Research Journal of Applied Sciences, Engineering and Technology 10(7): 758-769, 2015
DOI:10.19026/rjaset.10.2428
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: July 24, 2014 Accepted: October 12, 2014 Published: July 10, 2015

Research Article

Cost Optimization Using Hybrid Evolutionary Algorithm in Cloud Computing

[1]B. Kavitha and [2]P. Varalakshmi
[1]Anna University, Chennai, India
[2]Department of Information Technology, MIT, Anna University, Chennai, India

Abstract: The main aim of this research is to design the hybrid evolutionary algorithm for minimizing multiple problems of dynamic resource allocation in cloud computing. The resource allocation is one of the big problems in the distributed systems when the client wants to decrease the cost for the resource allocation for their task. In order to assign the resource for the task, the client must consider the monetary cost and computational cost. Allocation of resources by considering those two costs is difficult. To solve this problem in this study, we make the main task of client into many subtasks and we allocate resources for each subtask instead of selecting the single resource for the main task. The allocation of resources for the each subtask is completed through our proposed hybrid optimization algorithm. Here, we hybrid the Binary Particle Swarm Optimization (BPSO) and Binary Cuckoo Search algorithm (BCSO) by considering monetary cost and computational cost which helps to minimize the cost of the client. Finally, the experimentation is carried out and our proposed hybrid algorithm is compared with BPSO and BCSO algorithms. Also we proved the efficiency of our proposed hybrid optimization algorithm.

Keywords: Binary cuckoo search, binary particle swarm optimization, computational cost, levy flights, monetary cost

INTRODUCTION

With the rapid development of processing and storage technologies and the success of the Internet, computing resources have become cheaper, more powerful and more ubiquitously available than ever before. This technological trend has enabled the realization of a new computing model called Cloud computing (Zhang *et al*., 2010). As a realization of utility computing, cloud computing aims to provide computing resources to customers like public utilities such as water and electricity. In a cloud computing environment, an Infrastructure-as-a-Service (IaaS) provider packages its physical resources (e.g., CPU, memory disk) into distinct types of Virtual Machines (VMs) in terms of their sizes and features and offers them as services to the general public (Zhang *et al*., 2011). Also it delivers an infrastructure, platform and software (applications) as Services that are made available to consumers in a pay-as-you-go model. In industry these services are referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) respectively. Many computing service providers including Google, Microsoft, Yahoo and IBM are rapidly deploying data centers in various locations around the world to deliver Cloud computing services (Beloglazov *et al*., 2012). At the same time computing and information processing requirements of various public organizations and private corporations have also been increasing rapidly. Examples include digital services and functions required by the various industrial sectors, ranging from manufacturing to housing, from transportation to banking (Goudarzi and Pedram, 2011).

Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995; Eberhart and Kennedy, 1995), is one of the most important swarm intelligence paradigms. The PSO uses a simple mechanism that mimics swarm behavior in birds flocking and fish schooling to guide the particles to search for globally optimal solutions. As PSO is easy to implement, it has rapidly progressed in recent years and with many successful applications seen in solving real-world optimization problems (Ho *et al*., 2008; Zhan *et al*., 2009). PSO is distinctly different from other evolutionary-type methods in that it does not use the filtering operation (such as crossover and/or mutation) and the members of the entire population are maintained through the search procedure. In PSO algorithm, each member is called "particle" and each particle flies around in the multi-dimensional search space with a velocity, which is constantly updated by the particle's own experience and the experience of the particle's neighbors. The main idea behind the development of PSO is the social sharing of information among individuals of population. In PSO algorithms, search is conducted by using a population of particles, corresponding to individuals as in the case of evolutionary algorithms. Each particle adjusts its

Corresponding Author: B. Kavitha. MIT Campus, Anna University, Chennai, India

own position towards its previous experience and towards the best previous position obtained in the swarm. Memorizing its best own position establishes the particle's experience implying a local search along with global search emerging from the neighboring experience or the experience of the whole swarm. According to the global neighborhood, each particle moves towards its best previous position and towards the best particle in the whole swarm, called G-best model. On the other hand, according to the local variant, called L-best model, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood (Kennedy *et al*., 2001; Taşgetiren and Liang, 2003). The Cuckoo Search (CS) (Yang and Deb, 2009, 2010), is a new meta-heuristic algorithm imitating animal behavior. The optimal solutions obtained by the CS are far better than the best solutions obtained by efficient particle swarm optimizers and genetic algorithms (Yang and Deb, 2009). The CS models such breeding behavior and, thus, can be applied to various optimization problems. Cuckoo Search Algorithm is based on the brood parasitism of some cuckoo species (Brajevic *et al*., 2012; Yang and Deb, 2009, 2010; Layeb and Boussalia, 2012; Valian *et al*., 2011a), discovered that the performance of the CS can be improved by using Levy Flights instead of simple random walk. The CS was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of host birds. Some cuckoos have evolved in such a way that female parasitic cuckoos can imitate the colors and patterns of the eggs of a few chosen host species (Valian *et al*., 2011b). This reduces the probability of the eggs being abandoned and, therefore, increases their re-productivity. It is worth mentioning that several host birds engage direct conflict with intruding cuckoos (Jati *et al*., 2012; Valian *et al*., 2011a). In this case, if host birds discover the eggs are not their own, they will either throw them away or simply abandon their nests and build new ones, elsewhere (Dhivya *et al*., 2011; Babukartik and Dhavachelvan, 2012). For simplicity in describing the cuckoo search, consider the following three idealized rules:

- Each cuckoo lays one egg at a time and dump its egg in randomly chosen nest.
- The best nests with high quality of eggs will carry over to the next generations.
- The number of available host nests is fixed and the egg laid by a cuckoo is discovered by the host bird (Rani *et al*., 2012; Noghrehabadi *et al*., 2011).

The effective way of choosing an optimization algorithm can further improve the results, specifically for resource allocation.

In this study, we develop hybrid optimization algorithm for dynamic resource allocation in cloud computing for minimizing the cost of the client. The main task of the client is divided into many subtasks and allocates the resource for each subtask instead of selecting the single resource for the whole entire task. The total cost of the client is calculated using the proposed fitness function taking the monetary cost and computational cost into account. Here, we hybrid two optimization algorithm such as Binary Cuckoo Search Optimization (BCSO) algorithm and Binary Particle Swarm Optimization (BPSO) algorithm to assign cloud resources to the tasks so as to minimize the cost of the client.

In BCSO algorithm, initialization matrix is generated based on the number of resources and number of subtasks. Each cuckoo particle is evaluated in the initialization matrix through the objective function. The cuckoo particle is updated based on levy flights and the fitness function is evaluated. This process is repeated for a given number of iterations. The fitness value is compared with the previous fitness value in each iteration and the minimum fitness value is considered to be the best solution.

In BPSO algorithm, the initialization matrix is generated as the number of resources and subtasks. A velocity matrix is generated randomly for the initialization matrix and the fitness function is evaluated. The G-best and P-best values are set with the initial P-best value and then the particle's velocity and position are updated in the matrix. The iteration starts with the computation of fitness function using the updated matrix and arrived result is the current P-best value. If this P-best value, is lesser than the previous G-best value, then G-best is updated with that new P-best value. This cycle is repeated for a given number of iterations. The final G-best value obtained after all iterations is the optimal solution.

In the proposed algorithm, initially the BCSO algorithm is used which makes the initialization matrix based on the number of subtask at first subsequently it calculates the fitness value for each subtask of the initialization matrix and assigns fitness values to G-best of BPSO algorithm. Then, BPSO algorithm updates the initialization matrix through the levy flights and sigmoid function. The updation matrix of the BCSO algorithm is assigned to BPSO algorithm as initialization matrix. Now the BPSO algorithm starts its process by generating the velocity matrix randomly for the initialization matrix and consequently it calculates the fitness for each particle of the initialization matrix and also updates the values of G-best and P-best. After the updation of G-best and P-best, the BPSO algorithm updates its velocity matrix through the updated velocity and sigmoid function. The updated initialization matrix is assigned to BCSO algorithm. This cyclic process repeats up to K number of iterations in order to obtain an optimal resource allocation for the client's task. The main objective of this research is to meet the multiple objectives through hybrid evolutionary algorithm for effective resource allocation scheme for cloud computing which is attained through proposed fitness function, which includes of monetary cost and computational cost as multiple objectives.

LITERATURE REVIEW

Here, a review of some of the works are presented for resource allocation in cloud computing. Yin *et al*. (2012) have proposed a multi-dimensional resource allocation scheme for cloud computing that dynamically allocates the virtual resources for the cloud computing applications with reduced cost by using fewer nodes to process the applications. They adopted a two-stage algorithm using multi-constraint integer programming problem. Experimental results shows the improved resource utilization and reduced cost of data center. Warneke and Kao (2011) have discussed the opportunities and challenges for efficient parallel data processing in clouds and present their research project Nephele. Nephele was the first data processing framework to explicitly exploit the dynamic resource allocation offered by today's IaaS clouds for both, task scheduling and execution. Particular tasks of a job could be assigned to different types of virtual machines which were automatically instantiated and terminated during the job execution. Based on these frameworks, they perform the extended evaluations of Map Reduce-inspired processing jobs on an IaaS cloud system and compared the results to the popular data processing framework Hadoop.

Stillwell *et al*. (2010) have proposed a formulation of the resource allocation problem in shared hosting platforms for static workloads with servers that provide multiple types of resources. Their formulation supports a mix of best effort and QoS scenarios and, via a precisely defined objective function, promotes performance, fairness and cluster utilization. Further, these formulations make it possible to compute a bound on the optimal resource allocation. They have several classes of resource allocation algorithms, which have been evaluated in simulation. They were able to identify an algorithm that achieved average performance close to the optimal across many experimental scenarios.

Xiao *et al*. (2013) have presented a system that uses virtualization technology for allocating data center resources dynamically based on application demands and support green computing by optimizing the number of servers in use. They introduced the concept of "skewness" to be measured the unevenness in the multi-dimensional resource utilization of a server. By minimizing skewness, they combined different types of workloads and improved the overall utilization of server resources. Finally, they developed a set of heuristics that prevent the overload in the system effectively while saving the energy used. Trace driven simulation and experiment results was demonstrated to show case the algorithm achieved a good performance.

Onat *et al*. (2010) have investigated a dynamic autonomous resource management in cloud computing. The main contribution has two-fold. First, they adopt a distributed architecture where resource management has decomposed into independent tasks, each of which was performed by Autonomous Node Agents that were tightly coupled with the physical machines in a data center. Second, the Autonomous Node Agents carry out configurations in parallel through Multiple Criteria Decision Analysis. Simulation results showed that the promising effects in terms of scalability, feasibility and flexibility.

Gogulan *et al*. (2012) have introduced a algorithm called Multiple Pheromone Algorithm (MPA) which has belongs to Ant Colony Optimization Algorithm. The objective of MPA algorithm was to dynamically generate an optimal schedule so as to be completed the task in minimum period of time as well as utilizing the resources in an efficient way. They have three different Quality of Service (QoS) make span, cost and reliability constraints were considered as performance measure for scheduling. Finally, the algorithm was compared with normal Ant Colony Algorithm (ACO) and Genetic Algorithm (GA). With the implementation of the Multiple Pheromone Algorithm (MPA), it reached an optimal solution as well as obtained the better QoS than ACO and GA.

Pawar and Wagh (2012) have evaluated various policies for resource allocation in cloud computing based on Service-Level-Agreement (SLA), centralized decision and distributed multiple criteria decision. And also how different services i.e., Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) contribute in resource allocation. Lastly they discussed the pros and cons of each policy. An *et al*. (2010) have proposed an alternative approach where providers and consumers automatically negotiate resource leasing contracts. Since resource demand and supply could been dynamic and uncertain, they distributed negotiation mechanism where agents negotiate over both a contract price and a de-commitment penalty, which allows agents to de-commit from contracts at a lower cost. They compared their approach experimentally, for used representative scenarios and workloads, to both combinatorial auctions and the fixed-price model used by Amazon's Elastic Compute Cloud and showed that the negotiation was achieved a higher social welfare.

In cloud computing, the resource allocation for a task with the minimum cost is a challenging criterion. This problem is solved by computing the total cost using the monetary cost and the computational cost for the task. Evolutionary algorithms like Cuckoo Search and Particle Swarm Optimization can be utilized to achieve the minimum cost. Here we propose a hybrid optimization algorithm using BCSO and BPSO algorithms so as to achieve better results.

PROPOSED HYBRID OPTIMIZATION ALGORITHM

In this study, we develop hybrid optimization algorithm for dynamic resource allocation in cloud computing. Here, we hybrid two optimization algorithms such as Binary Cuckoo Search Optimization (BCSO) algorithm and Binary Particle Swarm Optimization (BPSO) algorithm to schedule tasks to cloud resources, that takes into account both monetary cost and computational cost. Monetary Cost (MC) is the cost which is allotted by the service provider as the maintenance cost of the resources which includes electricity, building, cooling etc., to the client to utilize the resources to do the client's task. This cost may differ from one service provider to another for the same task. Computational Cost (CC) is also assigned by the service provider based on the computational complexity of the task assigned by client to the resource. The computational cost also differs from one service provider to another since the computational complexity includes the number of core required, required memory space, bandwidth etc. Cloud service providers invest 70% of the cost for the maintenance of the data center where as only 30% of the cost is spent on actual IT resources.

To reduce the overall cost of the resource allocation for the task of the clients, here we divide the main task into many subtasks instead of selecting the best resource among the available resources for the whole task. Also the proposed algorithm selects and allocates the best resources for each subtask from the available resources through the hybrid optimization algorithm which includes BPSO and BCSO algorithm. This hybrid optimization algorithm helps to reduce the monetary cost and computational cost of each subtask.

The architecture of the proposed system is shown in Fig. 1. The client submits a task T and desires to complete the task with less cost and efficient computation through the cloud service providers. Consider there are n number of available resources $R = \{R_1, R_2 \ldots R_i \ldots R_n\}$ where $1 \leq i \leq n$ in the cloud to do

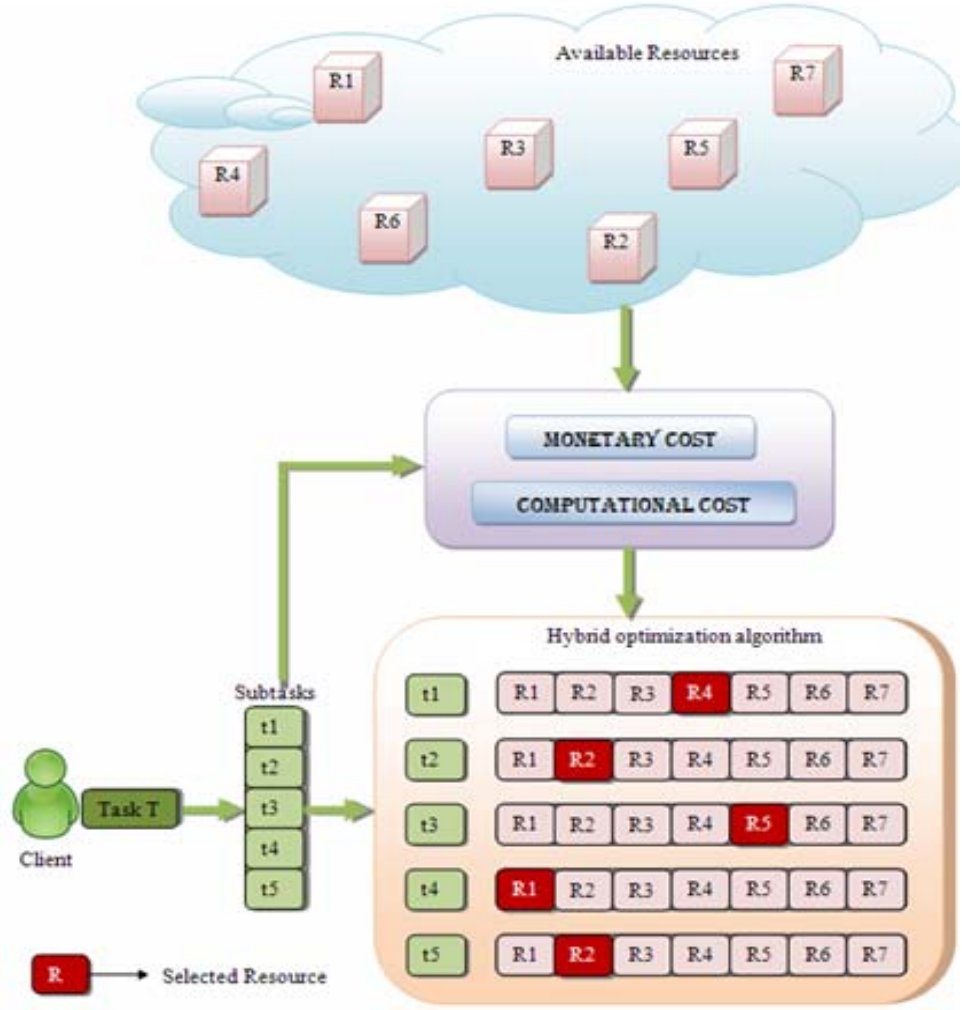


Fig. 1: Architecture of the proposed system

Table 1: Monetary cost of resources for each subtask

	Resource				
Subtask	R_1	R_2	R_i	R_{n-1}	R_n
t_1	$MC_{1,1}$	$MC_{1,2}$	$MC_{1,i}$	$MC_{1,n-1}$	$MC_{1,n}$
t_2	$MC_{2,1}$	$MC_{2,2}$	$MC_{2,i}$	$MC_{2,n-1}$	$MC_{2,n}$
t_j	$MC_{j,1}$	$MC_{j,2}$	$MC_{j,i}$	$MC_{j,n-1}$	$MC_{j,n}$
t_{m-1}	$MC_{m-1,1}$	$MC_{m-1,2}$	$MC_{m-1,i}$	$MC_{m-1,n-1}$	$MC_{m-1,n}$
t_m	$MC_{m,1}$	$MC_{m,2}$	$MC_{m,i}$	$MC_{m,n-1}$	$MC_{m,n}$

Table 2: Computational cost of resources for each subtask

	Resource				
Subtask	R_1	R_2	R_i	R_{n-1}	R_n
t_1	$CC_{1,1}$	$CC_{1,2}$	$CC_{1,i}$	$CC_{1,n-1}$	$CC_{1,n}$
t_2	$CC_{2,1}$	$CC_{2,2}$	$CC_{2,i}$	$CC_{2,n-1}$	$CC_{2,n}$
t_j	$CC_{j,1}$	$CC_{j,2}$	$CC_{j,i}$	$CC_{j,n-1}$	$CC_{j,n}$
t_{m-1}	$CC_{m-1,1}$	$CC_{m-1,2}$	$CC_{m-1,i}$	$CC_{m-1,n-1}$	$CC_{m-1,n}$
t_m	$CC_{m,1}$	$CC_{m,2}$	$CC_{m,i}$	$CC_{m,n-1}$	$CC_{m,n}$

the task *T*. To improve the efficiency and reduce the total cost of the task *T*, we divide the main task *T* into *m* number of subtasks $T = \{t_1, t_2 .. t_j .. t_m\}$ where $1 \leq j \leq m$. To complete the task with efficient way along with less cost, we consider the two costs such as monetary cost and computational cost at the same time. To achieve this, we develop hybrid optimization algorithm for dynamic resource allocation in cloud.

Resource allocation based on the input demand is a challenging issue in cloud, since both resource allocation and input demand must be satisfied for multiple constraints. So the allocation of resources for an input demand is increasing day-by-day in cloud computing resource management system. But the major problem is handling of real time constraints such as, monetary cost and computational cost. To handle all these criteria, we develop a resource allocation algorithm called hybrid optimization algorithm to do resource allocation dynamically in cloud computing. For example, let us consider that, the main task T is divided into 5 subtasks t_1, t_2, t_3, t_4, t_5 and there are seven resources R_1, R_2, R_3, R_4, R_5, R_6, R_7 available from the cloud. For each subtask our proposed algorithm helps to select the better resource based on the monetary cost and computational cost.

Hybrid optimization algorithm: Normally client desires to access the resources with minimum cost, so that the client will be benefited. Here, we consider monetary cost and computational cost for computing the total cost of the resources. For that we hybrid the two algorithms such as BCSO and BPSO to achieve the minimum total cost.

Each of the resources in the cloud assigns the Monetary Cost (MC) and Computational Cost (CC) for each subtask t_j. Table 1 and 2 represent the monetary cost and computational cost of *m* number of subtask with respect to *n* number of resources.

Figure 2 represents the work flow of the proposed hybrid optimization algorithm. This hybrid optimization algorithm contains two optimization algorithms such as

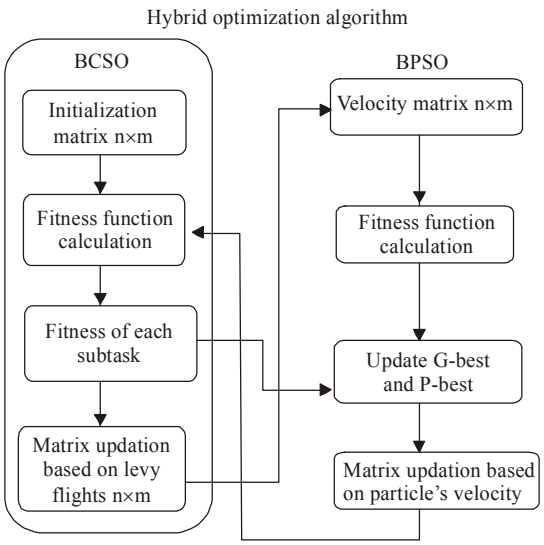


Fig. 2: Work flow of the proposed hybrid optimization algorithm

BCSO and BPSO. To begin with, the initialization matrix $n \times m$ is generated and assigned to the BCSO algorithm and then the fitness value is calculated to find the best resource for each subtask. The fitness value of the BCSO algorithm is assigned as initial G-best and P-best values of the BPSO algorithm. Now initialization matrix of the BCSO algorithm is updated via levy flights. After that, the updated matrix of the BCSO algorithm is handed over to BPSO algorithm as initialization matrix. Now at this time, the BPSO algorithm initializes its process by generating the velocity matrix randomly for its initialization matrix as the size $n \times m$. Subsequently BPSO find the best resource for each subtask through a fitness function. The arrived fitness value (P-best) is compared with the existing G-best value. If the fitness value is less than the G-best value, then G-best and P-best values are updated with the fitness value. Otherwise only P-best value is updated with the fitness value. The matrix $n \times m$ of BPSO is updated through the velocity calculation

Table 3: Parameters used

Symbol	Functionality of the symbol
n	Number of resources
m	Number of subtasks
MC	Monetary cost
CC	Computation cost
K	Iteration counter of BCSO
τ	Iteration counter of BPSO
K	Maximum number of iterations
$F(t^k_i)$	Fitness of j^{th} subtask at k^{th} iteration
CC_{ji}	Communication cost of j^{th} task with i^{th} resource
MC_{ji}	Monetary cost of j^{th} task with i^{th} resource

and particle's position updation. The cycle is completed after submitting the updated matrix of BPSO algorithm to BCSO algorithm as its initialization matrix. This process is repeated for *K* number of iterations. Finally, the best resource for each subtask is selected from the G-best value of BPSO which is given below. Parameters used are given in Table 3.

Algorithm:

Input: *m* number of subtasks, *n* number of resources, monetary cost, computational cost of *n* number of resources for *m* number of subtasks.
Output: Minimum cost of resource allocation for each subtask.

```
Begin
 BCSO algorithm
  Set k = 0
   Generate initialization matrix n×m
    For each particle
      Get the value of MC and CC
        Calculate fitness f (t^k_j) = MC_j+CC_ji
          Call Particle updation of BCSO
     End for
 Assign the fitness of each particle as G-best of BPSO algorithm

BPSO algorithm
Get the updation matrix from BCSO algorithm as initialization matrix n×m
   Set τ = 0
   Generate the velocity matrix of the initialization matrix n x m
       For each particle
         Get the value of MC and CC
           Calculate fitness f (t^k_j) = MC_ji+CC_ji
             Update G-best and P-best
               Call Particle updation of BPSO
      End for
End
```

Hybrid optimization algorithm:

```
Subroutine: Particle updation of BCSO
 Set k = k+1
 Calculate σ
 Calculate U and V
 Calculate step
        Calculate step size
        Apply the step size value of particle to sigmoid function
Generate rand (0, 1)
                If sigmoid function>1
                        Particle become 1
      Else
                        Particle become 0

Subroutine: Particle updation of BPSO
Set  τ = τ+2
Calculate Δv
Calculate updated velocity
Apply updated velocity to sigmoid function
Generate rand (0, 1)
If sigmoid function>1
                Particle become 1
   Else
                Particle become 0
```

BCSO and BPSO procedures are explained briefly.

Binary cuckoo search optimization: BCSO consists of four steps such as generation of initialization matrix, computing fitness function, particle updation and position updation which are explained in this section.

Initialization matrix: Initially the value of iteration counter k is assigned as zero. After the every iteration the value of the iteration counter gets increase by one. Generate the initialization matrix n x m where n and m represent, the number of resources and number of subtasks. In the initialization matrix if the resource R_i is allocated to the subtask t_j in k^{th}, it is represented as '1', otherwise '0'. An example initialization matrix is shown in Table 4.

Fitness function: To evaluate fitness of each cuckoo particle from the initialization matrix, the fitness function is computed. The fitness function includes the monetary cost and computational cost of the resources (dimension) for each subtask (cuckoo particle). Equation (1) is used to calculate the fitness value based on the monetary cost and the computational cost:

Table 4: Initialization matrix

	Resource					
Subtask	x/d	R_1	R_2	R_i	R_{n-1}	R_n
t^k_1	x^k_{1d}	1	0	0	0	0
t^k_2	x^k_{2d}	0	0	0	0	1
t^k_j	x^k_{jd}	0	0	1	0	0
t^k_{m-1}	x^k_{m-1d}	1	0	0	0	0
t^k_m	x^k_{md}	0	1	0	0	0

Table 5: Example initialization matrix

	Resource							
Subtask	x/d	R_1	R_2	R_3	R_4	R_5	R_6	R_7
t^0_1	x^0_{1d}	0	0	0	1	0	0	0
t^0_2	x^0_{2d}	1	0	0	0	0	0	0
t^0_3	x^0_{3d}	0	0	0	0	1	0	0
t^0_4	x^0_{4d}	1	0	0	0	0	0	0
t^0_5	x^0_{5d}	0	1	0	0	0	0	0

Table 6: Example initialization matrix with corresponding fitness

	Resource							
Subtask	x/d	R_1	R_2	R_3	R_4	R_5	R_6	R_7	Fitness
t^0_1	x^0_{1d}	0	0	0	1	0	0	0	1.486
t^0_2	x^0_{2d}	1	0	0	0	0	0	0	0.542
t^0_3	x^0_{3d}	0	0	0	0	1	0	0	1.286
t^0_4	x^0_{4d}	1	0	0	0	0	0	0	1.651
t^0_5	x^0_{5d}	0	1	0	0	0	0	0	1.812

$$f\left(t_j^k\right) = CC_{ji} + MC_{ji} \tag{1}$$

In the Eq. (1), the value of $f(t^k_j)$ represents the total cost of j^{th} subtask at the i^{th} resource in the k^{th} iteration.

As an example, in the Table 5, the subtask t_1 is assigned to the resource R_4. Based on this data, we select the values MC_{14} and CC_{14} as the value of the monetary cost and the computational cost of the subtask t_1 with respect to the resource R_4 and the computed fitness values are shown in Table 6.

The fitness of the each cuckoo particle is assigned as G-best to the BPSO algorithm for the further use. If the calculated fitness value of any particle is less than the existing fitness value, G-best value is updated with that of the calculated fitness value.

Cuckoo particle updation: In nature, animals search for food in a random or quasi-random manner. Generally, the foraging path of an animal is effectively a random walk because the next move is based on both the current location/state and the transition probability to the next location. The chosen direction implicitly depends on a probability, which can be modeled mathematically. Various studies have shown that the flight behavior of many animals and insects demonstrates the typical characteristics of Levy flights (Dhivya *et al*., 2011). A Levy flight is a random walk in which the step-lengths are distributed according to a heavy-tailed probability distribution. After a large number of steps, the distance from the origin of the random walk tends to be a stable distribution. Each dimension of the every cuckoo particle is updated through levy flights. The Eq. (2) to (6) are used to update the every dimension of each particle of the initialization matrix. The values of γ, β, rand are the random values between 0 and 1. The value of R (n) in the Eq. (3) represents the random value between 0 to n where n represents the total number of available resources to do the m number of subtasks:

$$\sigma = \left[\frac{\gamma(1+\beta)\sin\left(\frac{\pi * \beta}{2}\right)}{\gamma\left(\frac{1+\beta}{2}\right) * (\beta) * \left(2^{\frac{\beta-1}{2}}\right)}\right]^{\frac{1}{\beta}} \tag{2}$$

$$U = R(n) * \sigma \tag{3}$$

$$V = R(n) \tag{4}$$

$$step\left(x^k_{jd}\right) = \frac{U}{\sim V^{1/\beta}} \tag{5}$$

$$step\ size\left(x^k_{jd}\right) = rand * step \tag{6}$$

Here x^k_{jd} is the j^{th} particle position at the k^{th} iteration in the dimension d, where d represents the resource number, U and V are normally distributed stochastic variables with standard deviation (σ). Equation (6) represents the calculation of step size value necessary to update the dimension.

Position updation in BCSO algorithm: The dimension of the each cuckoo particle is updated based on step size value of that particle. The step size value of the each dimension of every cuckoo particle is applied into the sigmoid function, given in the Eq. (7) and the updated value becomes "0" or "1" through the condition which is represented in the Eq. (8):

$$S\left(step\ size\left(x^k_{jd}\right)\right) = \frac{1}{1 + e^{-step\ size\left(x^k_{jd}\right)}} \tag{7}$$

$$if\ \left[(rand(0,1)) < S\left(stepsize\left(x^k_{jd}\right)\right)\right]\ then\ x^k_{jd} = 1 \quad else\ x^k_{jd} = 0 \tag{8}$$

Table 7: Fitness for each swarm particle in BPSO algorithm

	Resource							
Subtask	x/d	R_1	R_2	R_3	R_4	R_5	R_6	R_7	Fitness
t^1_1	x^1_{1d}	1	0	0	0	0	0	0	1.480
t^1_2	x^1_{2d}	0	1	0	0	0	0	0	1.002
t^1_3	x^1_{3d}	0	0	0	1	0	0	0	0.283
t^1_4	x^1_{4d}	0	1	0	0	0	0	0	0.893
t^1_5	x^1_{5d}	0	0	0	1	0	0	0	0.578

Table 8: Values of G-best and p-best values for each iteration

	Iteration counter $\tau = 1$		Iteration counter $\tau = 2$	
Particles	G-best	P-best	G-best	P-best
x_1	1.486	1.486	1.480	1.480
x_2	0.542	0.542	0.542	1.002
x_3	1.286	1.286	0.283	0.283
x_4	1.651	1.651	0.893	0.893
x_5	1.812	1.812	0.578	0.578

While updating the particle in each iteration, if the subtask is assigned to more than 1 resource at a time, the fitness value of each resource is calculated first and then the resource having a minimum fitness value is selected.

Binary particle swarm optimization: There are four steps involved in BPSO. They are, generating initialization matrix, generating velocity matrix, computing fitness values, velocity updation and position updation which are explained in this section briefly.

Initialization matrix: The updation matrix n x m of the BCSO algorithm is given to the BPSO algorithm as initialization matrix. The BPSO algorithm process the received updated matrix from the BCSO algorithm to find the global best value (G-best) and local best (P-best).

Velocity matrix: The updating process of the BPSO algorithm is based on the velocity matrix n x m which is generated randomly as the same size of the initialization matrix. The generation of velocity matrix is completed before the calculation of fitness function. The velocity of the particle is represented by v^{τ}_{jd} in which j represents number of subtask, d represents dimension of the particle and τ represents the iteration counter value. The value of the iteration counter τ is initiated from 0 and is incremented by two (i.e., 0, 2, 4 and 6, respectively) every time.

Fitness function: The fitness function includes the monetary cost and computational cost of the resources for each subtask. The fitness function used in the BCSO algorithm is also used in BPSO algorithm as represented in the Eq. (1). Fitness values for the initialization matrix of the BPSO algorithm are given in Table 7.

Table 8 represents the values of G-best and P-best for two iterations. In the first iteration, the value obtained from the output of the BCSO algorithm, is assigned as the G-best as well as P-best of BPSO i.e., both G-best and P-best values are same initially. Now the value of τ is initialized to 0. The updation matrix of the BCSO algorithm is assigned as initialization matrix to the BPSO algorithm and the velocity matrix is generated randomly for the assigned initialization matrix. Subsequently the fitness value for each particle of the initialization matrix is calculated. If arrived fitness value is less than the existing fitness value, G-best and P-best values are updated with the current fitness value. Otherwise only the value of P-best gets updated with the arrived fitness value.

For example in Table 8, at the initial iteration the particle x_1 has the fitness value 1.486 and the value of G-best and P-best both becomes the same (1.486). But in the second iteration, the particle x_1 has the fitness value 1.480 which is less than previous iteration fitness value. Now G-best value becomes 1.480 and the P-best also becomes 1.480. Whereas in the initial iteration, the particle x_2 has the fitness value 0.542 and the value of G-best and P-best both becomes same (0.542). But in the second iteration the particle x_2 has the fitness value 1.002 which is higher than previous fitness value (0.542). So G-best is not changed (remains as 0.542) but the P-best value is changed into 1.002.

Velocity updation: After calculating the fitness of every particle of the matrix, the next step is to update particle position. Before that velocity of the particle should be updated. The Eq. (9) and (10) are used to update the (particle) velocity. w is used to control the impact of the previous velocities on the current velocity:

$$\Delta v^{\tau}_{jd} = c_1\, r_1 \left(pb^k_{jd} - x^k_{jd}\right) + c_2 r_2 \left(gb^k_{jd} - x^k_{jd}\right) \tag{9}$$

$$v^{\tau+2}_{jd} = w\left[v^{\tau}_{jd}\right] + \Delta v^{\tau}_{jd} \tag{10}$$

where,

Δv_{jd}^{τ} : Change in the Velocity of j[th] particle at τ[th] iteration in dimension d
c_1, c_2 : Acceleration constants
r_1, r_2 : Random numbers in the interval (0, 1)
pb_{jd}^{k} : P-best value of j[th] particle at k[th] iteration in dimension d
x_{jd}^{k} : Position of j[th] particle at k[th] iteration in the dimension d
gb_{jd}^{k} : G-best value of j[th] particle at k[th] iteration in dimension d
v_{jd}^{τ} : Updated Velocity of j[th] particle at τ iteration in dimension d
$v_{jd}^{\tau+2}$: Updated Velocity of j[th] particle at τ+2 iteration in dimension d
w : Inertia weight

Position updation in BPSO algorithm: With the help of updated velocity of each dimension of the particle, the BPSO algorithm updates the every dimension of the particle. The Eq. (11) and (12) are used to update the each dimension of the particle matrix. The updated matrix is given as the next input of BCSO algorithm:

$$S\left(v_{jd}^{\tau+2}\right)=\frac{1}{1+e^{-\left(v_{jd}^{\tau+2}\right)}} \tag{11}$$

$$if\ \left[\left(rand\ (0,1)\right)<S\left(v_{jd}^{\tau+2}\right)\right]\ then\ \ x_{jd}^{\tau}=1$$
$$else\ \ x_{jd}^{\tau}=0 \tag{12}$$

While updating the particle in each iteration, if the subtask is assigned to more than 1 resource at a time, the fitness value of each resource is calculated at first and then the resource having a minimum fitness value is selected. The proposed hybrid algorithm stops when a predefined number of iterations are completed.

IMPLEMENTATION AND RESULTS

The experimental results of the proposed technique for hybrid optimization algorithm for dynamic resource allocation are described in this section. In this study, we compare our proposed hybrid algorithm with BPSO algorithm and BCSO algorithm.

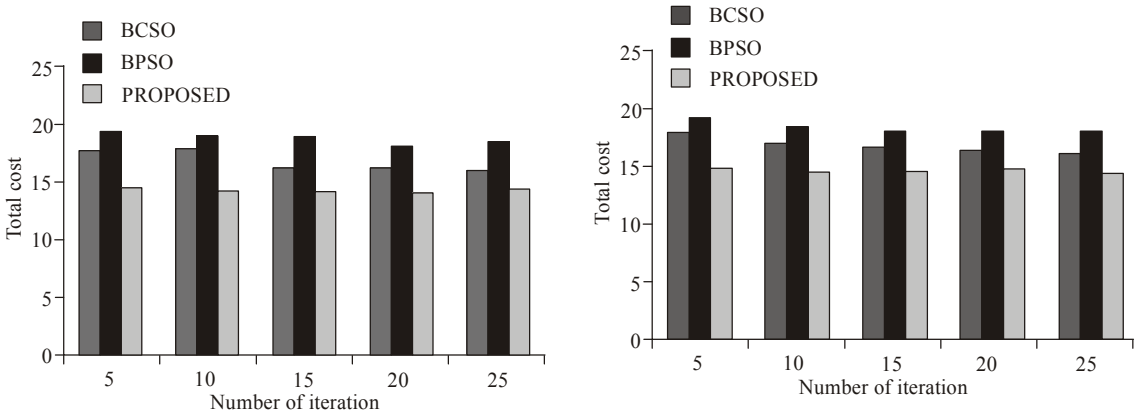
The proposed hybrid optimization algorithm is implemented with Cloudsim version 2.1.1. The experimentation has been carried out using the synthetic dataset with i3 processor PC machine with 4 GB main memory and 32 bit version of windows 7 operating system. The two synthetic datasets are generated in times of interval based on the number of subtask and available resources. Also each of the synthetic dataset consists of monetary cost and computational cost of each task with respect to all resources.

In this study, our main aim is to allocate the resource with minimum amount of monetary cost as well as computational cost. Here the total cost of the task is taken as the evaluation metrics. The total cost is calculated by the summation of fitness (G-best) value of every subtask. The evaluation is carried out with the total cost by varying the number resources and number of subtasks. Equation (13) is used to find the Total Cost (TC) of the task:

$$TC=\sum_{j=1}^{m} f\left(t_j\right) \tag{13}$$

The performance evaluation of the proposed hybrid optimization algorithm is prepared by comparing with the binary cuckoo search algorithm and binary particle swarm optimization.

Figure 3a and b represents the performance of the 25 number of resources and 20 numbers of subtasks



(a) (b)

Fig. 3: Cost estimation for 25 resources and 20 subtasks, (a) dataset D1, (b) dataset D2

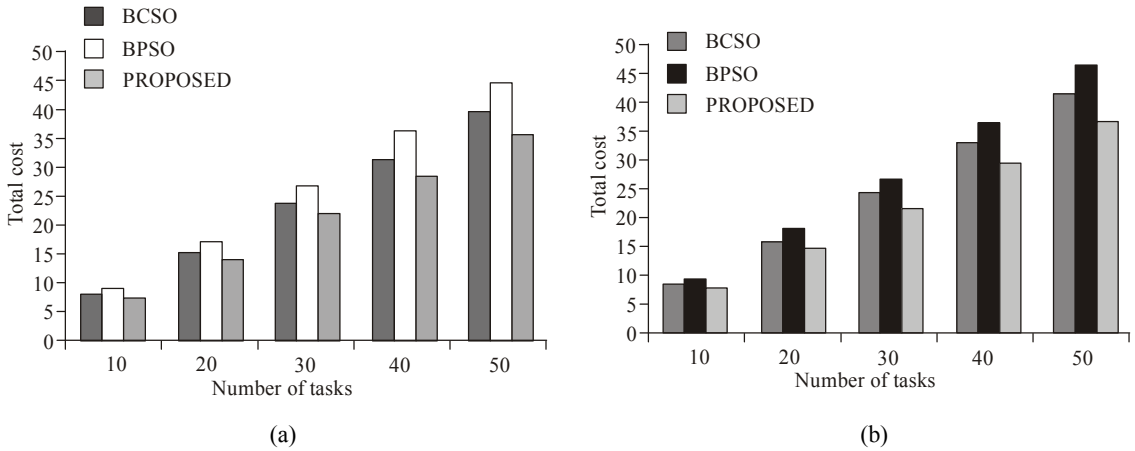


Fig. 4: Cost estimation for 25 resources (a) dataset D1, (b) dataset D2

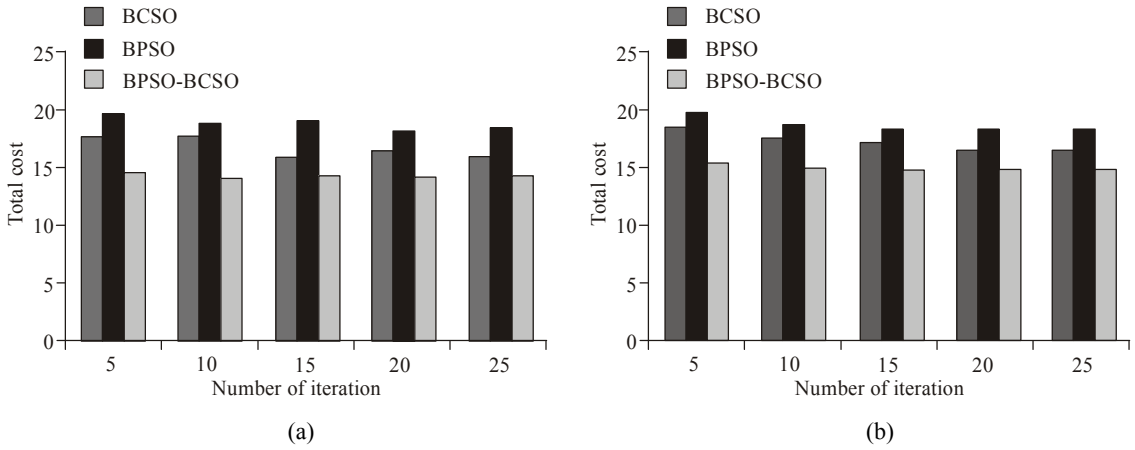


Fig. 5: Cost estimation for 25 resources and 20 subtasks, (a) dataset D1, (b) dataset D2

respectively. For the dataset D1, the proposed algorithm performed 14.87% less than BCSO and 24.14% less than BPSO. For the dataset D2, the proposed algorithm is 13.17% less than BCSO and 20.58% less than BPSO algorithm.

From the available results, it is concluded that the total cost of the proposed algorithm is less than the cost of BCSO algorithm and BPSO algorithm. At each level, the value of total cost is high in the BPSO algorithm when compared with other two algorithms. In both datasets D1 and D2, for a given number of resources and subtasks, as the number of iterations increase, the cost of the task reduces slowly in the case of BPSO algorithm and BCSO algorithm. But in the case of the proposed algorithm the cost remains almost constant. The proposed algorithm gives the minimum cost for the task "T" with minimum number of iterations by selecting the resources which has the minimum cost from the available resources during the updation process of particle at the every iteration.

Figure 4a and b represents the performance of the BCSO, BPSO and the proposed algorithm for 25 resources, 35 iterations for the datasets D1 and D2 respectively with varying subtasks. As the number of subtasks increases, the total cost of the task keeps increasing in all the three algorithms. At any level of subtasks, for a given number resources the cost achieved by the proposed method is less than the cost achieved by BCSO and BPSO. For the dataset D1, the proposed algorithm obtains the cost 10.80% less than BCSO algorithm as well as 21.10% less than BPSO algorithm, similarly for the dataset D2, the proposed algorithm is 10.51% less than BCSO and 19.76% less than BPSO algorithm.

Figure 5a and b represents the performance of the BPSO, BSCO and BPSO-BCSO algorithms for the datasets D1 and D2 with 25 numbers of resources and 20 number of subtasks respectively. In BPSO-BCSO algorithm, the output of BPSO is given as the input to BCSO to find the total cost which is a reverse procedure of the proposed hybrid algorithm.

Results obtained shows that, even if the hybrid combination is reversed the outcome is almost same as the proposed algorithm, since the G-best value is updated in each iteration.

Table 9: Average performance of the three algorithms

Number of resources	D1 dataset			D2 dataset		
	BCSO	BPSO	Proposed	BCSO	BPSO	Proposed
25	23.84	26.96	21.27	24.48	27.31	21.91
50	22.99	26.84	20.46	23.73	26.92	21.15
100	22.64	26.54	19.84	23.44	26.83	20.05

Table 9 represents the average performance of the proposed hybrid algorithm, BCSO and BPSO algorithms for the numbers of resources, n = 25, 50 and 100 and 35 number of iterations.

From the Table 9, it is predicted that, if the number of resources are increased, the total cost reduces. The proposed algorithm also demonstrates the robustness of the results obtained.

CONCLUSION

We proposed a hybrid optimization algorithm for dynamic resource allocation in cloud computing. Initially, the BCSO algorithm generates the initialization matrix as per the number of subtasks and resources. Subsequently it calculates the fitness value for each particle of the initialization matrix based on the monetary cost and computational cost. The fitness of the particle is assigned to BPSO algorithm as G-best and P-best, then the initialization matrix of the BCSO algorithm gets updated through the levy flights and sigmoid function and then updated matrix is assigned to BPSO algorithm as initialization matrix. The BPSO algorithm generates the velocity matrix randomly for the initialization matrix and then calculates the fitness value. Based on the arrived fitness value, the values of G-best and P-best are updated. The BPSO algorithm updates the matrix through the updated velocity and sigmoid function. The updated matrix is assigned then to BCSO algorithm. This cyclic process is repeated up to *k* number of iterations. The experimentation is carried out and the results of the proposed hybrid algorithm are compared with that of the BCSO and BPSO algorithm. The proposed hybrid algorithm gives better cost optimized results compared to BCSO and BPSO algorithms. The hybrid combination of other evolutionary algorithms can be carried out for further scope.

REFERENCES

An, B., V. Lesser, D. Irwin and M. Zink, 2010. Automated negotiation with decommitment for dynamic resource allocation in cloud computing. Auton. Agent. Multi-Ag., 1(1): 981-988.

Babukartik, R.G. and P. Dhavachelvan, 2012. Hybrid algorithm using the advantage of ACO and cuckoo search for job scheduling. Int. J. Inform. Technol. Convergence Serv., 2(4): 25-34.

Beloglazov, A., J. Abawajy and R. Buyya, 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comp. Sy., 28: 755-768.

Brajevic, I., M. Tuba and N. Bacanin, 2012. Multilevel image thresholding selection basedon the cuckoo search algorithm. Proceeding of the Advances in Sensors, Signals, Visualization, Imaging and Simulation (VIS, 2012), pp: 217-222.

Dhivya, M., M. Sundarambal and L.N. Anand, 2011. Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach. Int. J. Commun. Network Syst. Sci., 4: 249-255.

Eberhart, R.C. and J. Kennedy, 1995. A new optimizer using particle swarm theory. Proceeding of 6th International Symposium Micromachine Human Science. Nagoya, Japan, pp: 39-43.

Gogulan, R., A. Kavitha and U.K. Kumar, 2012. An multiple pheromone algorithm for cloud scheduling with various QOS requirements. Int. J. Comput. Sci. (IJCSI), 9(3).

Goudarzi, H. and M. Pedram, 2011. Maximizing profit in cloud computing system via resource allocation. Proceeding of 31st International Conference on Distributed Computing Systems Workshops (ICDCSW, 2011), pp: 1-6.

Ho, S.Y., H.S. Lin, W.H. Liauh and S.J. Ho, 2008. OPSO: Orthogonal particle swarm optimization and its application to task assignment problems. IEEE T. Syst. Man, Cy. A, 38(2): 288-298.

Jati, G.K., H.M. Manurung and S. Suyanto, 2012. Discrete cuckoo search for traveling salesman problem. Proceeding of 7th International Conference on Computing and Convergence Technology (ICCCT, 2012), pp: 993-997.

Kennedy, J. and R.C. Eberhart, 1995. Particle swarm optimization. Proceeding of IEEE International Conferences on Neural Networks, Perth, Australia, 4: 1942-1948.

Kennedy, J., R.C. Eberhart and Y.H. Shi, 2001. Swarm Intelligence. Morgan Kaufmann, San Mateo, CA.

Layeb, A. and S.R. Boussalia, 2012. A novel quantum inspired cuckoo search algorithm for bin packing problem. Int. J. Inform. Technol. Comput. Sci., 4(5): 58-67.

Noghrehabadi, A., M. Ghalambaz, M. Ghalambaz and A. Vosough, 2011. A hybrid power series-cuckoo search optimization algorithm to electrostatic deflection of micro fixed-fixed actuators. Int. J. Multidisciplinary Sci. Eng., 2(4): 22-26.

Onat, Y.Y., C. Matthews, F. Roozbeh and S. Neville, 2010. Dynamic resource allocation in computing clouds through distributed multiple criteria decision analysis. Proceeding of IEEE 3rd International Conference on Cloud Computing (CLOUD), pp: 91-98.

Pawar, C.S. and R.B. Wagh, 2012. A review of resource allocation policies in cloud computing. J. Sci. Technol., 2(3): 165-167.

Rani, K.N., M.F. Malek and N.S. Chin, 2012. Nature-inspired cuckoo search algorithm for side lobe suppression in a symmetric linear antenna array. Radioengineering, 21(3): 865-874.

Stillwell, M., D. Schanzenbach, F. Vivien and H. Casanova, 2010. Resource allocation algorithms for virtualized service hosting platforms. J. Parallel Distr. Com., 70(9): 962-974.

Taşgetiren, M.F. and Y.C. Liang, 2003. A binary particle swarm optimization algorithm for lot sizing problem. J. Econ. Soc. Res., 5(2): 1-20.

Valian, E., S. Mohanna and S. Tavakoli, 2011a. Improved cuckoo search algorithm for feed forward neural network training. Int. J. Artif. Intell. Appl., 2(3): 36-43.

Valian, E., S. Mohanna and S. Tavakoli, 2011b. Improved cuckoo search algorithm global optimization. Int. J. Commun. Inform. Technol., 1(1): 31-44.

Warneke, D. and O. Kao, 2011. Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. IEEE T. Parall. Distr., 3(3): 1-3.

Xiao, Z., W. Song and Q. Chen, 2013. Dynamic resource allocation using virtual machines for cloud computing environment. IEEE T. Parall. Distr., 24(6): 1107-1117.

Yang, X.S. and S. Deb, 2009. Cuckoo search via levy flights. Proceeding of World Congress on Nature and Biologically Inspired Computing. India, pp: 210-214.

Yang, X.S. and S. Deb, 2010. Engineering Optimization by Cuckoo Search. Int. J. Math. Model. Numer. Optim., 1(4): 330-343.

Yin, B., Y. Wang, L. Meng and X. Qiu, 2012. A multi-dimensional resource allocation algorithm in cloud computing. J. Inform. Comput. Sci., 9(11): 3021-3028.

Zhan, Z.H., J., Zhang, Y. Li and H.S. Chung, 2009. Adaptive particle swarm optimization. IEEE T. Syst. Man Cy. B, 39(6): 1362-1381.

Zhang, Q., L. Cheng and R. Boutaba, 2010. Cloud computing: State-of-the-art and research challenges. J. Int. Serv. Appl., 1(6): 7-18.

Zhang, Q., Q. Zhu and R. Boutaba, 2011. Dynamic resource allocation for spot markets in cloud computing environments. Proceeding of 4th IEEE International Conference on Utility and Cloud Computing (UCC, 2011), pp: 178-185.