Comput Optim Appl
DOI 10.1007/s10589-008-9176-7

Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions

Akbar Karimi · Hadi Nobahari · Patrick Siarry

Received: 1 August 2006 / Revised: 6 March 2008

Abstract A new hybrid optimization method, combining Continuous Ant Colony System (CACS) and Tabu Search (TS) is proposed for minimization of continuous multi-minima functions. The new algorithm incorporates the concepts of promising list, tabu list and tabu balls from TS into the framework of CACS. This enables the resultant algorithm to avoid bad regions and to be guided toward the areas more likely to contain the global minimum. New strategies are proposed to dynamically tune the radius of the tabu balls during the execution and also to handle the variable correlations. The promising list is also used to update the pheromone distribution over the search space. The parameters of the new method are tuned based on the results obtained for a set of standard test functions. The results of the proposed scheme are also compared with those of some recent ant based and non-ant based meta-heuristics, showing improvements in terms of accuracy and efficiency.

Keywords Ant colony optimization · Tabu search · Hybrid meta-heuristics · Global optimization · Continuous optimization

A. Karimi · H. Nobahari
Department of Aerospace Engineering, Sharif University of Technology, P.O. Box 11365-8639, Tehran, Iran

A. Karimi
e-mail: a_karimi@ae.sharif.edu

H. Nobahari
e-mail: nobahari@sharif.edu

P. Siarry (✉)
Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi), Université Paris 12 Val-de-Marne, 94010 Créteil, France
e-mail: siarry@univ-paris12.fr

1 Introduction

The global minimization of continuous multi-minima functions consists in finding the global minima without being trapped into any of the local minima. Various meta-heuristic approaches have been developed to solve these problems, such as Simulated Annealing, Genetic Algorithms (GA), Tabu Search, Ant Colony Optimization and so on. A big challenge in developing global optimization approaches is to compromise the contradictory requirements, including accuracy, robustness and computation time. It is difficult to meet all these requirements by concentrating on a sole meta-heuristic. In recent years, there has been an up-growing interest in hybridization of different meta-heuristics to provide more efficient algorithms. In this paper a hybridization of Tabu Search and Ant Colony Optimization meta-heuristics is proposed.

Tabu Search (TS) was originally developed by Glover [1, 2]. This meta-heuristic has been successfully applied to a variety of combinatorial optimization problems. The extension of TS to continuous optimization problems has been investigated notably in [3–8]. The proposed algorithm in [6], called Continuous Tabu Search (CTS), starts from a randomly generated initial solution, called the initial current solution. From this current solution, a set of neighbors are generated. To avoid the appearance of a cycle, the neighbors of the current solution, which belong to a subsequently defined 'tabu list', are systematically eliminated. The objective function is evaluated for each neighbor. The best neighbor becomes the new current solution, even if it is worse than the initial current solution. This allows escaping from the local minima of the objective function. The consequently generated current solutions are put into a circular list of tabu solutions, called tabu list. When the tabu list becomes full, it is updated by removing the first solution entered. Then a new iteration is performed. The previous procedure is repeated by starting from the new current point, until some stopping condition is reached. Usually, the algorithm stops after a given number of iterations for which no improvement of the objective function occurs.

To improve the accuracy of CTS, Chelouah and Siarry [7, 8] proposed a variant of TS, called Enhanced Continuous Tabu Search (ECTS). This scheme divides the optimization process into two sequential phases, namely diversification and intensification. In diversification, the algorithm scans the whole solution space and detects the areas, which are likely to contain a global minimum. The centers of these promising areas are stored in a list, called the promising list. When diversification ends, the intensification starts. In this phase, the search is concentrated within the most promising area by making the search domain smaller and gradually reducing the neighborhood structure. This strategy improves the performance of the algorithm and allows exploiting the most promising area with more accuracy [7, 8].

The other meta-heuristic, utilized in this paper, is Ant Colony Optimization (ACO), which was first proposed by Marco Dorigo and colleagues [9, 10] as a multi-agent approach to solve difficult combinatorial optimization problems. The first algorithm inspired from the ant colony functioning, is Ant System (AS) [9, 11], which is the base of many other approaches such as Max-Min AS (MMAS) [12], Ant Colony System (ACS) [13], Ant-Q [14] and ANTCOL [15]. The main idea utilized in all of these algorithms has been adopted from the ants' pheromone trails-laying behavior, which is an indirect form of communication mediated by modifications of the environment.

The application of ant algorithms to optimization problems can be divided into two main categories, including discrete and continuous problems. Most of the primary algorithms were developed to solve discrete optimization problems such as Traveling Salesman Problem, Quadratic Assignment Problem, Job-Shop scheduling, Vehicle Routing, Sequential Ordering, Graph Coloring, Routing in Communications Networks and so on [16, 17]. However, there have been several attempts to adapt ACO for continuous optimization problems. Bilchev and Parmee [19] and Wodrich and Bilchev [18] proposed a method called Continuous Ant Colony Optimization (CACO), which uses ant colony framework to perform local searches whereas the global search is handled by a genetic algorithm. Another idea, known as API method [20], was proposed based on a kind of recruitment process. This method was inspired by a primitive ant's recruitment behavior. It performs a tandem-running which involves two ants and leads them to gather on a same hunting site. The authors use this particular recruitment technique to make the population proceed towards the optimum solution, by selecting the best point among those evaluated by the ants.

A recent research on modeling ants' behavior [21, 22] has shown that it is possible to start a recruitment sequence even without taking pheromone trails into account. In this model, the stigmergic process is considered jointly with inter-individual relationships. The model is inspired from the flow of ants exiting the nest after the entry of a scout who has discovered a new food source. To differentiate this process from the recruitment, the authors have called it mobilization. This algorithm is called Continuous Interacting Ant Colony (CIAC). Another approach to solve continuous optimization problems is by converting them to discrete form so that the discrete versions of ant algorithms can be used [23, 24].

One pure pheromone based method for global minimization of continuous optimization problems, called Continuous Ant Colony System (CACS), was proposed by Pourtakdoust and Nobahari [25]. To deal with a continuous function, the pheromone distribution over the search space was modeled in the form of a normal Probability Distribution Function (PDF), the center of which is the last best global solution, found from the beginning of the trial, and its variance depends on the aggregation of the other promising areas around the best one. In this context a random generator with normal probability distribution function is utilized as the state transition operator. In another work, published concurrently by Socha and Dorigo (first in [26] and then in [27, 28]), a combination of several normal PDF was utilized instead of the discrete PDF used in the original ACO formulations. They called this method as Ant Colony Optimization for Real domains (ACO_R). In recent years, CACS has been successfully applied to several practical optimization problems such as fuzzy rule learning and shape optimization [29–31].

It should also be noted that among aforementioned ant-related algorithms, CACO, API and CIAC all have conceptual differences with the original ACO formulations, regarding the operators they use. Therefore, as discussed in [28], they do not qualify to be extensions of ACO.

In this work the authors hybridize their previously proposed methods to introduce a new hybrid optimization scheme. The proposed algorithm is called Tabu Continuous Ant Colony System (TCACS). The basic structure of TCACS is very similar to the original CACS, while it incorporates the concepts of tabu and promising lists,

used in CTS and ECTS, into the framework of CACS to improve the performance. The use of the promising list improves the convergence rate while the utilization of the so called tabu balls guides the ants toward the solutions far enough away from the worst regions of the search space. On the other hand, the use of promising and tabu lists allows to dynamically adjust the size of the tabu balls during the execution of the algorithm. Furthermore, to handle the undesired correlations between the optimization variables and have a more effective sampling, all pheromone calculations are performed in a separate coordinate system whose orientation is specified in a non-deterministic way.

The paper is organized as follows: Sect. 2 is devoted to a detailed description of the new optimization method. The experimental results are provided in Sect. 3. Here, the parameters of TCACS are tuned based on the results obtained for a set of standard test functions and the performance of the new algorithm is compared with those of some other meta-heuristics, namely four ant-based (ACO_R, CIAC, CACO, and CACS) and four non ant-based methods (CGA, CHA, ECTS, and ESA). The final conclusion is made in Sect. 4.

2 Tabu continuous ant colony system

It is desired to find the global minimum of the function f, within a given interval $[a, b]$, in which the minimum occurs at a point $\boldsymbol{x}_s$. In general, f can be a multi-variable function, defined on a subset of R^n delimited by n intervals $[a_i, b_i], i = 1, \ldots, n$.

2.1 General setting out of the algorithm

Figure 1 shows the general iterative structure of TCACS. A high level description of the sequential steps is shown in this figure. In the following subsections, these steps are discussed in detail. As in CACS, a continuous pheromone model is used to gradually guide the ants toward the global minimum point. This pheromone model is in fact a strategy to assign a continuous probability distribution to the whole solution space and to methodically update it as the algorithm progresses. During any iteration, ants move from their current positions to the new destinations according to the current pheromone distribution. The destinations are chosen using a random generator with normal probability distribution function. The values of the objective function are calculated in these new points and some knowledge about the problem is acquired, which is used to update the pheromone distribution.

Like CACS, for each dimension of the solution space a normal probability distribution function is used to model the pheromone aggregation around the last best point found from the beginning of the trial. Therefore, for $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ being an arbitrary point within the solution space, and $\boldsymbol{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ being the last best point, the pheromone intensities are given by n normal distribution functions in the following form:

$$\tau(x_i) = e^{-\frac{(x_i - x_i^*)^2}{2\sigma_i^2}}, \tag{1}$$

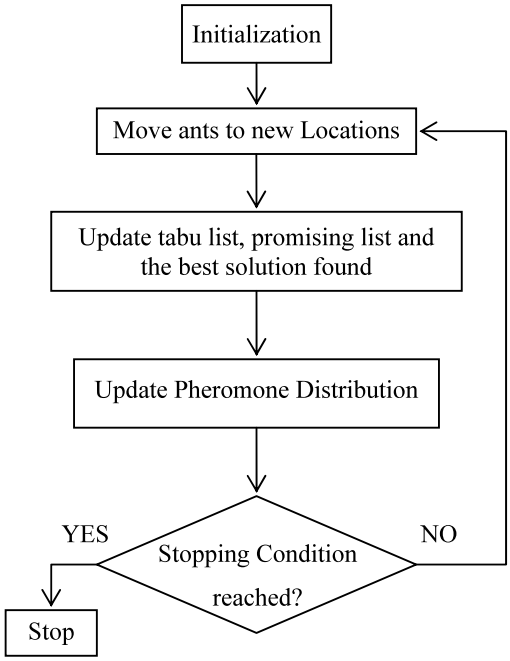

Fig. 1 General flowchart of TCACS

where is the variance of the normal distribution corresponding to the i-th dimension of the solution space. The algorithm updates and σ as it proceeds and the pheromone distribution over the whole solution space gradually changes.

A major difference between TCACS and CACS is the use of tabu and promising lists. Simply a specified number of the best points found from the beginning of the algorithm up to the current iteration form the promising list. Likewise, a specified number of the worst points found so far constitute the tabu list. Each member of the tabu list is the center of a *tabu ball*, the size of which is updated during iterations. Tabu balls specify circular, spherical and in general hyper-spherical regions within the solution space. The ants are not allowed to select any point inside the tabu balls, while they are choosing their new destinations. In other words, an *acceptable movement* is made when the new randomly selected destination does not lie within any of the tabu balls. Figure 2 lists the pseudo code of TCACS algorithm.

2.2 Initialization

The new algorithm has some control parameters, defined in the next subsections, which must be set before the execution of the algorithm. As it will be discussed later, pheromone distribution is computed in a temporary coordinate system, which is initially the same as the original coordinate system. Moreover, both the tabu and promising lists must be initially empty.

```
procedure TCACS()
    initialize()
    while (termination condition not satisfied)
        if (iteration_number=1)
            uniformly select initial position of ants
        else
            move_ants_to_new_locations()
            update tabu list
        end if
        update promising list
        update tabu ball size
        update coordinates transformation matrix
        update_pheromone_distribution()
    end while
end procedure
procedure initialize()
    set the parameters of algorithm
    set tabu and promising lists to empty tables
    set current coordinates system to the original axes
end procedure
procedure move_ants_to_new_locations()
    for i=1, k
        repeat
            for j=1, n
                choose the new x_j for the i-th ant
                using a normal random generator
            end for
        until (x_j is not included in a tabu ball)
    end for
end procedure
procedure update_pheromone_distribution()
    update the globally best point
    for j=1, n
        update the value of σ_j
    end for
end procedure
```

Fig. 2 Pseudo-code of TCACS

2.3 Movement of the ants

Each iteration starts by the movement of the ants to new locations $\boldsymbol{x}^j = (x_1, x_2, \ldots, x_n)$, $j = 1, \ldots, k$. Chosen randomly within the solution space, these new points should not belong to any of the tabu balls. It should be noted that if a random gen-

erated value for the i-th component of $\boldsymbol{x}^j$ lies outside the specified Interval $[a_i, b_i]$, then a new one is generated and the process is repeated until an acceptable value is obtained.

In order to be able to use the coordinate correlation handling routine described in Sect. 2.7, we describe and calculate the pheromone distribution in a separate coordinate system, denoted by Z, whose orientation is updated at any iteration based on the current distribution of the individuals. Thus, to move each ant to a new location, a vector of random increments in Z axes is generated based on the current pheromone distribution. This vector is then transformed into the original coordinate system, $\boldsymbol{X}$, and added to the best found solution as follows:

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}^* + \mathrm{R}^{XZ}\boldsymbol{N}(0, \boldsymbol{\sigma}). \tag{2}$$

Where $\boldsymbol{x}^*$ is the best solution found so far, R^{XZ} is the rotation matrix from Z to X axes, and $\boldsymbol{N}(0, \boldsymbol{\sigma})$ is a vector of random values generated according to a normal probability distribution with mean of zero and standard deviations specified in the vector $\boldsymbol{\sigma}$. As mentioned before, $\boldsymbol{\sigma}$ states the pheromone distributions according to Z axes.

After selection of all destinations, the objective function is evaluated at these points. The results are stored in a vector $\boldsymbol{y}$, where $y^j = f(\boldsymbol{x}^j)$, $j = 1, \ldots, k$. At the first iteration, both the tabu and promising lists are empty, therefore the initial position of the ants is selected using a uniform random generator, whereas for all subsequent iterations, the (2) is used to select the random points.

2.4 Updating tabu list, promising list and tabu ball radius

Tabu and promising lists are $k \times n$ matrices that store the worst and the best points visited from the beginning of the algorithm up to the current iteration. The parameters k and n are the number of ants and the dimension of the problem, respectively. Therefore, each row of the tabu or promising lists represents a point within the solution space.

In order to update the tabu and promising lists, the new points generated in the current iteration and the current contents of the tabu and promising lists are first combined to form a set of $3 \times k$ individuals. Then all individuals falling outside a hyper-rectangular neighborhood of the current best solution (x^*), defined as (3), are discarded from the set. It should be noted that the area outside of this region is not effectively covered by the sampling process and hence it is not useful to have any tabu balls within it.

$$x^* - 3 \times \max\{\boldsymbol{\sigma}\} \leq x_i \leq x^* + 3 \times \max\{\boldsymbol{\sigma}\}, \quad i = 1, \ldots, n. \tag{3}$$

Now the k number of the best and the worst individuals among the new set are specified as the new promising and tabu points, respectively. The promising list is filled in the first iteration, after the initial locations of the ants have been selected. It is updated in the later iterations and is used to update pheromone distributions. Moreover, it should be noted that the tabu list is empty in the first iteration and is filled in the second iteration after the algorithm has evaluated $2k$ points within the solution

space. The tabu list is used in the optimization process from the third iteration to the end.

During any iteration, the radius of the tabu balls is calculated as half of the smallest distance between the tabu points and the promising points ($d_{\min}$) as follows:

$$r_{\mathrm{T}} = \frac{1}{2} d_{\min} = \frac{1}{2} \min_{j=1}^{k} \left\| \boldsymbol{x}_{\mathrm{T}}^{j} - \boldsymbol{x}_{\mathrm{P}}^{j} \right\|, \tag{4}$$

where subscripts P and T represent the promising and the tabu lists, respectively. The parameter r_{T} specifies the minimum Euclidean distance that must exist between each random point and tabu points. It is important to note that using this scheme, the size of the tabu balls dynamically changes based on the distribution of the promising and the tabu points in the solution space.

2.5 Updating pheromone distributions

TCACS utilizes the pheromone updating rule of CACS as the basic structure. However some modifications are made to improve the performance. Pheromone updating rule of CACS can be stated as follows: the value of the objective function is evaluated for the new selected points by the ants. Then the best point found from the beginning of the trial is assigned to $\boldsymbol{x}^*$. Also, $\boldsymbol{\sigma}$ is updated based on the evaluated points during the last iteration and the aggregation of those points around $\boldsymbol{x}^*$. To satisfy simultaneously the fitness and aggregation criteria, CACS uses the concept of weighted variance as follows:

$$\sigma_i^2 = \frac{\sum_{j=1}^{k} \frac{1}{y^j - y^*} [x_i^j - x_i^*]^2}{\sum_{j=1}^{k} \frac{1}{y^j - y^*}}, \quad \text{for all } j \text{ in which } y^j \neq y^*. \tag{5}$$

However, in TCACS, the members of the promising list are used instead of the current positions of the ants, and some new elements are introduced. The pheromone updating rule of TCACS is stated as follows:

$$\sigma_i^2 = \frac{\sum_{j=1}^{k} (\gamma w_f^j + (1-\gamma) w_d^j)[(x_i^j)_{\mathrm{P}} - x_i^*]^2}{\sum_{j=1}^{k} (\gamma w_f^j + (1-\gamma) w_d^j)}, \quad \text{for all } j \text{ in which } y_{\mathrm{P}}^j \neq y^*, \tag{6}$$

where w_f^j and w_d^j are two weighting indices for the j-th promising point. The first index, w_f is a measure of the optimality of the point and w_d is a measure of how far the point is from the current optimal point. We have considered two ways for calculating these indices: the rank method and the roulette method.

When the rank method is used, the objective function values corresponding to the current points, except the current best point, are first sorted in descending order. Then, the rank of each point within the list is assigned to w_f. Therefore, the point with the minimum objective function value will receive a w_f of $k-1$. The points are again sorted according to their distance from the current best point and the rank of each point within the list is assigned to w_d. This time, the farthest point from the best one receives a w_d of $k-1$.

When Roulette method is used, w_f and w_d are calculated as follows:

$$w_f^j = \frac{\max(y_P) - y_P^j}{\sum_{j=1}^{k}(\max(y_P) - y_P^j)}, \quad \text{for all } j \text{ in which } y_P^j \neq y^*, \tag{7}$$

$$w_d^j = \frac{d_P^j - \min(d_P)}{\sum_{j=1}^{k}(d_P^j - \min(d_P))}, \quad \text{for all } j \text{ in which } y_P^j \neq y^*, \tag{8}$$

where the current best point is again discarded from the promising list. In the later equation, d_P^j represents the distance of the j-th point of the list from the current best point, and $\boldsymbol{d}_P$ is a vector containing the distance values associated with the list.

Regardless of the method used to calculate weighting indices, the collective weight values can be obtained by blending these two indices using a weighting factor γ as in (6). In fact, the points with better objective function values are preferred while at the same time choosing the points far from the current best solution, allows a more distributed promising list and helps the algorithm to avoid premature convergences.

2.6 Stopping conditions

Various stopping conditions can be applied to the algorithm. The algorithm may stop when a maximum number of evaluations, a minimum value of the weighted variance or a maximum number of iterations, without any significant improvement in the objective function, is reached. In this paper, we have two experimental setups, each with its own specific stopping condition. In the first setup, the algorithm stops when the Euclidean distance between the best found solution and the other individuals falls below a certain threshold. In the second setup, the algorithm stops when a predefined level of accuracy is reached. The details of both criteria are described in Sect. 3.2.

2.7 Variable correlation handling

Sometimes the variables of the problem are highly correlated, which is mainly due to the inherent rotation of the function landscape with respect to the reference coordinate system. In such cases, it is required to perform sampling in different directions according to the distribution of the individuals. As mentioned earlier, in this paper a temporary coordinate system, Z, is utilized to describe the pheromone distributions while sampling new points. Then during any iteration, it is needed to determine the orientation of Z axes so that they coincide with the principal directions among the current population. Principal Component Analysis (PCA) is a well-known method in this regard. However, Socha and Dorigo [27, 28] have stated that the use of PCA in ACO_R has not proven to be successful and often leads to stagnation. Therefore they propose their own method, which performs the similar task in a stochastic manner [27, 28]. Unfortunately, the details of this method are not reported completely.

In this work, the authors first employed PCA to determine the current principal coordinate system; however, like in ACO_R, it did not prove to be an efficient approach, and the same problem of premature convergence was experienced, as Socha and Dorigo [27, 28] reported. Therefore, an alternative stochastic method is designed,

which may have similarities with that used in [27, 28]. This method consists of the following steps:

1. The origin of the coordinate system is translated to the mean of the current individuals.
2. One of the present individuals, $\boldsymbol{x}^j$, is randomly selected, with a selection probability proportional to $|\boldsymbol{x}^j|^m$, where $|\boldsymbol{x}^j|$ is the Euclidean norm of the j-th individual and m is a parameter of the algorithm. The position vector of the selected individual is then considered as the first axis of the new coordinate system, denoted by $\boldsymbol{v}^1$.
3. For remaining $n-1$ individuals, the equation set $(\boldsymbol{x}^j + \alpha^j \boldsymbol{v}^1) \cdot \boldsymbol{v}^1 = 0$ is solved for the values of α^j. The new individuals, $\boldsymbol{x}^j + \alpha^j \boldsymbol{v}^1$, obtained in this way, represent the projection of the position vectors $\boldsymbol{x}^j$ on the plane perpendicular to the vector $\boldsymbol{v}^1$.
4. Given the new individuals, $\boldsymbol{x}^j + \alpha^j \boldsymbol{v}^1$, the second selection is made with a selection probability proportional to $|\boldsymbol{x}^j + \alpha^j \boldsymbol{v}^1|^m$. The second axis $\boldsymbol{v}^2$, which is normal to the first axis, is obtained in this way.
5. For remaining $n-2$ individuals, the equation set $(\boldsymbol{x}^j + \alpha^j \boldsymbol{v}^1 + \beta^j \boldsymbol{v}^2) \cdot \boldsymbol{v}^2 = 0$ is solved for the values of β^j. The third axis $\boldsymbol{v}^3$, is then selected with a selection probability proportional to $|\boldsymbol{x}^j + \alpha^j \boldsymbol{v}^1 + \beta^j \boldsymbol{v}^2|^m$.
6. This pattern is repeated until $n-1$ principal axes are obtained.
7. The n-th axis is specified using the orthogonality conditions:

$$\begin{cases} \boldsymbol{v}^1 \cdot \boldsymbol{v}^n = 0, \\ \boldsymbol{v}^2 \cdot \boldsymbol{v}^n = 0, \\ \vdots \\ \boldsymbol{v}^{n-1} \cdot \boldsymbol{v}^n = 0. \end{cases} \tag{9}$$

However, one more equation is needed for the system (9) to be determined. So the sum of the elements of the n-th vector is assumed to be equal to an arbitrary value s and form the following matrix equation:

$$\begin{bmatrix} v_1^1 & v_2^1 & \cdots & v_n^1 \\ v_1^2 & v_2^2 & \cdots & v_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ v_1^{n-1} & v_2^{n-1} & \cdots & v_n^{n-1} \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_1^n \\ v_2^n \\ \vdots \\ v_n^n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ s \end{bmatrix}. \tag{10}$$

Solving (10) gives the n-th axis $\boldsymbol{v}^n$. The vectors $\boldsymbol{v}^1, \boldsymbol{v}^2, \ldots, \boldsymbol{v}^n$ are then normalized to form the rotation matrix R^{XZ}.

3 Results and discussion

In this section, the parameters of the algorithm and their suggested optimal values are introduced in the first subsection. A comparison of the algorithm with some other

Table 1 Optimal parameter settings for TCACS

Problem dimension	k	Weighting strategy	γ	m
$n < 4$	10	Rank	1	1
$n \geq 4$	15	Roulette	0.5	2

Table 2 The algorithms present in the first experimental category

Abbr.	Complete name	Type	Reference
TCACS	Tabu Continuous Ant Colony System	Hybrid: Ant Colony + Tabu Search	This work
CGA	Continuous Genetic Algorithm	Genetic Algorithm	[32]
CHA	Continuous Hybrid Algorithm	Hybrid: Genetic Algorithm Simplex Search	[34]
ECTS	Enhanced Continuous Tabu Search	Tabu Search	[7]
ESA	Enhanced Simulated Annealing	Simulated Annealing	[33]
CIAC	Continuous Interacting Ant Colony	Ant Colony	[21]
CACO	Continuous Ant Colony Algorithm	Ant Colony	[19]

methods is presented in the next subsection, and a graphical demonstration of the algorithm makes up the last subsection.

3.1 Parameter setting

TCACS has four control parameters to set, the number of ants (k), the weighting strategy which can be either Roulette or Rank, the weighting factor (γ), and the parameter m used in correlation handling method. To find the optimal values of these parameters, the solutions obtained for a set of standard test functions (as listed in the Appendix) with different settings were studied. For each setting, the algorithm was applied to the test functions in 200 different runs. Finally, the optimal parameter setting was obtained as in Table 1, where there are given two optimal sets for the parameters of the algorithm based on the problem dimension, n.

3.2 Comparison with other methods

The new algorithm was tested over a set of standard test functions, listed in the Appendix. Since in the literature the performance of the proposed algorithms has been analyzed in different ways, in this paper the experimental setup was divided into two categories. Each category contains the results of a set of competing algorithms (including TCACS), for which the performance analysis is performed in a similar way.

In the first set of experiments, TCACS is compared with six other meta-heuristics, two ant based and four non-ant based methods, as listed in Table 2. Table 3 presents the results obtained by these methods. Dash signs correspond to cases for which there is no reported value available in the literature. The results for TCACS are obtained using 100 different executions of the algorithm. The algorithm stops when the

Table 3 Comparison of TCACS with other meta-heuristics. Reported are the average number of function evaluations (topmost number), the average error between the solution and the known optimum value (middle number), and the rate of successful minimizations (bottom number in parenthesis). Unavailable data have been represented by dash signs

Test function	TCACS	CGA	CHA	ECTS	ESA	CIAC	CACO
RC	390	620	295	245	–	–	–
	3.9736e-7	1e-4	1e-4	0.05			
	(100%)	(100%)	(100%)	(100%)			
B_2	415	430	132	–	–	11827	–
	6.0263e-9	3e-4	1e-7			2e-8	
	(96%)	(100%)	(100%)			(100%)	
ES	509	1504	952	1284	–	–	–
	6.1284e-10	1e-3	1e-3	0.01			
	(100%)	(100%)	(100%)	(100%)			
GP	239	410	259	231	783	23391	5330
	1.0135e-5	1e-3	1e-3	2e-3	0.86e-2	1.51	–
	(97%)	(100%)	(100%)	(100%)	(100%)	(56%)	(100%)
MG	310	–	–	–	–	11751	1688
	1.3347e-008					0.34	–
	(100%)					(20%)	(100%)
SH	2883	575	345	370	–	–	–
	7.7519e-6	5e-3	5e-3	1e-3			
	(81%)	(100%)	(100%)	(100%)			
R_2	533	960	459	480	–	11797	6842
	1.7776e-8	4e-3	4e-4	0.02		3e-3	0.00
	(81%)	(100%)	(100%)	(100%)		(100%)	(100%)
Z_2	261	620	215	195	–	–	–
	5.2345e-8	3e-6	3e-6	2e-7			
	(100%)	(100%)	(100%)	(100%)			
DJ	309	750	371	338	–	–	–
	4.3096e-10	2e-4	2e-4	3e-8			
	(100%)	(100%)	(100%)	(100%)			
$H_{3,4}$	531	582	492	548	698	–	–
	4.5866e-6	5e-3	5e-3	0.09	0.49e-3		
	(100%)	(100%)	(100%)	(100%)	(100%)		
$S_{4,5}$	1287	610	698	825	1487	39311	–
	9.6728e-6	0.14	9e-3	0.01	0.42e-2	6.34	
	(59%)	(76%)	(85%)	(75%)	(54%)	(5%)	
$S_{4,7}$	1176	680	620	910	1661	–	–
	1.0376e-5	0.12	0.01	0.01	0.84e-2		
	(68%)	(83%)	(85%)	(80%)	(54%)		
$S_{4,10}$	1231	650	635	989	1363	–	–
	1.1399e-5	0.15	0.015	0.01	0.43e-1		
	(74%)	(81%)	(85%)	(75%)	(50%)		

Table 3 (*Continued*)

Test function	TCACS	CGA	CHA	ECTS	ESA	CIAC	CACO
R_5	3315	3990	3290	2142	–	39792	–
	1.3628e-9	0.15	0.018	0.08		8e-3	
	(87%)	(100%)	(100%)	(100%)		(100%)	
Z_5	1557	1350	950	2254	–	–	–
	1.6994e-7	4e-4	6e-5	4e-6			
	(100%)	(100%)	(100%)	(100%)			
SM_6	1646	–	–	–	–	50000	22050
	1.5488e-007					9e-10	–
	(100%)					(100%)	(100%)
$H_{6,4}$	1313	970	930	1520	1638	–	–
	4.6371e-6	0.04	8e-3	0.05	0.59e-1		
	(55%)	(100%)	(100%)	(100%)	(100%)		
R_{10}	22781	21563	14563	15720	12403	–	–
	1.6419	0.02	8e-3	0.02	0.44e-1		
	(100%)[a]	(80%)	(83%)	(85%)	(100%)		
Z_{10}	6680	6991	4291	4630	15820	–	–
	3.0698e-8	1e-6	1e-6	2e-7	0.15e-2		
	(97%)	(100%)	(100%)	(100%)	(100%)		
GR_{10}	3259	–	–	–	–	50121	50000
	4.6009e-006					0.05	0.0
	(26%)					(52%)	(100%)

[a]Successfulness condition ignored (see Sect. 3.2)

Euclidean distance between the best found solution and the other individuals falls below a certain threshold, which is typically 10^{-4}. For each problem, three numbers are reported: the rate of successful minimizations in a sense defined shortly, and the average number of objective function evaluations and the average error between the best found solution and the known global optimum computed for successful minimizations only. Following the literature, a minimization can be considered successful when the following inequality holds:

$$|F_{\text{obj}-}F_{\text{anal}}| \leq \varepsilon_{\text{rel}} \times F_{\text{anal}} + \varepsilon_{\text{abs}}, \tag{11}$$

where F_{obj} is the value of the objective function corresponding to the best found solution and F_{anal} is the known analytical optimum. However, a problem arises here, where different values have been used for ε_{abs}. In fact, in all works, ε_{rel} is set to 10^{-4}, while the value of ε_{abs} in some articles is 10^{-6} and in others it is 10^{-4}. This makes it difficult to compare success rate values especially for functions with zero minimum values. In case of TCACS, we have used values of ε_{rel} and ε_{abs} equal to 10^{-4} and 10^{-6}, respectively.

Another problem with the results reported in the literature is an inconsistency between the successfulness criterion mentioned above and the error values reported.

For instance, when a success rate of 100% is reported for a function with a zero minimum value, one expects the average error to be less than or equal to ε_{abs}; while for many of the problems, this is not the case (even when assuming $\varepsilon_{\text{abs}} = 10^{-4}$) and the average error is of a considerably higher order than ε_{abs} (see Table 3).

Despite these problems, since for most methods both the average number of objective function evaluations and the average error are available in Table 3, it is still possible to compare the algorithms; although not much credit can be given to the success rate values. Since each method uses its own stopping condition allowing different orders of accuracy, to make a fair comparison, one has to simultaneously take both the number of evaluations and the average error values into account.

From Table 3 it is clear that TCACS exhibits superior results and for most problems its performance seems to be better than those of the competing methods. It reaches more accurate solutions with less number of function evaluations. However, there are also areas of weakness. For example, in case of R_{10}, the algorithm was not able to present the level of accuracy required by the successfulness condition used, so the authors have reported the average of all 100 runs regardless of the quality of the results. In this problem, it is observed that the average error of TCACS is higher than those of the other methods. If the competing algorithms were consistent in their definition of the successful minimization, it could also be possible to compare the robustness of TCACS with the other methods. However, as discussed above, it is clearly seen by comparing the reported average error and success rate values for competing methods that they are inconsistent with the definition of the successful minimization mentioned in the corresponding papers.

In the second set of experiments, TCACS is compared with CACS [25] and ACO_{R} [27, 28]. It should be noted that the results of ACO_{R}, reported in [27, 28], are obtained with the successfulness criterion of (10) employed as its stopping condition, which is not a common practice since all the competing methods use stopping conditions independent of the quality of the current solution. Therefore the comparisons made in those papers seem to be unfair. This is especially true when we note that only the number of objective function evaluations is reported and there is no indication of the quality of the solutions provided by each method. Furthermore, the domain of definition for the test functions reported in [27, 28] are in some cases different to those available in the current literature. Nevertheless, in order to provide a fair comparison, TCACS was applied to the problems for which the results of ACO_{R} were available and with the same search domains and stopping condition, i.e. the algorithm stops after (11) is held. The values of ε_{abs} and ε_{rel} were both set to 10^{-4} following the values used in [27, 28].

In addition, the results reported for CACS were too limited and incomplete; therefore, since the authors had access to the original code for CACS, it was applied to the related problems with the same stopping condition and search domain. In this case we examined different numbers of ants equal to 10, 20, 50, and 100. The results of the three methods are presented in Table 4. Reported are the average (arithmetic mean) number of objective function evaluations and the rate of successful minimizations displayed in parentheses. Both for TCACS and CACS we have examined 100 independent runs of the algorithms.

Interesting observations can be made by comparing TCACS with CACS. As seen, almost for all problems, the success rate values of TCACS are considerably higher

Table 4 Comparison of the results obtained by TCACS with ACO_R and CACS. Reported are the average number of function evaluations (topmost number) and the rate of successful minimizations (bottom number in parenthesis). Dash signs represent the cases where no solution was found with the required accuracy

Test function	TCACS	ACO_R	CACS Na = 10	CACS Na = 20	CACS Na = 50	CACS Na = 100
RC	239	735	131	222	437	827
	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
B_2	238	560	215	358	714	1298
	(94%)	(100%)	(79%)	(97%)	(100%)	(100%)
ES	287	772	631	1765	5777	8601
	(99%)	(98%)	(90%)	(97%)	(96%)	(11%)
GP	167	392	152	198	407	684
	(98%)	(100%)	(83%)	(89%)	(99%)	(100%)
MG	157	345	164	231	427	736
	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
R_2	206	816	274	307	551	923
	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
Z_2	138	292	124	188	343	576
	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
DJ	194	400	195	294	573	1003
	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
$H_{3,4}$	259	342	136	207	404	695
	(100%)	(100%)	(94%)	(100%)	(100%)	(100%)
$S_{4,5}$	768	793	432	842	2367	5274
	(63%)	(57%)	(35%)	(32%)	(34%)	(45%)
$S_{4,7}$	684	748	441	856	2219	4920
	(74%)	(79%)	(43%)	(47%)	(60%)	(68%)
$S_{4,10}$	738	715	471	791	2237	4898
	(75%)	(81%)	(22%)	(31%)	(49%)	(62%)
R_5	2356	2487	–	–	–	–
	(91%)	(97%)	(0%)	(0%)	(0%)	(0%)
Z_5	735	727	861	820	1437	2491
	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
SM_6	744	781	540	767	1568	2810
	(100%)	(100%)	(100%)	(100%)	(100%)	(100%)
$H_{6,4}$	621	722	405	568	1159	2147
	(71%)	(100%)	(61%)	(64%)	(51%)	(58%)
GR_{10}	1473	1390	908	1149	2575	4890
	(37%)	(61%)	(7%)	(17%)	(25%)	(28%)

than those of CACS. In case of R_5, being a highly correlated problem, CACS completely fails to find the global minimum, while TCACS tackles it with a good performance. This can be directly attributed to the variable correlation handling scheme

implemented in TCACS. It is also observable that the 20 number of ants reported in [25] as the optimal number of ants, proves to be optimal also for this set of test functions. Thus, considering the results of CACS with 20 ants, we can conclude that almost in all test cases TCACS is faster and more robust than CACS. This can be linked to the use of tabu and promising balls. A clear evidence is the far superior performance of TCACS in solving Easom problem that offers an extremely uniform landscape with a very narrow valley within, which makes it a hard one. The use of tabu balls blocks large portion of this uniform area and increases the chance of hitting the narrow valley.

Comparing TCACS with ACO_R, it can be seen that for almost all test functions used, TCACS is faster than or at least as fast as ACO_R. This is especially true for the functions of fewer dimensions, where TCACS reaches the desired solution with about half of the number of function evaluations needed by ACO_R. On the other hand, it is observable that ACO_R generally shows higher success rates than TCACS, particularly in cases of functions $H_{6,4}$ and GR_{10}, where a considerable difference is seen. Therefore, it can be concluded that TCACS is a faster algorithm than ACO_R, while the latter shows better robustness. This is also in accordance with the trade-off which always exists between opposing characteristics, efficiency and robustness of the algorithms.

3.3 Demonstration of the algorithm

In this section a graphical representation, on how the algorithm progresses, is demonstrated. Figure 3 shows the variation of a 2-variable function B_2 as defined in Appendix versus x_1 and x_2, where in this demonstration have been limited between -1 and 1.

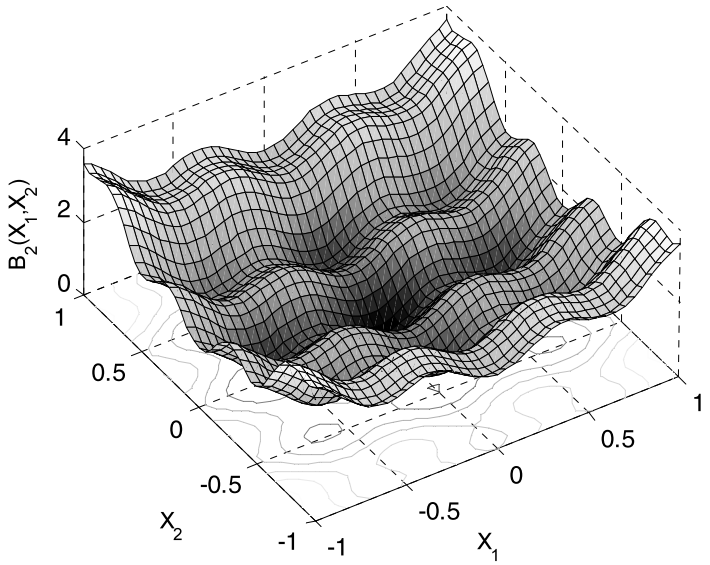

Fig. 3 Graphical representation of function B_2

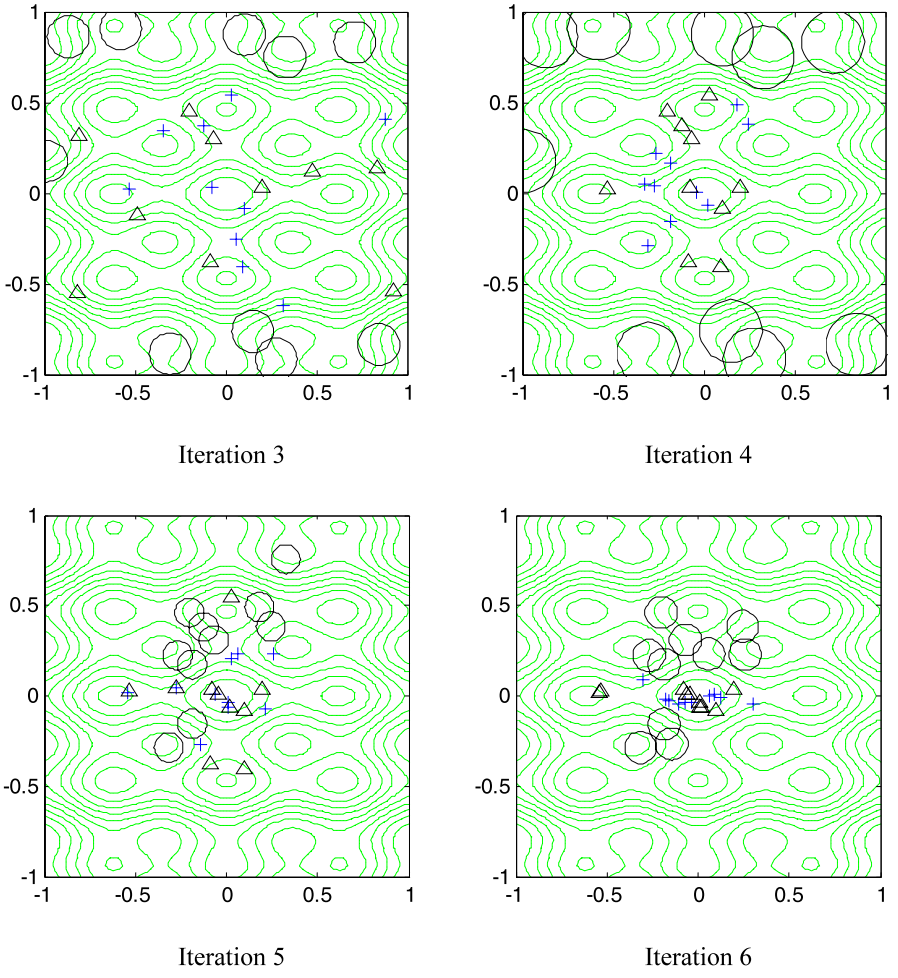

Fig. 4 A demonstration for function B_2 on how the tabu (O) and promising (Δ) points and the current ($+$) solutions are updated as TCACS progresses

Figure 4 represents the arrangement of the tabu balls, the promising points and the current locations of the ants during iterations 3, 4, 5, and 6. The first interesting observation is that as algorithm progresses, tabu and promising points gather in the worst and the best regions of the search space, respectively. Also it can be observed how the radius of tabu balls varies between iterations. Generally, the size of tabu balls decreases with the progress of the search and aggregation of the ants around the global optimum. However, based on the arrangement of the tabu and promising points, it can also be increased from one iteration to the next. This prevents the ants to choose their destinations within the tabu areas and pushes them toward the promising points.

4 Conclusion

In this paper a new hybrid optimization method, combining Continuous Ant Colony System (CACS) and Tabu Search (TS) was proposed for minimization of continuous multi-minima functions. The basic structure of the proposed scheme, called Tabu Continuous Ant Colony System (TCACS), is similar to CACS while it borrows the concepts of tabu and promising lists from TS. The so called tabu balls prevent the ants to choose their destination within the tabu regions. A new strategy was also proposed to dynamically tune the radius of the tabu balls and also to handle the variable correlations. TCACS was tested over a set of standard test functions to tune its control parameters and to compare its results with those of other meta-heuristics. The overall results show that the use of tabu and promising lists is beneficial regarding the accuracy and robustness of the method.

A graphical demonstration on the performance of TCACS was also made. It shows that as the algorithm progresses, the members of the tabu and promising points gather in the worst and the best regions, respectively. Also it can be observed how the radius of tabu balls varies between iterations. This prevents the ants to choose their destinations within the tabu areas and pushes them toward the promising points.

Appendix: List of test functions

Branin RCOS (RC) (2 variables):

$$RC(x_1, x_2) = \left(x_2 - \left(\frac{5}{(4\pi^2)}\right)x_1^2 + \left(\frac{5}{\pi}\right)x_1 - 6\right)^2 + 10\left(1 - \left(\frac{1}{8\pi}\right)\right)\cos(x_1) + 10,$$

search domain: $-5 < x_1 < 10$, $0 < x_2 < 15$; no local minimum; 3 global minima: $(x_1, x_2)^* = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$; $RC((x_1, x_2)^*) = 0.397887$.

B_2 (2 variables):

$$B_2(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - -0.4\cos(4\pi x_2) + 0.7,$$

search domain: $-100 < x_j < 100$; $j = 1, 2$; several local minima (exact number unspecified in usual literature); 1 global minimum: $(x_1, x_2)^* = (0, 0)$; $B_2((x_1, x_2)^*) = 0$.

Easom (ES) (2 variables):

$$ES(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp\big(-\big((x_1 - \pi)^2 + (x_2 - \pi)^2\big)\big),$$

search domain: $-100 < x_j < 100$; $j = 1, 2$; several local minima (exact number unspecified in usual literature); 1 global minimum: $(x_1, x_2)^* = (\pi, \pi)$; $ES((x_1, x_2)^*) = -1$.

Goldstein and Price (GP) (2 variables):

$$\begin{aligned} GP(x_1, x_2) &= \left[1 + (x_1 + x_2 + 1)^2 \times (19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \\ &\quad \times \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2)\right], \end{aligned}$$

search domain: $-2 < x_j < 2$, $j = 1, 2$; 4 local minima; 1 global minimum: $(x_1, x_2)^* = (-1, 0)$; $GP((x_1, x_2)^*) = 3$.

Shubert (SH) (2 variables):

$$SH(x_1, x_2) = \left\{\sum_{j=1}^{5} j \cos[(j+1)x_1 + j]\right\} \times \left\{\sum_{j=1}^{5} j \cos[(j+1)x_2 + j]\right\},$$

search domain: $-10 < x_j < 10$, $j = 1, 2$; 760 local minima; 18 global minima: $SH((x_1, x_2)^*) = -186.7309$.

De Joung (DJ) (3 variables):

$$DJ(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2,$$

search domain: $-5.12 < x_j < 5.12$; $j = 1, 3$; 1 single minimum (local and global): $(x_1, x_2, x_3)^* = (0, 0, 0)$; $DJ((x_1, x_2, x_3)^*) = 0$.

Hartmann ($H_{3,4}$) (3 variables):

$$H_{3,4}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right],$$

search domain: $0 < x_j < 1$, $j = 1, \ldots, 3$; 4 local minima: $\boldsymbol{p}_i = (p_{i1}, p_{i2}, p_{i3}) = i$th local minimum approximation; $f((\boldsymbol{p}_i)) \approx -c_i$; 1 global minimum: $\boldsymbol{x}^* = (0.11, 0.555, 0.855)$; $H_{3,4}(\boldsymbol{x}^*) = -3.86278$.

i	a_{ij}			c_i	p_{ij}		
1	3.0	10.	30.	1.0	0.3689	0.1170	0.2673
2	0.1	10.	35.	1.2	0.4699	0.4387	0.7470
3	3.0	10.	30.	3.0	0.1091	0.8732	0.5547
4	0.1	10.	35.	3.2	0.0381	0.5743	0.8828

Shekel (S4;n) (4 variables):

$$S_{4,n}(x) = -\sum_{i=1}^{n}\left[(x - a_i)^T(x - a_i) + c_i\right]^{-1}; \quad x = (x_1, x_2, x_3, x_4)^T;$$

$$a_i = \left(a_i^1, a_i^2, a_i^3, a_i^4\right)^T,$$

3 functions $S_{4,n}$ were considered: $S_{4,5}$, $S_{4,7}$ and $S_{4,10}$; search domain: $0 < x_j < 10$, $j = 1, \ldots, 4$; n local minima ($n = 5$, 7 or 10): $\boldsymbol{a}_i^T = i$th local minimum approximation; $S_{4,n}((\boldsymbol{a}_i^T)) \approx -1/c_i$;

$S_{4,5}$	$n = 5$	5 minima with 1 global minimum: $S_{4,5}(\boldsymbol{x}^*) = -10.1532$
$S_{4,7}$	$n = 7$	7 minima with 1 global minimum: $S_{4,5}(\boldsymbol{x}^*) = -10.40294$
$S_{4,10}$	$n = 10$	10 minima with 1 global minimum: $S_{4,5}(\boldsymbol{x}^*) = -10.53641$

i	a_{ij}				c_i
1	4.	4.	4.	4.	.1
2	1.	1.	1.	1.	.2
3	8.	8.	8.	8.	.2
4	6.	6.	6.	6.	.4
5	3.	7.	3.	7.	.4
6	2.	9.	2.	9.	.6
7	5.	5.	3.	3.	.3
8	8.	1.	8.	1.	.7
9	6.	2.	6.	2.	.5
10	7.	3.6	7.	3.6	.5

Hartmann (H6; 4) (6 variables):

$$H_{6,4}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right],$$

search domain: $0 < x_j < 1$; $j = 1, \ldots, 6$; 4 local minima: $\boldsymbol{p}_i = (p_{i1}, \ldots, p_{i6}) = i$th local minimum approximation; $f(p_i) \approx -c_i$; 1 global minimum: $H_{6,4}(\boldsymbol{x}^*) = -3.86278$.

i	a_{ij}						c_i	p_{ij}					
1	10	3	17	3.5	17	8	1.0	0.1312	0.1696	0.5569	0.0124	0.8283	0.58886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3.0	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Rosenbrock (Rn) (n variables):

$$R_n(x) = \sum_{j=1}^{n-1}\left[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2\right],$$

3 functions were considered: R_2; R_5; R_{10}; search domain: $-5 < x_j < 10$, $j = 1, \ldots, n$, several local minima (exact number unspecified in usual literature); 1 global minimum: $\boldsymbol{x}^* = (1, \ldots, 1)$, $R_n(\boldsymbol{x}^*) = 0$.

Zakharov (Zn) (n variables):

$$Z_n(x) = \left(\sum_{j=1}^{n} x_j^2\right) + \left(\sum_{j=1}^{n} 0.5 j x_j\right)^2 + \left(\sum_{j=1}^{n} 0.5 j x_j\right)^4,$$

3 functions were considered: Z_2; Z_5; Z_{10}; search domain: $-5 < x_j < 10$, $j = 1, \ldots, n$, several local minima (exact number unspecified in usual literature); 1 global minimum: $\boldsymbol{x}^* = (0, \ldots, 0)$, $Z_n(\boldsymbol{x}^*) = 0$.

Martin & Gaddy (MG) (2 variables):

$$MG(x) = (x_1 - x_2)^2 + \big((x_1 + x_2 - 10)/3\big)^2,$$

search domain: $-20 < x_j < 20, j = 1, 2$; 1 global minimum: $\boldsymbol{x}^* = (5, 5)$, $MG(\boldsymbol{x}^*) = 0$.

Sphere Model (SM) (6 variables)

$$SM(x) = \sum_{j=1}^{n} x_j^2,$$

search domain: $-5.12 < x_j < 5.12$, $j = 1, \ldots, 6$; 1 global minimum: $\boldsymbol{x}^* = (5, 5)$, $MG(\boldsymbol{x}^*) = 0$.

Griewangk (Gr_n) (n variables):

$$Gr_n(x) = \left(0.1 + \left(\sum_{i=1}^{n} (x_i^2/4000) - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1\right)\right)^{-1},$$

Gr_{10} were used; search domain: $-5.12 < x_i < 5.12$, $i = 1, \ldots, 10$; 1 global minimum at origin; $Gr_{10}(\boldsymbol{x}^*) = 0.475072$.

References

1. Glover, F.: Tabu search: Part I. ORSA J. Comput. **3**, 190–206 (1989)
2. Glover, F.: Tabu search: Part II. ORSA J. Comput. **1**, 4–32 (1990)
3. Hu, N.: Tabu search method with random moves for globally optimal design. Int. J. Numer. Methods Eng. **35**, 1055–1070 (1992)
4. Cvijovic, D., Klinowski, J.: Taboo search: an approach to the multiple minima problem. Science **667**, 664–666 (1995)
5. Battiti, R., Tecchiolli, G.: The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. Ann. Oper. Res. **63**, 53–188 (1996)

6. Siarry, P., Berthiau, G.: Fitting of tabu search to optimize functions of continuous variables. Int. J. Numer. Methods Eng. **40**, 2449–2457 (1997)
7. Chelouah, R., Siarry, P.: Enhanced continuous tabu search: an algorithm for the global optimization of multiminima functions. In: Voss, S., Martello, S., Osman, I.H., Roucairol, C. (eds.) Meta-Heuristics, Advances and Trends in Local Search Paradigms for Optimization, vol. 4, pp. 49–61. Kluwer Academic, Dordrecht (1999)
8. Chelouah, R., Siarry, P.: Tabu search applied to global optimization. Eur. J. Oper. Res. **123**, 256–270 (2000)
9. Dorigo, M.: Optimization, learning and natural algorithms. Ph.D. thesis, Univ. of Milan, Milan (1992)
10. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life, pp. 134–142. Elsevier, Amsterdam (1992)
11. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B **1**, 29–41 (1996)
12. Stutzle, T., Hoos, H.: The MAX–MIN ant system and local search for the traveling salesman problem. In: Proceedings of IEEE International Conference on Evolutionary Computation and Evolutionary Programming, pp. 309–314 (1997)
13. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**, 53–66 (1997)
14. Gambardella, L.M., Dorigo, M.: Ant-Q: a reinforcement learning approach to the traveling salesman problem. In: Proceedings of the Twelfth International Conference on Machine Learning, Palo Alto, pp. 252–260 (1995)
15. Costa, D., Hertz, A.: Ants can colour graphs. J. Oper. Res. Soc. **48**, 295–305 (1997)
16. Dorigo, M., Caro, G.D., Gambardella, L.M.: Ant algorithms for discrete optimization. Artif. Life **3**, 137–172 (1999)
17. Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. Future Gener. Comput. Syst. **16**, 851–871 (2000)
18. Wodrich, M., Bilchev, G.: Cooperative distributed search: the ants' way. Control Cybern. **26**(3), 413–445 (1997)
19. Bilchev, G., Parmee, I.C.: The ant colony metaphor for searching continuous design spaces. Lect. Notes Comput. Sci. **993**, 25–39 (1995)
20. Monmarché, N., Venturini, G., Slimane, M.: On how Pachycondyla apicalis ants suggest a new search algorithm. Future Gener. Comput. Syst. **16**, 937–946 (2000)
21. Dréo, J., Siarry, P.: Continuous interacting ant colony algorithm based on dense heterarchy. Future Gener. Comput. Syst. **20**, 841–856 (2004)
22. Dréo, J., Siarry, P.: A new ant colony algorithm using the heterarchical concept aimed at optimization of multi-minima continuous functions. Lect. Notes Comput. Sci. **2463**, 216–221 (2002)
23. Ling, C., Jie, S., Ling, O., Hongjian, C.: A method for solving optimization problems in continuous space using ant colony algorithm. Lect. Notes Comput. Sci. **2463**, 288–289 (2002)
24. Jun, L.Y., Jun, W.T.: An adaptive ant colony system algorithm for continuous-space optimization problems. J. Zhejiang Univ. Sci. **1**, 40–46 (2003)
25. Pourtakdoust, S.H., Nobahari, H.: An extension of ant colony system to continuous optimization problems. Lect. Notes Comput. Sci. **3172**, 294–301 (2004)
26. Socha, K.: ACO for continuous and mixed-variable optimization. Lect. Notes Comput. Sci. **3172**, 25–36 (2004)
27. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. IRIDIA Technical Report, TR/IRIDIA/2005-037
28. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. Eur. J. Oper. Res. **185**, 1155–1173 (2008)
29. Nobahari, H., Pourtakdoust, S.H.: Optimization of fuzzy rule bases using continuous ant colony system. In: Proceedings of the First International Conference on Modeling, Simulation and Applied Optimization, Sharjah, U.A.E., Paper No. 243 (2005)
30. Nobahari, H., Pourtakdoust, S.H.: Optimal fuzzy CLOS guidance law design using ant colony optimization. Lect. Notes Comput. Sci. **3777**, 95–106 (2005)
31. Nobahari, H., Nabavi, S.Y., Pourtakdoust, S.H.: Aerodynamic shape optimization of unguided projectiles using ant colony optimization. In: Proceedings of ICAS 2006, Hamburg, Germany, 3–8 Sept. 2006

32. Chelouah, R., Siarry, P.: A continuous genetic algorithm designed for the global optimization of multimodal functions. J. Heuristics **6**, 191–213 (2000)
33. Siarry, P., Berthiau, G., Durbin, F., Haussy, J.: Enhanced simulated annealing for globally minimizing functions of many continuous variables. ACM Trans. Math. Softw. **23**(2), 209–228 (1997)
34. Chelouah, R., Siarry, P.: Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. Eur. J. Oper. Res. **148**, 335–348 (2003)