Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 188 (2007) 129–142
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

# Particle swarm and ant colony algorithms hybridized for improved continuous optimization

P.S. Shelokar [a,*], Patrick Siarry [b], V.K. Jayaraman [a], B.D. Kulkarni [a]

[a] Chemical Engineering & Process Development Division, National Chemical Laboratory, Dr. Homi Bhabha Road, Pune 411008, India
[b] Université Paris 12, LISSI, 61, Avenue du Généralde Gaulle, 94010 Créteil, France

**Abstract**

This paper proposes PSACO (particle swarm ant colony optimization) algorithm for highly non-convex optimization problems. Both particle swarm optimization (PSO) and ant colony optimization (ACO) are co-operative, population-based global search swarm intelligence metaheuristics. PSO is inspired by social behavior of bird flocking or fish schooling, while ACO imitates foraging behavior of real life ants. In this study, we explore a simple pheromone-guided mechanism to improve the performance of PSO method for optimization of multimodal continuous functions. The proposed PSACO algorithm is tested on several benchmark functions from the usual literature. Numerical results comparisons with different metaheuristics demonstrate the effectiveness and efficiency of the proposed PSACO method.


*Keywords:* Particle swarm optimization; Ant colony; Metaheuristics; Global optimization; Multimodal continuous functions

## 1. Introduction

Swarm intelligence metaheuristics, namely, particle swarm optimization (PSO) and ant colony optimization (ACO) are recently proved to be successful approaches to solve complex optimization problems. PSO algorithm, whose concept began as a simulation of a simplified social milieu, is a powerful optimization technique for solving multimodal continuous optimization problems [1–3]. While ACO approaches that imitate foraging behavior of real life ants, are known to be efficient and robust for solution of combinatorial optimization problems [4–7]. In this study, one of the ways of integrating the concepts of these two swarm intelligence metaheuristics for solution of multimodal continuous problems is explored.

Consider the following continuous global optimization problem as

$$(F) = \begin{cases} \min\ f(x) \\ \text{s.t. } S \subset \Re^n, \end{cases} \tag{1}$$

* Corresponding author.
*E-mail addresses:* ps.shelokar@gmail.com (P.S. Shelokar), siarry@univ-paris12.fr (P. Siarry).

0096-3003/$ - see front matter 
doi:10.1016/j.amc.2006.09.098

where $f(x)$ is a highly non-convex function, which will in general have several local minima. In this study, we consider only minimization of problem ($F$), without loss of generality, for maximization, $\max f(x) = -\min[-f(x)]$. $x$ is a real valued vector in a bounded space $S = \{x \in \Re^n, a_j \leqslant x_j \leqslant b_j; j = 1, \ldots, n\} \subset \Re^n$. For solution of problem ($F$) in Eq. (1), until now there exist few hybrid implementations of PSO algorithm with other local search methods. The first hybrid algorithm is called, NM-PSO [8] (Nelder–Mead–particle swarm optimization), which comprises NM method at the top level, and PSO at the lower level. In NM-PSO method $3n + 1$ particles are initially randomly generated, where, $n$ is the size of solution vector. 'Best' $n + 1$ solutions of total solutions are sent as initial points to NM method and remaining (worst) $2n$ solutions to PSO method to generate a total of $3n + 1$ new solutions. Global best particle is selected from $3n + 1$ particles and neighborhood best particles by evenly dividing the $2n$ particles into $n$ neighborhoods according to objective function value and velocity update is applied on the $2n$ particles. CPSO [9] (chaotic particle swarm optimization) algorithm applies PSO to perform global exploration and chaotic local search to perform local search on the solutions produced in the global exploration process.

This paper proposes improved particle swarm optimization hybridized with an ant colony approach, called PSACO (particle swarm ant colony optimization), for optimization of multimodal continuous functions. The proposed method applies PSO for global optimization and the idea of ant colony approach to update positions of particles to attain rapidly the feasible solution space. The implementation of PSACO algorithm consists of two stages. In the first stage, it applies PSO, while ACO is implemented in the second stage. ACO works as a local search, wherein, ants apply pheromone-guided mechanism to update the positions found by the particles in the earlier stage. The implementation of ACO in the second stage of PSACO is based on the studies by Angeline [10] which showed that (1) PSO discovers reasonable quality solutions much faster than other evolutionary algorithms, and (2) if the swarm is going to be in equilibrium, the evolution process will be stagnated as time goes on. Thus, PSO does not possess the ability to improve upon the quality of the solutions as the number of generations is increased. The proposed PSACO method is tested on several widely used benchmark multimodal continuous functions. Numerical results are compared with the some other hybrid PSO methods and non-PSO methods available in the usual literature. The performance study demonstrates the effectiveness and efficiency of the proposed PSACO approach.

The remaining content of this paper is organized as follows: In Section 2, particle swarm optimization algorithm and the concept of ant colony method are briefly explained before delineating the proposed PSACO algorithm. The computational results are discussed in Section 3 and conclusions and directions of future work are given in Section 4. In Appendix A, several widely used test functions are given. They are applied in this paper to study the performance of the proposed PSACO method.

## 2. Improved particle swarm optimization

To make the paper self-explanatory, before actually proposing the improved particle swarm optimization using ant colony approach, the characteristics of particle swarm optimization algorithm and the principle of an ant colony approach are briefly explained in the following two sections as:

### 2.1. Particle swarm optimization (PSO)

PSO is a population-based, co-operative search metaheuristic introduced by Kennedy and Eberhart [11]. In PSO, candidate solutions of a population, called particles, coexist and evolve simultaneously based on knowledge sharing with neighboring particles. While flying through the problem search space, each particle generates a solution using directed velocity vector. Each particle modifies its velocity to find a better solution (position) by applying its own flying experience (i.e. memory having best position found in the earlier flights) and experience of neighboring particles (i.e. best-found solution of the population). Particles update their positions and velocities as shown below:

$$v_{t+1}^i = w_t v_t^i + c_1 r_1 (p_t^i - x_t^i) + c_2 r_2 (p_t^g - x_t^i), \tag{2}$$

$$x_{t+1}^i = x_t^i + v_{t+1}^i, \tag{3}$$

where $x_t^i$ represents the current position of particle $i$ in solution space and subscript $t$ indicates an iteration count; $p_t^i$ is the best-found position of particle $i$ up to iteration count $t$ and represents the cognitive contribution to the search velocity $v_t^i$. Each component of $v_t^i$ can be clamped to the range $[-v_{\max}, v_{\max}]$ to control excessive roaming of particles outside the search space; $p_t^g$ is the global best-found position among all particles in the swarm up to iteration count $t$ and forms the social contribution to the velocity vector; $r_1$ and $r_2$ are random numbers uniformly distributed in the interval $(0,1)$, while $c_1$ and $c_2$ are the cognitive and social scaling parameters, respectively; $w_t$ is the particle inertia, which is reduced dynamically to decrease the search area in a gradual fashion [12]. The variable $w_t$ is updated as

$$w_t = (w_{\max} - w_{\min}) * \frac{(t_{\max} - t)}{t_{\max}} + w_{\min}, \tag{4}$$

where, $w_{\max}$ and $w_{\min}$ denote the maximum and minimum of $w_t$ respectively; $t_{\max}$ is a given number of maximum iterations. Particle $i$ flies toward a new position according to Eqs. (2) and (3). In this way, all particles $P$ of the swarm find their new positions and apply these new positions to update their individual best $p_t^i$ points and global best $p_t^g$ of the swarm. This process is repeated until iteration count $t = t_{\max}$ (a user-defined stopping criterion is reached). The pseudo-code of PSO is given in Fig. 1, where, $P$ denotes the number of particles in the population; $f(x_t^i)$ represents the objective function value of particle $i$ at position $x$, while $f_t^{\text{best}}(x_t^{\text{best}})$ represents the best function value in the population of solutions $P$ at iteration count $t$.

step 1. Initialize Optimization.

step 1.1 Initialize algorithm constants, $t_{\max}$, $P$.

step 1.2 Initialize randomly all particles positions $x_t^i$ and velocities $v_t^i$.

step 1.3 Evaluate objective function value as $f\left(x_t^i\right)$ in equation (1).

step 1.4 Assign best positions $p_t^i = x_t^i$ with $f\left(p_t^i\right) = f\left(x_t^i\right)$, $i = 1,\ldots,P$.

step 1.5 Find $f_t^{best}\left(p_t^{best}\right) = \min\left\{f\left(p_t^1\right),\ldots,f\left(p_t^i\right),\ldots,f\left(p_t^P\right)\right\}$ and initialize $p_t^g = p_t^{best}$ and $f\left(p_t^g\right) = f^{best}\left(p_t^{best}\right)$.

step 2. Perform Optimization

**while** $(t \le t_{\max})$

step 2.1 Update particle velocity $v_t^i$ and position $x_t^i$, according to equation (2) and equation (3) of all $P$ particles.

step 2.2 Evaluate objective function value as $f\left(x_t^i\right)$ in equation (1).

step 2.3 Update particle best position if $f\left(p_t^i\right) > f\left(x_t^i\right)$ then $p_t^i = x_t^i$ with $f\left(p_t^i\right) = f\left(x_t^i\right)$.

step 2.4 Find $f_t^{best}\left(p_t^{best}\right) = \min\left\{f\left(p_t^1\right),\ldots,f\left(p_t^i\right),\ldots,f\left(p_t^P\right)\right\}$; if $f\left(p_t^g\right) > f\left(p_t^{best}\right)$ then $p_t^g = p_t^{best}$ and $f\left(p_t^g\right) = f^{best}\left(p_t^{best}\right)$.

step 2.5 Increment iteration count $t$=$t$+1.

**end while**

step 3. Report best solution $p^g$ of the swarm with objective function value $f\left(p^g\right)$.

Fig. 1. Pseudo-code for PSO algorithm.

### 2.2. Ant colony optimization (ACO)

ACO is a multiagent approach that simulates the foraging behavior of ants for solving difficult combinatorial optimization problems, such as, the traveling salesman problem and the quadratic assignment problem [5]. Ants are social insects whose behavior is directed more toward the survival of the colony as a whole than that of a single individual of the colony. An important and interesting behavior of an ant colony is its indirect co-operative foraging process. While walking from food sources to the nest and vice versa, ants deposit a substance, called pheromone on the ground and form a pheromone trail. Ants can smell pheromone, when choosing their way, they tend to choose, with high probability, paths marked by strong pheromone concentrations (shorter paths). Also, other ants can use pheromone to find the locations of food sources found by their nest mates. In fact, ACO simulates the optimization of ant foraging behavior. Recently, there are few adaptations of ACO for solution of continuous optimization problems [13,14]. In this work, a simple pheromone-guided search mechanism of ant colony is implemented which acts locally to synchronize positions of the particles of PSO to quickly attain the feasible domain of objective function.

step 1. Initialize Optimization.

step 1.1 Initialize constants for PSO and ACO processes, $P$, $t_{\max}$.

step 1.2 Initialize randomly all particles positions $x_t^i$ and velocities $v_t^i$.

step 1.3 Evaluate objective function value as $f\left(x_t^i\right)$ in equation (1).

step 1.4 Assign best positions $p_t^i = x_t^i$ with $f\left(p_t^i\right) = f\left(x_t^i\right)$, $i = 1,\ldots,P$.

step 1.5 Find $f_t^{best}\left(p_t^{best}\right) = \min\left\{f\left(p_t^1\right),\ldots,f\left(p_t^i\right),\ldots,f\left(p_t^P\right)\right\}$ and initialize $p_t^g = p_t^{best}$ and $f\left(p_t^g\right) = f^{best}\left(p_t^{best}\right)$.

step 2. Perform Optimization

**while** $(t \le t_{\max})$

step 2.1 Update particle velocity $v_t^i$ and position $x_t^i$, according to equation (2) and equation (3) of all $P$ particles.

step 2.2 Evaluate objective function value as $f\left(x_t^i\right)$ in equation (1).

step 2.3 Generate $P$ solutions $z_t^i$ using equation (5).

step 2.4 Evaluate objective function value as $f\left(z_t^i\right)$ in equation (1) and if $f\left(z_t^i\right) < f\left(x_t^i\right)$ then $f\left(x_t^i\right) = f\left(z_t^i\right)$ and $x_t^i = z_t^i$.

step 2.5 Update particle best position if $f\left(p_t^i\right) > f\left(x_t^i\right)$ then $f\left(p_t^i\right) = f\left(x_t^i\right)$ and $p_t^i = x_t^i$.

step 2.6 Find $f_t^{best}\left(p_t^{best}\right) = \min\left\{f\left(p_t^1\right),\ldots,f\left(p_t^i\right),\ldots,f\left(p_t^P\right)\right\}$; if $f\left(p_t^g\right) > f\left(p_t^{best}\right)$ then $f\left(p_t^g\right) = f^{best}\left(p_t^{best}\right)$ and $p_t^g = p_t^{best}$.

step 2.7 Increment iteration count $t$=$t$+1;

**end while**

step 3. Report best solution $p^g$ of the swarm with objective function value $f\left(p^g\right)$.

Fig. 2. Pseudo-code for PSACO algorithm.

### 2.3. Hybridization of particle swarm optimization with an ant colony approach

This section describes the implementation of proposed improvement in particle swarm optimization using an ant colony approach. The proposed method, called, PSACO (particle swarm ant colony optimization) is based on the common characteristics of both PSO and ACO algorithms, like, survival as a swarm (colony) by coexistence and cooperation, individual contribution to food searching by a particle (an ant) by sharing information locally and globally in the swarm (colony) between particles (ants), etc. The implementation of PSACO algorithm consists of two stages. In the first stage, it applies PSO, while ACO is implemented in the second stage. ACO works as a local search, wherein, ants apply pheromone-guided mechanism to refine the positions found by particles in the PSO stage. In PSACO, a simple pheromone-guided mechanism of ACO is proposed to apply as local search. The proposed ACO algorithm handles $P$ ants equal to the number of particles in PSO. Each ant $i$ generates a solution $z_t^i$ around $p_t^g$ the global best-found position among all particles in the swarm up to iteration count $t$ as

$$z_t^i = \mathcal{N}\left(p_t^g, \sigma\right). \tag{5}$$

In Eq. (5), we generate components of solution vector $z_t^i$, which satisfy Gaussian distributions with mean $p_t^g$ and standard deviation $\sigma$, where, initially at $t = 1$ value of $\sigma = 1$ and is updated at the end of each iteration as $\sigma = \sigma \times d$, where, $d$ is a parameter in $(0.25, 0.997)$ and if $\sigma < \sigma_{\min}$ then $\sigma = \sigma_{\min}$, where, $\sigma_{\min}$ is a parameter in $(10^{-2}, 10^{-4})$. Compute objective function value $f(z_t^i)$ using $z_t^i$ and replace position $x_t^i$ the current position of particle $i$ in the swarm if $f(z_t^i) < f(x_t^i)$ as $x_t^i = z_t^i$ and $f(x_t^i) = f(z_t^i)$. This simple pheromone-guided mechanism considers, there is highest density of trails (single pheromone spot) at the global best solution $p_t^g$ of the swarm at any iteration $t + 1$ in each stage of ACO implementation and all ants $P$ search for better solutions in the neighborhood of the global best solution. In the beginning of the search process, ants explore larger search area in the neighborhood of $p_t^g$ due to the high value of standard deviation $\sigma$ and intensify the search around $p_t^g$ as the algorithm progresses. Thus, ACO helps PSO process not only to efficiently perform global exploration for rapidly attaining the feasible solution space but also to effectively reach optimal or near optimal solution.

The pseudo-code of PSACO method is given in Fig. 2. The algorithm starts with initializing parameters of both PSO and ACO methods. The first stage consists of PSO, which generates $P$ solutions using Eqs. (2) and (3). Objective function values are computed as $f(x_t^i)$. ACO is applied in the second stage to update the positions of particles in the swarm. This process is repeated until iteration count $t = t_{\max}$.

## 3. Results and discussion

The performance of the proposed PSACO algorithm for global optimization of continuous function is tested on several well-known benchmark multimodal problems. They are listed in Appendix A. All the test functions are multimodal in nature. Because of the characteristics, it is difficult to seek for the global minima. PSACO algorithm parameter settings used in all the simulations is given as: number of particles, $P = 10$; cognitive and social scaling parameters, $c_1 = 2$, $c_2 = 2$; maximum and minimum values of inertia weights, $w_{\max} = 0.7$, $w_{\min} = 0.4$; maximum number of iterations, $t_{\max} = 100 * n$, $n$ is the size of solution vector. To gauge the performance of the proposed PSACO method with the performance of some other existing hybrid PSO methods and non-PSO methods, several sets of experiments are carried out and results are given in Tables 1–6. Table 7 lists different global optimization methods used for performance analysis. The details of experiments conducted are given below.

### 3.1. Results with fixed number of iterations

To compare the performance of PSACO with the performance of CPSO [9] (chaotic particle swarm optimization), Table 1 reports the results produced by PSACO after 2000 function evaluations in terms of mean value of objective function over 50 independent runs. The performance of CPSO, PSO and GA algorithms reported in [9] is also shown in Table 1. From Table 1, it can be observed that the results obtained by both

Table 1
Fixed iteration experiments results with PSACO algorithm

| Function | Average objective function value | | | |
|---|---|---|---|---|
| | PSACO | CPSO | PSO | GA |
| *GP* | 3.0000 | 3.0000 | 4.6202 | 3.1471 |
| *BR* | 0.3979 | 0.3979 | 0.4960 | 0.4021 |
| $HR_{3,4}$ | −3.8628 | −3.8610 | −3.8572 | −3.8571 |
| $HR_{6,4}$ | −3.3198 | −3.1953 | −2.8943 | −3.0212 |
| $RA_2$ | −1.9999 | −1.9940 | −1.9702 | −1.9645 |
| *SH* | −186.7309 | −186.7274 | −180.3265 | −182.1840 |

Table 2
Results of robustness analysis for PSACO algorithm

| Function | PSACO | | CPSO | | PSO | | GA | |
|---|---|---|---|---|---|---|---|---|
| | SR | AVEN | SR | AVEN | SR | AVEN | SR | AVEN |
| *GP* | 100 | 157 | 100 | 192 | 98 | 1397 | 98 | 536 |
| *BR* | 100 | 156 | 100 | 154 | 94 | 743 | 92 | 1682 |
| $HR_{3,4}$ | 100 | 159 | 90 | 119 | 96 | 183 | 16 | 112 |
| $HR_{6,4}$ | 98 | 263 | 96 | 2551 | 26 | 3796 | 94 | 5727 |
| $RA_2$ | 100 | 112 | 98 | 653 | 100 | 1160 | 84 | 238 |
| *SH* | 100 | 307 | 100 | 360 | 98 | 1337 | 98 | 1516 |
| Overall | 99 | 192 | 97 | 672 | 85 | 1436 | 80 | 1635 |

Table 3
Performance of PSACO and NM-PSO on 17 test functions

| Function | Required accuracy | % Success rate | | Average number of function evaluations | | Average error | |
|---|---|---|---|---|---|---|---|
| | | PSACO | NM-PSO | PSACO | NM-PSO | PSACO | NM-PSO |
| *BR* | 1e−3 | 100 | 100 | 209 | 151 | 2.6185e−13 | 0.00003 |
| *ES* | 1e−3 | 100 | 100 | 254 | 165 | 0.0000000 | 0.00004 |
| *GP* | 1e−3 | 100 | 100 | 240 | 217 | 0.0000000 | 0.00003 |
| *B*2 | 1e−2 | 100 | 100 | 370 | 240 | 5.5511e−17 | 0.00003 |
| *SH* | 1e−2 | 100 | 100 | 534 | 400 | 1.0239e−09 | 0.00002 |
| $RS_2$ | 1e−3 | 100 | 100 | 327 | 339 | 1.7152e−10 | 0.00003 |
| $ZA_2$ | 1e−4 | 100 | 100 | 167 | 135 | 5.7061e−27 | 0.00003 |
| *DJ* | $1e^{-4}$ | 100 | 100 | 190 | 291 | 7.6900e−29 | 0.00005 |
| $H_{3,4}$ | 1e−4 | 100 | 100 | 592 | 271 | 2.0755e−11 | 0.00024 |
| $S_{4,5}$ | 1e−4 | 100 | 100 | 482 | 1177 | 5.8229e−11 | 0.00020 |
| $S_{4,7}$ | 1e−4 | 100 | 100 | 483 | 1130 | 1.8134e−10 | 0.00017 |
| $S_{4,10}$ | 1e−4 | 100 | 100 | 489 | 1179 | 3.0795e−10 | 0.00015 |
| $RS_5$ | 1e−2 | 100 | 100 | 517 | 3308 | 1.8538e−04 | 0.00560 |
| $ZA_5$ | 1e−4 | 100 | 100 | 516 | 1394 | 3.6352e−17 | 0.00026 |
| $H_{6,4}$ | 1e−3 | 96 | 100 | 529 | 1541 | 4.4789e−11 | 0.00250 |
| $GR_8$ | 1e−3 | 87 | 100 | 1081 | 1354 | 6.2311e−22 | 0.00041 |
| $GR_{10}$ | 1e−3 | 86 | 100 | 1634 | 2024 | 1.2197e−15 | 0.00058 |

Table 4
Summary of results reported in Table 3

| Dimension | % Success rate | | Average number of function evaluations | | Average error | |
|---|---|---|---|---|---|---|
| | PSACO | NM-PSO | PSACO | NM-PSO | PSACO | NM-PSO |
| $n < 4$ | 100 | 100 | 286 | 245 | 1.4946e−10 | 0.00006 |
| $n \geqslant 4$ | 97 | 100 | 703 | 1638 | 2.0598e−05 | 0.00123 |
| overall | 99 | 100 | 495 | 901 | 1.0299e−05 | 0.00061 |

Table 5
Summary of results reported in Table 3

| Function | Required accuracy | % Success rate | | | Average number of function evaluations | | | Average error | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | PSACO | CTSS | CHA | PSACO | CTSS | CHA | PSACO | CTSS | CHA |
| *BR* | 1e−3 | 100 | 100 | 100 | 209 | 125 | 295 | 2.6185e−13 | 0.005 | 0.0001 |
| *ES* | 1e−3 | 100 | 100 | 100 | 254 | 325 | 952 | 0.0000000 | 0.005 | 0.001 |
| *GP* | 1e−3 | 100 | 100 | 100 | 240 | 119 | 259 | 0.0000000 | 0.001 | 0.001 |
| *B*2 | 1e−2 | 100 | 100 | 100 | 370 | 98 | 132 | 5.5511e−17 | 5e−6 | 2e−7 |
| *SH* | 1e−2 | 100 | 100 | 100 | 534 | 283 | 345 | 1.0239e−09 | 0.001 | 0.005 |
| $RS_2$ | 1e−3 | 100 | 100 | 100 | 327 | 369 | 459 | 1.7152e−10 | 0.004 | 0.004 |
| $ZA_2$ | 1e−4 | 100 | 100 | 100 | 167 | 78 | 215 | 5.7061e−27 | 3e−7 | 3e−6 |
| *DJ* | 1e−4 | 100 | 100 | 100 | 190 | 155 | 371 | 7.6900e−29 | 0.0002 | 0.0002 |
| $H_{3,4}$ | 1e−4 | 100 | 100 | 100 | 592 | 225 | 492 | 2.0755e−11 | 0.005 | 0.005 |
| $S_{4,5}$ | 1e−4 | 100 | 75 | 85 | 482 | 538 | 598 | 5.8229e−11 | 0.007 | 0.007 |
| $S_{4,7}$ | 1e−4 | 100 | 77 | 83 | 483 | 590 | 620 | 1.8134e−10 | 0.001 | 0.01 |
| $S_{4,10}$ | 1e−4 | 100 | 74 | 81 | 489 | 555 | 635 | 3.0795e−10 | 0.001 | 0.015 |
| Overall | 2e−3 | 100 | 94 | 96 | 361 | 288 | 448 | 1.4700e−10 | 0.0025 | 0.004 |

PSACO and CPSO are much closer to the theoretical optima, and the proposed PSACO method is superior to all the methods in terms of the searching quality and derivation of results.

### 3.2. Robustness analysis of the proposed PSACO

Table 2 shows the robustness analysis results for the proposed PSACO. For comparison with CPSO [9], this paper has adopted two robustness measures, called, succeed ratio (SR) and average valid evaluation number (AVEN) given as follows:

$$\mathrm{SR} = \frac{100N_s}{50}, \tag{6}$$

$$\mathrm{AVEN} = \frac{\sum_{i=1}^{N_s} N_i}{N_s}, \tag{7}$$

where $N_s$ is the total number of success runs from 50 independent runs. $N_i$ is the function evaluation number of the $i$th success run. In case a solution obtained has the objective function value within 3.5% of its global optimum then it is called a success run and its function evaluation number is stored. Table 2 lists the SR and AVEN values produced by PSACO algorithm. For comparison purpose, Table 2 also shows the SR and AVEN values for CPSO, PSO and GA methods reported in [9]. From Table 2 it can be seen that PSACO can find global optima with very high probability for every function even with small function evaluation number. Besides, for those valid runs, it costs the least average function evaluation number. Thus, indirect co-operative search of ant colony approach has helped PSO to improve its effectiveness and reliability for complex numerical optimization.

### 3.3. Performance comparison of PSACO with NM-PSO

The results presented in Tables 3 and 4 are based on 100 independent runs of the proposed PSACO method on each of 17 test functions. All the test examples are multimodal functions and are given in Appendix A. In order to have comparable results, the accuracy was chosen based on the results of NM-PSO algorithm reported in [8]. Results reported in Table 3 are in terms of, the rate of successful minimizations, the average of the objective function evaluation numbers, and the average error. The average of the objective function evaluation numbers is evaluated in relation to only the 'successful minimizations.' The mean error is defined as the average of the difference between the best successful point found and the known global optimum, where only the 'successful minimizations' achieved by the algorithm are considered. The term 'successful minimizations' is the number of successful runs. In the successful runs the algorithm generates the best solution of

Table 6
Comparison of results of PSACO and non-PSO methods on 10–100 dimension functions

| Function | Required accuracy | % Success rate | | | | | Average number of function evaluations | | | | | Average error | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PSACO | NHGA | CHA | CGA | ECTS | PSACO | NHGA | CHA | CGA | ECTS | PSACO | NHGA | CHA | CGA | ECTS |
| $RS_{10}$ | 1e−2 | 95 | 100 | 83 | 80 | 85 | 1541 | 6257 | 14563 | 21563 | 15720 | 4e−04 | 3e−03 | 8e−3 | 0.02 | 0.02 |
| $RS_{50}$ | 1e−2 | 88 | 100 | 79 | 78 | 75 | 10433 | 44706 | 55356 | 78356 | 63210 | 3e−03 | 4e−03 | 5e−3 | 0.05 | 0.02 |
| $RS_{100}$ | 1e−2 | 86 | 100 | 72 | 66 | 75 | 24236 | 87582 | 124302 | 194302 | 162532 | 4e−03 | 4e−03 | 8e−3 | 0.07 | 0.05 |
| $ZA_{10}$ | 1e−3 | 100 | 100 | 100 | 100 | 100 | 2299 | 10734 | 4291 | 6991 | 4630 | 2e−08 | 1e−6 | 1e−6 | 1e−6 | 2e−7 |
| $ZA_{50}$ | 1e−3 | 100 | 100 | 100 | 100 | 100 | 47288 | 84327 | 75520 | 75201 | 63970 | 4e−06 | 1e−5 | 1e−5 | 1e−5 | 2e−7 |
| $ZA_{100}$ | 1e−3 | 100 | 100 | 100 | 100 | 100 | 145648 | 141430 | 95246 | 195246 | 152030 | 4e−05 | 1e−3 | 1e−3 | 1e−3 | 1e−3 |

Table 7
Global optimization methods used for performance analysis

| Method | Reference |
|---|---|
| Particle swarm ant colony optimization (PSACO) | This paper |
| Chaotic particle swarm optimization (CPSO) | [9] |
| Particle swarm optimization (PSO) | [9] |
| Genetic algorithm (GA) | [9] |
| Nelder–Mead–particle swarm optimization (NM-PSO) | [8] |
| Continuous hybrid algorithm (CHA) | [15] |
| Continuous tabu simplex search (CTSS) | [16] |
| Niche hybrid GA (NHGA) | [18] |
| Continuous GA (CGA) | [19] |
| Enhanced continuous tabu search (ECTS) | [20] |

'required accuracy', where, 'required accuracy' is a given maximum value to consider that the algorithm has successfully found the solution. It is calculated as the absolute difference in the best solution found and the global optimum. The results obtained by NM-PSO method are also given in Table 3. Results reported in Table 3 show that NM-PSO method seems to converge more quickly in cases of lower dimensions ($n < 4$), and in cases of higher dimensions ($n \geqslant 4$) the proposed PSACO converges more effectively to the global optimum than the NM-PSO method. In order to gain a better knowledge of how PSACO algorithm performs under different problem sizes, a summary report of the three algorithms based on the results shown in Table 3 is tabulated in Table 4. The problems are further divided into two groups $n < 4$ and $n \geqslant 4$ based on their sizes. As shown in Table 4, for larger dimensions, PSACO algorithm presents better saving of function evaluations over NM-PSO algorithm. Regarding the average error, PSACO method finds solutions with higher accuracy than NM-PSO algorithm in both categories.

### 3.4. Additional comparison with non-PSO hybrid methods

The performance of PSACO algorithm has been further compared with two recently published hybrid algorithms, including, CHA [15] (continuous hybrid algorithm) and CTSS [16] (continuous tabu simplex search). The experimental results obtained for 12 test functions are given in Table 5, which reports the average number of function evaluations, the average of the objective function evaluation numbers, and the average error for 100 runs of each function. From Table 5, it can be observed that PSACO obtained solutions with highest average error accuracy of 2e−10 with 100% performance on all examples. In view of the algorithm efficiency and effectiveness in term of this 'smaller' set of test functions, it can be anticipated that PSACO approach remains quite competitive as compared to the other existing methods.

### 3.5. Experiments to study the effect of high dimension

To investigate the effect of high dimension on searching quality of the proposed PSACO method, Ackley function is chosen for the test whose global minimum is $f(x) = 0$ at $x = 0$. This benchmark example has thousands of minima in the region and is very difficult to be optimized [17]. Liu et al. [9] also used this function to investigate the curse of dimensionality on the CPSO algorithm, which was run 10 times for Ackley function with different dimensions. A total number of function evaluations were set as $4 \times 10^5$. Fig. 3 illustrates the varying curve of the mean objective value obtained by PSACO in 10 independent runs with respect to different dimensions. The PSACO was also run for a maximum of $4 \times 10^5$ function evaluations. From Fig. 3, it can be seen that the performance of PSACO algorithm is very good even for high dimension of problem. For comparison purpose, the average objective value reported by PSACO method is close to three units for Ackley function of dimension equal to 1000 while that reported by CPSO, PSO and GA methods in [9] is more than six units. The effect of dimension is also tested on two widely used difficult benchmark examples, viz., Rosenbrock and Zakharov with dimension of 10–100. Table 6 reports the average number of function evaluations, the average of the objective function evaluation numbers, and the average error for 100 runs of each function.

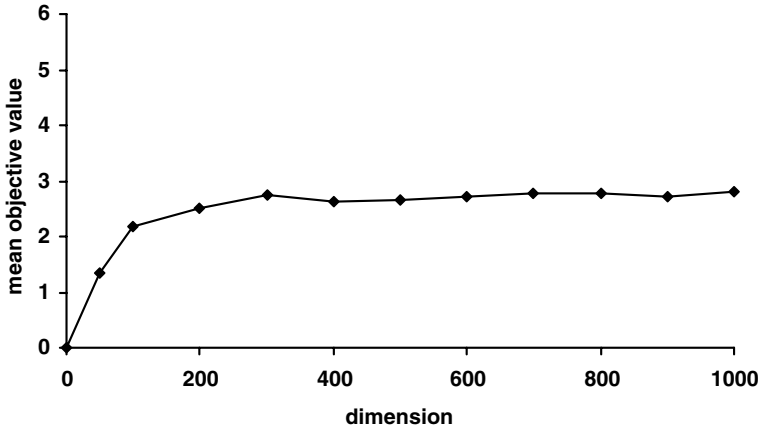


Fig. 3. Performance of PSACO in term of mean objective value for different dimension of Ackley function.

For comparison purpose, the results produced by other methods, CHA [15], NHGA [18] (Niche Hybrid GA), CGA [19] (continuous GA) and ECTS [20] (enhanced continuous tabu search) are also given in Table 6. The results show that PSACO method is a viable alternative to other metaheuristics to solve even moderate to high dimension multimodal functions.

## 4. Conclusions

This paper has proposed PSACO (particle swarm ant colony optimization) algorithm for solution of highly non-convex problems. A simple pheromone-guided local search is implemented to improve the performance of particle swarm optimization algorithm. The results show that ACO helps PSO process not only to efficiently perform global exploration for rapidly attaining the feasible solution space but also to effectively reach optimal or near optimal solution. The comparisons of numerical results with other hybrid PSO and non-PSO methods show that there is a scope of research in hybridizing swarm intelligence methods to solve difficult continuous optimization problems.

## Appendix A

*Goldstein–Price* (*GP*) (*2 variables*)

$$GP(x_1,x_2) = (1+(x_1+x_2+1)^2(19-14(x_1+x_2)+3(x_1^2+x_2^2)+6x_1x_2)) \times (30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)),$$

- search domain: $-2 \leqslant x_1, x_2 \leqslant 2$;
- four local minima;
- global minimum: $(x_1,x_2) = (0,-1)$, $GP(x_1,x_2) = 3$.

*Branin* (*BR*) (*2 variables*)

$$BR(x_1,x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)\cos(x_1) + 10,$$

- search domain: $-5 \leqslant x_1 \leqslant 10$, $0 < x_2 < 15$;
- no local minimum;
- three global minima: $(-\pi, 12.275)$, $(\pi, 2.275)$, $(3\pi, 2.475)$;
- $BR(x_1,x_2) = 5/4\pi$.

*Rastrigin* (*RA*) (*2 variables*)

$$RA(x_1, x_2) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2),$$

- search domain: $-1 \leqslant x_1,\ x_2 \leqslant 1$;
- 50 local minima;
- global minimum: $(x_1, x_2) = (0, 0)$, $RA(x_1, x_2) = -2$.

*Shubert* (*SH*) (*2 variables*)

$$SH(x_1, x_2) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right),$$

- search domain: $-10 \leqslant x_1,\ x_2 \leqslant 10$;
- 760 local minima;
- 18 global minima with $SH(x_1, x_2) = 0$.

*Hartman* ($HR_{3,4}$) (*3 variables*)

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

- search domain: $0 \leqslant x_j \leqslant 1,\ j = 1, \ldots, 3$;
- four local minima;
- one global minimum: $x = (0.11, 0.555, 0.855)$; $HR_{3,4}(x) = -3.86278$.

| $i$ | $\alpha_{i1}$ | $\alpha_{i2}$ | $\alpha_{i3}$ | $c_i$ | $p_{i1}$ | $p_{i2}$ | $p_{i3}$ |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 10 | 30 | 1 | 0.3689 | 0.1170 | 0.2673 |
| 2 | 0.1 | 10 | 35 | 1.2 | 0.4699 | 0.4387 | 0.7470 |
| 3 | 3 | 10 | 30 | 3 | 0.1091 | 0.8742 | 0.5547 |
| 4 | 0.1 | 10 | 35 | 3.2 | 0.03815 | 0.5743 | 0.8828 |

*Hartman* ($HR_{6,4}$) (*6 variables*)

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

- search domain: $0 \leqslant x_j \leqslant 1,\ j = 1, \ldots, 6$;
- four local minima;
- one global minimum: $x = (0.201, 0.150, 0.477, 0.275, 0.311, 0.657)$; $HR_{6,4}(x) = -3.32$.

| $i$ | $\alpha_{i1}$ | $\alpha_{i2}$ | $\alpha_{i3}$ | $\alpha_{i4}$ | $\alpha_{i5}$ | $\alpha_{i6}$ | $c_i$ |
|---|---|---|---|---|---|---|---|
| 1 | 10.0 | 3.0 | 17.0 | 3.5 | 1.7 | 8.0 | 1.0 |
| 2 | 0.05 | 10.0 | 17.0 | 0.1 | 8.0 | 14.0 | 1.2 |
| 3 | 3.0 | 3.5 | 1.7 | 10.0 | 17.0 | 8.0 | 3.0 |
| 4 | 17.0 | 8.0 | 0.05 | 10.0 | 0.1 | 14.0 | 3.2 |

| $i$ | $p_{i1}$ | $p_{i2}$ | $p_{i3}$ | $p_{i4}$ | $p_{i5}$ | $p_{i6}$ |
|---|---|---|---|---|---|---|
| 1 | 0.1312 | 0.1696 | 0.5569 | 0.0124 | 0.8283 | 0.5886 |
| 2 | 0.2329 | 0.4135 | 0.8307 | 0.3736 | 0.1004 | 0.9991 |
| 3 | 0.2348 | 0.1451 | 0.3522 | 0.2883 | 0.3047 | 0.6650 |
| 4 | 0.4047 | 0.8828 | 0.8732 | 0.5743 | 0.1091 | 0.0381 |

*De Joung* (*DJ*) (*3 variables*)

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

- search domain: $-5.12 \leqslant x_j \leqslant 5.12$, $j = 1, \ldots, n$;
- global minimum: $x = (0.0, 0.0, \ldots, 0.0)$; $DJ(x) = 0.0$.

*Ackley* ($AK_n$) (*n variables*)

$$AK_n(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e,$$

- 11 functions were considered: $AK_{50}, AK_{100}, AK_{200}, \ldots, ZA_{1000}$;
- search domain: $-32 \leqslant x_j \leqslant 32$, $j = 1, \ldots, n$;
- several local minima (exact number of local minima unspecified in the usual literature);
- one global minimum: $x = (0, \ldots, 0)$; $AK_n(x) = 0$.

*Rosenbrock* ($RS_n$) (*n variables*)

$$RS_n(x) = \sum_{j=1}^{n-1}[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2].$$

- Five functions were considered: $RS_2$, $RS_5$, $RS_{10}$, $RS_{50}$ and $RS_{100}$;
- search domain: $-5 \leqslant x_j \leqslant 10$, $j = 1, \ldots, n$;
- several local minima (exact number of local minima unspecified in the usual literature);
- one global minimum: $x = (1, \ldots, 1)$; $RS_n(x) = 0$.

*Zakharov* ($ZA_n$) (*n variables*)

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

- Five functions were considered: $ZA_2$, $ZA_5$, $ZA_{10}$, $ZA_{50}$ and $ZA_{100}$;
- search domain: $-5 \leqslant x_j \leqslant 10$, $j = 1, \ldots, n$;
- several local minima (exact number of local minima unspecified in the usual literature);
- one global minimum: $x = (0, \ldots, 0)$; $ZA_n(x) = 0$.

*Griewank* (*GRn*) (*n variables*)

$$GR_n(x) = \sum_{j=1}^{n} x_j^2 \Big/ 4000 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$$

- two functions were considered: $GR_8$, $GR_{10}$;
- search domain: $-300 \leqslant x_j \leqslant 600$, $j = 1, \ldots, n$;
- several local minima (exact number of local minima unspecified in the usual literature);
- one global minimum: $x = (0, \ldots, 0)$; $GR_n(x) = 0$.

*Shekel* ($S_4$, $n$) (*4 variables*)

$$S_{4,n}(x) = -\sum_{i=1}^{n}[(x-a_i)^{\mathrm{T}}(x-a_i)+c_i]^{-1},$$

$$x = (x_1, x_2, x_3, x_4); \quad a_i = (a_i^1, a_i^2, a_i^3, a_i^4)^{\mathrm{T}},$$

- three functions $S_4$, $n$ were considered: $S_{4,5}$, $S_{4,7}$, $S_{4,10}$;
- search domain: $0 \leqslant x_j \leqslant 10$, $j = 1, \ldots, n$;
- $n$ local minima ($n = 5, 7$, or 10): $a_i^{\mathrm{T}} = i$th local minimum approximation:
- $S_{4,n}(a_i^{\mathrm{T}}) \cong -1/c_i$;
- $S_{4,5}(n = 5)$5 minima with one global minimum: $S_{4,5}(x) = -10.1532$;
- $S_{4,7}(n = 7)$7 minima with one global minimum: $S_{4,7}(x) = -10.40294$;
- $S_{4,10}(n = 7)$7 minima with one global minimum: $S_{4,10}(x) = -10.53641$.

| $i$ | $a_i^{\mathrm{T}}$ | | | | $c_i$ |
|---|---|---|---|---|---|
| 1 | 4.0 | 4.0 | 4.0 | 4.0 | 0.1 |
| 2 | 1.0 | 1.0 | 1.0 | 1.0 | 0.2 |
| 3 | 8.0 | 8.0 | 8.0 | 8.0 | 0.2 |
| 4 | 6.0 | 6.0 | 6.0 | 6.0 | 0.4 |
| 5 | 3.0 | 7.0 | 3.0 | 7.0 | 0.4 |
| 6 | 2.0 | 9.0 | 2.0 | 9.0 | 0.6 |
| 7 | 5.0 | 5.0 | 3.0 | 3.0 | 0.3 |
| 8 | 8.0 | 1.0 | 8.0 | 1.0 | 0.7 |
| 9 | 6.0 | 2.0 | 6.0 | 2.0 | 0.5 |
| 10 | 7.0 | 3.6 | 7.0 | 3.6 | 0.5 |

*B2* (*2 variables*)

$$B2(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7,$$

- search domain: $-100 \leqslant x_j \leqslant 100$, $j = 1, \ldots, n$;
- several local minima (exact number of local minima unspecified in the usual literature);
- one global minimum: $(x_1, x_2) = (0, 0)$; $B2(x_1, x_2) = 0$.

*Easom* (*ES*) (*2 variables*)

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

- search domain: $-100 \leqslant x_j \leqslant 100$, $j = 1, \ldots, n$;
- several local minima (exact number of local minima unspecified in the usual literature);
- one global minimum: $(x_1, x_2) = (\pi, \pi)$; $B2(x_1, x_2) = -1$.

## References

[1] S. Paterlini, T. Krink, Differential evolution and particle swarm optimisation in partitional clustering, Computational Statistics & Data Analysis 50 (5) (2006) 1220–1247.

[2] Y. Dong, J. Tang, B. Xu, D. Wang, An application of swarm optimization to nonlinear programming, Computers & Mathematics with Applications 49 (11–12) (2005) 1655–1668.

[3] C.O. Ourique, E.C. Biscaia, J.C. Pinto, The use of particle swarm optimization for dynamical analysis in chemical processes, Computers & Chemical Engineering 26 (12) (2002) 1783–1793.

[4] P.S. Shelokar, V.K. Jayaraman, B.D. Kulkarni, An ant colony classifier system: application to some process engineering problems, Computers & Chemical Engineering 28 (9) (2004) 1577–1584.
[5] M. Dorigo, C. Blum, Ant colony optimization theory: a survey, Theoretical Computer Science 344 (2–3) (2005) 243–278.
[6] P.-Y. Yin, J.-Y. Wang, Ant colony optimization for the nonlinear resource allocation problem, Applied Mathematics & Computation 174 (2) (2006) 1438–1453.
[7] S.J. Shyu, B.M.T. Lin, T.-S. Hsiao, Ant colony optimization for the cell assignment problem in PCS networks, Computers & Operations Research 33 (6) (2006) 1713–1740.
[8] S.-K.S. Fan, Y.-C. Liang, E. Zahara, Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions, Engineering Optimization 36 (4) (2004) 401–418.
[9] B. Liu, L. Wang, Y.-H. Jin, F. Tang, D.-X. Huang, Improved particle swarm optimization combined with chaos, Chaos Solitons & Fractals 25 (2005) 1261–1271.
[10] P.J. Angeline, Evolutionary optimization versus particle swarm optimization: philosophy and performance difference, in: V.W. Porto et al. (Eds.), Proceedings of 7th Annual Conference on Evolutionary Programming, Lecture Notes in Computer Science, vol. 1447, Springer, Berlin, 1998, pp. 601–610.
[11] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[12] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, 1998.
[13] J. Dréo, P. Siarry, Continuous interacting ant colony algorithm based on dense heterarchy, Future Generation Computer Systems 20 (2004) 841–856.
[14] K. Socha, ACO for continuous and mixed-variable optimization, in: M. Dorigo et al. (Eds.), Lecture Notes in Computer Science, vol. 3172, Springer-Verlag, Berlin, 2004, pp. 25–36.
[15] R. Chelouah, P. Siarry, Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions, European Journal of Operational Research 148 (2003) 335–348.
[16] R. Chelouah, P. Siarry, A hybrid method combining continuous tabu search and Nelder–Mead simplex algorithms for the global optimization of multiminima functions, European Journal of Operational Research 161 (2005) 636–654.
[17] T. Bäck, Evolutionary Algorithms in Theory and Practice, Oxford University Press, New York, 1996.
[18] L. Wei, M. Zhao, A niche hybrid genetic algorithm for global optimization of continuous multimodal functions, Applied Mathematics & Computation 160 (2005) 649–661.
[19] R. Chelouah, P. Siarry, A continuous genetic algorithm designed for the global optimization of multimodal functions, Journal of Heuristics 6 (2000) 191–213.
[20] R. Chelouah, P. Siarry, Tabu search applied to global optimization, European Journal of Operational Research 123 (2000) 256–270.