Available online at www.sciencedirect.com

Applied Mathematics and Computation 197 (2008) 604–613

www.elsevier.com/locate/amc

A new charged ant colony algorithm for continuous dynamic optimization

Walid Tfaili, Patrick Siarry [*]

Université Paris XII Val-de-Marne, LISSI (E.A. 3956), 61 av. du Général de Gaulle, 94010 Créteil, France

Abstract

Real world problems are often of dynamic nature. They form a class of difficult problems that metaheuristics try to solve at best. The goal is no longer to find an optimum for a defined objective function, but to track it in the search space. In this article we introduce a new ant colony algorithm aimed at continuous and dynamic problems. To deal with the changes in the dynamic problems, the diversification in the ant population is maintained by attributing to every ant a repulsive electrostatic charge, that allows to keep ants at some distance from each other. The algorithm is based on a continuous ant colony algorithm that uses a weighted continuous Gaussian distribution, instead of the discrete distribution, used to solve discrete problems. Experimental results and comparisons with two competing methods available in the literature show best performances of our new algorithm called CANDO on a set of multimodal dynamic continuous test functions.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Ant colony optimization; Continuous optimization; Dynamic optimization

1. Introduction

To find the shortest way between the colony and a source of food, ants adopt a particular collective organization technique (see Fig. 1). The first algorithm inspired from ant colonies, called ACO (refer to [1,2]), was proposed as a multi-agent approach to solve hard combinatorial optimization problems. It was applied to discrete problems like the traveling salesman, routing and communication problems.

Some ant colony techniques aimed at dynamic optimization have been described in the literature. In particular, Johann Dréo and Patrick Siarry introduced in [3,4] the DHCIAC (Dynamic Hybrid Continuous Interacting Ant Colony) algorithm, which is a multi-agent algorithm, based on the exploitation of two communication channels. This algorithm uses the ant colony method for global search, and the Nelder and Mead dynamic simplex method for local search. However, a lot of research is still needed to obtain a general purpose tool.

[*] Corresponding author.
E-mail addresses: tfaili@univ-paris12.fr (W. Tfaili), siarry@univ-paris12.fr (P. Siarry).

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.08.087

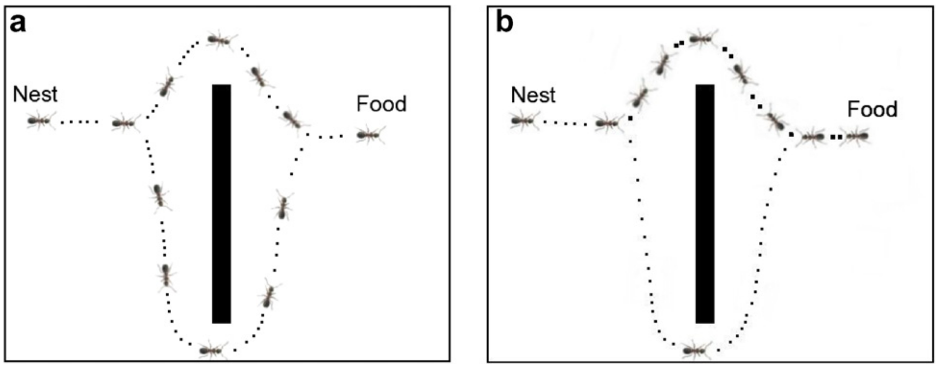

Fig. 1. An example illustrating the capability of ant colonies of finding the shortest path, in the case there are only two paths of different lengths between the nest and the food source. (a) When the foraging starts, the probability that ants take the short or the long path to the food source is 50%. (b) The ants that arrive by the short path return earlier. Therefore, the probability to take again the short path is higher.

We introduce in that paper a new dynamic optimization method, based on the original ACO. A repulsive charge is assigned to every ant in order to maintain diversification inside the population. To adapt ACO to the continuous case, we make use of continuous probability distributions.

We will recall in Section 2 the biological background that justifies the use of metaheuristics, and more precisely those inspired from nature, for solving dynamic problems. In Section 3 some techniques encountered in the literature to solve dynamic problems are exposed. The principle of ant colony optimization is resumed in Section 4. We describe our new method in Section 5. Experimental results are reported in Section 6. We conclude the paper in Section 7.

2. Biological background

In the late 1980s, a new research domain in distributed artificial intelligence has emerged, called swarm intelligence. It concerns the study of the utility of mimicking social insects for conceiving new algorithms.

In fact, the ability to produce complex structures and find solutions for non-trivial problems (sorting, optimal search, task repartition ...), using simple agents having neither a global view of their environment nor a centralized control or a global strategy, has intrigued researchers. Many concepts have then been defined, like auto-organization and emergence. Computer science has used the concepts of auto-organization and emergence, found in social insects societies, to define what we call swarm intelligence.

The application of these metaphors related to swarm intelligence to the design of methods and algorithms shows many advantages:

- flexibility in dynamic landscapes,
- better performance than with isolated agents,
- more reliable system (the loss of an agent does not alter the whole system),
- simple modeling of an agent.

Nevertheless certain problems appear:

- difficulty to anticipate a problem solution with an emerged intelligence,
- formulation problem, and convergence problem,
- necessity of using a high number of agents, which induces conflict risks,
- possible oscillating or blocking behaviors,
- no intentional local cooperation, which means that there is no voluntary cooperative behaviors (in case of emergence).

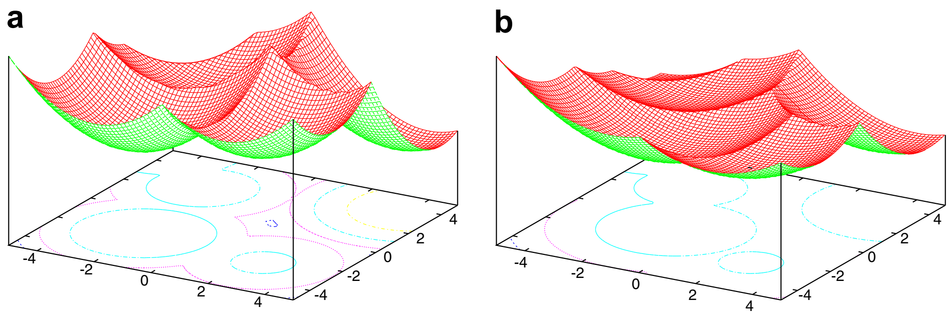

Fig. 2. Plotting of a dynamic test function representing local optima that change over time. (a) Time t_1 and (b) time $t_2 > t_1$.

One of the major advantages of algorithms inspired from nature, such as the ant colony algorithm, is flexibility in dynamic environments. Nevertheless few works deal with applications of ant colony algorithms to dynamic continuous problems (see Fig. 2). In the next section, we will briefly expose some techniques found in the literature.

3. Some techniques aimed at dynamic optimization

Evolutionary algorithms were largely applied to the dynamic landscapes, in the discrete case. Jürgen Branke (refer to [5,6]) classifies the dynamic methods found in the literature as follows:

1. The reactive methods (refer to [7,8]), which react to changes (i.e. by triggering the diversity). The general idea of these methods is to do an external action when a change occurs. The goal of this action is to increase the diversity. In general the reactive methods based on populations loose their diversity when the solution converges to the optimum, thus inducing a problem when the optimum changes. By increasing the diversity the search process may be regenerated.
2. The methods that maintain the diversity (refer to [9,10]). These methods maintain the diversity in the population, hoping that, when the objective function changes, the distribution of individuals within the search space permits to find quickly the new optimum (see [11,12]).
3. The methods that keep in memory “old” optima. These methods keep in memory the evolution of different optima, to use them later. These methods are specially effective when the evolution is periodic (refer to [13–15]).
4. The methods that use a group of sub-populations distributed on different optima (refer to [16–18]), thus increasing the probability to find new optima.

Michael Guntsch and Martin Middendorf solved in [19] two dynamic discrete combinatorial problems, the dynamic traveling salesman problem (TSP), and the dynamic quadratic assignment problem, with a modified ant colony algorithm. The main idea consists in transferring the whole set of solutions found in an iteration to the next iteration, then calculating the pheromone quantity needed for the next iteration.

The same authors proposed in [20] a modification of the way in which the pheromone is updated, to permit keeping a track of good solutions until a certain time limit, then to explicitly eliminate their influence from the pheromone matrix. The method was tested on a dynamic TSP.

In [21], Michael Guntsch and Martin Middendorf proposed three strategies; the first approach allows a local re-initialization at a same value of the pheromone matrix, when a change is detected. The second approach consists in calculating the value of the matrix according to the distance between the cities (this method was applied to the dynamic TSP, where a city can be removed or added). The third approach uses the value of the pheromone at each city.

The last three strategies were modified in [22], by introducing an elitist concept: the best ants are only allowed to change the pheromone at each iteration, and when a change is detected. The previous good solutions are not forgotten, but modified as best as possible, so that they can become new reasonable solutions.

Daniel Merkle and Martin Middendorf studied in [23] the dynamics of ACO, then proposed a deterministic model, based on the expected average behavior of ants. Their work highlights how the behavior of the ants is influenced by the characteristics of the pheromone matrix, which explains the complex dynamic behavior. Various tests were carried out on a permutation problem. But the authors did not deal with really dynamic problems.

In Section 5 we will present a new ant colony algorithm aimed at dynamic continuous optimization.

4. From nature to discrete optimization

Artificial ants coming from the nest pass through different paths until all paths are visited (Fig. 1). Pheromones have equal value initially. When coming back from the food source, ants deposit pheromone. The pheromone values are updated following the equation:

$$\tau_i = \tau_i + \frac{Q}{l_i},$$

with Q a constant and l_i the length of the path, which means that the new pheromone value is inversely proportional to the path length. A single trajectory from the nest to the food source is a complete construction of a solution. The process is iterated. The choice of a path is done following the probability:

$$P_{e_i} = \frac{\tau_i}{\sum_{i=1}^{n} \tau_i}.$$

Let us consider an example (Fig. 3); an ant will choose its path according to three probabilities

$$P_{e_1} = \frac{\tau_1}{\tau_1 + \tau_2 + \tau_3}; \quad P_{e_2} = \frac{\tau_2}{\tau_1 + \tau_2 + \tau_3}; \quad P_{e_3} = \frac{\tau_3}{\tau_1 + \tau_2 + \tau_3}.$$

If we suppose that $l_1 < l_2 < l_3$, then $\tau_1 > \tau_2 > \tau_3$, which implies that $P_{e_1} > P_{e_2} > P_{e_3}$. The chosen path will be e_1. To prevent the pheromone value from increasing infinitely and to decrease the importance of some solutions, the pheromone values are decreased (evaporated) with time following the equation:

$$\tau_i = \tau_i.(1-\rho) \quad \text{with} \quad \rho \in [0,1],$$

where ρ is a pheromone regulation parameter; by consequence this parameter will influence the convergence speed towards a single solution.

Ant colony algorithms were initially dedicated to discrete problems (the number of variables is finite), in this case we can define a set of solution components; the optimal solution for a given problem will be an organized set of those components. We consider the Traveling Salesman Problem "TSP": the salesman has to visit all the N cities and must not visit a city more than once, the goal is to find the shortest path. In practice, the TSP can be represented by a connected graph (a graph where nodes are interconnected). Nodes represent the N cities with $i = \{1,\ldots,N\}$ and N the total number of nodes. A path between two nodes (i.e. N_i and N_j) is denoted by e_{ij}. So a solution construction consists in choosing a starting node (city), adding to the current partial solution a new node according to a certain probability and repeating the process until the components

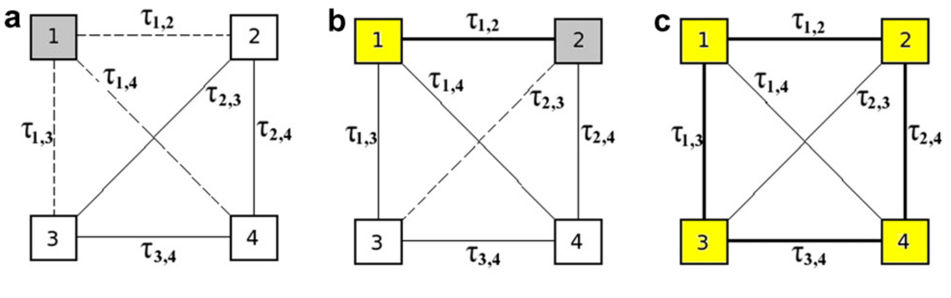

Fig. 3. An example showing the construction of the solution for a TSP problem with four cities. (a) At the beginning, (b) choosing a path and (c) a complete path.

number is equal to N. In discrete problems, solutions are not known in advance, which means that pheromones cannot be attributed to a complete solution, but to solution components.

5. CANDO: charged ants for continuous dynamic optimization

For problems with discrete variables, problem components are finite, therefore the pheromone values can be attributed directly to solution components. A deterministic solution to a discrete problem exists, even if its search may be expensive in calculation time. But for continuous problems, pheromone values cannot be attributed directly to solution components. The approach that we use is a diversification keeping technique (refer to Tim Blackwell in [24]), that consists in attributing to each ant an electrostatic charge, $a_i = \sum_{l=1;l\neq i}^{k} a_{il}$

$$a_{il} = \begin{cases} \frac{Q_i Q_l}{|x_i - x_l|^3}(x_i - x_l) & \text{if } R_c \leqslant |x_i - x_l| \leqslant R_p, \\ \frac{Q_i Q_l}{R_c^2 |x_i - x_l|}(x_i - x_l) & \text{if } |x_i - x_l| < R_c, \\ 0 & \text{if } R_p < |x_i - x_l|, \end{cases}$$

where Q_i and Q_l are the initial charges of ants i and l, respectively, $|x_i - x_l|$ is the Euclidean distance, R_p and R_c are the "perception" and the "core" radius respectively (see Fig. 4). The perception radius is attributed to every ant, while the core radius is the perception radius of the best found ant in the current iteration. These charge values can be positive or negative. Our artificial ants are dispersed within the search space (see Fig. 5), the values of charges change during the algorithm execution, in function of the quality of the best found solution.

The adaptation to the continuous domain that we use was introduced by Krzysztof Socha [25]. The algorithm iterates over a finite and constant ant population of k individuals, where every ant represents a solution or a variable X^i, with $i = \{1, \ldots, n\}$, for a n dimensional problem. The set of ants can be presented as follows (see Fig. 6):

k is the ants number and n is the problem dimension. At the beginning, the population individuals are equipped with random solutions. This corresponds to the pheromone initialization used for discrete problems. At each iteration, the new found solutions are added to the population and the worst solutions are removed in equal number; this corresponds to the pheromone update (deposit/evaporation) in the classical discrete case, the final goal is to bias the search towards the best solutions.

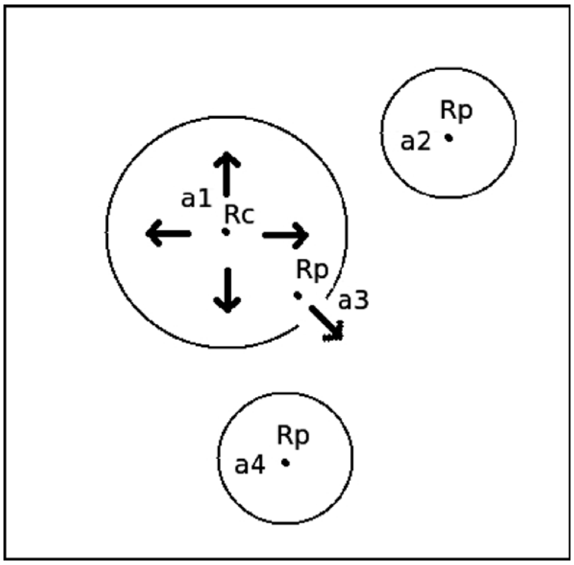

Fig. 4. An illustration of the repulsion of ants from the "core" of radius R_c of the best found solution. Every ant has its "perception" radius R_p.

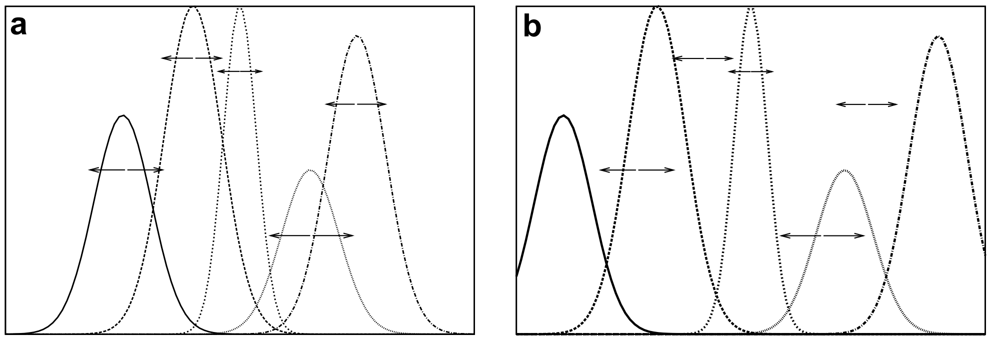

Fig. 5. A repulsion force is attributed to every ant, in practice to every Gaussian distribution. It results from the modification of (a) into (b).

The construction of a solution is done by components like in the original ACO algorithm. For a single dimension (for example dimension *j*), an ant *i* chooses only one value (the *j*th value of the vector $\langle X_1^i, X_2^i, \ldots, X_j^i, \ldots, X_n^i \rangle$). Every ant has a repulsive charge value, and a weighted Gaussian distribution. The choice of a single ant for a given dimension is done as follows: an ant chooses one of the Gaussian distributions according to its perception radius following the probability:

$$p_j^i = \frac{w_j^i}{\sum_{l=1}^{k} w_l^i}.$$

Once a single Gaussian distribution is chosen, the ant *i* generates a random value according to the distribution:

$$G(X_j^i) = \frac{1}{\sigma_j \sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{(x-\mu_j)^2}{2.\sigma_j^2}}$$

where μ and σ are the mean and the standard deviation vectors respectively. For a problem with *n* dimensions, the solution construction is done by dimension, which means that every solution component represents exactly one dimension. While the algorithm proceeds on a dimension, other dimensions are left apart. The final algorithm is presented in Algorithm 1:

Algorithm 1 Continuous charged ant colony algorithm

```
While stop criteria are not met do
  For 1 to $m$ ants do
    Calculate the $a_i$ value for the ant $i$
    Ant movement based on the charges values of neighbors
    $s^0 = \phi$
    Repeat for each of $n$ dimensions
    Choose a single distribution $G(X_j^i)$ only among neighbors according to $p_j^i$
    Choose randomly the $X_j^i$ value using the chosen $G(X_j^i)$
    $s^i = s^i \bigcup \{X_j^i\}$
    End
    Update ant charge based on the best found solution
    Update the weight (pheromone) based on the best found solution
  End
  Choose the best solution among the $m$ ants
  Choose the best solution among current and old solutions
End while
```

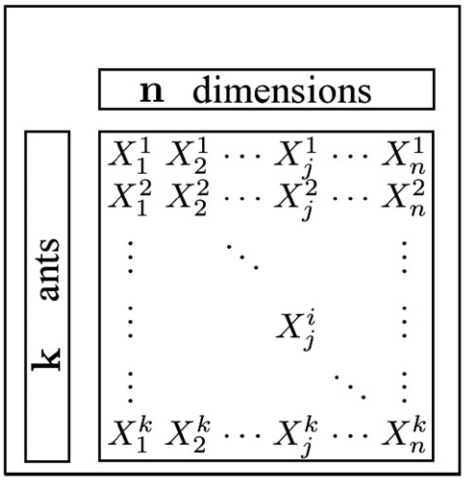
n dimensions
k ants
$X_1^1\ X_2^1 \cdots X_j^1 \cdots X_n^1$
$X_1^2\ X_2^2 \cdots X_j^2 \cdots X_n^2$
X_j^i
$X_1^k\ X_2^k \cdots X_j^k \cdots X_n^k$

Fig. 6. Every ant constructs a complete solution which includes *n* dimensions.

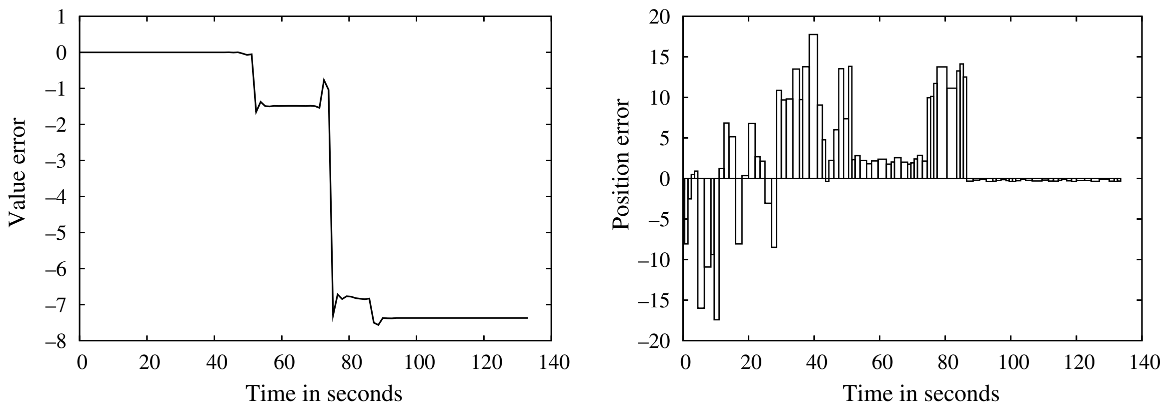

Fig. 7. Value and position errors results on the dynamic function AbPoP (*All but Position Periodic*), based on the static Morrison function.

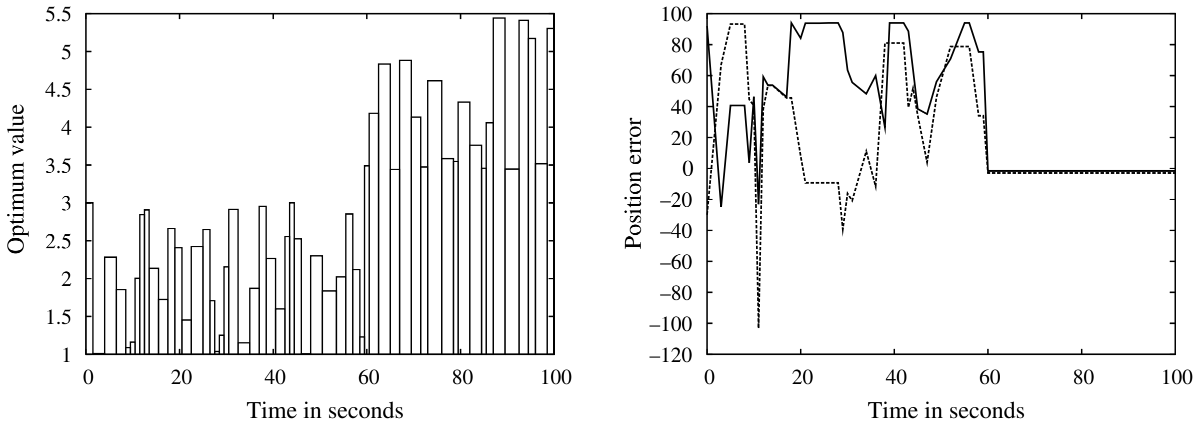

Fig. 8. Optimum value and position error results on the dynamic function AbVP (*All but Value Periodic*), based on the static Martin-Gady function.

6. Experimental results

Tests (Figs. 7–9) were performed using the Open Metaheuristic project platform (see [26,27] for more information). We used a set of dynamic continuous test functions that we introduced in [26]. Adding electrostatic charges to the ants has enhanced the found solution on the totality of the dynamic test functions (see Tables 1–3). We tested the competing algorithm eACO on the same dynamic functions to make a comparison with our approach. Our algorithm outperformed the classical algorithm eACO. This is due mainly to the continuous diversification in the search space over time. Ant electrostatic charges prevent ants from crowding around the found solutions thus preventing from a premature convergence. We also notice that CANDO shows better performance than DHCIAC on the functions having many optima, due to the fact that charged ants form groups around different optima, which permits a better tracking of the changing optima, while DHCIAC

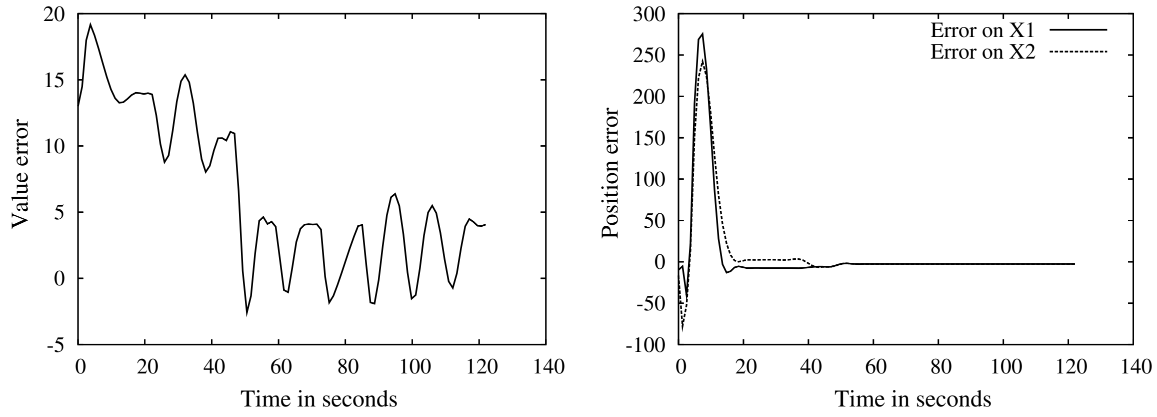

Fig. 9. Value and position errors results on the dynamic function OVP (*Optimum Value Periodic*), based on the static Morrison function.

Table 1
Comparison of the error and, in brackets, the standard deviation related to the value for CANDO, eACO and DHCIAC (best results are in bold)

	CANDO	eACO	DHCIAC
AbPoP	4.27(4.02)	4.34(4.15)	**4.09(3.86)**
AbVP	11.54(5.2)	12.03(5.45)	**9.73(3.49)**
APhL	**0.29(4.07)**	0.39(4.1)	0.38(4.18)
OVP	**1.78(8.09)**	1.84(9.11)	1.97(9.14)
OPoL	12.3(0.37)	13.61(0.42)	**10.62(0.14)**
AVP	**4.77E−005(0)**	5.43E−005(0)	6.54E−005(0)
APoL	2.87(2.24)	2.91(2.4)	**2.42(2.05)**

Table 2
Comparison of the error and, in brackets, the standard deviation related to the position X1 value for CANDO, eACO and DHCIAC (best results are in bold)

	CANDO	eACO	DHCIAC
AbPoP	0.043(4.29)	0.06(4.50)	**0.04(3.99)**
AbVP	−0.29(2.98)	−0.37(3.01)	**−0.21(2.54)**
APhL	**−3.04(1.56)**	−3.37(1.9)	−3.29(1.7)
OVP	**24.01(43.14)**	28.8(47.9)	31.49(52.76)
OPoL	−98.01(55.43)	−98.9(56.2)	**−92.95(50.1)**
AVP	**−17(19.6)**	−17.39(19.7)	−18(21.34)
APoL	−7.02(3.13)	−7.19(3.25)	**−5.8(2.96)**

Table 3
Comparison of the error and, in brackets, the standard deviation related to the position X2 value for CANDO, eACO and DHCIAC (best results are in bold)

	CANDO	eACO	DHCIAC
AbPoP	−0.91(2.74)	−0.97(2.81)	**−0.8(2.56)**
AbVP	−0.36(2.69)	−0.39(2.64)	**−0.27(2.62)**
APhL	**−3.02(1.51)**	−3.24(1.63)	−3.18(1.79)
OVP	**13.7(28.4)**	14.02(30.05)	15.55(31.36)
OPoL	−98.1(43.06)	−98.9(45)	**−97.91(40.16)**
AVP	**−16.5(18.3)**	−18.84(21.3)	−18.79(20.15)
APoL	−6.7(3.05)	−7.04(3.27)	**−5.9(2.88)**

converges more quickly on the dynamic functions with single optimum, due to the local search (through the Nelder–Mead simplex method). We notice that CANDO unlike DHCIAC does not use communication between ants and constructs the solution iteratively, which is the essence of the ant colony algorithms.

7. Conclusion

We presented in this article a new ant colony algorithm aimed at solving dynamic continuous optimization problems. Very few works were performed until now to adapt the ant colony algorithms to dynamic problems, and most of them concern only the discrete case. To deal with dynamic problems, we chose to keep ants simple and reactive, the final solution being obtained only through a collective work. The new algorithm consists in attributing to every ant a repulsive electrostatic charge, that is a function of the quality of the found solution. The adaptation of the ant colony algorithm to the handling of continuous problems is done by replacing the discrete probability distribution by a continuous one. This approach is interesting because it is very close to the original ACO. Experimental results on a set of dynamic continuous test functions proved the effectiveness of our new approach.

References

[1] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[2] M. Dorigo, L.M. Gambardella, Guest editorial special on ant colony optimization, IEEE Transactions on Evolutionary Computation 6 (4) (2002) 317–319.
[3] J. Dréo, P. Siarry, Dynamic optimization through continuous interacting ant colony, in: M. Dorigo, et al., (Eds.), ANTS 2004. LNCS 3172, 2004, pp. 422–423.
[4] W. Tfaili, J. Dréo, P. Siarry, Fitting of an ant colony approach to dynamic optimization through a new set of test functions, International Journal of Computational Intelligence Research 3 (3) (2007) 205–218.
[5] J. Branke, Evolutionary Approaches to Dynamic Environments – updated survey, in: GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2001.
[6] J. Branke, Evolutionary Approaches to Dynamic Optimization Problems – Introduction and Recent Trends, in: J. Branke, (Ed.), GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2003.
[7] J.J. Grefenstette, C.L. Ramsey, An approach to anytime learning, in: D. Sleeman, P. Edwards (Eds.), International Conference on Machine Learning, Morgan Kaufman, 1992, pp. 189–195.
[8] R. Tinos, A. de Carvalho, A genetic algorithm with gene dependent mutation probability for non-stationary optimization problems, in: Congress on Evolutionary Computation, vol. 2, 2004.
[9] H.G. Cobb, J.J. Grefenstette, Genetic algorithms for tracking changing environments, in: International Conference on Genetic Algorithms, Morgan Kaufman, 1993, pp. 523–530.
[10] T. Nanayakkara, K. Watanabe, K. Izumi, Evolving in dynamic environments through adaptive chaotic mutation, in: Third International Symposium on Artificial Life and Robotics, vol. 2, 1999, pp. 520–523.
[11] S.M. Garrett, J.H. Walker, Genetic algorithms: combining evolutionary and 'non'-evolutionary methods in tracking dynamic global optima, in: Genetic and Evolutionary Computation Conference, Morgan Kaufman, 2002, pp. 359–374.
[12] A. Simões, E. Costa, Using biological inspiration to deal with dynamic environments, in: Proceedings of the Seventh International Conference on Soft Computing (MENDEL01), Brno, Czech Republic, 2001.
[13] C.N. Bendtsen, T. Krink, Dynamic Memory Model for Non-Stationary Optimization, in: Congress on Evolutionary Computation, IEEE, 2002, pp. 145–150.

[14] K. Trojanowski, Z. Michalewicz, Evolutionary optimization in non-stationary environments, Journal of Computer Science and Technology 1 (2) (2000) 93–124.
[15] C.N. Bendtsen, Optimization of non-stationary problems with evolutionary algorithms and dynamic memory, Ph.D. thesis, University of Aarhus, Department of Computer Science Ny Munkegade 8000 Aarhus C, 2001.
[16] F. Oppacher, M. Wineberg, The shifting balance genetic algorithm: Improving the GA in a dynamic environment, Genetic and Evolutionary Computation Conference (GECCO), vol. 1, Morgan Kaufman, San Francisco, 1999, pp. 504–510.
[17] W. Cedeno, V.R. Vemuri, On the use of niching for dynamic landscapes, in: International Conference on Evolutionary Computation, IEEE, 1997.
[18] R.K. Ursem, Multinational GA optimization techniques in dynamic environments, in: D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, H.G. Beyer (Eds.), Genetic and Evolutionary Computation Conference, Morgan Kaufman, 2000, pp. 19–26.
[19] M. Guntsch, M. Middendorf, Applying population based ACO to dynamic optimization problems, in: Third International Workshop ANTS, LNCS, 2463, Springer-Verlag, 2002, pp. 111–122.
[20] M. Guntsch, M. Middendorf, A population based approach for ACO, in: 2nd European Workshop on Evolutionary Computation in Combinatorial Optimization, LNCS, 2279, Springer-Verlag, 2002, pp. 72–81.
[21] M. Guntsch, M. Middendorf, Pheromone modification strategies for ant algorithms applied to dynamic TSP, in: EvoWorkshop, 2001, pp. 213–222.
[22] M. Guntsch, M. Middendorf, H. Schmeck, An ant colony optimization approach to dynamic TSP, in: L., Spector, E.D. Goodman, A. Wu, W.B. Langdon, H.M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M.H. Garzon, E. Burke, (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), Morgan Kaufman, San Francisco, California, USA, 2001, pp. 860–867.
[23] D. Merkle, M. Middendorf, Studies on the dynamics of ant colony optimization algorithms, in: The Genetic and Evolutionary Computation Conference, (GECCO), New York, 2002, pp. 105–112.
[24] T.M. Blackwell, Particle swarms and population diversity I: Analysis, in: J. Branke, (Ed.), GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2003, pp. 9–13.
[25] K. Socha, ACO for continuous and mixed-variable optimization, in: M. Dorigo, et al., (Eds.) ANTS 2004, LNCS 3172, 2004, pp. 25–36.
[26] J. Dréo, J.P. Aumasson, W. Tfaili, P. Siarry, Adaptive learning search, a new tool to help comprehending metaheuristics, International Journal on Artificial Intelligence Tools 16 (3) (2007) 483–505.
[27] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard, Metaheuristics for hard optimization methods and case studies, Springer, 2005.