Computing (2019) 101:1609–1632
https://doi.org/10.1007/s00607-018-0674-x

Generalized Ant Colony Optimizer: swarm-based meta-heuristic algorithm for cloud services execution

Ajay Kumar[1] · Seema Bawa[1]

Received: 1 May 2018 / Accepted: 8 November 2018 / Published online: 15 November 2018

Abstract
This work presents a swarm-based meta-heuristic technique known as Generalized Ant Colony Optimizer (GACO). It is a hybrid approach which consists of Simple Ant Colony Optimization and Global Colony Optimization concepts. The main concept behind GACO is the foraging behavior of ants. GACO operates in the following four phases: Creation of a new colony, search of nearest food location, balance the solution, and updating of pheromone. GACO has been tested on seventeen well recognized standard benchmark functions and its results have been compared with three different meta-heuristic algorithms namely as Genetic Algorithm, Particle Swarm Optimization and Artificial Bee Colony. The performance metrics such as average and standard deviation are computed and evaluated with respect to these metrics. The proposed GACO performs better in comparison to the aforementioned algorithms. The proposed algorithm optimizes the cloud resource allocation problem and gives better results with unknown search spaces.

Keywords Ant algorithms · Meta-heuristics · Cloud computing · Optimization

Mathematics Subject Classification 91B32 · 68T20 · 90C26

1 Introduction

The cloud is an emerging computing technology that operates at large scales. It is the development of grid and service-oriented computing [1]. It supports virtualization technology and enables to lease on computing resources as a service. These resources are deployed in the form of different Virtual Machines (VMs). The VMs are dynami-

✉ Ajay Kumar
akumarphd@thapar.edu

Seema Bawa
seema@thapar.edu

1 Thapar Institute of Engineering and Technology, Patiala, Punjab, India

cally allocated to the user based on their Service Level Agreement (SLA) [2–4]. The SLA is established through negotiation between user and service provider in terms of the deadline, consistency, security, network congestion and performance limit [5,6]. The main goal of any service providers is to gain maximum profit and efficient usage of resources. There are many issues in cloud computing such as limited bandwidth, poor resource allocation, security concerns, data integration problem, migration concerns etc. Lack of optimized resource allocation is one of the key issues of the cloud computing [7]. The primary objective of resource allocation process is that the Quality of Service (QoS) [2] constraints must be satisfied so that service provider profit should be maximized. The SLA violations affect the QoS guarantees and may be a profit loss of a service provider due to certain penalty cost. This manuscript is confined only two approaches of QoS: Cloud Service Provider(CSP) and Cloud Service Consumer(CSC). The CSP and CSC mutually support the SLA in three aspects: services, profits and resources. Each service provider has many service consumers and it is necessary to ensure that the QoS requirements of all customers are achieved. Generally, the cloud provider applied a conventional auction mechanism periodically to allocate the instances of VMs [8,9]. The existing schemes of resource allocation are not much more effective to reduce the SLA penalty costs. The allocation of resources in cloud environment is an NP-hard problem. There are several meta-heuristic optimization algorithms proposed to solve an optimal resource allocation in cloud computing.

This paper propose a hybridization of SACO [10] and GCO [11,12] known as Generalized Ant Colony Optimizer (GACO), which is based on distance matrix, new colony creation, foraging behavior and continuous efforts of ants. The performance of GACO algorithm is measured on 17 benchmark test functions and compared with Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Artificial Bee Colony (ABC) algorithms.

The rest of the research article is organized as follow: Sect. 2 explores the comprehensive literature review on meta-heuristic optimization and ant algorithms. The proposed GACO algorithm has been explained in Sect. 3. An experiment, results and discussion are presented in Sect. 4. In Sect. 5, we apply GACO to optimization of resource allocation in cloud computing environment and Sect. 6 followed by conclusion and future scopes.

2 Literature review

This section explores a systematic review of meta-heuristic optimization algorithms, the emergence and recent applications of ant algorithms.

2.1 Meta-heuristics optimization algorithms

In the last three decades, the meta-heuristic optimization algorithms became popular. These algorithms have been used for getting optimal possible results for large-scale

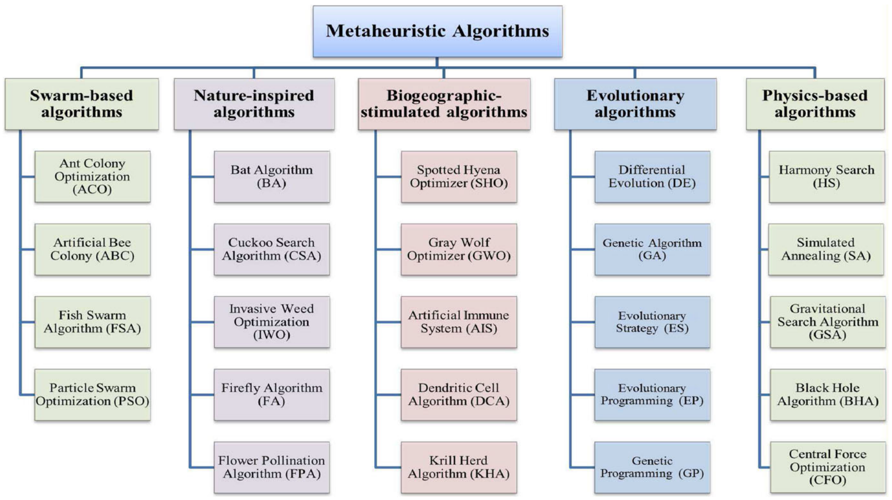
Metaheuristic Algorithms
Swarm-based algorithms
Ant Colony Optimization (ACO)
Artificial Bee Colony (ABC)
Fish Swarm Algorithm (FSA)
Particle Swarm Optimization (PSO)
Nature-inspired algorithms
Bat Algorithm (BA)
Cuckoo Search Algorithm (CSA)
Invasive Weed Optimization (IWO)
Firefly Algorithm (FA)
Flower Pollination Algorithm (FPA)
Biogeographic-stimulated algorithms
Spotted Hyena Optimizer (SHO)
Gray Wolf Optimizer (GWO)
Artificial Immune System (AIS)
Dendritic Cell Algorithm (DCA)
Krill Herd Algorithm (KHA)
Evolutionary algorithms
Differential Evolution (DE)
Genetic Algorithm (GA)
Evolutionary Strategy (ES)
Evolutionary Programming (EP)
Genetic Programming (GP)
Physics-based algorithms
Harmony Search (HS)
Simulated Annealing (SA)
Gravitational Search Algorithm (GSA)
Black Hole Algorithm (BHA)
Central Force Optimization (CFO)

Fig. 1 Taxonomy of meta-heuristic algorithms

scientific compute-intensive problems. Meta-heuristic techniques are more relevant as compare to tradition optimization techniques to solve the optimization problem in large search spaces:

(i) Reduced complexity
(ii) In general faster
(iii) Global searching ability
(iv) Enhance performance and efficiency
(v) No fixed iteration
(vi) Simplicity and flexibility

Mostly, these algorithms are inspired by nature, physical phenomena, animal behavior, insect behavior, swarm intelligence and evolutionary concepts [13]. Meta-heuristic techniques are divided into five categories: Swarm-based, Nature-inspired, Biogeographic-stimulated, Evolutionary, and Physics-based. This classification is not unique; it may also happen that some algorithms have been included in another category. It depends on what the emphasis is and what the perspective may be. Figure 1 shows the taxonomy of meta-heuristic algorithms.

The first meta-heuristic technique is a swarm-based which is inspired by the mutual behavior of particles, insects and social creatures. The most popular swarm-based algorithms are Fish Swarm Algorithm (FSA) [14], Particle Swarm Optimization (PSO) [15] Artificial Bee Colony (ABC) [16] and Ant Colony Optimization (ACO) [17]. A comprehensive literature survey of ant algorithms is provided in Sects. 2.2 and 2.3. Following are the advantages of swarm-based algorithms:

(i) Swarm-based algorithms have very less number of parameters to amend.
(ii) A swarm-based algorithm does not have a significant number of operators as compared to other approaches.
(iii) Swarm-based algorithms collect facts about search space over the number of iterations.

(iv) Very easy to implement.

Nature-inspired is the second category of meta-heuristic technique. It is based on guided search procedure. Population always adapts from environment. In some cases nature-inspired algorithm can be chemical and biological system depending on source of inspiration. Such meta-heuristic algorithms include Cuckoo Search Algorithm (CSA) [18], Bat Algorithm (BA) [19], Flower Pollination Algorithm (FPA) [20], Invasive Weed Optimization (IWO) [21], and Firefly Algorithm (FA) [22].

The third subclass of meta-heuristic is biogeographic-stimulated algorithms based on hunting and foraging behavior of some animals and sea creatures. The well-known algorithm of bio-inspired is Spotted Henya Optimizer (SHO) [23], Grey Wolf Optimizer (GWO) [24], Emperor Penguin Optimizer (EPO) [25] and Krill Herd Algorithm (KHA) [26]. These algorithms have their substantial ability to impersonate the best efforts.

The fourth subclass of meta-heuristic is evolutionary algorithms. These are based on the natural selection theory. The population tries to continue constructed on the fitness measure in the environment. It performs well and gives a nearest optimal solution. This method does not yield any hypothesis about adaptive nature and fitness. The most popular evolutionary algorithms are Deferential Evolution (DE) [27], Genetic Algorithm (GA) [28], Evolutionary Strategy (ES) [29], Evolutionary Programming (EP) [30], and Genetic Programming [31].

The last one is physics-based algorithms. These algorithms are typically based on physical rules. Some popular algorithms are Black Hole Algorithm (BHA) [32], Harmony Search (HS) [33], Simulated Annealing (SA) [34], and Central Force Optimization (CFO) [35], Gravitational Search Algorithm (GSA) [35]. These algorithms are different from swarm-based and nature-inspired algorithms. There are undefined usual of search agents are communicating and moving from one place to another due to physical rules. This communication and movement can be easily implemented by BH,CFO, GSA

2.2 Emergence of ant algorithm

This section provides an overview of various ant colony optimization meta-heuristic basic principles such as Ant Q, Max Min ant system, Ant-tabu, Continuous colony optimization, Fast ant system etc. Ants are social insects and living collectively in the colony. The collective behaviors of ants are more important to the survival of colony as well as the individual. Many researchers are working on it and many research on ant colonies intended at a better indulgent of ants behaviors. Many algorithms have been developed include the division of worker, foraging behavior, brood care, cemetery organization and colony construction. Table 1 shows the comprehensive review on emergence of different ant algorithms developed by many researchers.

2.3 Recent applications using ant algorithms

Over the last three decades, ACO algorithms have developed as a dominant meta-heuristic technique to solve the various optimization problems [44–46]. The ACO

Table 1 Emergence of ant algorithms

Variant	Authors	Description
ACO metaheuristic (ACO-MH)	Marco Dorigo	It is a first algorithmic optimization model of ants based on the foraging behavior and continuous effort of ants [10]
Ant colony system (ACS)	Gambardella and Marco Dorigo	Improve the performance of ant system in four aspects: (1) used transition rules (2) defined different pheromone update rules (3) introduced local pheromone update (4) used candidate list [1]
Elitist ant system (EAS)	Marco Dorigo	[17,36]
Max–min ant system (MMAS)	Stutzle and Hoos	[17,32,36,37]
Fast ant system (FANT)	Taillard and Gambardella	Specifically to find the best solution of quadratic assignment problem. It uses only one ant knowingly moderates the complexity [38]
Ant-Q	Gambardella and Marco Dorigo	Q-learning inspires ants. In ant-Q, the pheromone update substituted by ant-Q value [39]
Ant Tabu (Antabu)	Roux and Kaji	Comprise a local search space using Tabu search and refine the solutions of alliterations [40]
Lumer–Faieta algorithm (LFA)	Lumer and Faieta	It is a generalized form of the BACCM to cluster data with real-valued components [41,42]
Basic ant colony clustering model (BACCM)	Deneubourg et al.	It is the Chretiens observations inspired the algorithmic simulations [43]
Continuous ant colony optimization (CACO)	Bilchev and Parmee	It has accomplished two different search spaces. A local search to act as good province by local optima and global search act as bad province explored by the global optima [11]

meta-heuristic algorithms [17,45] have applied in different domains such as supply chain management [47], a dynamic scheduling problem [48,49], communication network design [50], multi-objective problem [51,52], clustering problem [53], optimal truss design [54] etc. Table 2 shows the detailed review of recent applications solved by different authors.

Dorigo et al. [10] has explained the recent trends of ant algorithms theoretically. The paper highlights how biological and nature insects inspiration can be transferred into an algorithm for distinct optimizations and also outlined the ACO in more precise regarding discrete optimization. Authors have summarized theoretical concepts and real optimization problems such as data mining, load balancing, graph coloring, etc.

Zeng et al. [75] have given some direction about the usage of ACO algorithm in the cloud storage system. The user can select a cloud storage resource randomly at the initial stage, and preserve the experiences and satisfactory information in service path as a pheromone. After over time the next user will be referred the past user experience and related value (pheromone deposit) from current cloud storage service. All these information are controlled by some protocols according to the situation.

An optimal solution of load balancing problem in cloud environments using ACO algorithm has explained in [76]. This article described how ACO algorithm maximizes and minimizes the different performance parameters like load, capacity, CPU, memory for the cloud. It has proposed a heuristic based ACO to initiate the service load distribution in the cloud environments. It also proved the efficiency and effectiveness of load balancing by their pheromone updating mechanism. The fault tolerance mechanism not considered in this article.

In [77] authors have studied two different approaches: MAX–MIN Ant System (MMAS) and Graph-Based Ant System (GBAS). They resolve the limitation pheromone updating of both approaches. In this article, authors have used two alternative methods to update pheromone (i) GBAS with lower pheromone bound and (ii) GBAS with evaporation factor.

3 Proposed Generalized Ant Colony Optimizer (GACO) algorithm

This section explains proposed GACO algorithm and technical specification the same. In GACO algorithm ants forging behavior plays an important role. The next section describing the same.

3.1 Ants foraging behavior

The existence of ants appeared on the earth around 10^7 years back and current population density is projected 10^6 individuals. Many entomologists studied about foraging behavior of ant and their ability to select the optimal path, without any coordination, advice, corporation and central approach. These emergent behaviors of ants located a food origin and influenced other ants also towards food. Ants are trailing pheromone on the path during food search. These pheromone deposits reinforce the ants to decide the better path. The higher pheromone deposited path attracts more ants to use that

Table 2 Recent applications of ACO

Applications	Authors	References
Microscopic model of a multi-agent interaction network	Kasprzok, Ayalew and Lau	[55]
Online scientific cloud scheduling	Pacini, Mateos and Garino	[56]
Dynamic problems	Mavrovouniotis, Müller and Yang	[57]
	Merkle and Middendorf	[58]
Shortest path problem	W.J. Gutjahr	[59]
	Kolavali and Bhatnagar	[60]
	Jiping, Shenghua, Zhang and Wang	[61]
Data communication design	Bonabeau, Henaux, Guérin, Snyers, Kuntz and Theraulaz	[50]
	Caro and Dorigo	[50]
Sequential ordering	Gambardella and Marco Dorigo	[62]
Graph coloring	Hertz and Costa	[63]
	Aleksander Vesel and Janez Zerovnik	[64]
Traveling Salesman Problem	L Bianchi, LM Gambardella and Marco Dorigo	[65]
Vehicle routing problem	Reimann, Doerner and Hartl	[66]
Bioinformatics problems	Moss and Johnson	[67]
Constraint satisfaction problems	Solnon C	[68]
Multi-objective problems	Gao, Yongqiang, Guan, Zhengwei, Yang and Liang Liu	[52]
	Ahuja, Ashish, Sanjoy Das, and Anil Pahwa	[51]
Data mining	Parpinelli, Rafael, Lopes and Alex	[69]
Tasks scheduling	Bhupinder Singh and Seema Bawa	[49]
Quadratic assignment problem	Merz,, and Freisleben	[70]
	Stützle and Dorigo	[71]
Data clustering	Runkler and Thomas	[53]
Cloud computing initiative	Banerjee, Soumya, Mukherjee and Mahanti	[72]
Mixed-value optimization	Socha and Krzysztof	[73]
Routing problem	Lu, THNguyen, Ngo, Nguyen and Nguyen.	[74]

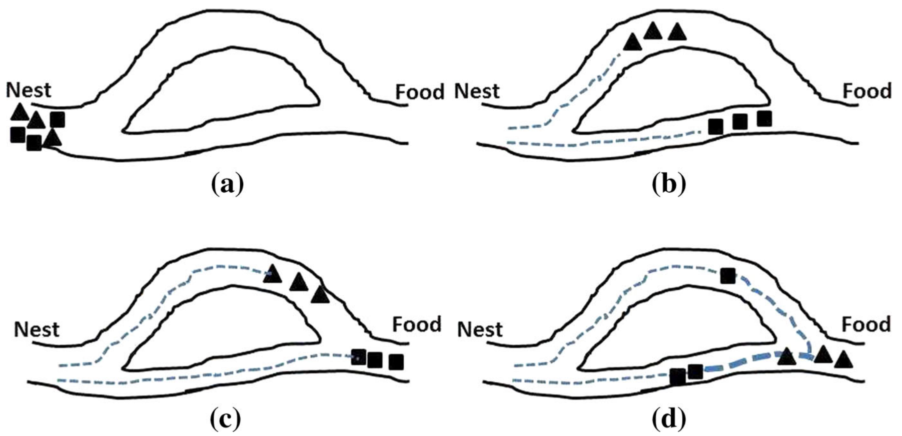

Fig. 2 The capability of binary path finding by ants. The pheromone deposits are shown as a dotted lines whose thickness denotes the pheromone strength, **a** all ants are in the nest. At initial stage there is no pheromone deposit on the path, **b** all ants move towards food. There may be probability 50% ants select shorter path and 50% ants take longer path. All ants trail same amount of pheromone on there respective path, **c** the ants have selected the shorter path, have arrived earlier. In case of returning, the probability of selecting shorter path again is higher, **d** the shorter path has got more pheromone deposited as compare to longer path. The shorter path has reinforced, and the probability to select this path increases. As the time increases the whole colony will use shorter path

path. Here, the indirect communication modifies their path to affects the behavior of the other ants also. Dorigo and Blum [1] have studied about ants foraging behavior and explained a formal model of binary path selection process. This model assumed each ant deposits the same amount of pheromone. The mathematical formulation of the binary path selection had defined as follows:

Let $n_{e1}(k)$ and $n_{e2}(k)$ indicate the number of ants moving on path e_1 and e_2 respectively at time k. After time $(k+1)$, the probability of the upcoming ants to select e_1 path as follow:

$$\begin{aligned} P_{e1}(k+1) &= \frac{(n_{e1}(k)+c)^{\alpha}}{(n_{e1}(k)+c)^{\alpha}+(n_{e2}(k)+c)^{\alpha}} \\ &= 1-P_{e2}(k+1) \end{aligned} \tag{1}$$

where c is the quantifier that shows the degree of the attraction of path, and α indicates the pheromone deposit. If the value of α is increased, then probability may also be increased and reinforced to follow this path.

When ants move from nest to food source, an amount of pheromone deposited by each ant. Over time nearest path will have deposited more pheromone, and other ants move quickly on this path. The finding capability of binary path by ant's colony is illustrated in Fig. 2.

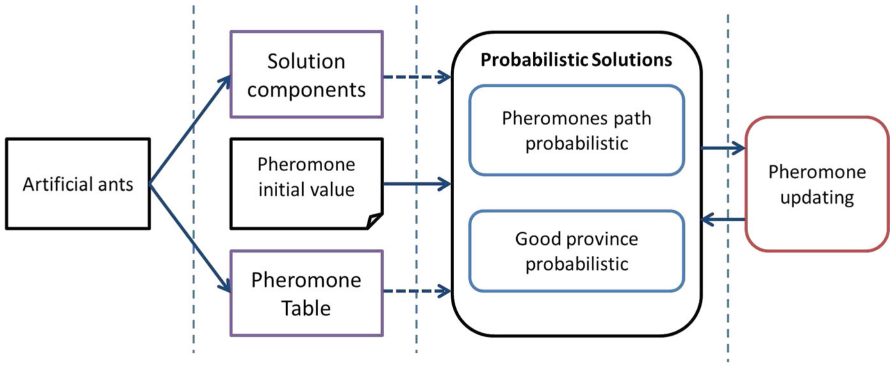

Fig. 3 Working model of GACO algorithm

3.2 Working model of GACO algorithm

The simple ACO algorithm was explained by Marco Dorigo et al. in the early 1990s. The SACO is an algorithmic implementation of binary path model illustrated in Equation 1. Artificial ants will find the nearest node randomly first time and the ants who have successfully reached the destination will update the total path length (L) as a pheromone trail of the link visited in pheromone table. Here, actually routing table is known as pheromone table that contains for each destination with real valued information, on for each known adjacency node. This information is a remedy of goodness of traversing over that adjacency node on the way to the destination. This table is continuously updated conferring to the quality of solution by the artificial ants. The next ant's group will explore the pheromone trails with the help of pheromone table to reach the destination. The working model of proposed algorithm graphically exhibited in Fig. 3.

Let, number of artificial ant sets $m = \{1, 2, \ldots, n_m\}$ are located in the colony. We assume ant m is currently situated at node i then the transitive probability of the ants to choose next node $j \epsilon v_i^m$ is as follows:

$$P_{ij}^m(t) = \begin{cases} \frac{z_{ij}^{\alpha}(t)}{\sum_{j \epsilon v_i^m} z_{ij}^{\alpha}(t)} & if\ j \in v_i^m \\ 0 & if\ j \notin v_i^m \end{cases} \tag{2}$$

where v_i^m is the set of feasible path linked with v_i with respect to m number of ants. Each ant has their own decision policy to choose the next node. The z_{ij} indicates the pheromone concentration on corresponding edge (*i, j*). If any v_i and ant m does not has any adjacent $v_i^m = \emptyset$, then the predecessor to v_i is included in v_i^m. There is always a probability of getting a cyclic path. In this case we can remove the cycle once the destination node has been selected. The α denoted as a constant (positive) value to increase the impact of pheromone deposit. Once ants have built a complete path between source to destination nodes then entire cycle has been removed and individual path can be retraced to their source and retain a pheromone amount on each edge (i, j)

of their respective path. The total deposited pheromone by ant m is $\delta z_{ij}^{m}(t)$ on path length $L^{m}(t)$. The inverse if the path length $L^{m}(t)$ consider as quality path as follows:

$$\delta z_{ij}^{m}(t) \propto \frac{1}{L^{m}(t)} \tag{3}$$

$$therefore, \qquad z_{ij}(t+1) = z_{ij}(t) + \sum_{m=1}^{n_m} \delta z_{ij}^{m}(t) \tag{4}$$

where n_m denotes the population. We assume if $O^{m}(t)$ denotes the optimal solution during time t then $f(O^{m}(t))$ depicts the quality solution. It is more important to emphasize on the optimal path selection process as a result of coordination which are evaluated from the modest behaviors of individual ants. In multiple global colony optimization [4,11,36,73] for artificial ants n_m, we consider both local as well as global search spaces. We assume n_l and n_g ants perform local and global search respectively. Let r_i denotes the province and $f(r_i)$ is the fitness of province r_i. All provinces must be assigned with quality solution dynamically and z_i initializes the pheromone amount for each province. At starting point global search space is considered as weak because $\forall n_m$ unknown about global province to explore new path. Here, 95% of the n_g perform crossover to struggle for new province and remaining 5% involved in pheromone trail distribution. The probability of each m of the local ants n_l selects r_i province which partially towards the good province as follow:

$$p_i^{m}(t) = \frac{z_i^{\alpha}(t)\mu_i^{\beta}(t)}{\sum_{j\epsilon N_i^m} z_j^{\alpha}(t)\mu_j^{\beta}(t)} \tag{5}$$

where $\mu_i^{\beta}(t)$ denotes the a priori attractiveness of the move from source to destination, N_i^m indicate the feasible nodes. If province find better fitness then ant moves in the same direction otherwise increase the age of province and choose new direction randomly. The age of province denotes the weakness of the particular solution. The pheromone is modified by adding the amount of each z_i proportionate to the fitness of relative province. Some of other techniques to solve continuous optimization are mentioned in [11,37,73]. The technical specification of GACO algorithm is explained in Algorithm 1.

4 Experimental evaluation and discussions

This section explains seventeen well-known benchmark functions that are used to evaluate the efficiency and performance of GACO algorithm. The suggested benchmark test functions are exhibited in Sects. 4.1. Section 4.2 evaluates the performance metrics such as average and standard deviation compare with PSO, GA and ABC meta-heuristic algorithms.

Algorithm 1 : Generalized Ant Colony Optimizer

Input: the number of artificial ants $n_m(m = 1, 2, 3, \ldots n_m)$, number of province $r_k(k = 1, 2, 3, \ldots r_k)$
Output: Optimal path with quality solution, Province with best fitness value

```
1: Procedure GACO
2: Set $d = 0$, $z_{ij}(0) = 0.01$ (to small default values), $z_k(0) = 1$
3: do
4:     for each $m = 1, 2, 3, \ldots n_m$ do
5:         $O^m(t) = NULL$
6:         do
7:             Pick next node using Eq. 2
8:             Link edge $(i, j)$ to path $O^m(d)$
9:         while till reach on destination node
10:    end for
11:    for each ant $m = 1, 2, 3, \ldots n_m$ do
12:        for each edge $(i, j)$ of $O^m(d)$ do
13:            $\delta z^m = \frac{1}{f(O^m(d))}$
14:            Add pheromone $z_{ij}$ using Eq. 4
15:        end for
16:    end for
17:    d=d+1
18: while exit condition is true
19: Update $O^m(d)$ as optimal with quality solution $f(O^m(d))$
20: do
21:    Calculate $f(r_i)$
22:    Sort fitness of $\forall r_i$ in decreasing order
23:    Post 95% of $n_g$ for mutation and crossover
24:    Post 5% of $n_g$ for trail diffusion
25:    Update pheromone and age of $n_g$ week province
26:    Send $n_l$ ants to picked good province by Eq. 5
27:    for each $n_l$ do
28:        if improve fitness of $r_i$ then
29:            Update pheromone and move to good province
30:        else
31:            Increase province age and select new direction randomly
32:        end if
33:    end for
34: while exit condition is true
35: End Procedure
```

4.1 Test benchmark functions and competitor algorithms

There are seventeen benchmark test functions have been applied on GACO algorithm and evaluate the performance. These test functions are categorized into two classes: (a) uni-modal, (b) multi-model. The modality of benchmark function defines the number of local minima and global minima locations. Table 3 shows the complete details of test functions along with search space range. These benchmark test functions evaluate the behavior of an algorithm in difficult situation and sometime diverse [78]. The seven functions (T1 – T7) are uni-modal and ten functions (T8 – T17) are included in multi-modal test benchmark. The first group of uni-model benchmark functions is more relevant for evaluating the utilization capability and convergence rapidness of an algorithm. There is no local optima and having single global optimum. Another group of multi-domain benchmark functions provide multiple local solutions and test the finding ability of an algorithm.

The population size/search agent of each competitive algorithm is set to 20. We observed that 20 is an equitable value of population size for several optimization problems. Generally, large size is not suitable for determining the global optima because it reflects the higher probability.

4.2 Performance evaluation

The parameters description of proposed algorithm and competitive algorithms are mentioned in Table 4. These parameters are fixed on the basis of reported literature review. The algorithms illustrated in this paper are implemented in MATLAB R2017b-9.3.0713579 version and then executed on MS Windows 7 Professional N with 64 bits. It is deployed on hardware configuration such as Intel Xeon e5 2650 2.6 GHz and 8 GB memory. The performance metrics are computed as average (AVG) and standard deviation (STDEV) of best solution till last iteration. Tables 5 and 6 show the evaluation results of test benchmark functions (T1–T7) and (T8–T17) respectively. As per reported results, the GACO is more efficient algorithm for test functions T1, T2, T5–T8, T10–T14 and T16. Remaining functions produce competitive results. We have generated and reported the result of each benchmark separately. These reports are mentioned in Figs. 4 and 5 of uni-modal and multi-model test functions respectively.

5 Optimization of cloud resource allocation using GACO

In this section proposed GACO has been applied to optimize the cloud resource allocation problem. The main motivation of doing this is that GACO provides best solution in larger search space as compare to competitors. The cloud resource allocation problem has been mathematically formulated which is based on biding concepts [8]. The main objective of this resource allocation process is the Quality of Service (QoS) constraints must be satisfied and maximized the service provider profit. The sole motivation behind this formulation is to reduced the unnecessary SLA violations penalty cost.

We assume cloud provider offers $\mathrm{VM} = \{\mathrm{vm}_1, \mathrm{vm}_2 \ldots \mathrm{vm}_t\}$ computing services to user on t different types of Virtual Machines (VMs). The number of existing identical service capacity of a type $\mathrm{vm}_{k=\{1,2,3,\ldots t\}}$ is W_k. We consider that W_t is maximum number of VM instances provisioned by cloud-bid provider. Let, $\mathrm{Z} = \{\mathrm{z}_1, \mathrm{z}_2, \ldots \mathrm{z}_i\}$ be the set of cloud-bid providers. $\mathrm{A}_i = \{\mathrm{a}_1, \mathrm{a}_2, \ldots \mathrm{a}_i\}$ is a set of bids submitted by cloud-bid provider z_i. So A_1 is the set of bids submitted by cloud-bid provider z_1. Each bid define in pairs as $\mathrm{a}_i = (\mathrm{R}_i, \mathrm{c}_i)$, where, R is the requested VM resources set and c is the cost of respective bid. The requested VM set further divided in different pairs of individual resource (r) and their unit (u) as shown in Fig. 6. It is defined as:

$$\mathrm{R}^{i}_{\{1,\ 2,\ldots n\}} = \left\{\left(\mathrm{r}^{i}_{1}, \mathrm{u}^{i}_{1}\right),\ \left(\mathrm{r}^{i}_{2}, \mathrm{u}^{i}_{2}\right), \ldots \left(\mathrm{r}^{i}_{n}, \mathrm{u}^{i}_{n}\right)\right\}$$

Table 3 Test benchmark functions [79,80]

Test functions	Dim	Range	f_{min}
$T1\left(x\right)=\sum_{k=1}^{N}x_k^2$	20	[− 100, 100]	0
$T2\left(x\right)=\sum_{k=1}^{N}\lvert x_k\rvert+\prod_{k=1}^{N}\lvert x_k\rvert$	20	[− 10, 10]	0
$T3\left(x\right)=\sum_{k=1}^{N}\left(\sum_{l=1}^{k}x_l\right)^2$	20	[− 100, 100]	0
$T4\left(x\right)=max_k\left\{\lvert x_k\rvert,\ 1\le k\le N\right\}$	20	[− 100, 100]	0
$T5\left(x\right)=\sum_{k=1}^{N-1}\left[100\left(x_{k+1}-x_k^2\right)^2+\left(x_k-1\right)^2\right]$	20	[− 30, 30]	0
$T6\left(x\right)=\sum_{k=1}^{N}\left(\lvert x_k+0.5\rvert\right)^2$	20	[− 100, 100]	0
$T7\left(x\right)=\sum_{k=1}^{N}kx_k^4+rand[0,1]$	20	[− 1.28,1.28]	0
$T8\left(x\right)=\sum_{k=1}^{N}-x_k sin\left(\sqrt{\lvert x_k\rvert}\right)$	20	[− 500, 500]	-418.9829×5
$T9\left(x\right)=\sum_{k=1}^{N}\left[x_k^2-10\cos\left(2\pi x_k\right)+10\right]$	20	[− 5.12, 5.12]	0
$T10\left(x\right)=-20exp\left(-0.2\sqrt{\frac{1}{N}\sum_{k=1}^{N}x_k^2}\right)$ $-exp\left(\frac{1}{N}\sum_{k=1}^{N}cos\left(2\pi x_k\right)\right)+20+e$	20	[− 32,32]	0
$T11\left(x\right)=\frac{1}{4000}\sum_{k=1}^{N}x_k^2-\prod_{k=1}^{N}cos\left(\frac{x_k}{\sqrt{k}}\right)+1$	20	[− 600, 600]	0
$T12\left(x\right)=$ $\frac{\pi}{N}\left\{10sin\left(\pi z_1\right)+\sum_{k=1}^{N-1}\left(z_k-1\right)^2\left[1+10sin^2\left(\pi z_{k+1}\right)\right]+\left(z_N-1\right)^2\right\}$ $+\sum_{k=1}^{N}u\left(x_k,10,100,4\right)\ z_k=1+\frac{x_k+1}{4}\quad u\left(x_k,\ p,\ q,r\right)$ $=\begin{cases}q(x_k-p)^r & x_k>p\\ 0 & -p<x_k<p\\ q(-x_k-p)^r & x_k<-p\end{cases}$	20	[− 50, 50]	0

Table 3 continued

Test functions	Dim	Range	f_{min}
$T13\,(x) = 0.1\left\{sin^2\,(3\pi x_1) + \sum_{k=1}^{N}(x_k - 1)^2\left[1 + sin^2\,(3\pi x_k + 1)\right] + (x_k - 1)^2\left[1 + sin^2\,(2\pi x_k)\right]\right\} + \sum_{k=1}^{N} u\,(x_k,\ 5,\ 100,\ 4)$	20	[− 50, 50]	0
$T14\,(x) = -\sum_{k=1}^{N}\sin\,(x_k)\,.\left(sin\left(\frac{kx_k^2}{\pi}\right)\right)^{2r},\ r = 10$	4	[0, π]	− 4.687
$T15(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[− 5, 5]	− 1.0316
$T16\,(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cos x_1 + 10$	2	[− 5, 10]	0.398
$T17\,(x) = \left[1 + (x_1 + x_2 + 1)^2 * \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] * \left[30 + (2x_1 - 3x_2)^2 * \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	3	[− 2, 2]	2

Table 4 Test parameters setting of algorithms

Algorithms	Parameters	Values
GACO	Number of Iteration	100
	Population size	20
	Enforcement factor	0.1
PSO	Number of iteration	100
	Number of particles	20
	Inertia coefficient	0.50
	Cognitive and social coefficient	1.7, 1.5
GA	Number of iteration	100
	Population size	20
	Mutation and crossover	0.02, 0.8
ABC	Number of iteration	100
	Population size	20
	Acceleration coefficient	1

The satisfaction levels conditionally define as:

$$R^{i}_{\{1,\ 2,\ldots,n\}} = \begin{cases} \left\{\left(r^{i}_{1}, u^{i}_{1}\right)\cap \left(r^{i}_{2}, u^{i}_{2}\right)\cap\ldots\cap\left(r^{i}_{n}, u^{i}_{n}\right)\right\}\neq\emptyset, & \text{satisfiable for } \forall R_i \\ \left\{\left(r^{i}_{1}, u^{i}_{1}\right)\cup \left(r^{i}_{2}, u^{i}_{2}\right)\cup\ldots\cup\left(r^{i}_{n}, u^{i}_{n}\right)\right\}\neq\emptyset, & \text{satisfiable for any } R_i \end{cases}$$

The set $P = \{p_1,\ p_2,\ \ldots . p_t\}$ specify the SLA violation penalty cost to be determined for each VM. The problem is formulated using linear programming with the help of three optimization variables. For description of variable refer Table 7.

The problem formulation as follows:

$$\mathrm{MAX}\sum_{a_i\in A} c_i\alpha_i - \sum_{a_j\in A} p_{jj} \tag{6}$$

Subject to

$$\sum_{a_i\in A\ \wedge(r^{i}_{n},u^{i}_{n})\in R^{i}_{\{1,\ 2,\ldots n\}}} \beta^{j}_{i,n} + {}_{j} = W_j \quad \left(vm_j\in VM\right) \tag{7}$$

$$\sum_{vm_j\in VM} \beta^{j}_{i,n} - u^{i}_{n}\alpha_i = 0 \qquad \left(a_i\in A\wedge\left(r^{i}_{n}, u^{i}_{n}\right)\in R^{i}_{\{1,\ 2,\ldots n\}}\right) \tag{8}$$

$$\beta^{j}_{i,n} = 0 \quad \left(\left[a_i\in A\wedge\left(r^{i}_{n}, u^{i}_{n}\right)\in R^{i}_{\{1,\ 2,\ldots n\}}\right]\wedge[vm_j\in VM]\wedge[vm_j\notin r^{i}_{n}]\right) \tag{9}$$

$$\alpha_i = \{0,\ 1\} \tag{10}$$

$$\beta^{j}_{i,n},\ j\in\mathbb{Z}^{+} \tag{11}$$

The objective of above formulation maximizes the profit if we are considering SLA violation cost along with unallocated resources. The Eq. 6 is an objective line that

Table 5 Results of uni-model test benchmark functions

Performance metrics	Test benchmark functions (uni-model)						
	T1	T2	T3	T4	T5	T6	T7
AVG							
GACO	– 0.62797019	0.000221190	1.748032206	0.391470966	0.455660092	– 0.4997646	0.002744267
PSO	– 0.02911386	0.364284527	– 0.13791292	– 0.21619396	0.637147652	– 0.52166412	0.031880239
GA	1.194915470	0.034654050	0.619005273	– 3.924912955	1.289315143	1.62667031	0.018838751
ABC	0.921683838	0.010837072	2.065509962	– 4.078155224	1.932765879	– 0.416151698	0.003724579
STDEV							
GACO	5.382147	0.000377	20.67296	4.907896	0.516724	0.010934	0.087702
PSO	0.18897	1.507345	4.767657	2.910768	0.868995	0.106138	0.222223
GA	5.777799	0.356267	20.61802	22.43178	3.755351	6.344261	0.245649
ABC	4.812106	0.991172	55.35147	34.39109	4.860732	5.04982	0.168650

Table 6 Results of multi-model test benchmark functions

Performance metrics	Test benchmark functions (multi-model)									
	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
AVG										
GACO	6.6361E+22	0.07516	0.00045	0.68084	− 1.0027	1.00593	− 122610.8	2.26764	− 43.38227	− 0.41502
PSO	22.62901	0.48929	0.04936	11.78129	− 0.55434	0.91878	− 2.07970	0.03114	0.27082	− 0.05000
GA	− 16.19261	0.18020	− 0.36855	0.80121	0.38154	1.26491	0.65199	0.45261	2.95325	0.08616
ABC	− 4.5215E+11	0.14332	− 1.23789	− 6.86156	1.82284	− 1.00734	2.04154	− 8.67197	− 10.8251	2.54838
STDEV										
GACO	2.97E+23	1.005825	0.002141	3.85908	0.018907	0.072945	969763.2	31.73859	254.8488	24.96636
PSO	319.0099	1.279973	0.148833	43.55997	3.544811	0.311512	5.449231	0.161662	0.845367	0.223607
GA	323.1886	0.83332	1.525809	19.47741	13.62174	4.460204	2.237045	2.940258	3.954134	0.940207
ABC	2.02E+12	1.639552	2.609788	22.64074	7.838364	7.311913	54.12545	37.46899	70.15135	12.33269

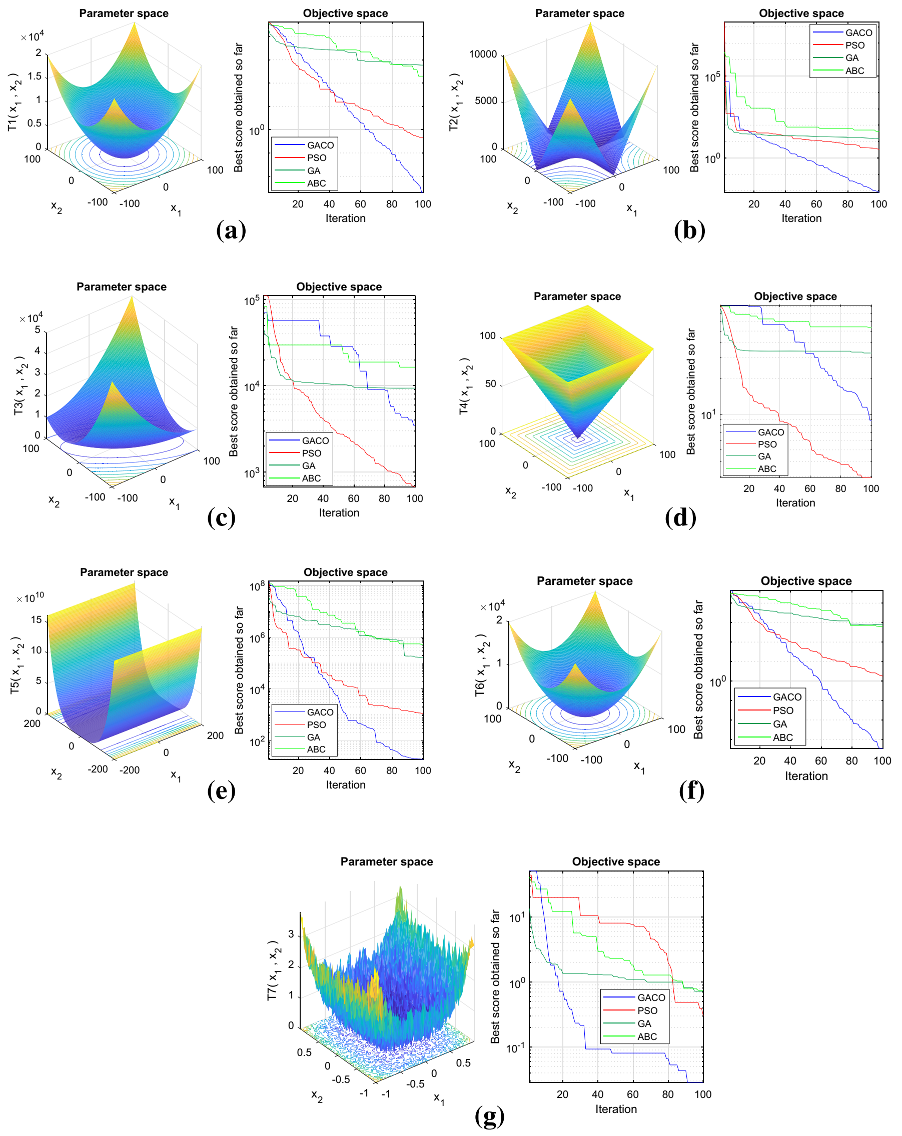

Fig. 4 Results of uni-model test benchmark functions, **a** T1, **b** T2, **c** T3, **d** T4, **e** T5, **f** T6, **g** T7

maximizes the total profit if it is considering the cost of SLA violation along with underutilized resources. Equation 7 identifies the total number of allocated VM instances of each vm_j type which never exceed the number of available instances. Equation 8 recognizes the successful bids for each pair of resource type. Equation 9 inhibits all underutilized resources to be assigned to particular bid.

We have considered all constraints and implemented above resource allocation optimization problem using MATLAB. Here, we have defined set of all the possible

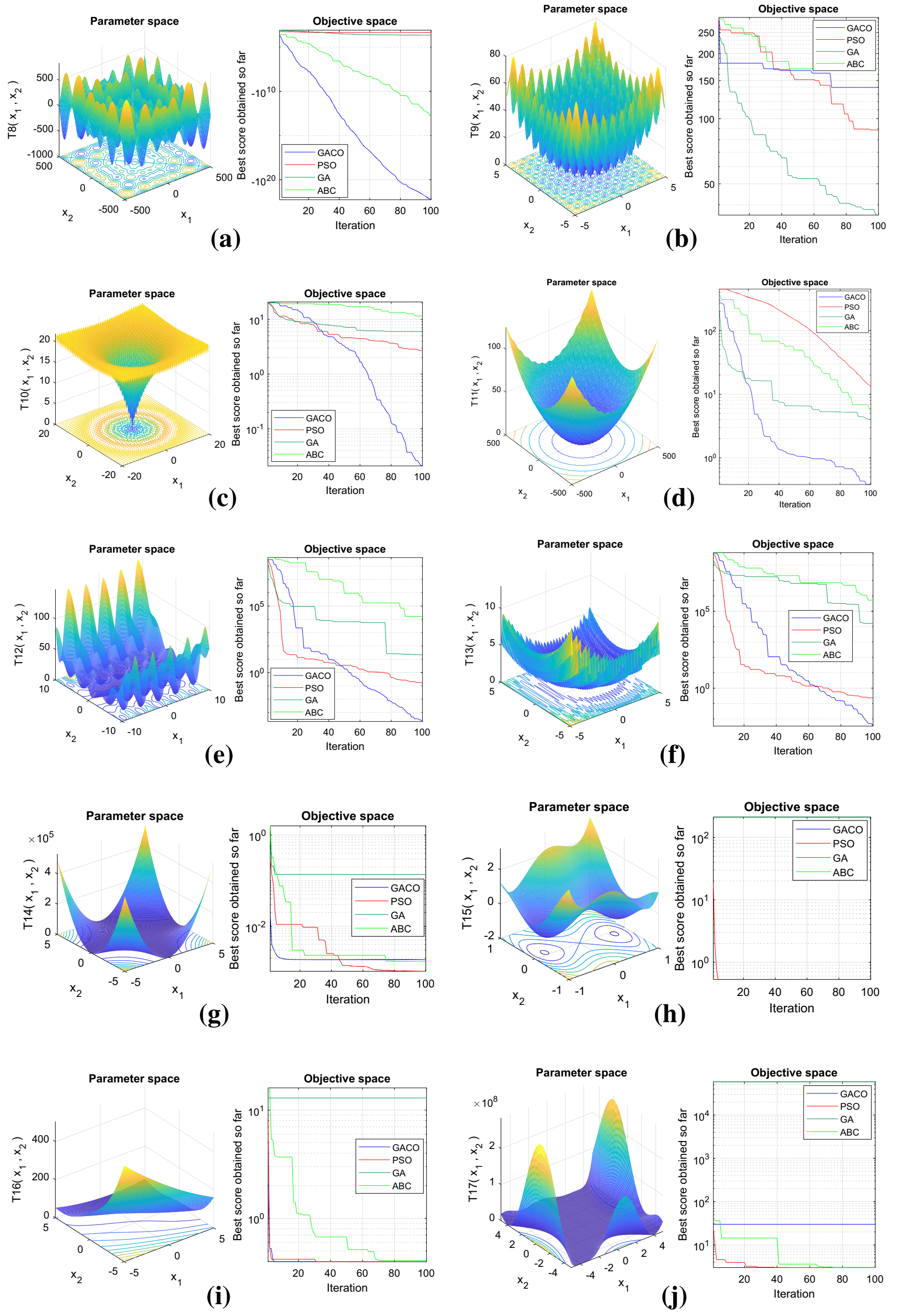

Fig. 5 Results of multi-model test benchmark functions, **a** T8, **b** T9, **c** T10, **d** T11, **e** T12, **f** T13, **g** T14, **h** T15, **i** T16, **j** T17

$$
\boxed{
\begin{array}{l}
R_1 = \{[(CPU32bit2.5GHz, 10), (Storage, 300) \dots\dots\dots..], \$150\} \\
R_2 = \{[(CPU64bit2.0GHz, 5), (Storage, 450) \dots\dots\dots..], \$250\} \\
. \\
. \\
. \\
R_{n-1} = \{[(CPU32bit3.5GHz, 15), (Storage, 500) \dots\dots\dots..], \$450\} \\
R_n = \{[(CPU64bit2.5GHz, 8), (Storage, 350) \dots\dots\dots..], \$350\}
\end{array}
}
$$

Fig. 6 Sample bid format

Table 7 Description of optimization variables

Variables	Description
α	It is a binary decision of bid success. Success bid denoted by '1' and not success by '0'
β	It denotes an integer variable. This variable contains the number of resources that are allocated to the bid
γ	It is a positive integer variable. This variable contains the number of resources that are not allocated to any bid

Table 8 Comparison of proposed GACO statistical for resource allocation

Algorithms	Worst case	Best case	Average case	STDEV
GACO	0.013792	43701.79	2796.085	8053.906
PSO	0.342633	42021.53	3271.718	8175.305
GA	146.2593	23769.62	1347.263	3141.235
ABC	919.4537	39802.37	7302.713	7546.448

bids orderings as search spaces that will be used by all meta-heuristic algorithms. Each bid will be converted to a solution and checked it will feasible or not. As per bid policy there must be at least one ordering that reflects the optimal solution. The proposed GACO algorithm has been executed 100 times and the comparative statistics of proposed GACO for resource allocation in unidentified search spaces are reported in Table 8. Figure 7 shows the comparative best solution with other algorithms. Moreover, the results of the resource allocation problems in cloud computing showed the better performance in unidentified search spaces. The results of this problem are improved as compared to existing techniques.

6 Conclusion

In this paper a swarm-based optimization algorithm inspired by ants forging behavior has been proposed. The algorithm is named as Generalized Ant Colony Optimizer (GACO). The algorithms demonstrated in this paper are implemented in MATLAB R2017b version and executed on MS Windows 7. The performance of GACO algo-

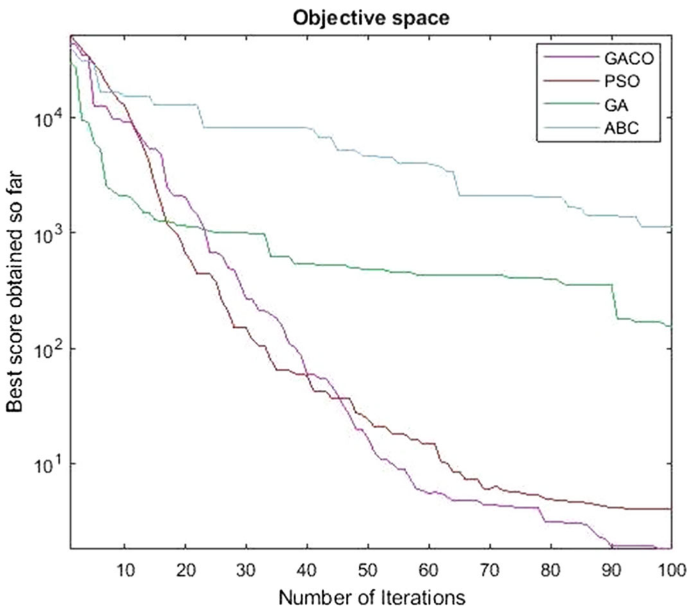

Fig. 7 Comparative best solutions with other algorithms

rithm is measured on 17 benchmark test functions and compared with three well-known algorithms naming PSO, GA, and ABC. The test results of the proposed GACO algorithm are better as compared to other algorithms. Future scopes of GACO algorithm in the cloud computing include efficient resource allocation and load balancing. This algorithm can also be extended for multi-objective optimization problems.

References

1. Kumar A, Bawa S (2012) Distributed and big data storage management in grid computing, arXiv preprint arXiv:1207.2867
2. Choi Y, Lim Y (2016) Optimization approach for resource allocation on cloud computing for IoT. Int J Distrib Sens Netw 12(3):3479247
3. Leitner P, Ferner J, Hummer W, Dustdar S (2013) Data-driven and automated prediction of service level agreement violations in service compositions. Distrib Parallel Databases 31(3):447
4. Leitner P, Hummer W, Dustdar S (2013) Cost-based optimization of service compositions. IEEE Trans Serv Comput 6(2):239
5. Farahnakian F, Ashraf A, Pahikkala T, Liljeberg P, Plosila J, Porres I, Tenhunen H (2015) Using ant colony system to consolidate VMs for green cloud computing. IEEE Trans Serv Comput 8(2):187
6. Riveni M, Nguyen TD, Dustdar S (2017) In: International conference on business process management. Springer, pp 361–373
7. Ranjan R, Wang L, Chen J, Benatallah B (2011) Cloud computing: methodology, systems, and applications. CRC Press, Boca Raton
8. Özer AH, Özturan C (2009) In: Fifth international conference on soft computing, computing with words and perceptions in system analysis, decision and control, 2009. ICSCCW 2009. IEEE, pp 1–4
9. Li W, Liu X, Zhang X, Zhang X (2015) Dynamic fair allocation of multiple resources with bounded number of tasks in cloud computing systems. Multiagent Grid Syst 11(4):245
10. Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344(2–3):243

11. Socha K, Dorigo M (2008) Ant colony optimization for continuous domains. Eur J Oper Res 185(3):1155
12. Merkle D, Middendorf M (2003) Ant colony optimization with global pheromone evaluation for scheduling a single machine. Appl Intell 18(1):105
13. Liu X, Zhang X, Li W, Zhang X (2017) Swarm optimization algorithms applied to multi-resource fair allocation in heterogeneous cloud computing systems. Computing 99(12):1231
14. Neshat M, Sepidnam G, Sargolzaei M, Toosi AN (2014) Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. Artif Intell Rev 42(4):965
15. Kennedy J (2010) Particle swarm optimization. In: Sammut C, Webb GI (eds) Encyclopedia of machine learning. Springer US, Boston, MA, pp 760–766. https://doi.org/10.1007/978-0-387-30164-8_630
16. Karaboga D, Ozturk C (2011) A novel clustering approach: artificial bee colony (ABC) algorithm. Appl Soft Comput 11(1):652
17. Dorigo M, Caro G Di (1999) In: Proceedings of the 1999 congress on evolutionary computation, 1999. CEC 99, vol 2, IEEE, pp 1470–1477
18. Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17
19. Yang XS (2011) Bat algorithm for multi-objective optimisation. Int J Bio-Inspir Comput 3(5):267
20. Yang XS, Karamanoglu M, He X (2014) Flower pollination algorithm: a novel approach for multiobjective optimization. Eng Optim 46(9):1222
21. Karimkashi S, Kishk AA (2010) Invasive weed optimization and its features in electromagnetics. IEEE Trans Antennas Propag 58(4):1269
22. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspir Comput 2(2):78
23. Dhiman G, Kumar V (2017) Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Adv Eng Softw 114:48
24. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46
25. Dhiman G, Kumar V (2018) Emperor penguin optimizer: a bio-inspired algorithm for engineering problems. Knowl-Based Syst 159:20–50
26. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831
27. Yan JY, Ling Q, Sun DM (2006) In: International conference on machine learning and cybernetics, 2006, IEEE, pp 2103–2106
28. Weile DS, Michielssen E (1997) Genetic algorithm optimization applied to electromagnetics: a review. IEEE Trans Antennas Propag 45(3):343
29. Du D, Simon D, Ergezer M (2009) In: IEEE international conference on systems, man and cybernetics, 2009. SMC 2009, IEEE, pp 997–1002
30. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evolut Comput 3(2):82
31. Koza JR (1994) Genetic programming as a means for programming computers by natural selection. Stat Comput 4(2):87
32. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175
33. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60
34. Van Laarhoven PJ, Aarts EH (1987) Simulated annealing: theory and applications. Springer, New York, pp 7–15
35. Yang Ll, Qian Wy, Zhang Q (2011) Central force optimization. J Bohai Univ (Nat Sci Ed) 3:001
36. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B (Cybern) 26(1):29
37. Blum C (2005) Ant colony optimization: introduction and recent trends. Phys Life Rev 2(4):353
38. Taillard E (1998) FANT: fast ant system, Technical report
39. Dorigo M, Caro GD, Gambardella LM (1999) Ant algorithms for discrete optimization. Artif Life 5(2):137
40. Kaji T (2001) In: IEEE international conference on systems, man, and cybernetics, 2001, vol. 5, IEEE, pp 3429–3434
41. Boryczka U (2009) Finding groups in data: cluster analysis with ants. Appl Soft Comput 9(1):61
42. Giraldo LF, Lozano F, Quijano N (2011) Foraging theory for dimensionality reduction of clustered data. Mach Learn 82(1):71

43. Deneubourg J, Goss S, Franks N, Sendova-Franks A, Detrain C, Chretien L (1992) In: From animals to animats: proceedings of the first international conference on simulation of adaptive behavior, pp 353–363
44. Dorigo M, Birattari M (2011) Encyclopedia of machine learning. Springer, New York, pp 36–39
45. Dorigo M, Stützle T (2003) Handbook of metaheuristics. Springer, New York, pp 250–285
46. Shtovba SD (2005) Ant algorithms: theory and applications. Program Comput Softw 31(4):167
47. Silva CA, Sousa J, Runkler TA, Da Costa JS (2009) Distributed supply chain management using ant colony optimization. Eur J Oper Res 199(2):349
48. Lorpunmanee S, Sap MN, Abdullah AH, Chompoo-inwai C (2007) An ant colony optimization for dynamic job scheduling in grid environment. Int J Comput Inf Sci Eng 1(4):207
49. Singh B, Bawa S (2007) In: Proceedings of the third conference on IASTED international conference, pp 283–286
50. Di Caro G, Dorigo M (1998) In: International conference on parallel problem solving from nature, Springer, pp 673–682
51. Ahuja A, Das S, Pahwa A (2007) An AIS-ACO hybrid approach for multi-objective distribution system reconfiguration. IEEE Trans Power Syst 22(3):1101
52. Gao Y, Guan H, Qi Z, Hou Y, Liu L (2013) A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J Comput Syst Sci 79(8):1230
53. Runkler TA (2005) Ant colony optimization of clustering models. Int J Intell Syst 20(12):1233
54. Luh GC, Lin CY (2008) Optimal design of truss structures using ant algorithm. Struct Multidiscip Optim 36(4):365
55. Kasprzok A, Ayalew B, Lau C (2018) An ant-inspired model for multi-agent interaction networks without stigmergy. Swarm Intell 12(1):53
56. Pacini E, Mateos C, Garino CG (2016) Multi-objective swarm intelligence schedulers for online scientific clouds. Computing 98(5):495
57. Mavrovouniotis M, Müller FM, Yang S (2017) Ant colony optimization with local search for dynamic traveling salesman problems. IEEE Trans Cybern 47(7):1743
58. Merkle D, Middendorf M (2002) Modeling the dynamics of ant colony optimization. Evolut Comput 10(3):235
59. Gutjahr WJ (2000) A graph-based ant system and its convergence. Future Gener Comput Syst 16(8):873
60. Kolavali SR, Bhatnagar S (2008) In: International conference on network control and optimization, Springer, pp 37–44
61. Liu J, Xu S, Zhang F, Wang L (2017) A hybrid genetic-ant colony optimization algorithm for the optimal path selection. Intell Autom Soft Comput 23(2):235
62. Gambardella LM, Dorigo M (2000) An ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS J Comput 12(3):237
63. Costa D, Hertz A (1997) Ants can colour graphs. J Oper Res Soc 48(3):295
64. Žerovnik J, Vesel A (2000) How well can ants color graphs? J Comput Inf Technol 8(2):131
65. Bianchi L, Gambardella LM, Dorigo M (2002) In: International conference on parallel problem solving from nature, Springer, pp 883–892
66. Reimann M, Doerner K, Hartl RF (2004) D-ants: savings based ants divide and conquer the vehicle routing problem. Comput Oper Res 31(4):563
67. Moss J, Johnson CG (2003) In: Artificial neural nets and genetic algorithms, Springer, pp 182–186
68. Solnon C (2002) Ants can solve constraint satisfaction problems. IEEE Trans Evolut Comput 6(4):347
69. Parpinelli RS, Lopes HS, Freitas AA (2002) Data mining with an ant colony optimization algorithm. IEEE Trans Evolut Comput 6(4):321
70. Merz P, Freisleben B (1999) In: Proceedings of the 1999 congress on evolutionary computation, 1999. CEC 99. vol. 3, IEEE, pp 2063–2070
71. Stutzle T, Dorigo M (1999) Aco algorithms for the quadratic assignment problem. New Ideas Optim C(C50):33
72. Banerjee S, Mukherjee I, Mahanti P (2009) Cloud computing initiative using modified ant colony framework. World Acad Sci Eng Technol 56(32):221
73. Socha K (2004) In: International workshop on ant colony optimization and swarm intelligence, Springer, pp 25–36
74. Lu DN, Nguyen TH, Nguyen DN, Nguyen HN et al. (2017) In: International conference on information networking (ICOIN), 2017, IEEE, pp 584–589

75. Zeng W, Zhao Y, Ou K, Song W (2009) In: Proceedings of the 2nd international conference on interaction sciences: information technology, culture and human, ACM, pp 1044–1048
76. Mishra R, Jaiswal A (2012) Ant colony optimization: a solution of load balancing in cloud. Int J Web Semant Technol 3(2):33
77. Gutjahr WJ (2002) ACO algorithms with guaranteed convergence to the optimal solution. Inf Process Lett 82(3):145
78. Nakib A, Ismail B, Ouchraa S, Schmitt L et al (2017) Metaheuristics for intelligent electrical networks, vol 10. Wiley, Hoboken
79. Molga M, Smutnicki C (2005) Test functions for optimization needs, Test Funct Optim Needs 101
80. Jamil M, Yang XS (2013) A literature survey of benchmark functions for global optimisation problems. Int J Math Modell Numer Optim 4(2):150