Multivariate Ant Colony Optimization in Continuous Search Spaces

Fabrício O. de França, Guilherme P. Coelho, Fernando J. Von Zuben
Laboratory of Bioinformatics and Bioinspired Computing (LBiC)
School of Electrical and Computer Engineering (FEEC)
University of Campinas (Unicamp)
Campinas, SP, Brazil
P.O. Box 6101
Zip Code 13083-970
{olivetti, gcoelho, vonzuben}@dca.fee.unicamp.br

Romis R. de Faissol Attux
Department of Computer Engineering and Industrial Automation (DCA)
School of Electrical and Computer Engineering (FEEC)
University of Campinas (Unicamp)
Campinas, SP, Brazil
P.O. Box 6101
Zip Code 13083-970
attux@dca.fee.unicamp.br

ABSTRACT

This work introduces an ant-inspired algorithm for optimization in continuous search spaces that is based on the generation of random vectors with multivariate Gaussian pdf. The proposed approach is called MACACO – Multivariate Ant Colony Algorithm for Continuous Optimization – and is able to simultaneously adapt all the dimensions of the random distribution employed to generate the new individuals at each iteration. In order to analyze MACACO's search efficiency, the approach was compared to a pair of counterparts: the Continuous Ant Colony System (CACS) and the approach known as Ant Colony Optimization in $\Re^n$ (ACO$_R$). The comparative analysis, which involves well-known benchmark problems from the literature, has indicated that MACACO outperforms CACS and ACO$_R$ in most cases as the quality of the final solution is concerned, and it is just about two times more costly than the least expensive contender.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – heuristic methods

General Terms

Algorithms

Keywords

Ant Colony Optimization, Continuous Optimization, Multivariate Normal Distribution

1. INTRODUCTION

Ant Colony Optimization (ACO) was first proposed by Dorigo [4] as an attempt to use the ant foraging behavior as a source of inspiration for the development of new search and optimization techniques. By using the pheromone trail as a reinforcement signal for the choice of which path to follow, ants tend to find minimal routes from the nest to the food source. The system is based on the fact that each ant, while foraging, deposits a chemical substance known as pheromone on the path adopted.

Initially, this framework was proposed to solve combinatorial problems such as the Traveling Salesman Problem (TSP) [5] and the Quadratic Assignment Problem (QAP) [15], which is justifiable in light of the fact that the inspiration took form in the context of graph optimization, although real pheromone trails are deposited on a continuous space.

In 1995, Bilchev and Parmee first proposed an adaptation of this framework to continuous search spaces [1], which was named CACO (*Continuous Ant Colony Optimization*). Their proposal initializes a nest on a given point of the search space and generates random vectors corresponding to the directions that will be followed by each ant in its quest for better solutions. If an ant is successful in such pursuit, the direction vector chosen is updated. In [11], another ant-inspired approach for continuous optimization, called API (from the ant *Pachycondyla apicalis*), was proposed. In this approach, each ant searches independently for a solution, though they all start from the same point (the nest). This algorithm also uses a recruitment strategy to refine the search. Later, in 2002, Dréo and Siarry created the CIAC (*Continu-*

ous Interacting Ant Colony) [7] algorithm, which uses some spots on the search space to which the ants are attracted and a certain degree of direct communication between ants to improve the exploration.

Pourtakdoust and Nobahari proposed in 2004 another Ant System approach for this problem - the algorithm called CACS (*Continuous Ant Colony System*) [12]. In their approach, the discrete pheromone probabilistic function is replaced by a Gaussian (normal) probability density function (pdf) whose mean and variance parameters are dynamically adapted by the ants at each iteration.

Finally, in 2006, Socha and Dorigo [14] proposed another approach, inspired by PB-ACO (*Population-Based ACO*) [8], in which the solutions are built according to an archive of the n best solutions found so far. This algorithm also employs the same idea of CACS [12], working with a Gaussian pdf as the pheromone, and was named ACO$_R$ (Ant Colony Optimization in $\Re^n$). In [14], Socha and Dorigo compared the performance of ACO$_R$ with those of CACO, API and CIAC, together with some evolutionary approaches, and ACO$_R$ led to the best results in most cases.

It can be inferred from the comparative experimental results shown in [14] that there is room for improvement of the performance achieved by ACO algorithms for continuous optimization. A hitherto unexplored possibility is to consider the existence of dependence between variables: all ant-colony-based algorithms proposed so far treat them as being mutually independent, which can reduce the capability of the employed sampling mechanism.

In order to take a step in this direction, we propose a new ant-based technique that uses the information contained in the covariance matrix of the ant population in order to create a multivariate Gaussian random pdf to be employed in the displacement of the ants along the continuous search space. By doing so, we intend to map and explore the search space more effectively. This proposal was applied to several well-known benchmark problems and the obtained results were compared to those of the CACS and ACO$_R$ algorithms. As will be shown, the introduced multivariate distribution mechanism improves the solution quality and the exploration of the search space, and, consequently, leads to a significant performance improvement in comparison with the CACS and ACO$_R$ algorithms.

The paper is organized as follows. Section 2 outlines the basic aspects of the Ant Colony Optimization algorithm together with the CACS and ACO$_R$ extensions to deal with continuous search space problems, and also briefly explains some related approaches. In Section 3, we introduce a new algorithm based on covariance information extracted from the solutions already proposed by the ants. The algorithms are evaluated and compared in Section 4 with the aid of a diversified set of representative benchmark problems. Concluding remarks are presented in Section 5 together with a discussion on the formal and methodological contributions of this work. A proposal of several paths for further investigation is also part of Section 5.

2. ANT SYSTEM AND ANT COLONY OPTIMIZATION

The Ant Colony Optimization (ACO) [4] was the first ant-based algorithm applied to solve combinatorial optimization problems. More than one decade after this initiative, several different versions, improvements and applications have been presented (c.f. Dorigo and Stützle [6], de França *et al.* [3]). In this section, the original proposal of Dorigo [4] will be briefly reviewed, together with some adaptations presented in the literature to solve optimization problems in continuous search spaces.

2.1 Ant Colony Optimization

The standard ACO [4] is conceptually simple, as described in Algorithm 1.

Algorithm 1 Basic ACO Algorithm.

```
while $iteration < maximumNumberOfIterations$ do
    for each ant do
        build_solution();
        update_pheromone();
    end for
end while
```

In the classical ACO presented in Algorithm 1, the combinatorial problem to be solved is understood in terms of a graph, and the ants are placed on its nodes. A solution to this problem thus corresponds to a sequence of edges that are selected on a one-by-one basis by a given ant. When a complete solution is built, it is evaluated, and the edges belonging thereto have their pheromone concentration increased proportionally to a relative quality index. The pheromone concentration at each edge influences the probability of its selection by the ants in the next generation of the algorithm. Therefore, edges belonging to high-quality solutions will tend to have a higher probability of being selected during the construction of a new solution to the problem. Depending on the formulation of the problem, the way the candidate solutions are represented may force the pheromone to be associated with nodes, and not with edges of the graph.

In Algorithm 1, the procedure `build_solution()` builds a solution to the problem being solved based on a pheromone trail and on optional information heuristics which will not be included in the standard formulation described in this subsection. Each ant k traverses one edge per iteration t and, at each edge, the local information about its pheromone level, τ_{ij}, is used by the ant so that it can probabilistically decide the next node to move to, according to the following rule:

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}}{\sum_{j \in J^k} \tau_{ij}} & \text{if} \quad j \in J^k \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\tau_{ij}(t)$ is the pheromone level of edge (i, j), i.e. edge connecting nodes i and j, J^k is the list of nodes yet to be visited by ant k, and node i is the current node of ant k.

In the procedure `update_ pheromone()`, the pheromone level of edge (i, j) is updated according to Equation 2:

$$\tau_{ij} \leftarrow \rho.\tau_{ij} + \Delta\tau_{ij}, \tag{2}$$

where $\rho \in (0, 1]$ is the pheromone decay rate and $\Delta\tau_{ij}$ is the increment in the pheromone level. In minimization problems, the pheromone increment may be given by:

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{f(S)} & \text{if} \quad (i, j) \in S \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where S is the solution used to update the trail and $f(S)$ is a function that reflects the cost of a solution, i.e., the lower $f(S)$ the better the quality of proposal S.

2.2 Adapting the ACO Algorithm to deal with Continuous Search Spaces

In order to adapt the ACO algorithm to deal with continuous search spaces, it is necessary to modify the probabilistic model used in the process of sampling new candidate solutions (given in Equation 1) so that solutions be generated as real-valued parameter vectors. To do that, the pheromone must define a probability density function that determines the chance of choosing certain regions of the search space when random solution vectors are created. The chance tends to be proportional to the relative quality of such regions as observed in the past iterations of the algorithm.

In general, a Gaussian pdf (Figure 1) is adopted, since it is easily sampled by methods such as Box-Muller [2] and the Ziggurat algorithm [10]. However, it has the disadvantage of being able to enclose just one promising region of the search space, since it possesses a single maximum.

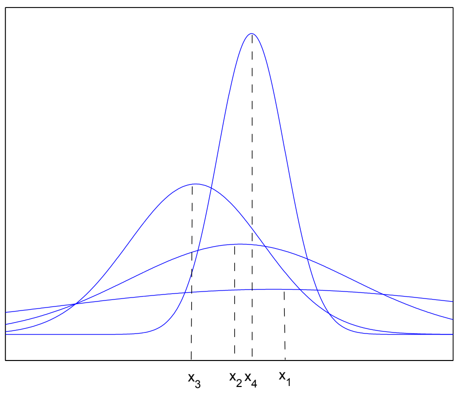


Figure 1: Four Gaussian probability density functions (centered at x_1, x_2, x_3 and x_4) enclosing four different regions of the problem.

The aperture (variance) of the Gaussians, as illustrated in Figure 1, determines the degree of dispersion of the new random solutions around the corresponding centers.

Therefore, in ACO algorithms for optimization in continuous search spaces, each new solution vector is created by sampling Gaussian random variables, one at a time, for each dimension of the problem. The differences between the various approaches reported in the literature lie in how they update the center and the variance of the Gaussian pdfs. In the next section, we will show how CACS [12] and ACO$_R$ [14] deal with such parameters.

2.3 CACS Algorithm

The CACS algorithm was first presented in [12], in which a continuous pheromone model consisting of a Gaussian pdf centered on the best solution found so far was proposed. The variance of such pdf starts with a value three times greater than the range of each variable of the problem (i.e., three times $(x_{max} - x_{min})$), and, during the evolution of the algorithm, it is modified according to a weighted average of the distance between each individual in the population and the best solution found so far, as shown in Equation 4, where K is the number of ants defined in the algorithm.

$$\sigma^2 = \frac{\sum_{j=1}^{K} \frac{1}{|f_j - f_{min}|}(x_j - x_{min})^2}{\sum_{j=1}^{k} \frac{1}{|f_j - f_{min}|}}. \tag{4}$$

The main advantages of this algorithm are that it requires the setting of just one parameter (the number of ants in the population) and presents a very simple mechanism to generate the ants of the next generation. On the other hand, a clear drawback is that it only investigates one promising region of the problem at a time, which means that the algorithm tends to concentrate the Gaussian pdf around local optima very quickly, thus leading to a premature convergence.

2.4 ACO$_R$ Algorithm

The ACO$_R$ [14] algorithm was conceived to deal with the aforementioned *premature convergence drawback* of proposals that use a single Gaussian function. This algorithm consists of an archive that holds the k best solutions found so far, being, in conceptual terms, each solution correspondent to the center of a different Gaussian pdf. Moreover, this archive is used to calculate the variance of each distribution, so that the whole process can be described as follows.

Initially, the whole archive is randomly created (using a uniform distribution), and the generated individuals are sorted in descending order of fitness (best-to-worst). Then, the main iteration starts by first attributing a solution of the archive to each ant of the problem, with probability proportional to the weight of the l-th archive solution ω_l (Equation 5):

$$\omega_l = \frac{1}{qK\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2K^2}}, \tag{5}$$

where K is the number of ants, l is the rank of the solution on the archive, and q is a variable called locality of the search process. The role of variable q is to balance exploitation and exploration. The center of each Gaussian is then defined as being the correspondent archive solution, and its variance is given by Equation 6:

$$\sigma_l^i = \xi \sum_{e=1}^{K} \frac{||s_e^i - s_l^i||}{K-1}, i = 1, ..., dim, \tag{6}$$

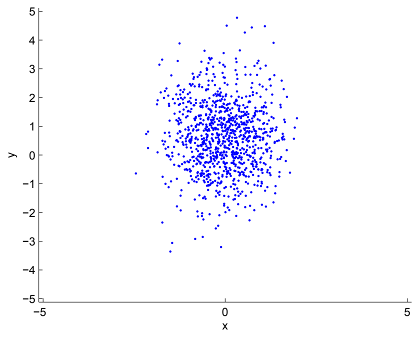
where ξ – called the speed of convergence – determines how fast the solutions of the archive will converge, s is a solution belonging to the archive, and dim is the dimension of the search space.

Finally, this Gaussian distribution with center and variance defined as described above is used to generate a new solution to the problem. After each ant has built a candidate solution, these candidate solutions are inserted into the archive, which is sorted again. The algorithm then iteratively removes the worst solutions until the archive returns to its original size. This approach has a clear advantage – the evolution of several Gaussian distributions in parallel – and an interesting feature: it reduces the selective pressure by adopting the rank-based selection mechanism.

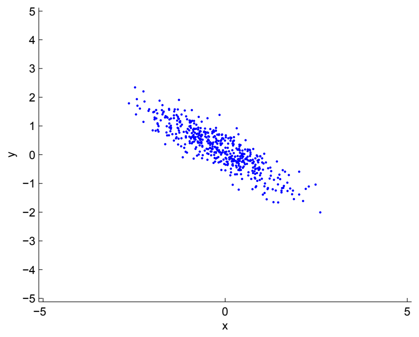
Even though this process reduces the chance of premature convergence, the algorithm may still converge to local optima, particularly when the elements of the archive are very close to each other. As will be shown in Section 4, ACO$_R$ has a poor performance when applied to problems with multiple optima and/or large plateaus. Furthermore, the inclusion of the sorting step and the calculation of the distance between each solution lead to a significantly higher computational demand when compared to the CACS algorithm.

3. MACACO: MULTIVARIATE ANT COLONY ALGORITHM FOR CONTINUOUS OPTIMIZATION

One aspect of the CACS and ACO$_R$ algorithms described on the previous section is that they both treat each dimension of the search space as an independent variable when generating new solutions using a Gaussian pdf. Whenever the variables are taken to be independent, the given distribution always takes the shape of a hypersphere (e.g. Figure 2(a)) but, when the covariance matrix is used to create a multivariate distribution, it is possible to mould the shape of the given samples into a hyperellipse (as in Figure 2(b)) that can be rotated by any angle in order to bias the sampling process toward the promising regions of the search space.



(a)



(b)

Figure 2: Differences on using univariate and multivariate distributions to generate samples on a 2-dimensional problem. (a) Samples generated using two univariate distributions; and (b) using a multivariate distribution.

Therefore, in an attempt to explore the higher flexibility provided by the multivariate distribution, we propose a multivariate ant colony algorithm for continuous optimization, denoted MACACO. To do so, given a covariance matrix Σ and center $\vec{\mu}$, we have chosen to create the solution vectors according to the transformation-based technique described in [9], which consists of three simple steps:

- First create a vector $\vec{x} \in \Re^n$ by means of a Gaussian distribution with independent variables (i.e., $\vec{x_i} = N(0,1), i = 1, ..., dim$), so that $\vec{x}$ obeys the distribution $N(\vec{0}, I)$, where I is an identity matrix.
- Next calculate the matrix Φ of normalized eigenvectors (where each eigenvector corresponds to each column of Φ) of the covariance matrix Σ, and create a diagonal matrix Λ whose values are the corresponding eigenvalues. Let:

$$Q = \Lambda^{\frac{1}{2}}\Phi. \tag{7}$$

- Finally the vector $\vec{x}$ is transformed by Equation 8 so that $\vec{y}$ obeys the distribution $N(\vec{\mu}, \Sigma)$:

$$\vec{y} = Q\vec{x} + \vec{\mu}. \tag{8}$$

It is important to remark that, although the process of performing the required eigendecomposition may be costly, the very essence of the modus operandi of the MACACO algorithm allows that both the covariance update and the computation of the transformation matrix Q be carried out only once per generation.

As MACACO works with a covariance matrix (instead of a simple variance vector like the other two contenders), the updating step for each ant should be made in a way that the new distribution of the solutions tends to the promising region supposedly indicated by the candidate solutions at the previous iteration. This can be accomplished by simply recalculating this matrix using just 70% (empirical value) of the best solutions found at the current iteration. By doing that, the generated distribution will close down until the covariance becomes almost zero and centered at the best local optima found so far. Unfortunately, this updating procedure has a drawback, since it requires a large number of candidate solutions in order to make a good sampling.

Empirically, it was observed that this updating procedure works better whenever the initial distribution is centered near the global optimum. Since this optimum is unknown in practice, MACACO was divided into two steps, being the first one devoted to finding a good initial center vector and the second one conceived to locally optimize the region identified in the previous step. The pseudocode of the proposed algorithm is shown in Algorithm 2.

In Algorithm 2, the function `generate_initial_ants()` creates n_ants candidate solutions with uniform distribution $U(0, range)$, where $range$ is the domain of the problem. This function also calculates the initial covariance matrix and the transformation matrix Q. During Phase 1, the solutions are built according to Equation 8, using the best candidate solution found so far as the center vector $\vec{\mu}$ (this is done in the procedure `build_solution_meanbest()`). In Phase 2, the procedure `build_solution()` considers as the center of the distribution the mean of the recently created population of individuals, in order to obtain the transformation

Algorithm 2 MACACO Algorithm.

```
generate_initial_ants();
phase = 1;
while $iteration < maximumNumberOfIterations$ do
    for each ant do
        if phase == 1 then
            build_solution_meanbest();
        else
            build_solution();
        end if
    end for
    update_pheromone();
    if best solution is no more improving then
        phase = 2;
        restart_covar();
    end if
end while
```

of the multivariate distribution. The `update_pheromone()` procedure works by taking the 70% best solutions found at the current iteration and considering such individuals in the recalculation of the covariance matrix, as described above. Additionally, it recalculates the transformation matrix Q for the next iteration. When the best solution stops improving for a given number of iterations (empirically set as 10 iterations), the covariance is recalculated by sampling n_ants vectors with distribution $U(ant\_best, range)$, where ant_best is the best solution found so far. And if the algorithm is still working in Phase 1, it turns to Phase 2.

4. PERFORMANCE EVALUATION

In the experimental section of this work, we will compare our proposal (the MACACO algorithm) with two of the existing continuous-domain ACO algorithms: CACS [12] and ACO$_R$ [14]. The rationale of our choice is twofold: *i*) in [14], a thorough comparison between ACO$_R$ and other ACO-based approaches has already been made, but CACS was not included therein; and *ii*) the majority of the approaches not cited in [14] are very similar to CACS.

To evaluate the performance of the three algorithms, we have chosen six well-known benchmark problems from the literature (given in Table 1) and calculated the mean and standard deviation of the fitness of the best individuals (function values) found by each algorithm in 30 independent runs.

The parameters of the ACO$_R$ were set as suggested in the original paper [14], vide Table 2. The number of iterations was not chosen a priori; it was defined instead, for the sake of fairness, that each algorithm should be allowed a maximum of 10^6 function evaluations for all the benchmark problems studied here, except for the Rosenbrock function, which could be evaluated up to 6×10^6 times.

A population of 1000 ants was defined for both CACS and MACACO algorithms. It is important do highlight here that, although those algorithms had much more ants than ACO$_R$, the latter was executed for a smaller total number of iterations to keep the total number of function evaluations the same.

Also, to infer that even though we maintain a higher number of ants when compared to ACO$_R$, it will be shown, through some experiments, that raising this value may even worsen its average results.

It should be notice that the adopted parameters for the ACO$_R$ algorithms were the optimum values reported by the respective authors on similar experiments.

All the simulations performed in this work were made on an AMD Athlon64 3500+, 2.2GHz, 1GB of RAM running Windows XP Professional SP2 and all the algorithms were programmed in C++ and compiled with GCC 3.4.2. In the following sub-section, we will describe the obtained experimental results and discuss the performance of each algorithm.

4.1 Experimental Results

For each benchmark problem considered in this work, we have, in Table 3, the mean and standard deviation produced by each algorithm after 30 independent runs, being the best results outlined in bold.

From Table 3, comparing firstly ACO$_R$ and CACS, it can be seen that the CACS algorithm was capable of outperforming the ACO$_R$ in five of the six functions, being just slightly worse than ACO$_R$ on the Schwefel function, albeit presenting a smaller variance for this function.

Considering now the results obtained with the MACACO algorithm, it can be seen that a significant overall improvement was achieved, both on the mean and variance values for all benchmark problems, except for the Rosenbrock (and the Sphere, for which our proposal has found the global optimum similarly to the other algorithms). This is even more noticeable in functions like Rastrigin and Schwefel, which give rise to highly multimodal problems that represent a challenging task in continuous optimization.

In order to attest that the MACACO and CACS algorithms are not just exploiting the high number of ants to find good solutions as a simple sampling method, the problems on Table 4 was tested on ACO$_R$ changing the number of ants to 1000 and the number of iterations accordingly. As we can see on Table 4, some of the presented ACO$_R$ results are worse than those obtained with the recommended parameters.

Let us now take a closer look at the Rosenbrock problem: in Figure 3 the box-plot for the results obtained by the three algorithms in the 30 independent executions is presented. Both CACS and MACACO present a very similar behavior, specially with respect to their medians, but MACACO has presented some outliers that have significantly increased the mean values given in Table 3. These results are corroborated by the high standard deviation also given in Table 3. This might be due to the fact that the Rosenbrock function for a number of dimensions greater than 2 becomes multimodal, as pointed out in [13], and to the nature of the MACACO algorithm itself. Although the covariance matrix is restarted from time to time in order to try to escape from local optima, since the center value will still correspond to the local minimum, the number of samples generated at each iteration may not be enough to reach a better local optimum, or even the global optimum, due to the dimension of the search space.

Table 5 presents the average execution time for each of the three algorithms studied in this work. As can be seen from this table, the MACACO algorithm presents an average execution time about two times higher than the CACS algorithm, but it is about ten times faster than ACO$_R$. Although an analysis of execution times is always subject to a number of technical issues - including implementation poli-

Table 1: Benchmark problems adopted in this work

Name	Problem	Dimension (n)	Range
Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$
Rosenbrock	$f(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	30	$[-30, 30]^n$
Rastrigin	$f(x) = 10n + \sum_{i=1}^{n} [x_1^2 - 10\cos(2\pi x_i)]$	30	$[-5.12, 5.12]^n$
Griewank	$f(x) = 1 + 0.00025 \sum_{i=1}^{n} x_1 - \prod_{i=1}^{n} \cos \frac{x_i}{\sqrt{i}}$	30	$[-500, 500]^n$
Schwefel	$f(x) = -\sum_{i=1}^{n} x_i \sin \sqrt{\lvert x_i \rvert}$	30	$[-500, 500]^n$
Salomon	$f(x) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^{n} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	30	$[-100, 100]^n$

Table 2: Suggested parameters for ACO$_R$

Parameter	Value
Number of Ants	2
Speed of Convergence (ξ)	0.85
Locality of the Search Process (q)	10^{-4}
Archive Size	50

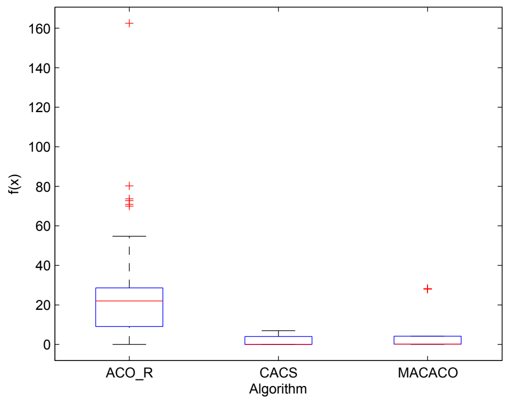


Figure 3: Boxplot for the Rosenbrock function.

cies - which demand great care, the results in Table 5 can be interpreted as a good indication of the comparative performance of the three algorithms studied in this work.

As a final experiment, the number of ants required by MACACO to achieve a pre-defined solution of the Rastrigin problem with higher dimensions (30, 50, 70 and 100) was empirically verified. The number of ants was obtained so that the algorithm could reach a mean solution of at least 0.5 for this problem, which is close to the global optimum, keeping the number of iterations fixed in 1000 (the same value adopted for the experiments performed before). Figure 4 shows the evolution of the required number of ants with the increase of the dimension of the problem, and suggests that these two variables are linearly related.

Additionally, the time required to perform the optimization process for each dimension of the Rastrigin problem was plotted (and it is shown in Fig. 5) in order to evaluate the impact on the algorithm when the number of ants and the problem dimension grow. Figure 5 shows that the initial growth is linear with the dimension of the problem but, after a number of 70 variables, the time requirement seems to grow steeper.

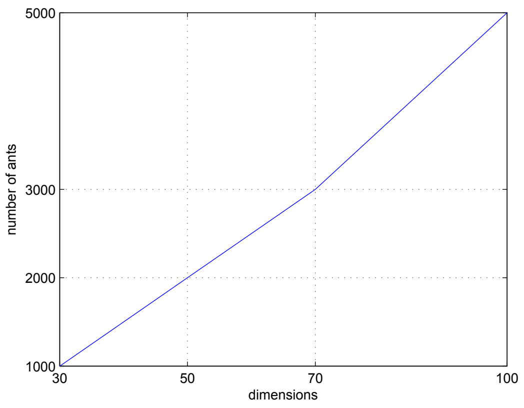


Figure 4: Evolution of the number of ants required to solve the Rastrigin problem with distinct dimensions.

5. DISCUSSION AND FUTURE WORKS

In this paper, a novel ant-based algorithm for continuous environments was proposed. The technique, called MACACO, which is based on the generation of random vectors from a multivariate Gaussian pdf, was compared to the ACO$_R$ and CACS approaches.

The reader was initially introduced to the concepts of Ant Systems and the subsequent adaptations to continuous environments. Two of these continuous-domain ACO algorithms were highlighted: CACS and ACO$_R$, which were explained and had their positive and negative aspects outlined.

The proposed algorithm was then thoroughly described and applied, together with CACS and ACO$_R$ algorithms, to six well-known benchmark optimization problems. The results have shown that the new proposal was able to pro-

Table 3: Mean and standard deviation values of the best individual obtained by ACO$_R$, CACS and MACACO, for the six benchmark problems studied in this paper (and given in Table 1). The best results are marked in bold.

	ACO$_R$	**CACS**	**MACACO**
Sphere	**0**	**0**	**0**
Rosenbrock	29.93 ± 35.14	$\mathbf{1.23 \pm 1.91}$	7.48 ± 11.68
Rastrigin	101.65 ± 21.01	57.17 ± 14.14	$\mathbf{0.00058 \pm 0.00009}$
Griewank	0.09 ± 0.180	0.02 ± 0.02	**0**
Schwefel	-8703.26 ± 721.53	-8934.57 ± 633.78	$\mathbf{-12569.49 \pm 0.01}$
Salomon	3.05 ± 1.43	0.33 ± 0.05	$\mathbf{0.15 \pm 0.05}$

Table 4: Results obtained by ACO$_R$ using a thousand ants on each test problem.

	ACO$_R$
Sphere	**0**
Rosenbrock	82818.94 ± 98832.77
Rastrigin	242.12 ± 38.18
Griewank	1.45 ± 0.42
Schwefel	-9484.07 ± 977.79
Salomon	0.33 ± 0.05

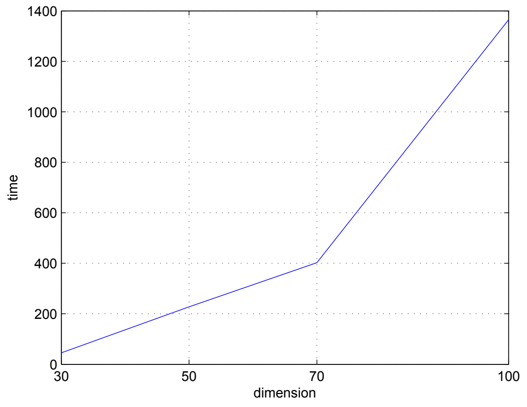


Figure 5: Evolution of the mean time (in seconds) required to solve the Rastrigin problem with distinct dimensions.

duce significantly better results in four of the six benchmark problems (for the Sphere problem, all the algorithms were capable of finding the global optimum). Only for the Rosenbrock problem the proposal presented an average performance worse than the one obtained by CACS. However, analyzing the box-plot for this problem we could see that both CACS and MACACO presented a very similar behavior, although MACACO has presented some outliers, probably due to its inability to hop from a local optimum to a better one when both are at a considerable distance from each other on the search space.

Although MACACO demands the calculation of the eigenvectors and eigenvalues of a correlation matrix at each iteration, its average computational time taken to optimize the benchmark problems was smaller than the one presented by ACO$_R$ and only about two times higher than the one presented by CACS. However, the significant improvement in the results presented by MACACO, when compared to CACS, indicates that it is a price that may be worth paying for.

This paper shows that ACO-like approaches to deal with optimization in continuous spaces are quite promising, since these algorithms can obtain good-quality solutions for challenging problems at a relatively low computational cost. In future works, we plan to investigate new ways to deal with local optima, better mechanisms to improve the exploration of the search space and compare our ant-based proposal with well-known continuous optimization algorithms based on different paradigms. We also plan to investigate the extension of such approaches to multi-objective optimization problems in continuous spaces.

6. ACKNOWLEDGMENTS

The authors would like to thank CAPES and CNPq for the financial support.

7. REFERENCES

[1] G. Bilchev and I. C. Parmee. The ant colony metaphor for searching continuous design spaces. In T. C. Fogarty, editor, *Evolutionary Computing, AISB Workshop*, volume 993 of *Lecture Notes in Computer Science*, pages 25–39. Springer, 1995.

[2] G. E. P. Box and M. A. Muller. A note on the generation of random normal deviates. *Annals. Math. Stat.*, 29:610–611, 1958.

[3] F. O. de França, F. J. Von Zuben, and L. N. de Castro. Max min ant system and capacitated p-medians: Extensions and improved solutions. *Informatica (Slovenia)*, 29(2):163–172, 2005.

[4] M. Dorigo. *Optimization, Learning and Natural Algorithms.* PhD thesis, Politecnico di Milano, Italy, 1992.

Table 5: Mean execution times (in seconds) obtained by ACO$_R$, CACS and MACACO, for the six benchmark problems studied in this paper.

	ACO$_R$	**CACS**	**MACACO**
Sphere	429.19	19.40	41.61
Rosenbrock	2113.82	110.56	222.75
Rastrigin	445.19	23.75	45.28
Griewank	385.26	17.52	47.26
Schwefel	411.11	18.93	47.30
Salomon	443.06	15.93	41.65

[5] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, 1999.

[6] M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In F. W. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 251–286. Kluwer Academic Press, 2003.

[7] J. Dréo and P. Siarry. A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions. In M. Dorigo, G. D. Caro, and M. Sampels, editors, *Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 216–221. Springer, 2002.

[8] M. Guntsch and M. Middendorf. A population based approach for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279 of *LNCS*, pages 72–81, Kinsale, Ireland, 3-4 2002. Springer-Verlag.

[9] I. T. Hernádvölgyi. Generating random vectors from the multivariate normal distribution. Technical Report TR-98-07, University of Ottawa, Aug. 20 1998.

[10] G. Marsaglia and W. W. Tsang. The ziggurat method for generating random variables. *Journal of Statistical Software*, 5(8):1–7, 2000.

[11] N. Monmarché, G. Venturini, and M. Slimane. On how *Pachycondyla apicalis* ants suggest a new search algorithm. *Future Generation Computer Systems*, 16(8):937–946, 2000.

[12] S. H. Pourtakdoust and H. Nobahari. An extension of ant colony system to continuous optimization problems. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *ANTS Workshop*, volume 3172 of *Lecture Notes in Computer Science*, pages 294–301. Springer, 2004.

[13] Y.-W. Shang and Y.-H. Qiu. A note on the extended Rosenbrock function. *Evolutionary Computation*, 14(1):119–126, March 2006.

[14] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, In Press, Corrected Proof, 2006.

[15] T. Stützle and M. Dorigo. ACO algorithms for the quadratic assignment problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 33–50. McGraw-Hill, London, 1999.