Algorithmic Operations Research

Hybrid Continuous Interacting Ant Colony aimed at enhanced global optimization

J. Dréo et P. Siarry

Volume 2, numéro 1, summer 2007

URI : https://id.erudit.org/iderudit/aor2_1art06

Aller au sommaire du numéro

Éditeur(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (numérique)

Découvrir la revue

Citer cet article

Dréo, J. & Siarry, P. (2007). Hybrid Continuous Interacting Ant Colony aimed at enhanced global optimization. *Algorithmic Operations Research*, *2*(1), 52–64.

Résumé de l'article

Ant colony algorithms are a class of metaheuristics which are inspired from the behaviour of real ants. The original idea consisted in simulating the stigmergic communication, therefore these algorithms are considered as a form of adaptive memory programming. A new formalization was proposed for the design of ant colony algorithms, introducing the biological notions of heterarchy and communication channels. We are interested in the way ant colonies handle the information. According to these issues, a heterarchical algorithm called “Continuous Interacting Ant Colony” (CIAC) was previously designed for the optimization of multiminima continuous functions. We propose in that paper an improvement of CIAC, by the way of a hybridization with the local search Nelder-Mead algorithm. The new algorithm called “Hybrid Continuous Interacting Ant Colony” (HCIAC) compares favorably with some competing algorithms on a large set of standard test functions.

Algorithmic Operations Research Vol.2 (2007) 52–64

Hybrid Continuous Interacting Ant Colony aimed at enhanced global optimization

J. Dréo [a] P. Siarry [a]

[a]Université Paris XII Val-de-Marne, Laboratoire Images, Signaux et Systèmes Intelligents (LISSI), 61 avenue du Général de Gaulle, 94010 Créteil, France.

Abstract

Ant colony algorithms are a class of metaheuristics which are inspired from the behaviour of real ants. The original idea consisted in simulating the stigmergic communication, therefore these algorithms are considered as a form of adaptive memory programming. A new formalization was proposed for the design of ant colony algorithms, introducing the biological notions of heterarchy and communication channels. We are interested in the way ant colonies handle the information. According to these issues, a heterarchical algorithm called "Continuous Interacting Ant Colony" (CIAC) was previously designed for the optimization of multiminima continuous functions. We propose in that paper an improvement of CIAC, by the way of a hybridization with the local search Nelder-Mead algorithm. The new algorithm called "Hybrid Continuous Interacting Ant Colony" (HCIAC) compares favorably with some competing algorithms on a large set of standard test functions.

Key words: Ant colony algorithm, metaheuristic, global optimization, continuous optimization, hybrid metaheuristic.

1. Introduction

Metaheuristics are very often used for discrete problems, but there is a class of problems often met in engineering, where the decision variables are continuous and for which metaheuristics can be of a great help (due to nonderivable function, multiple local minima, great number of variables, nonconvexity, etc). Having recently given rise to some new metaheuristics, the ant colony metaphor proved to be a successful approach to solve "difficult" optimization problems. The first algorithm inspired from ant colonies (the "Ant System" [7]), was successfully applied to various discrete optimization problems. So far, several attempts to adapt the ant colonies metaheuristic to the continuous field appeared. In addition to the usual problems related to the adaptation to the continuous case, the ant colonies metaheuristics pose some specific problems. Most of the metaheuristics are inspired by the characteristics of self-organization and external memory of the ant colonies, leaving aside the iterative construction of the solution. We listed four ant colonies algorithms for continuous optimization: CACO ("Continuous Ant Colony Algorithm" [1,17][11]), a hybrid algorithm not named, API (named from an ant species) and CIAC ("Continuous Interacting Ant Colony").

The first ant colony algorithm designed for the optimization of continuous functions is the CACO algorithm. CACO retains a particular feature of the behaviour of the real ants: the deposit of track of pheromone. Indeed, the colony of ants is often described like a distributed system which is able to solve complex problems with the use of "stigmergic" processes, a form of indirect communication, by the means of modifications of the environment. But the deposit of trail is also met in some processes of "recruitment", defined by the biologists as a mode of communication leading some individuals to meet in a place where a work is needed.

Another method of optimization taking this definition as a starting point was developed in the continuous case: the API metaheuristic, inspired by the behaviour of recruitment of a primitive ant [12]. However, the API algorithm uses little the memory structures which generally characterize the systems of colonies of ants [15], namely the presence of an external memory, shared by the agents.

A method using at the same time an approach of colonies of ants and an evolutionary algorithm was proposed by Ling et al.. [10], but few results are available

Email: J. Dréo [dreo@univ-paris12.fr], P. Siarry [siarry@univ-paris12.fr].

until now. The main idea of this method is to consider the differences between two individuals on each dimension as many parts of a way where the pheromones are deposited. The evolution of the individuals dealt with operators of mutation and crossing-over. This method thus tries to reproduce the mechanism of construction of the solution component by component.

Our point of view is that the ant colonies metaphor can be defined like a model considering not only stigmergic phenomena, but also, more largely, processes of recruitment. We developed, according to this idea, a method aimed at continuous optimization, inspired by ant colonies, which exploits the concept of interindividual communication [8]. This method, called CIAC, was tested on a large set of analytical functions, some of them of a great number of variables. It was also compared with some of the best methods published so far: other algorithms based on the colonies of ants, genetic algorithms, tabu search and simulated annealing.

Like the original "Ant System", it has been shown that CIAC is less competitive in local search [8], in spite of a relatively effective global search. To lessen the impact of this drawback, we chose to implement an hybridization with the Nelder-Mead "simplex" algorithm [13], we called this hybrid algorithm "HCIAC". The present paper describes the HCIAC metaheuristic.

This paper comprises five more sections. The basic algorithms that are used to build HCIAC are described in section 2.. The subsequent section 3. is devoted to the description of the HCIAC algorithm. Then we discuss in section 4. the tuning of the algorithm parameters using a particular technique that we called meta-setting of parameters. In section 5. we present and discuss experimental results. Conclusion makes up the last section.

2. Basic algorithms

2.1. *Heterarchical algorithm*

An algorithm, being focused on the principles of communication of the ant colonies, was proposed by the authors of this work [8]. It consists in adding to the stigmergic processes, i.e. a way to exchange informations by modifications of the environment (see [3]), the direct exchanges of information, while being inspired for that by the "heterarchical" approach, by opposition to a "hierarchical" approach (see [16,8]). Thus, a formalization of the exchanges of information is proposed around the concept of communication channels. A communication channel may be, to take an example in the "ant metaphor", the deposit of pheromonal trail. These channels transmit an information, here the localization of a food source, and have some properties, like stigmergy and memory. One can define various communication channels representing the transporting unit of informations. From the point of view of metaheuristics, there are three main characteristics:

Scope: the way the information goes through the population. A sub-group of the population, from one to n agents, can exchange informations with another group of agents.

Memory: the way the information persists in the system. The information can be stored during some period of time or be transitory.

Integrity: the way the information is evolving in the system. The information can be modified, by one or more agents, by an external process, or not.

These properties can be combined in the same channel, so that a large variety of different channels can be built.

The information transmitted during the communication can take many forms, from a simple value to a complex "object", therefore it is difficult to describe some particular classes. The more intuitive forms are for example the vector coordinates of a point and the value of the objective function at this location.

As an example, let us take a look at the properties of the "trail laying" channel. Basically, the scope is potentially the whole population, as each ant can perceive the trail pheromone. There is also a form of memory, as this is a stigmergic process, the trail persists in the environment during a certain period of time. Finally, the integrity of the channel permits that the informations are damaged by the time, as the pheromones evaporate.

2.2. *CIAC*

The CIAC algorithm (acronym for "Continuous Interacting Ant Colony") uses two channels of communication:

- The stigmergic channel calls upon spots of pheromone, deposited on the search space. These spots will be more or less attractive for the artificial ants, according to their concentrations and their distances. Here, a spot is defined as a point on the search space and a value associated with it. We use the verb "to deposit" when a new spot is created or modified by an ant. The characteristics of the stigmergic channel are the following ones. Firstly, the range is maximal, all the ants can potentially take into account information.

Secondly, there is use of memory since the spots persist on the search space. Finally information evolves with time since the spots evaporate. The information carried by a spot implicitly contains the position of a point and explicitly the value of the improvement found by the ant having deposited the spot.

- The direct channel is implemented in the form of messages exchanged between two individuals. An artificial ant has a stack of received messages and can send some messages to another ant. The range of this channel is of one since only one ant receives the message, the memory is implemented in the messages stack which the ant memorizes. Finally, information, here a couple position/value of a point, does not change with time.

The algorithm showed interesting characteristics, in particular a certain capacity to oscillate between a process of intensification and a process of diversification when the two channels of communication (stigmergic and direct) are used in synergy [8].

But CIAC is slower than other metaheuristics and is most times comparable in terms of precision (i.e. ability to find the optimum with only a small error). In fact, CIAC can be fruitfully used for a global search for the most "promising" regions within the search space, but it has to be hybridized with some local search classical algorithm for a faster and more accurate localization of the best solution.

Some other problems have been pointed out: CIAC allows ants to deposit as many spots as they want, using the addition of several spots to state the reinforcement of regions ; but this method sometimes leads to an excessive number of spots to be handled, which can slow down the algorithm. Another problem is the fact that the two channels are used in serial, ants are using first the stigmergic channel and then the direct channel ; this working sometimes leads to the loss of gathered information as the direct channel can overflow the stigmergic one.

2.3. *Nelder-Mead algorithm*

Many of the population-based algorithms are not very efficient for a fast and accurate finding of local optima, but are quite good to locate promising areas. The ant colonies algorithms perform better when they use a local search, and for discrete problems, this technique is often used to make the ant algorithms competitive [2].

The Nelder-Mead algorithm [13] is a simple algorithm that presents the advantage of being a derivative-free method that quickly finds local optima. Instead of using the derivatives of the objective function, the Nelder-Mead algorithm uses a little population of points to handle a non degenerate "simplex".

3. The HCIAC algorithm

3.1. *Improvements to the CIAC algorithm*

We found two drawbacks occurring in the design of the CIAC algorithm, for each one we incorporate a new feature that intends to solve the problem in order to improve the algorithm efficiency. The first idea is to use less spots to structure the search space as a way to speed up the algorithm and the second improvement concerns the use of thresholds for the regulation of the different strategies used.

3.1.1. *Spot management*

In the CIAC algorithm, the spots are used to describe the search space as a function of the potential interest of regions. Each ant can deposit a spot and the function of interest is more depending on the number of spots in a region than on the concentration of the spots. Thus for some test functions with a turmented landscape this can lead to an excessive number of spots. The problem in this case is the excessive use of computer memory needed to handle the vectors used to store the information, the time spent to use this information can be prohibitive, specially when the number of variables of the objective function is high.

The solution we chose consists of putting the information of interest on the concentration of pheromone and no more on the spatial repartition of the spots. Each ant that decides to deposit a pheromonal spot can then reinforce an existing spot instead of depositing another one. The use of computer memory is then limited as the number of vectors is maintained low.

In practice, the decision between the deposit of a new spot or the reinforcement of an existing one is made according to a resolution parameter. Here again, in order to maintain the parallel design of the algorithm, each ant has its own resolution parameter which we can call the "visible zone". The resolution parameter is dynamically set according to the ant's environnement during the ant evolution on the search space. Basically, when the ant ends a local search and finally finds a spot in its visible zone, it reinforces the spot and the visible zone is reduced to the distance from the farthest spots in the visible zone (see Figure 1). We talk about a "visible spot" when a pheromonal spot is comprised in this visible zone.

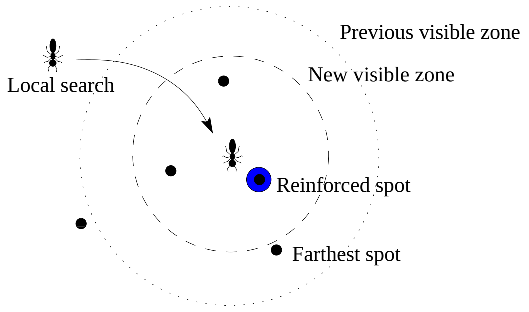


Fig. 1. Spot management and resolution parameter: after a local search, if the ant finds some spots, it reinforces the closest spot and its visible zone is reduced to the distance from the farthest spot.

With this mechanism, the visible zone of the ant is automatically set during the search process, according to the granularity of the search space it finds. From the point of view of the whole system, this is a resolution parameter as it describes the local granularity of the objective function and as it is decreasing during the search process. One can compare this behaviour with a cooling parameter as in simulated annealing, but with the advantage that the temperature is decreased automatically according to the informations gathered from the objective function's landscape (see Figure 2). One must notice that the exponential behaviour observed on figure 2b is not explicitely coded, but results from the whole system behaviour.

3.1.2. *Threshold choices*

In biological systems such as ant colonies, there is no choice made with a simple true/false test. In fact, the behaviour is based on what biologists call stimulus-response functions. These functions describe the fact that the decisions — at the individual level — are taken with a certain probability according to the internal state and to a stimulus.

The stimulus-response function is a simple equation, giving the probability of the decision $p(s)$ according to the stimulus s:

$$p(s) = \frac{1}{1 + e^{(-\rho \cdot s + \rho \cdot \tau)}}$$

with ρ the power, the "smoothness" of the choice and τ the threshold.

For example, an individual perceives a pheromonal trail (this is the stimulus), let say that this trail has an

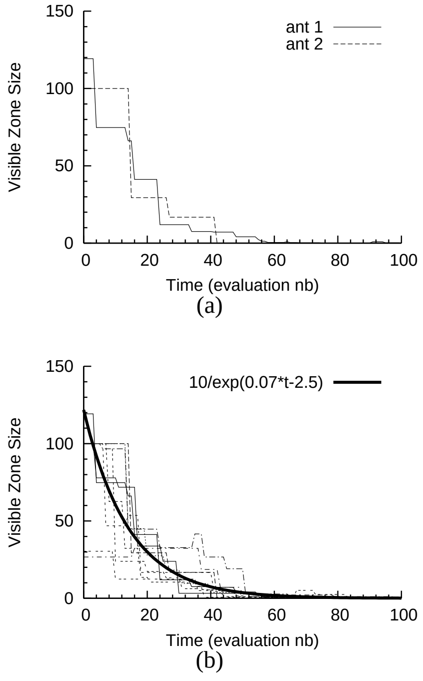


Fig. 2. Decreasing of the resolution parameter: evolution of the visible zone size of some ants acording to time. (a) for 2 ants, (b) with 10 ants the global exponential dynamics appears.

interest of $s = 0.5$ (in an arbitrary unit, typically in $[0, 1]$). This individual is quite sensible to pheromones of $\tau = 0.3$ (this is the internal state, the threshold). With $\rho = 10$ the probability of laying a trail, the decision in our example, is thus of $p(s) = 0.88$.

The interest of such functions for optimization algorithms is to increase the flexibility of the metaheuristic by adding more information in the system. As in biological systems, if we aim to build a generic algorithm, it is impossible to precisely tune parameters. We thus generally need to make a compromise in order to maintain a good average performance. But with threshold choices, a parallel metaheuristic such as an ant colony algorithm can avoid this delicate tuning. Indeed, an ant can make an interesting move, because of the probabilistic choice, where all the other ants failed. One can notice that with this technique we replace one parameter by two parameters (τ and ρ), that could be considered as a bad solution for a parameter setting problem. But we argue that the loss of difficulty in the parameter setting for the user of the metaheuristic and the gain of flexibility, in combination with a meta-setting of parameters (see Section 4.1.), justify this choice.

3.2. *Hybridization*

3.2.1. *Linking the two algorithms*

Two versions of the hybrid algorithm were implemented, because of the various possible relations between the two methods. Firstly the temporal relation, where local search is launched at regular intervals starting from the best solution. Secondly, space relation, where local search is launched independently by each ant. We will qualify the first algorithm of "simple", since it simply consists in improving the best solution at a given time. The second hybridization makes it possible to maintain the strongly decentralized feature which characterizes the ant colonies algorithms. Indeed, each agent decides to do a local search only on the basis of information it has, and not, as in the first case, on the basis of the whole system's information. One can qualify this hybridization of "decentralized".

The simple hybridization is easy to implement and leads to a simple behaviour of the algorithm. However, it loses the parallel structure of the algorithm as there must be a global control to determine which ant is the best among all the population. In addition, the simple hybridization does not permit to explore the search space with a better efficiency as it does not take into account the information found by the local search. In fact, the decentralized hybridization is more efficient as it takes into account the local search, and does not consider it as just a way to improve the final result, but as a way to simplify the landscape of the objective function.

3.2.2. *Motivation*

Given the idea that the decentralized hybridization is more interesting to preserve the ant colonies algorithms properties, we must consider a way to decide when the local search is launched. In order to maintain the parallel architecture of the algorithm, the decision must be taken by the individual, according to local informations. One other desirable behaviour consists in avoiding a simple periodicity, as it does not conform with the informations gathered by the ants. The local search must be used by a single ant, according to local information and only when necessary.

The rules we have chosen are based on the notion of motivation, which describes the probability of starting a local search. Each ant has an internal counter that is used as a stimulus in a threshold choice to decide if the ant makes a local search or sends a message. The motivation increases when the ant does not see any spot, has no message queueing and does not make a local search. At this moment, the motivation is increased by a small amount.

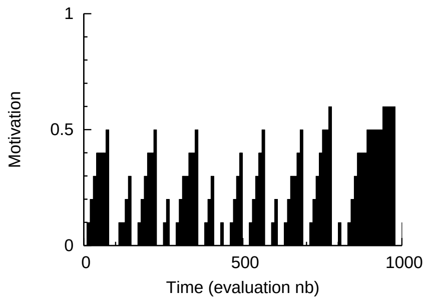


Fig. 3. Dynamics of the motivation parameter for an ant (the threshold here is of 0.5).

The dynamics of the motivation is shown on Figure 3. One can notice that the local search is launched (when the motivation returns to 0 on the figure) mainly when the motivation is of 0.5, as this is the threshold of the stimulus-response function. But the local search can also be launched with a motivation of 0.1, according to the probabilistic aspect of the decision process.

3.3. *Final algorithm*

The HCIAC algorithm is described on Figure 4.

The first step of the algorithm is to randomly put the η ants over the search space according to an uniform distribution and to initialize all parameters.

Then the pheromonal spots evaporate, the value $\tau_{j,t+1}$ of each spot j at time $t+1$ is set according to:

$$\tau_{j,t+1} = \rho \cdot \tau_{j,t}$$

with ρ the persistence parameter.

At this step, a decision is taken, according to a threshold choice function: the ant chooses to handle one of the two communication channels. Here the two parameters of the stimulus-response function are called χ_τ for the threshold and χ_ρ for the power, these parameters are set for the whole population, as each ant i has a particular stimulus parameter χ_i, initialized at the first step according to a normal distribution $\mathcal{N}_{\chi_m,\chi_d}$.

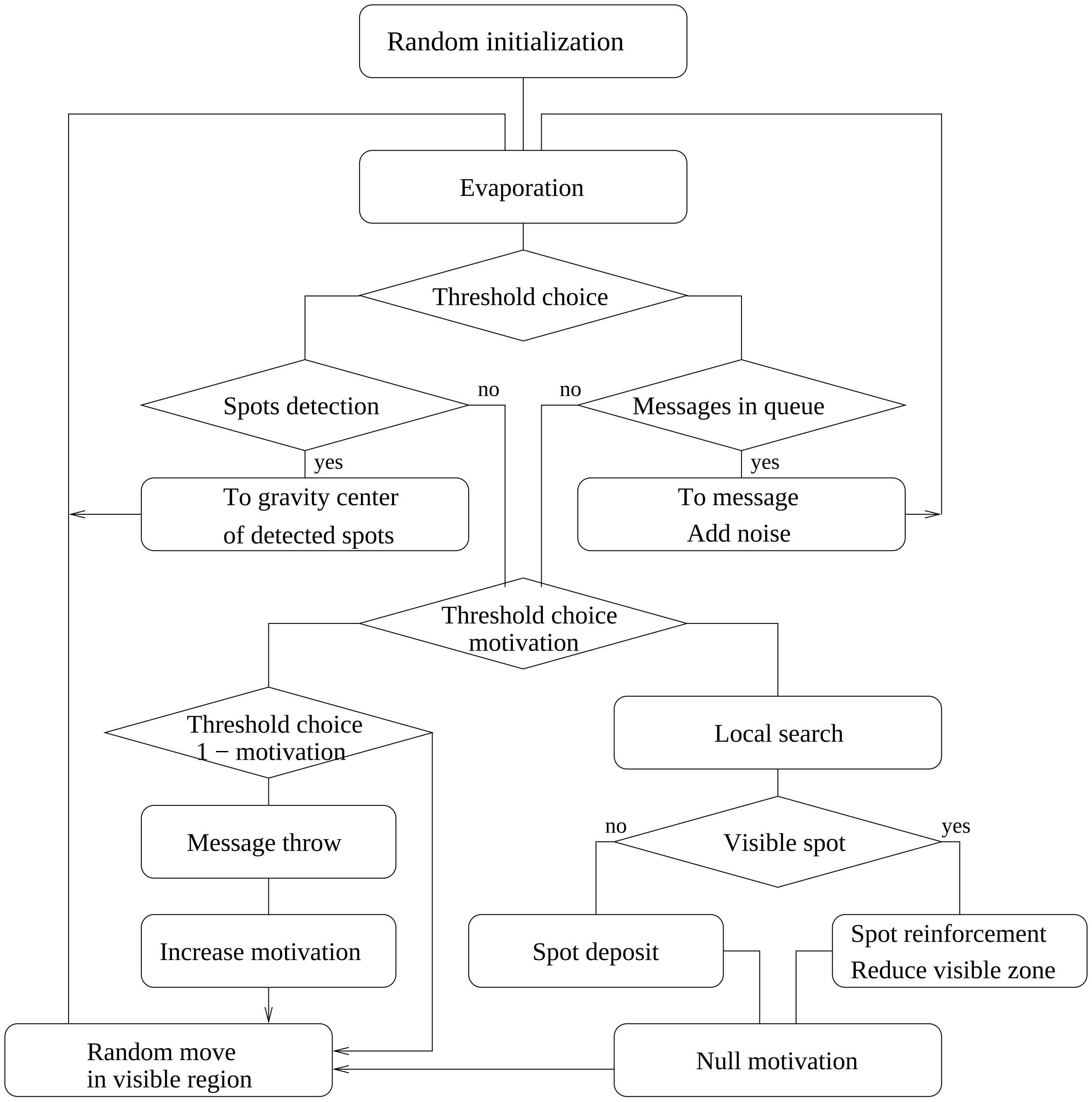


Fig. 4. The HCIAC algorithm

If the ant chooses the stigmergic channel, it looks for pheromonal spots in the visible zone π_i (see Section 3.1.1.) which is initialized according to $\mathcal{N}_{\pi_m,\pi_d}$ at the first step. If there is some spots, it moves towards the weighted gravity center of the visible spots, else it goes on the step of motivation decision.

On the contrary, if the ant chooses the direct channel, it looks for messages in its message queue (see section 2.2.). If there is some messages, it moves towards the location indicated by the message and then adds some noise to its new location (i.e. it moves randomly in the visible zone), else it goes on the step of motivation decision.

If there is no spot and no message, the ant takes a decision based on its motivation ω_i. The choice is made according to a stimulus-response function, the threshold ω_ρ and the power ω_τ are initialized at the first step for the whole population. The stimulus ω_i is initialized at 0 at the first step.

The first choice leads to the direct channel, with message handling. At this step, another decision is taken according to a stimulus-response function with the same parameter as in the previous step, except that the stimulus is $(1-\omega_i)$. If the choice is made to handle messages, a message is sent to a random ant and the motivation is increased of a small amount ω_δ. The second choice is to avoid messages, then the ant goes to the step of random move.

If the motivation choice is to handle local search, a Nelder-Mead search is launched with the location of the ant as base point and the radius of the visible zone as initial step length, the local search is limited to ν evaluations of the objective function. After this local search, the ant looks for visible spots. If there is a spot, it reinforces it according to:

$$\tau_{j,t+1} = \tau_{j,t} + \frac{\tau_{j,t}}{2}$$

and the radius of the visible zone is reduced to the distance to the farthest visible spot. Else if there is no spot, the ant deposits a new one, with a concentration set to the value of the objective function at this ant's current location. After the handling of the stigmergic channel, the motivation is set to zero.

The possible final step is to make a random move in the visible zone.

The algorithm stops if it cannot find a better optimum than the best found during θ evaluations of the objective function.

4. Parameter setting

4.1. *Meta-setting of parameter*

The parameter setting is always a delicate step in the conception of a metaheuristic. In fact this is an optimization problem, even in case of just one parameter. Here the objective function is the efficiency of the algorithm and the search space is defined by the parameters of the metaheuristic.

The objective function is difficult to be described, as it is difficult to describe the efficiency of an algorithm. An interesting point of view on metaheuristics has been proposed by Taillard [15] with the "Adaptive Memory Programming" theory. In this theory, there are two main aspects in the metaheuristics: intensification and diversification. In fact if we take a close look to this idea, this is exactly what is asked to the algorithms: to be fast (achieve the optimization with a minimum of computations of the objective function) and accurate (to find the global optimum with a minimal error). As these objectives are contradictory, this is a true biobjective problem.

This idea leads us to consider the parameter setting as a biobjective optimization problem. We argue that there is a loss of complexity, so that optimizing an optimization algorithm is not an endless reasoning. Using this technique, we do not need to carefully set the parameters of the biobjective algorithm, as the optimization problem is not necessary very difficult. Moreover, the setting of the parameters needs to be performed only one time. Once the parameter setting is performed, either for a general or a specific problem, it does not need to be reapeated. This approach is thus interesting for problems where the optimization needs to be performed several times.

To set the parameters of our algorithm, we use the MOGA algorithm, a multiobjective genetic algorithm described in [9] with standard parameters [6].

The Pareto front, showing the optima in the objective function space, shows the compromise performed by the algorithm between the two objectives and is a good indicator of the algorithm behaviour. The study of the locations of the optima in the parameter space is also of great interest as it is an indicator of the impact of the parameters on the algorithm behaviour.

Moreover, getting the Pareto front permits to reduce the multi-parameter setting problem to the setting of only one parameter, that we can call an *intensification/diversification* index. Indeed, with the repetition of the biobjective optimization, the convex Pareto front can generally be well sampled, leading to a scatter plot that can be approximated by an exponential function. In order to get a single index gathering all the parameters sets, we just need to choose a certain number of points homogeneously spread over the exponential function. Given these points, we can associate them with an index, for example on a 10 points scale.

Drawing the Pareto front is also an interesting way to understand the way the algorithm works. For example, drawing the Pareto front in the search space of two parameters helps to understand in which way these two parameters influence the algorithm, it is then easier for instance to see if parameters are linked. One can also draw the Pareto front for one parameter in function of one objective function, showing here how the parameter affects the efficiency of the algorithm. As the different representations of the Pareto fronts are not necessarily essential to do the parameter setting, we will not insist on them in this paper.

4.2. *Parameter setting of HCIAC*

We will use the following notation for the objective functions used by the multiobjective optimization program: F_i the quickness, which needs more intensification, and F_d the accuracy, which needs more diversification.

The Pareto front achieved by the multiobjective optimization of HCIAC parameters is considered as representative of a mean parameter setting for classical optimization problems. If an user needs to handle a more specific problem then it is necessary to run again the meta-setting of parameters.

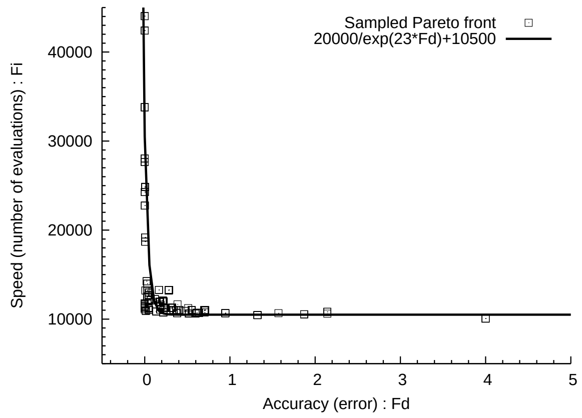


Fig. 5. Superposition of 20 samples of the Pareto fronts of the HCIAC parameters in the objective function space.

As it is shown on Figure 5, the set of the Pareto fronts found by the bi-objective algorithm is well approximated (square error: $R^2 = 0.95$) by the exponential function:

$$F_i = \frac{2 \cdot 10^4}{e^{(23 \cdot F_d)}} + 105 \cdot 10^2$$

In order to handle a simple parameter setting, we have sampled the Pareto front with simple values on the exponential function, drawing an easy to use intensification/diversification index. For further reference to the parameter setting, we will use the notation P_i to refer to the point i of the sampling of the approximated Pareto front, each P_i point is associated to values of the corresponding parameters of HCIAC, as shown in Table 1. We have rounded the parameters associated to the Pareto front when the influence on the algorithm efficiency was weak.

Table 1

Sampling of the Pareto front (intensification/diversification index) for the parameters of the HCIAC algorithm.

P_i	1	2	3	4	5	6	7	8	9	10
F_i	1.9e-5	1e-4	5e-3	5e-3	0.16	0.2	0.7	1.3	1.8	4
F_d	44000	33800	20000	19000	14000	13200	11000	10500	10500	10000
ν	950	930	910	890	500	400	400	340	264	17
η	950	360	100	100	150	80	150	45	12	12

As shown by Figure 5, the Pareto front shows a sharp shape. Therefore there exists a satisfying compromise between the two objectives, so that most of users should choose an intensification/diversification index between 4 and 7.

We have not discussed all the parameters as only the maximum number of the local search evaluations (ν) and the number of ants (η) are significant, as they have a great variance along the Pareto front. All the other parameters have only a little influence on the algorithm performances (little variance), comparatively to ν and η. We have found that approximately the same values are obtained all along the Pareto fronts and are thus well suited for default values: $\rho = 0.5$, $\chi_m = 0.5$, $\chi_d = 0.2$, $\pi_m = 0.8$, $\pi_d = 0.5$, $\omega_\delta = 0.1$, $\chi_\tau = \omega_\tau = 0.5$, $\chi_\rho = \omega_\rho = 10$.

For the rest of this paper, we will use only one value of the intensification/diversification index for the comparison with competing algorithms. The reason of this choice is that the results in the literature are shown for only one parameter set. To make a fair comparison, we have chosen the index P_4 that achieves the compromise we often see with our classical continuous problems: a greater importance is given to the accuracy over the rapidity.

Finally, one of the most sensible parameters is θ, the sensitivity of the stopping criterion. The stopping criterion is not considered here as a parameter of the HCIAC algorithm, as it is very dependent on the optimization problem. Therefore we have chosen to initialize it with a single value, and not to use the meta-setting of parameters to tune it. We have thus achieved a balance for the whole set of the test functions, and chosen the value of 10000.

5. Experimental results and discussion

5.1. *HCIAC behaviour*

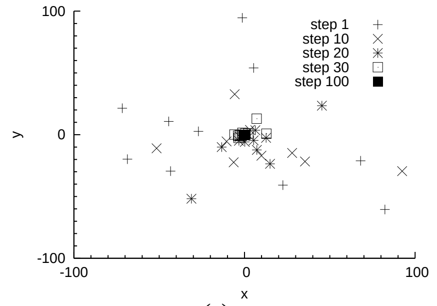
To illustrate the behaviour of HCIAC, we have first applied it on the B_2 function.

We used only 10 ants to make the figures more readable, all the other parameters values are those described above.

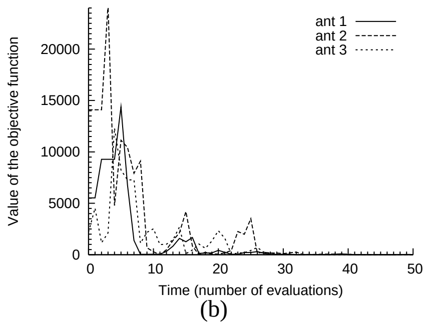
Figure 6 shows that the global dynamics of the algorithm consists of a convergence to the global optimum. Indeed, the local search permits to the ants to quickly walk through local optima, the direct channel permits to force the ants to get out of local optima in addition to the spot channel, which elaborates a description of the search space. As described in [8], both channels work together to adapt the algorithm behaviour to the objective function. Indeed, with the concomitant use of both channels appear oscillations of the standard deviation of the values of the objective functions found by all the ants. This phenomenon is still present in HCIAC as shown by figure 7.

5.2. *Tests results*

We have tested the HCIAC algorithm over a set of analytical test functions found in the literature. In order to compare HCIAC with other continuous optimization algorithms,we have chosen a set of 13 test functions partly in the literature devoted to continuous ant colonies algorithms [1,11,12], and partly in papers dealing with other continuous optimization algorithms [14][4][5]. It can be pointed out that HCIAC can handle as well functions to be minimized as functions to be maximized.



(a)



(b)

Fig. 6. The HCIAC algorithm optimizing the $B2$ function. (a) x and y positions of the 10 ants at different steps of the optimization, (b) values of the objective function found by 3 ants during the 50 first evaluations.

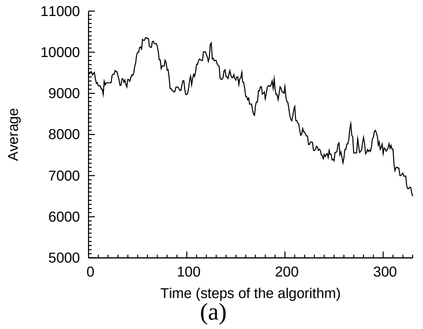
To avoid any problem due to the choice of an initial population, we performed each test 100 times with a different random seed at each test. The values of the objective function evaluation number (“evals”) and of the average error (“err”) are obtained through averaging the results over all the tests. The standard deviations are given in parenthesis. A test is considered to be successful, thus contributing to “% ok”, if the following condition 1 held.

$$|f^{\alpha} - f^{*}| < \epsilon_1 \cdot f^{*} + \epsilon_2 \tag{1}$$

with:

- f^* the global optimum of the objective function ;
- f^{α} the optimum found by the algorithm ;
- ϵ_1 and ϵ_2 accuracy parameters: in all tests discussed in that paper, $\epsilon_1 = \epsilon_2 = 10^{-4}$.

In order to compare HCIAC with two competing continuous ant colony algorithms, we have performed some tests with the same test functions than those used in the



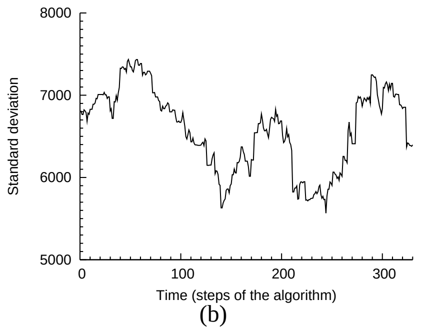


Fig. 7. The HCIAC algorithm optimizing the $B2$ function. (a) average of the values of the function found by the ants, (b) standard deviation of the same values.

related articles. Results are presented in Tab. 2. The values given in square brackets describe tests with fixed numbers of evaluations, without any other stopping criterion and empty cells stand when values are not given in the literature [12,1].

HCIAC outperforms CIAC for most of the test functions, except the Baluja test suite and the Gr_{10} function. Concerning the other ant colony algorithms, we can observe that the rapidity of HCIAC tends to be similar, but due to the lack of complete data in the literature, the comparison is difficult. What can be said is that, comparatively to other ant colony algorithms, HCIAC performs a precise and efficient optimization, but needs a lot of objective function evaluations.

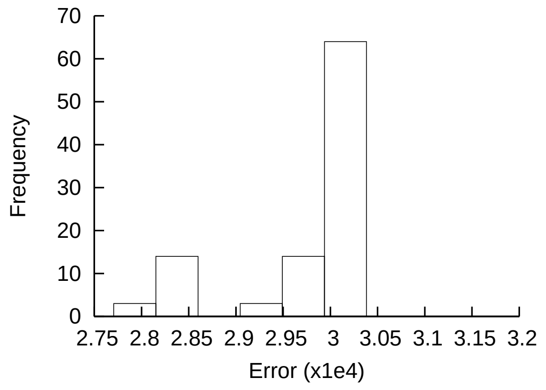
We finally compared HCIAC with other continuous optimization algorithms, the results are shown in Tab.3.

We can observe that if HCIAC does not compete with the other algorithms on rapidity, it gains speed and efficiency comparatively to CIAC. In addition, HCIAC outperforms other algorithms concerning precision and efficiency, except for the Griewangk function, which seems to be really difficult for HCIAC.

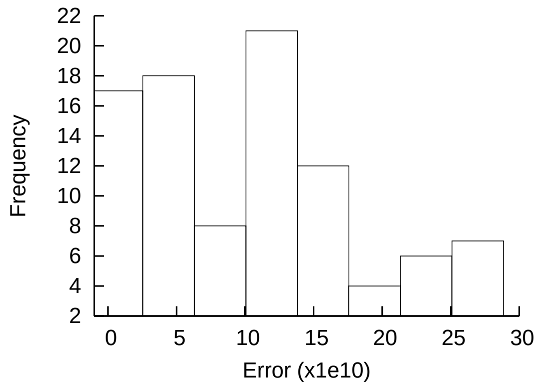
To test if the behaviours of the CIAC and the HCIAC algorithm are statistically different, we used the non-parametric Kruskal-Wallis test. These tests were achieved using the *R* environment for statistical computing, version 2.0.0.

For each of the tests functions listed in tables 2 and 3, we test if the distributions of errors are different. For each problem, we obtain $p < 0.005$, this means that there is a statistically signifiant difference in the errors of CIAC and HCIAC.

To illustrate this result, we have plotted the error distributions of both algorithms on figure 8. This figure shows that on the R_2 problem, the distributions are radically different.



(a) CIAC



(b) HCIAC

Fig. 8. Error distributions of CIAC and HCIAC for the Rosenbrock R_2 problem.

Table 2

Results of HCIAC against competing ant colony algorithms.

	CACO			API		
Function	% ok	evals	err	% ok	evals	err
$R2$	100	6842	0.00		[10000]	0.00 (0.00)
SM	100	22050			[10000]	0.00 (0.00)
$Gr5$					[10000]	0.18 (0.04)
$Gr10$	100	50000	0.0			
GP	100	5330				
MG	100	1688				
St		[6000]	0.0			
B_{f1}		[200000]	99456			
B_{f2}		[200000]	99123			
B_{f3}		[200000]	45769			

	CIAC			HCIAC		
Function	% ok	evals	err	% ok	evals	err
$R2$	100	11797	3e-3(3e-3)	100	18747(6697)	1e-8(4e-9)
SM	100	50000	9e-10(1e-11)	100	18616(6715)	5e-8(1e-8)
$Gr5$	63	48402	0.01(9e-3)	75	10870(1347)	1e-4(2e-4)
$Gr10$	52	50121	0.05(0.05)	18	23206(13132)	1e-3(4e-4)
GP	56	23391	1.51(2.33)	100	34533(4086)	0(0)
MG	20	11751	0.34(0.19)	100	24596(11413)	4e-9(4e-9)
St	94	28201	1.96(1.61)	100	10726(219)	0(0)
B_{f1}	0	50000	99521(463)	0	20388(7466)	9999(3e-3)
B_{f2}	0	50000	99635(512)	0	18734(6902)	9999(2e-3)
B_{f3}	0	50000	99895(86)	0	19685(6863)	9999(1e-3)

Table 3

Results of HCIAC against four competing continuous optimization algorithms.

	CGA			ECTS			DE		
Function	% ok	evals	err	% ok	evals	err	% ok	evals	err
St							100	1300	
Gr_{10}							100	12804	
SM	100	750	0.0002	100	338	3e-8	100	392	
R_2	100	960	0.004	100	480		100	615	
R_5	100	3990	0.15	100	2142	0.08			
GP	100	410	0.001	100	231	2e-3			
$S_{4,5}$	76	610	0.14	75	825	0.01			

	CIAC			HCIAC		
Function	% ok	evals	err	% ok	evals	err
St	94	8201	1.96(1.61)	100	726(219)	0(0)
Gr_{10}	52	50121	0.05(0.05)	18	23206(13132)	1e-3(4e-4)
SM	100	50000	9e-10(1e-11)	100	18616(6715)	5e-8(1e-8)
R_2	100	11797	3e-3(3e-3)	100	18747(6697)	1e-8(4e-9)
R_5	90	39546	8e-3(7e-3)	100	19469(6606)	6e-8(2e-8)
GP	56	23391	1.51(2.33)	100	34533(4086)	0(0)
$S_{4,5}$	5	39311	6.34(1.01)	100	17761(5976)	0(0)

6. Conclusion

We have shown that the heterarchical concept can be interesting to design new ant colony algorithms, in particular aimed at the optimization of continuous multiminima functions. We have proposed to extend the ant colony metaphor to take into account several communication processes. We have proposed an Interacting Ant Colony Algorithm hybridized with a local search to improve its performance. We found that the new HCIAC algorithm performs better than CIAC, especially in terms of accuracy and efficiency.

When comparing HCIAC with competing algorithms, we pointed out the high number of evaluations, inherited from the ant colony algorithms concept and from the use of a less sensible stopping criterion in our tests. But we have also shown the efficiency and the accuracy of the algorithm. If a particular problem requires an increased speed, we recommend to make use of the intensification/diversification index. Indeed, it is a good tool to tune the algorithm according to a given problem. An end user who needs to fit the algorithm to his problem can simply decide if he wants a fast, or a precise solution. HCIAC also inherits the qualities of the ant colony concept, such as self-management and flexibility.

Furthermore, if we take a look at both the advantages and the drawbacks of the ant colony metaphor, we can see that these algorithms bear medium performance when handling static test functions. Conversely they seem to be well suited to dynamical problems, due to their distributed and adaptive features [2].

We believe that HCIAC can be more efficient on dynamical problems, probably even on continuous spaces. Indeed, HCIAC takes advantage of the heterarchical framework and makes use of communication channels — which can be used to rapidly diffuse an information about some variation in the objective function — ; besides HCIAC involves an efficient local search, which can be easily adapted to dynamical functions, as shown by [18].

References

[1] G. Bilchev and I.C. Parmee. The Ant Colony Metaphor for Searching Continuous Design Spaces. *Lecture Notes in Computer Science*, 993:25–39, 1995.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence, From Natural to Artificial Systems*. Oxford University Press, 1999.

[3] S. Camazine, J.L. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, 2000.

[4] R. Chelouah and P. Siarry. A Continuous Genetic Algorithm Designed for the Global Optimization of Multimodal Functions. *Journal of Heuristics*, 6:191–213, 2000.

[5] R. Chelouah and P. Siarry. Tabu Search Applied to Global Optimization. *European Journal of Operational Research*, 123:256–270, 2000.

[6] Y. Collette. *Contribution à l'évaluation et au perfectionnement des méthodes d'optimisation multiobjectif. Application à l'optimisation des plans de rechargement de combustible nucléaire*. PhD thesis, Université de Paris XII Val-de-Marne, December 2002.

[7] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. In F. Varela and P. Bourgine, editors, *Proceedings of ECAL'91 - First European Conference on Artificial Life*, pages 134–142, Paris, France, 1992. Elsevier Publishing.

[8] J. Dréo and P. Siarry. Continuous Interacting Ant Colony Algorithm Based on Dense Heterarchy. *Future Generation Computer Systems*, 20(5):841–856, 2004.

[9] C. M. Fonseca and P. J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993.

[10] 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 on Ant Algorithms (ANTS'2002)*, volume 2463 of *Lecture Notes in Computer Science*, pages 288–289, Brussels, Belgium, September 2002. Springer Verlag.

[11] M. Mathur, S. B. Karale, S. Priye, V. K. Jyaraman, and B. D. Kulkarni. Ant Colony Approach to Continuous Function Optimization. *Ind. Eng. Chem. Res.*, 39:3814–3822, 2000.

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

[13] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[14] Rayner Storn and Kenneth Price. Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR95 -012, International Computer Science Institute, Berkeley, California, March 1995.

[15] É. D. Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Potvin. Adaptive Memory Programming: A Unified View of Meta-Heuristics. *European Journal of Operational Research*, 135(1):1–16, 1998.

[16] E.O. Wilson and B. Hölldobler. Dense Heterarchy and mass communication as the basis of organization in ant colonies. *Trend in Ecology and Evolution*, 3:65–68, 1988.

Received 7 July 2005; revised 5 April 2006; accepted 29 April 2006

[17] M. Wodrich and G. Bilchev. Cooperative distributed search: the ant's way. *Control and Cybernetics*, 26(3), 1997.

[18] Q. Xiong and A. Jutan. Continuous optimization using a dynamic simplex method. *Chemical Engineering Science*, 58(16):3817–3828, 2003.