# Swarm intelligence-based task scheduling algorithm for load balancing in cloud system

D. Komalavalli*

Bharathiar University,
Coimbatore, India
Email: komsdayalan@gmail.com
*Corresponding author

T. Padma

Department of Master of Computer Applications,
Sona College of Technology,
Salem, India
Email: padmatheagarajan@gmail.com

**Abstract:** As on miniature devices to military applications, the cloud computing plays a vital role. Building efficient cloud management systems will lead to improve extraordinary features in the applications of cloud services. In cloud atmosphere, enormous tasks are performed simultaneously; an effectual task scheduling is very important role to get better performance of the cloud system. An assortment of cloud-based task scheduling algorithms is offered that schedule the user's task to resources for implementation. The innovation of cloud computing, conventional scheduling algorithms cannot gratify the cloud's requirements, the researchers are frustrating to modify conventional algorithms that can accomplish the cloud needs similar to rapid elasticity, resource pooling and on-demand self-service. Also, the priority becomes an important task when dealing with critical functionality systems. Real world invocations are needed to make an efficient selection in cloud services through a collection of functionally equivalent services. This research aims to detect a novel method to predict the system functionality without consuming more time and less expensive for implementation. Investigation on swarm intelligence-based task scheduling is presented. This will improve the power consumption by reducing overloads when more services opt for a single load. Experiments were carried out to test the effectiveness of the method.

**Keywords:** task scheduling; particle swarm optimisation; PSO; bat algorithm; swarm intelligence; cloud computing; load balancing; low power.

**Reference** to this paper should be made as follows: Komalavalli, D. and Padma, T. (2021) 'Swarm intelligence-based task scheduling algorithm for load balancing in cloud system', *Int. J. Enterprise Network Management*, Vol. 12, No. 1, pp.1–16.

**Biographical notes:** D. Komalavalli is a Research Scholar at the Bharathiar University, Coimbatore, India. She received her Postgraduate in Computer Applications from the Indira Gandhi National Open University, India and Computer Science and Engineering from the Anna University, Coimbatore. Her research interests include cloud computing, virtualisation and swarm intelligence. She has nine years of teaching experience. She is a life member of Indian Society for Technical Education.

T. Padma is a Professor and the Head at the Department of Computer Applications, Sona College of Technology, India. Her research interests include machine learning, data analytics and interaction design. She has executed research projects as a Principal Investigator funded by AICTE, UGC and NCW; published around 60 journal papers, books and book chapters. She is very instrumental in setting up the institution's management information system; Certified Six Sigma Black Belt; recipient of Shayesta Akhtar Memorial National Award of the ISTE in 2015. Her biography was included in the 2010 edition of the Marquis Who's Who in the World, New Jersey, USA.

## 1 Introduction

Several IT enterprises in the vein of IBM, Microsoft, Google, Amazon and alike provides subscription access to its network and software services. They offer cloud services to their customers through cloud IT infrastructure. The cloud service vastness creates difficulty among the customer's point of view to choose and no framework exist to allow customers to evaluate the cloud services provided by the IT enterprises. The main problem will be on the understanding the user requirement towards quality of service (QoS). Success of cloud computing has made cloud technology research into setting a new age. Usually the applications and methodology designed for cloud computing are larger and complicated. There is no best solution for the addressed problem in hand. Since in a single application multiple users and equipment's are connected in large-scale the issue is getting complex. The entity involved in the cloud are the software been shared between the users, hardware holding the software or controlled by the software. Energy utilisation is developing as a novel key issue for cloud computing atmospheres like data centres. Especially in the sense of scientific workflow implementation in the cloud, the issue of power consumption is more challenging as they cause severe computational tasks and data management measures which produce excessive information movement operations through communiqué networks. For example, network devices have been reported to consume up to one-third of cloud data centre's total energy consumption.

The paper contributes towards designing of a hybrid swarm intelligence-based task scheduling algorithm for load balancing in cloud system, using the principles of particle swarm optimisation (PSO) and bat algorithms and analyses it. Load balancing eliminates and reduces the power consumed by unwanted or idle processing of a load. The nature inspired swarm optimisation methods enhance the performance of load balancing process.

## 2 Literature survey

In literature, many works has been carried out in cloud computing, its methods and applications. The next section deals with the detailed survey of various research carried out in past.

Agarwal and Raina (2012) proposed a scheme called open cloud migration of virtual machines (VMs). A VM migration simply moves the VM running on a physical device to another device. In order to increase the system's throughput, the VMs need to be

statically balanced in load, i.e., the load is distributed to each part of the system in proportion to their IO computing capacity.

Xing et al. (2016) proposed a scheme for power communication networks called a load balancing routing optimisation mechanism. It is a difficult to face effectively reduces the threat of dissimilar services to get better operation consistency in this type of networks. Service route distribution is a vital key factor in reflecting communication risk. Conversely, present routing algorithms never consider the degree of service value, resulting in unbalanced loading and increased risk of services and networks. Consequences of model on practical network topology prove to facilitate the method can be able to optimise the network threat.

Kaur and Rani (2015) proposed a scheme in cloud computing called virtual migration. This method assigns the computing tasks from a wide number of computers to the resource pool. Live migration allows the migration of VMs without significant downtime. A VM's transition actually refers to its state's transfer. It includes its memory, devices' internal state, and virtual CPU status. The most time-consuming of these is the transition of memory. During the live VM migration, two parameters are known to be downtime and migration time.

Bousselmi et al. (2016) proposed a scheme for systematic workflows in the cloud called energy proficient partition and scheduling method. Workflow partitioning for energy minimisation (WPEM) algorithm is proposed to facilitate the network energy utilisation of the workflow and the whole sum of data communication be minimised at the same time maintaining a high degree of parallelism.

Carrera et al. (2012) introduced the mixed batch and transactional workloads autonomous placement technique. These workloads often include both long-running and transactional analytic computations. In this paper, a methodology is proposed for long running jobs and transactional applications that allows existing middleware to equally handle mixed workloads. In the experimental results, including simulations shows that the methodology exploits mix workload efficiency at the same time as provided that product discrimination based on lofty performance objectives.

Li et al. (2016) proposed a dynamic load balancing algorithm in smart grid systems that would implement water-filing approach. Patel et al. (2014) have discussed about different techniques related VM migration in cloud.

König et al. (2011) proposed a framework called cloud infrastructure elastic monitoring framework. Scalable and elastic distributed network based on peer to peer architecture to control the cloud infrastructure. Its distributed nature allows long-lived queries to be deployed across the network to track a number of entities and metrics across all layers of a cloud stack that can rapidly change.

Li et al. (2012) proposed a scheme in an untrusted control environment called a secure VM. Jadhav and Deshpande (2014) discussed about dynamic balancing techniques to balance the load in cloud systems. Balancing the load in cloud systems is the central issues in distributed cloud environment.

Han et al. (2018) dynamically scheduled the tasks which are cached and then allocated the resources. Even though virtual memories are used in this process, the waiting time and memory usage are more.

Yin et al. (2018) explains about the scheduling methods are similar to genetic algorithm and multiple fitness functions (FFs) are required for convergence problems.

Tang et al. (2018) explained scheduling task in mobile cloud systems called as an NP hard problem, which have concerned a lot of efforts in the previous days. This proposal deals with the problem as an optimised energy utilisation problem, even if fascinating into description of task dependency, data transmission and a number of constriction circumstances such as response time deadline and cost, similarly resolve it with the aid of genetic algorithms.

Schäfer et al. (2018) presented a hybrid scheduling approach for cloud and edge computing. This structural design supports essential task scheduling to remote cloud and edge resources and a decentralised scheduling to nearby edging environments. But the remote edge systems were complicated in the online live process.

Comer and Karandikar (2019) proposes VM migration system based on data centric method. It preserves compatibility through existing hardware by means of the similar frame design as standard protocols. It does not need any modifications to applications and host operating systems or libraries. It presents the structural design of the proposed work illustrates a test bed used to assess DCnet and details the experimentation measurements. First come first serve (FCFS) principle is used in this work to schedule the process.

Wang et al. (2019) implemented path planning algorithms based on DPSO algorithms. Specific FFs are premeditated to meet up the various requirements of tactic intensions and needs of time and accuracy. The parameter selection method for PSO-based algorithm and the cooperation between a team of heterogeneous UAVs (i.e., speed difference, sensor difference) is to be considered.

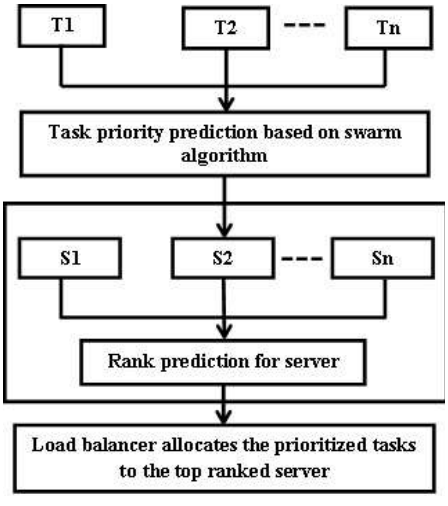
Shi et al. (2019) proposed a proficient digital modulation recognition method supports on profound neural network (DNN) model. Transformed signals are first pre-processed in this model as the input to the neural network. Consequently, PSO algorithm is concerned to optimise the structure of DNN. The time complexity is elevated in the case of large training datasets or constituent part quantity and this technique will leads to low performance when applied to unknown modulated signals.

Phuong et al. (2019) implemented a new automatic clustering algorithm for categorical data. This system is examined to recognise the highest number of clusters in a dataset stands on the local density. This work can be extended by developing the significant attributes which are represents for each cluster and outliers can be measured to be progressed before clustering.

## 3 Structure of the proposed system

The structure of the proposed system is shown in Figure 1. In this method is divided into three segments namely task scheduler, rank predictor and load balancer. The task scheduler has n number of tasks (t1, t2, …, tn) and the tasks are sending the load request to task scheduler swarm algorithm. The scheduler classifies the task load based on their priority vales, it predicted by swarm algorithm. Then the priority load tasks are assigned to the rank prediction server. The rank prediction server has S1, S2, …, Sn. The prioritised task loads are sending as an input of rank prediction servers. The rank prediction servers sends their ranking tasks output into the load balancer. The load balancer allocates prioritised tasks load to the top ranked server with QoS parameter. In this load balancing task schedule method is execute by first come and first serve manner.

**Figure 1** Structure of the proposed system




## *3.1 Task scheduling using nature-based computing*

### *3.1.1 Problem formulation in task scheduling*

The problem behind the task scheduling process was the mapping of tasks to its resources in a distributed manner. The ultimate aim was to define a clear workflow of an application with reduced cost of computation. Through the directed acyclic graph (DAG) the allocation of task and its work flow has been explained clearly. Consider that the graph $G = (A, D)$ where $A$ was the representation of set of tasks $\{X_1, X_2, \ldots, X_n\}$ and $D$ was the indication of data dependencies present among the tasks which is the $a_{i,j} = (X_i, X_j) \in D$. It was defined as that the data produced by $X_i$ was the input of $X_j$. The approach also contains the storage space $B = \{1, \ldots, n\}$, resource $VM = \{1, \ldots, k\}$. It was also considered that the average estimation for completion of a task to be $T_1$ when it was processed on the resource $VM_k$. Since the total computation cost of a task to a resource was conversely related to the time taken to complete a task therefore, the cost was assigned as to be $g_{i,j}$. It was also assumed by Pandey et al. (2010) that all these cost were non-negative and symmetric values, therefore it satisfies inequality triangle problem as $p_{i,j} = p_{j,i}$ for all $i, j \in N$ and $p_{i,j} + p_{j,k} \geq p_{i,k}$ for all $i, j, k \in N$.

The work flow process was related to the evolutionary multi-objective optimisation (EMO). The structure of workflow comprises of five different tasks and each has the different input file for access. Consider that the input file for the root task to be (a.in) and the final output file was (a.out). Each and every task gets processed and produces output file as $a_{12}, a_{13}, \ldots, a_{45}$. These numerical values are the representation of edge weight of nodes $(n_1, n_2)$ involving two tasks where $n_1 \in X$ and $n_2 \in X$. The resource are denoted as $VM_1$, $VM_2$, $VM_3$ with the buffer storage unit as $(B_1, B_2, B_3)$. It was assigned as the total completion cost of all task for the resource $VM_j$ after execution as to be $G_{comp}(M)_j$ denoted in equation (1).

The total access cost among the task which was allotted to the resource $VM_j$ as $G_{tx}(M)_j$ described in equation (2). The estimated cost of unit data transmission between

two entities is denoted as $M(n_1)$, $M(n_2)$ and the two tasks of node $n_1$ and $n_2$ are mapped. This work illustrates that when there are dependencies present among the two tasks, the communication cost can be computed $n_1$, $n_2 > 0$. Meanwhile the communication cost is 0 when there are two or more tasks are in need to be processed simultaneously.

$$G_{comp}(M)_j = \sum\nolimits_k X_{kj} \quad \forall M(k) = j \tag{1}$$

$$G_{tx}(M)_j = \sum\nolimits_{k_1 \in T} \sum\nolimits_{k_2 \in T} dM(k_1), M(k_2) ek_1, ek_2 \quad \forall M(k) = j \text{ and } M(k_2) \neq j \tag{2}$$

$$G_{tot}(M)_j = G_{comp}(M)_j + G_{tx}(M)_j \tag{3}$$

$$Cost(M) = Max\left(G_{tot}(M)_j\right) \quad \forall j \in p \tag{4}$$

$$Minimise\left(Cost(M) \forall M\right) \tag{5}$$

Each and every input task is allotted evenly to all resource and it is represented in equation (4). The equation (5) ($Minimise(Cost(M)\forall M)$) represents to find the minimum cost for the completion of distribution. The entire total cost, after completion of task allocation is related to resource $VM_j$, and the sum of execution cost and access cost is calculated in equation (3). After calculating the total cost, the highest cost is identified and reduced. This equation (5) represents that not more than one task is mapped to single resource.

### *3.1.2 Scheduling based on PSO*

Pandey et al. (2010) devised the task scheduling process by relating the PSO approach. This ensures that the optimisation was achieved by using the heuristic scheduling method for the task-resource mapping which was related to the PSO method. There are two stages developed for scheduling. First stage is scheduling based on heuristic method and the second stage is for optimisation in task resource mapping based on PSO.

For the task scheduling several methods were adopted in this work. A detail investigation on three methods has been carried out.

### *3.1.3 PSO-based heuristic methodology*

PSO is the swarm intelligence technique based on social behaviour of animals. It is similar to how birds are searching food source (or) like group of fish preventing themselves from an attack. This is because the particles in PSO are matched up with a bird flying for the search (problem) of food resource and the fish trying to prevent from the predator attack. This method is mainly based on relating the velocity of each particle which holds both the magnitude and direction. The particle positions are mainly affected by the best position of particle and its location in a problem space. Finally, all particles are evaluated by a fitness value, when it experiences the problem. Moreover, the PSO algorithm is alike as other algorithms. Here the populations are considered to the number of particles and it is also random process. The particles are evaluated by a fitness value, when it experiences the problem. Each particle contains an aptness value obtained from the aptness function from each formation. Mean square error (MSE) function is used as

FF when the function call is initiated. After measuring the fitness value, the value to the dimensions of the particles has been assigned based on the resource. Through these values the mapping of task to a resource takes place easily.

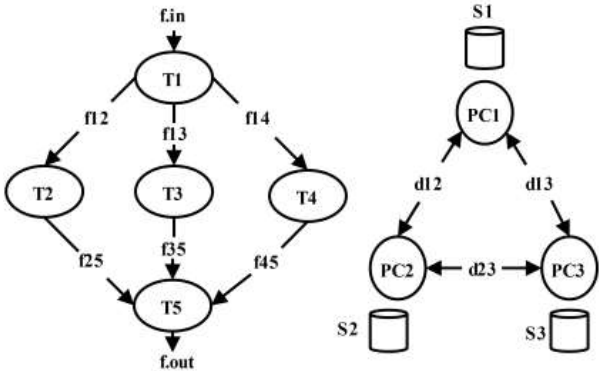
**Algorithm 1** Scheduling based on heuristic method

Estimate the average cost of computing to all resources for all tasks.
Measure the average cost (communication / size of data) among the resource.
Consider that the average computation cost of task node weight as $wk_j$.
Declare the edge weight $n_1$, $n_2$, as file size transmitted among the task.
Calculate $PSO(\{t_i\})$
Do
  for all 'ready' task $\{x_i\} \in A$: Do
    Allot task $\{x_i\}$ to resource $\{VM_j\}$ based on the solution obtained from PSO.
  end-for
  Dismiss all mapped task.
  Wait for the polling time.
  Modify the list of tasks prepared.
  Modify the average resource communication cost based on the actual load.
  Determine $PSO(\{t_i\})$.
Until the presence of unscheduled tasks.

### *3.1.3.1 Scheduling heuristic*

In order to identify the fast-completing resource, the average computation cost for all the resource allocated tasks are estimated (allotted as node weight in Figure 2). It was easy to compute cost of the resource since the computing cost of a task is conversely related to the time taken to complete. The greater resource cost would complete the task quickly through the calculation. Therefore, it was considered that for each task the input size and output value to be edge weight represented as $n_1$, $n_2$, in Figure 2.

**Figure 2** Block diagram of the PSO-based heuristic



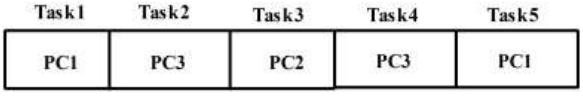
Mapping of tasks done and computation of PSO is done in initial step. The overall cost is optimised. PSO validates the 'ready' tasks to resources based on mapping. Once dispatching is done the scheduler enters pooling. The ready list on run is updated similarly the current network load. The PSO mappings are recomputed balancing of heuristic mapping. Tasks in loaded state are allotted to resources based on recomputed mappings of PSO. These steps are repeated.

**Algorithm 2** Resource mapping based on PSO

| | |
|---|---|
| 1 | The dimension of particle is equated on size of the ready task in $\{x_i\} \in A$. |
| 2 | Start the particles position to be $VM = \{1, \ldots, j\}$ and velocity $V_i$ in random process. |
| 3 | Measure the value of fitness function. |
| 4 | Define the value of present fitness as new pbest, only if value of the fitness is best than the previous better pbest value. |
| 5 | Choose gbest as preeminent particle. |
| 6 | Measure and update all particles' velocity and its positions. |
| 7 | If maximum iteration is not fulfilled or criteria for stopping are met, repeat the steps from step 3. |

This PSO algorithm is an active (online) process and updates the communication cost for each scheduling process. Since the approach is dynamic process it gets optimised depending upon the current or present network and source availability. In the above PSO algorithm, the process initialisation begins in random process with the position of the particles and its velocity. The scheduling problem in a workflow is addressed by assigning particles as tasks allocated and the particle size is represented as total number of tasks. After measuring fitness, the value of the dimensions to the particles has been assigned based on the resource. Through these values the mapping of task to a resource takes place easily. From Figure 2 it was observed that the particles are 5D as there are five tasks and resource assigned to the task are the dimensions of the particles which were depicted as shown in the Figure 3. From the equation (5) the FF is used to process the particle with the minimum cost for the completion of distribution. The velocity measure and position update of the particles are made by using equations (6) and (7). The final verification was carried out till the particular numbers of iterations are reached.

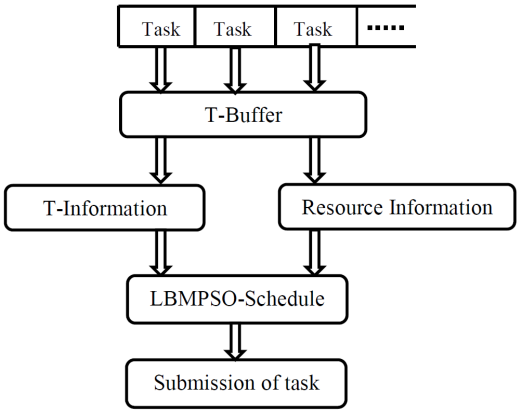
**Figure 3** Workflow of particle sample




### *3.1.4 Task scheduling using load mutation using enhanced PSO*

Awad et al. (2015) discussed about the development of scheduling of task process on cloud environments by improved PSO approach. To overcome the task scheduling issues the authors developed a model that comprised of five stages. The five phases of the model are represented in Figure 4 the stages involved are:

**Figure 4** Block diagram of the load mutation PSO




#### *3.1.4.1 Task buffer (T-buffer)*

Task buffer is the managing unit that controls all the incoming tasks from the users. Since there are a greater number of users were in need of implementing their task in cloud environment, the task buffer monitors the network traffic by collecting the task from the user's request.

#### *3.1.4.2 Task information (T-information)*

In this stage, the information regarding expected-execution-time (EET) and expected-transmission-time (ETT) are calculated. Resources required (RR) and round-trip-time (RTT) are also analysed.

#### *3.1.4.3 Resource information*

The resource information is the third stage that gathers entire data present in cloud environments. The information like datacentre, hosts and VMs are collected. The price of memory, band width (BW) and additional data are also the datacentre information. More than one VM is able to allocate to a single host. The resource information block transmits the data's like RAM, MIPS, bandwidth and other information to the next stage of the developed model.

#### *3.1.4.4 LBMPSO*

Load balancing mutation PSO was the main block of the model. The ultimate focus of this phase was to reschedule the failure tasks. This type of PSO reschedules the failed tasks. In conventional PSO problem arises on task allocation failure to VM and single task to multiple VM allocation happens. This problem solved by first rescheduling and then VM load balancing. By this reliability increases, avoids task failures and minimise execution and round-trip time.

#### 3.1.4.5 Task submission

After receiving the allocation plan from previous phase as per the plan the VMs are allocated.

#### 3.1.4.6 Formulation of problem in scheduling of tasks

Consider n number of tasks as $t$ and $m$ number of VMs and mapping of tasks $t_1 \ldots t_n$ allocated to $VM_1 \ldots VM_m$ VMs is made as shown in Figure 5. Since task to VM allocation is mono, the PSO optimises the task distribution to VMs. The modelling has an objective function (OF) to reduce completing time which is calculated on EET ($EET_{ij}$) of task $i$ in $VM_j$.

The expected execution time ($EET_{ij}$) is $length_i$ / $mips_j$.$length_i$.

Where $mips_j$ is the number of instructions executed per second by $VM$.

Second OF is to reduce the time of transmission ($ETRT_{ij}$).

$$Expected\text{-}Transmission\text{-}Time\left(ETRT_{ij}\right) = file\ size / bandwidth$$

$ERTT_{ij} = ETRT_{ij} + delay + EET_{ij} + delay$. $x_{ij}$ is allocating $task_i$ to $VM_j$ or not. The value of $x_{ij}$ may be 1 or 0.

A mathematical model through EET:

$$Min\ z = \sum_{i=0}^{n} \sum_{j=0}^{m} EET_{ij} * x_{ij} \tag{6}$$

Subject to:

$$\sum_{j=0}^{m} X_{ij} = 1 \quad \forall i \tag{7}$$

$$\sum_{j=0}^{m} cpu_j \leq total_{cpu} \tag{8}$$

$$\sum_{j=0}^{m} mem_j \leq total_{mem} \tag{9}$$

$$x_{ij} \geq 0 \quad \forall i, j \tag{10}$$

Second mathematical model:

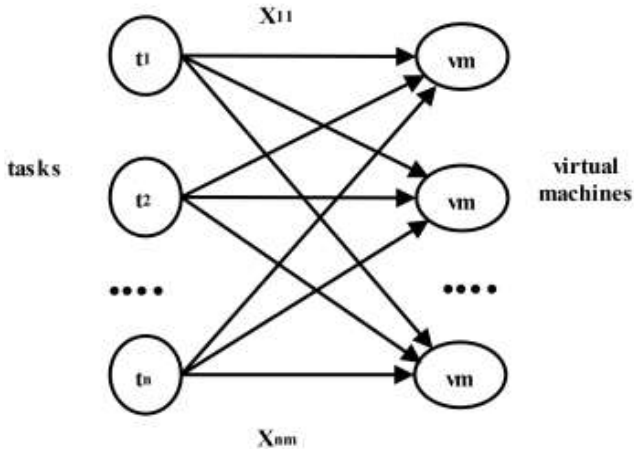
OF – ETT:

$$Min\ z = \sum_{i=0}^{n} \sum_{j=0}^{m} ETRT_{ij} * x_{ij} \tag{11}$$

OF – expected-round-trip-time:

$$Min\ z = \sum_{i=0}^{n} \sum_{j=0}^{m} ERRT_{ij} * x_{ij} \tag{12}$$

**Figure 5** Block diagram of the PSO-based heuristic




## 4 Proposed methodology – improvised bat algorithm

In the proposed methodology, the improved BAT algorithm is used to classify and reroute the cloud user requests based on its carbon emission level instead of classifying the cloud based on its QoS. The FF is allowed with respect to the speed of execution. The FF determines priority and load for the servers are made as per the priority value.

1 This method is functioning on the novel meta-heuristic method.

2 Using the min-min and max-min algorithm to construct an improved and optimised population of virtual bats and also the improvement is prepared with the process.

3 Min-min algorithm: it determines the minimum completion time of the task and then chooses the minimum value from them. Then the source takes minimum execution time based on the schedule and the time available is in addition to all other activities.

4 Max-min algorithm: it calculates the tasks' maximum completing time and then chooses the minimum value from them. Then the source's schedules have a fixed execution time, and the time available is addition to all.

5 Generation of population creates the population based on scheduling.

6 Enter the time and availability of each node for processing.

7 The additional preferable population is created for enhanced better results by implementing the min-min and max-min algorithms. For each node ($x_i$), pulse frequency ($f$) is initialised now; $f$ lies in $[0, f_{max}]$.

8 Pulse rates ($r$) and loudness ($A_i$) are too initialised.

   a '$r$' lies between [0, 1] – 0 represents empty pulse and 1 represents maximum pulse.

   b '$A_i$' lies between $A$[0 – 1] and $A$[min – 0].

9 The bat algorithm is currently being implemented until final result is achieved. It works on the echolocation possessions and the frequency of ultrasonic waves they emit.

10 It is designed to measure their target as well as any difficulties in the flying path by the level of emissions and waves they generate for assistance. It reduced loudness or sound frequency created by achieving their prey and temporarily stopping the wave emission.

---

**Pseudo code**

Objective function: $f(x) = [x_1, x_2, \ldots, x_d]^t$

**STEP 1:** Initialisation of population $X_i$: where $i = 1, 2 \ldots n$ and $V_i$ as velocity.

**STEP 2:** Apply the algorithms min-min and max-min.

For further process, reconsider the population.

**STEP 3:** Initialise the pulse rate as $r_i$ and loudness as $A_i$.

**STEP 4:** Pulse frequency $f_i$ is initialised to each node $X_i$.

for ($t <$ max no. of population)

Produce the solution by adjusting the frequency.

Based on the location/solution [from equations (2) and (4)] update the velocity.

if ($r_i <$ rand)

Find the optimum solution among the best solutions.

From the selected solution, a local solution must be entered.

end if

if ($rad. < A_i$ & $f(x_i) < f(x^*)$) Obtain

the new solution

and reduce $A_i$.

end-if

Arrange the bats and get the best $x^*$ currently available.

End-for

Visualisation and post process result.

Movement of virtual bats:

$F_i = f_{\min} + (f_{\max} - f_{\min}) * b$

$V_i(t) = V_i(t-1) + (x_i(t) - x^*) * f_i$

$x_i(t) = x_i(t-1) + V_i(t)$

$b = [0, 1]$

New solution:

$X_{new} = X_{old} + eA(t)$

$e = [-1, 1]$

$A(t) =< A(t) >$ – average loudness at time '$t$'.

**STEP 5:** Initially task and its dead line is defined.

**STEP 6:** Maximum user pay is calculated.

**STEP 7:** It is calculated, because when 2 tasks has the similar dead line, the highest pay cost of the user is used, and the task is processed.

**STEP 8:** Task in queue.

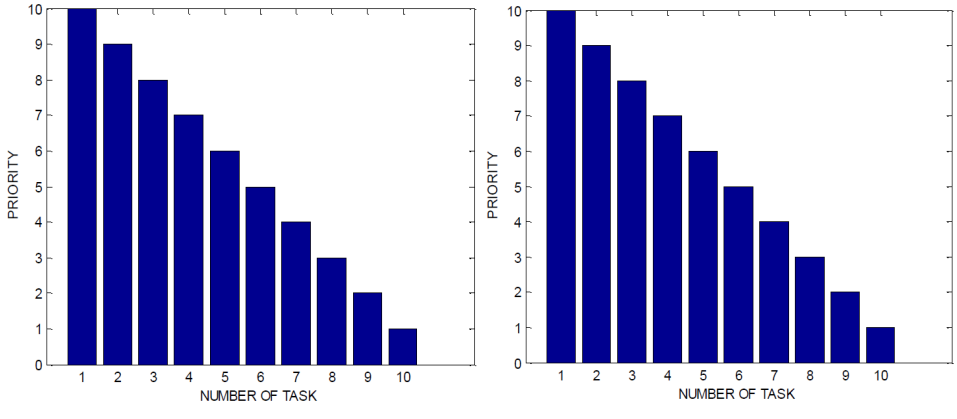
**STEP 9:** Sorting takes place as earliest dead line first.

**STEP 10:** Task is ready for mapping to the corresponding server using PSO approach.

## 5 Results and discussion

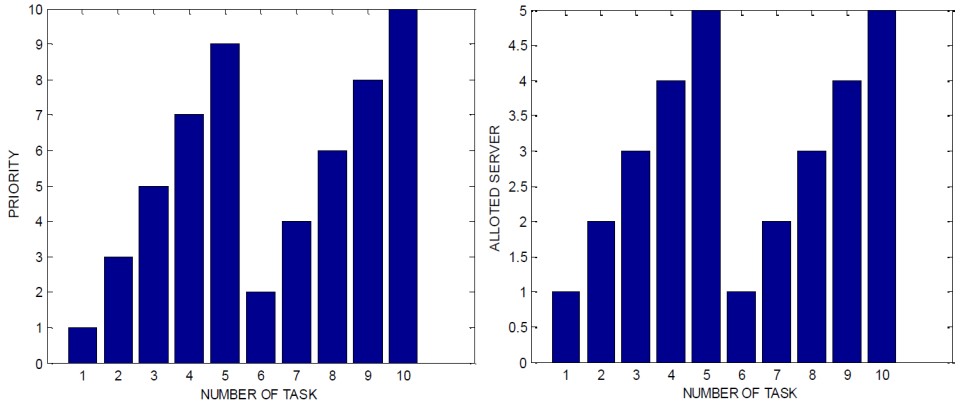
In this method the nature-based computing is applied to choose the optimal task. The creation of tasks is the initial stage in which the number of tasks is chosen to be 10 ($t_1$, $t_2$, $t_3$ … $t_{10}$) and number of servers is 5 ($s_1$, $s_2$, …, $s_5$). The priorities of the tasks are predetermined based on their importance. The tasks are assigned dynamically, and two tasks are allotted as maximum load of a server. The task scheduling is done based on the priority. The FF is allowed with respect to the execution speed. It determines priority and load for the servers are made as per the priority values as shown in Figures 6(a) and 6(b). Each allocation of task has been made by pheromone values. For the simulation MATLAB tool is used. The algorithm is tested with various task priority options. Figure 6 shows the results obtained for tasks with reverse priority that is t1 have priority 10, t2 has priority 9 and t10 is priority 1. Figure 7 shows the results obtained for tasks mixed priority that is t1 to t10 has priority t1, t3, t5, t7, t9, t2, t4, t6, t8 and t10. Figure 8 shows the results obtained for tasks priority t1 to t10 has priority t1, t2, t3, t4, t5, t6, t7, t8, t9 and t10 respectively.

**Figure 6** (a) Number of tasks with reverse priority (b) Allocation of tasks to server (see online version for colours)



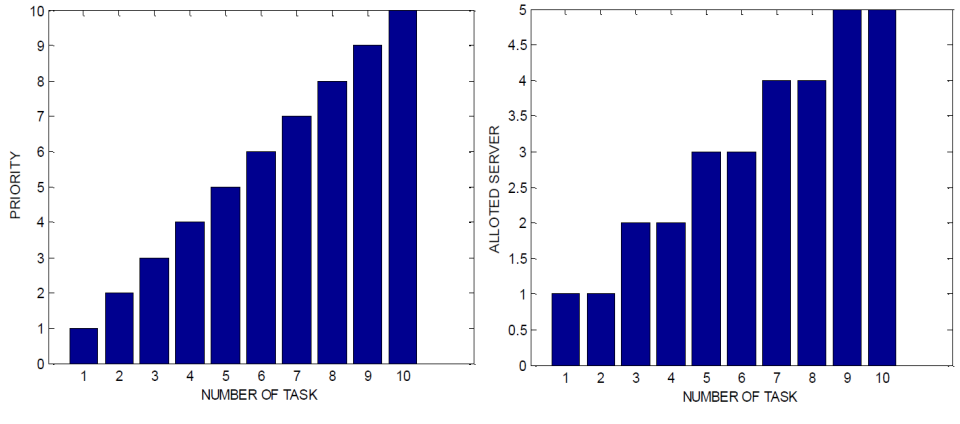

(a) (b)

**Figure 7** (a) Number of tasks with mixed priority (b) Allocation of tasks to server (see online version for colours)




(a) (b)

**Figure 8** (a) Number of tasks with regular priority (b) Allocation of tasks to server (see online version for colours)




(a) (b)

## 6 Conclusions

Task mapping to a server is an important work in load balancing. Several methods were addressed in this paper. Since cloud computing is used in every application today, new methodologies and an algorithm has to be implemented. Critical functionality systems need complete attention. This paper presented the several existing methods and proposed a new method to predict the system functionality without consuming more time. The proposed method is less expensive for implementation. Swarm intelligence-based task scheduling improves the power consumption by reducing overloads. For task scheduling simulations were conceded to test the efficiency of the method.

## References

Agarwal, A. and Raina, S. (2012) 'Live migration of virtual machines in cloud', *International Journal of Scientific and Research Publications*, Vol. 2, No. 6, pp.1–5.

Awad, A.I., El-Hefnawy, N.A. and Abdelkader, H.M. (2015) 'Enhanced particle swarm optimization for task scheduling in cloud computing environments', *International Conference on Communication, Management and Information Technology*, Vol. 65, pp.920–929.

Bousselmi, K., Brahmi, Z. and Gammoudi, M.M. (2016) 'Energy efficient partitioning and scheduling approach for scientific workflows in the cloud', *IEEE International Conference on Services Computing*, Vol. 9, pp.146–154.

Carrera, D., Steinder, M., Whalley, I., Torres, J. and Ayguade, E. (2012) 'Autonomic placement of mixed batch and transactional workloads', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, No. 2, pp.219–231.

Comer, D. and Karandikar, R.H. (2019) 'DCnet: a data centre network architecture that supports live VM migration', *IET Networks*, Vol. 8, No. 2, pp.114–125.

Han, B., Liu, L., Zhang, J., Tao, C., Qiu, C., Zhou, T., Li, Z. and Piao, Z. (2018) 'Research on resource migration based on novel RRH-BBU mapping in cloud radio access network for HSR scenarios', *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Vol. 7.

Jadhav, S.P. and Deshpande, P.R. (2014) 'Load balancing in cloud computing', *International Journal of Science and Research*, Vol. 3, No. 6, pp.2282–2285.

Kaur, P. and Rani, A. (2015) 'Virtual machine migration in cloud computing', *International Journal of Grid Distribution Computing*, Vol. 8, No. 5, pp.337–342.

König, B., Alcaraz Calero, J.M. and Kirschnick, J. (2011) 'Elastic monitoring framework for cloud infrastructures', *Information and Communication Engineering*, Vol. 10, No. 6, pp.1306–1315.

Li, C., Raghunathan, A. and Jha, N.K. (2012) 'Secure virtual machine execution under an untrusted management OS', *IEEE International Conference on Cloud Computing*, Vol. 3, pp.172–179.

Li, M., He, P. and Zhao, L. (2016) 'Dynamic load balancing applying water-filling approach in smart grid systems', *IEEE Internet of Things Journal*, Vol. 4, No. 1, pp.247–257.

Pandey, S., Wu, L., Guru, S.M. and Buyya, R. (2010) 'A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments', in *2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, IEEE, April, pp.400–407.

Patel, P.D., Karamta, M., Bhavsar, M.D. and Potdar, M.B. (2014) 'Live virtual machine migration techniques in cloud computing: a survey', *International Journal of Computer Applications*, Vol. 86, No. 16, pp.18–21.

Phuong, T., Nguyen, Q. and Kuo, R.J. (2019) 'Automatic fuzzy clustering using Non-dominated sorting particle swarm optimization algorithm for categorical data', *EEE Access*, p.1, DOI: 10.1109/access.2019.2927593.

Schäfer, D., Edinger, J., Eckrich, J., Breitbach, M. and Becker, C. (2018) 'Hybrid task scheduling for mobile devices in edge and cloud environments', *IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp.669–674.

Shi, W., Liu, D., Cheng, X., Li, Y. and Zhao, Y. (2019) 'Particle swarm optimization-based deep neural network for digital modulation recognition', *IEEE Access*, Vol. 7, pp.104591–104600, DOI: 10.1109/access.2019.2932266.

Tang, C., Xiao, S., Wei, X., Hao, M. and Chen, W. (2018) 'Energy efficient and deadline satisfied task scheduling in mobile cloud computing', *13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp.198–205.

Wang, Y., Bai, P., Liang, X., Wang, W., Zhang, J. and Fu, Q. (2019) 'Reconnaissance mission conducted by UAV swarms based on distributed PSO path planning algorithms', *IEEE Access*, p.1, DOI: 10.1109/access.2019.2932008.

Xing, N., Xu, S., Zhang, S. and Guo, S. (2016) 'Load balancing based routing optimization mechanism for power communication networks', *China Communications Networks*, Vol. 13, No. 8, pp.169–176.

Yin, S., Ke, P. and Tao, L. (2018) 'Dynamic cloud task scheduling based on a two-stage strategy', *IEEE Transactions on Automation Science and Engineering*, Vol. 15, No. 2, pp.772–783.