.742e+3	3.2984e+1	5.4189e+8	1.0859e+3	3.6405e+1	2.1175e+1
	0.01	1^{st} (Best)	**0**	**1.5818e+1**	**2.0430e-1**	**5.5227e+2**	**0**	1.7712e+1
		13^{th} (Median)	0	2.2484e+1	8.7483e+1	7.0987e+2	6.6830e-1	2.0944e+1
		25^{th} (Worst)	0.7562e+1	2.6203e+1	1.7667e+2	1.0275e+3	0.4357e+1	2.1158e+1
		p-values	**4.1391e-10**	**1.5127e-5**	**1.5967e-9**	**3.0245e-5**	**1.6377e-6**	0.6871

Table 4: Errors values for problems 1-6 ($D = 500$) with $q \in \{0.001, 0.01\}$ up to $25.00e + 6$ FES

FES	q	Stats	1	2	3	4	5	6
$2.50e+4$	0.001	1^{st} (Best)	2.3922e+5	1.5557e+2	**9.4274e+10**	7.2027e+3	**2.8067e+3**	2.1407e+1
		13^{th} (Median)	4.1694e+5	1.6275e+2	1.4279e+11	7.9304e+3	3.4474e+3	2.1451e+1
		25^{th} (Worst)	7.2040e+5	1.8704e+2	2.8528e+11	8.4763e+3	5.4779e+3	2.1474e+1
	0.01	1^{st} (Best)	2.8948e+5	1.5434e+2	1.2241e+11	6.8459e+3	2.8935e+3	2.1398e+1
		13^{th} (Median)	3.8390e+5	1.6051e+2	1.8207e+11	7.7979e+3	3.8204e+3	2.1450e+1
		25^{th} (Worst)	5.1286e+5	1.8704e+2	2.7521e+11	9.5348e+3	4.7482e+3	2.1475e+1
		p-values	0.4728	0.7934	**5.9408e-004**	0.1160	**0.0209**	0.6139
$2.50e+5$	0.001	1^{st} (Best)	5.6686e+1	1.4508e+2	**5.9546e+6**	6.2038e+3	0.28101e-1	**2.1357e+1**
		13^{th} (Median)	1.2089e+4	1.5043e+2	1.7269e+9	6.9793e+3	5.5363e+1	2.1437e+1
		25^{th} (Worst)	5.4493e+4	1.5438e+2	9.6816e+10	7.5168e+3	6.3174e+2	2.1463e+1
	0.01	1^{st} (Best)	**1.0082e+1**	**1.4187e+2**	1.3680e+7	**5.7009e+3**	0.1095e-1	2.1398e+1
		13^{th} (Median)	6.5797e+2	1.4935e+2	5.3053e+8	0.65447e+1	2.6645e+2	2.1450e+1
		25^{th} (Worst)	6.7055e+4	1.5564e+2	2.9510e+10	7.1207e+3	5.0414e+2	2.1475e+1
		p-values	**3.0746e-004**	**0.0457**	0.0808	**1.6516e-005**	0.1302	**0.0328**
$2.50e+6$	0.001	1^{st} (Best)	5.6633e+1	**1.0641e+2**	5.9496e+6	6.2038e+3	2.8098e+0	**1.9923e+1**
		13^{th} (Median)	1.2087e+4	1.1471e+2	1.7268e+009	6.9790e+3	5.5362e+1	2.1397e+1
		25^{th} (Worst)	5.4491e+4	1.2792e+2	9.6816e+10	7.5166e+3	6.3174e+2	2.1441e+1
	0.01	1^{st} (Best)	**0.7395e+1**	1.4508e+2	1.2545e+7	**5.7009e+3**	1.0643e+0	2.1357e+1
		13^{th} (Median)	6.5714e+2	1.5043e+2	5.3006e+8	6.5447e+3	2.6645e+1	21.4505
		25^{th} (Worst)	6.7053e+4	1.5438e+2	2.9509e+10	7.1207e+3	5.0414e+2	21.4755
		p-values	**2.8526e-004**	**0.0457**	0.0808	**1.6516e-5**	0.1302	0.5094

Table 5: Errors values ($f_i(\mathbf{x}) - f_i(\mathbf{x}^*)$), for $i = 1, \ldots, 6$; ($D = 1000$); $q \in \{0.001, 0.01\}$ up to $50.00e+6$ FES

FES	q	Stats	1	2	3	4	5	6
$5.00e+4$	0.001	1^{st} (Best)	**7.3434e+5**	1.6856e+2	**4.4900e+10**	**1.6311e+4**	**6.5600e+3**	2.1470e+1
		13^{th} (Median)	8.7221e+5	1.7305e+2	4.3077e+11	1.7290e+4	8.0630e+3	2.1502e+1
		25^{th} (Worst)	1.4940e+6	1.9101e+2	7.1003e+11	1.8404e+4	1.2989e+4	2.1524e+1
	0.01	1^{st} (Best)	8.5456e+5	1.6800e+2	5.1477e+11	1.6573e+4	7.9829e+3	2.1465e+1
		13^{th} (Median)	1.0237e+6	1.7535e+2	6.2513e+11	1.7799e+4	9.7774e+3	2.1500+1
		25^{th} (Worst)	1.2356e+6	1.8887e+2	1.2922e+12	1.9632e+4	1.1053e+4	2.1524e+1
		p-values	**9.7208e-004**	0.1936	**3.3440e-007**	**0.0055**	**3.0245e-005**	0.9381
$5.00e+5$	0.001	1^{st} (Best)	9.3070e+2	1.6286e+2	4.0001e+8	1.4238e+4	3.0609e+1	1.9980e+1
		13^{th} (Median)	2.1936e+4	1.6722e+2	6.1033e+9	1.4893e+4	1.4342e+2	2.1494e+1
		25^{th} (Worst)	1.5386e+5	1.7263e+2	8.8517e+10	1.5697e+4	1.0235e+3	2.1518e+1
	0.01	1^{st} (Best)	9.5884e+2	1.6284e+2	2.7486e+8	**1.3694e+4**	0.4625e-1	2.1462e+1
		13^{th} (Median)	1.4540e+4	1.6746e+2	2.2323e+9	1.4649e+4	8.8825e+1	2.1489e+1
		25^{th} (Worst)	4.5222e+4	1.7042e+2	1.1079e+11	1.5801e+4	4.5928e+2	2.1520e+1
		p-values	0.0842	0.7269	0.0598	**0.0099**	0.1206	0.9690
$5.00e+6$	0.001	1^{st} (Best)	9.2873e+2	1.5107e+2	3.9600e+8	1.4238e+4	3.0560e+1	**1.9924e+1**
		13^{th} (Median)	2.1936e+4	1.5500e+2	6.0860e+9	1.4893e+4	1.4340e+2	2.1483e+1
		25^{th} (Worst)	1.5385e+5	1.6461e+2	8.8517e+19	1.5697e+5	1.0235e+3	2.1520e+1
	0.01	1^{st} (Best)	9.4767e+2	1.4637e+2	3.4711e+7	**1.3694e+4**	4.6256e+0	2.1200e+1
		13^{th} (Median)	1.0929e+4	1.5476e+2	6.1684e+9	1.4648e+4	8.8636e+1	2.1330e+1
		25^{th} (Worst)	5.2064e+4	1.6461e+2	1.1077e+11	1.5801e+4	4.5928e+2	2.1499e+1
		p-values	0.1653	0.1837	0.6275	**0.0093**	0.4244	**0.0129**

Table 6: Comparison of $ACO_{\mathbb{R}}$ and $ACO_{\mathbb{R}}$-Div with $D = 100$ and $q \in \{0.001, 0.01\}$

FES	q	Stats	1	2	3	4	5	6
$5.00e+3$	0.001	Best	1.9326e+4	9.8291e+1	0.2217e+10	8.4231e+2	5.5226e+2	2.0307e+1
		$Best_{div}$	**1.6226e+4**	9.8291e+1	0.2217e+10	8.0361e+2	**1.6193e+2**	2.0307e+1
	0.01	Best	1.3583e+4	9.5175e+1	3.3488e+9	8.6345e+2	1.5270e+2	1.9405e+1
		$Best_{div}$	1.3583e+4	9.5175e+1	3.3488e+9	8.7386e+2	1.5270e+2	1.9405e+1
$5.00e+4$	0.001	Best	6.0000e-6	7.3209e+1	4.5120e+2	6.6045e+2	1.7520e-1	0.1888e-1
		$Best_{div}$	**2.0000e-6**	7.1295e+1	**1.5527e+1**	6.6043e+2	**1.0000e-6**	1.8871e+1
	0.01	Best	6.0000e-6	5.4617e+1	2.5061e+2	5.5227e+2	1.0000e-5	1.8871e+1
		$Best_{div}$	**3.0000e-6**	5.4617e+1	2.4322e+1	5.5226e+2	**5.0000e-6**	1.7648e+1
$5.00e+5$	0.001	Best	1.0000e-6	2.1856e+1	1.7257e+2	6.6049e+2	1.7460e-1	1.8884e+1
		$Best_{div}$	**0**	2.0334e+1	**8.4803e+1**	6.6045e+2	**0**	1.8857e+1
	0.01	Best	0	1.5818e+1	2.0430e-1	5.5227e+2	0	1.7712e+1
		$Best_{div}$	0	1.4853e+1	**0.1242 e+0**	5.5227e+2	0	1.7712e+1

Table 7: Comparison of $ACO_{\mathbb{R}}$ and $ACO_{\mathbb{R}}$-Div with $D = 500$ and $q \in \{0.001, 0.01\}$

FES	q	Stats	1	2	3	4	5	6
$5.00e+4$	0.001	Best	2.3922e+5	1.5557e+1	9.4274e+010	7.4522e+3	2.8067e+3	21.4077
		$Best_{div}$	**2.0584e+5**	1.5557e+1	9.0867e+10	7.2027e+3	**2.0983e+3**	21.4077
	0.01	Best	2.8948e+5	1.5434e+2	1.2241e+11	6.7664e+3	2.8935e+3	21.3983
		$Best_{div}$	2.8948e+5	1.5434e+2	1.2241e+01	6.8459e+3	2.8935e+3	21.3983
$5.00e+5$	0.001	Best	5.6686e+1	1.4508e+2	5.9546e+6	6.2038e+003	0.28101e+1	21.3579
		$Best_{div}$	**0.0898**	1.4508e+2	**9.4978e+003**	6.1980e+3	**0.00086+1**	21.3579
	0.01	Best	1.0082e+1	1.4187e+2	1.3680e+7	5.7009e+3	0.10951e+1	21.3983
		$Best_{div}$	**9.1070e-1**	1.4187e+2	**3.8123e+4**	5.7207e+3	**0.02307e+1**	21.3983
$5.00e+6$	0.001	Best	56.6338	1.0641e+2	1.2545e+7	6.2038e+3	2.8098e+0	1.9923e+1
		$Best_{div}$	**0**	**1.0000e+2**	**5.6733e+2**	6.1980e+3	**0**	1.9920e+1
	0.01	Best	7.3952	1.0438e+1	1.2545e+7	5.7007e+3	1.0643e+0	1.9889e+1
		$Best_{div}$	**0**	**1.0000e+1**	**5.5533e+2**	5.7005e+3	**0**	1.9881e+1

Table 8: Comparison of $\text{ACO}_{\mathbb{R}}$ and $\text{ACO}_{\mathbb{R}}$-Div with $D = 1000$ and $q \in \{0.001, 0.01\}$

FES	q	Stats	1	2	3	4	5	6
$5.00e+4$	0.001	Best	7.3434e+5	1.6856e+2	4.4900e+10	1.6311e+4	6.5600e+3	2.1470e+1
		Best_{div}	**5.9803e+5**	**2.1565e+0**	4.4018e+10	1.6078e+4	**5.2271e+3**	2.1470e+1
	0.01	Best	8.5456e+5	1.6800e+2	5.1477e+11	1.6573e+4	7.9829e+3	2.1465e+1
		Best_{div}	8.5456e+5	**4.8252e+1**	5.1477e+11	1.6694e+4	8.7005e+3	2.1465e+1
$5.00e+5$	0.001	Best	9.3070e+2	1.6286e+2	4.0001e+8	1.4238e+4	3.0609e+1	1.9980e+1
		Best_{div}	**2.1565e+0**	1.6286e+2	**4.0899e+5**	1.4245e+4	**0.1454e+0**	1.9998e+1
	0.01	Best	9.5884e+2	1.6284e+2	2.7486e+8	1.3694e+4	4.6256e+0	2.1462e+1
		Best_{div}	**4.8252e+1**	1.6284e+2	**6.7842e+6**	1.3678e+4	**1.4757e+0**	2.1462e+1
$5.00e+6$	0.001	Best	9.2873e+2	1.5107e+2	3.9600e+8	1.4238e+4	3.0560e+1	1.9924e+1
		Best_{div}	**0**	1.5010e+2	**7.0498e+5**	1.4011e+4	**0**	1.9290e+1
	0.01	Best	9.4767e+2	1.5637e+2	3.4711e+7	1.3694e+4	4.6256e+0	2.1200e+1
		Best_{div}	**0.4022e+0**	**1.5089e+2**	**4.7675e+6**	**1.4340e+3**	0.0212e+0	2.0988e+1