Trust-based fruit fly optimisation algorithm for task scheduling in a cloud environment

Priya Govindaraj and Jaisankar Natarajan*

School of Computer Science and Engineering,
VIT, Vellore, India
Email: gpriya@vit.ac.in
Email: njaisankar@vit.ac.in
*Corresponding author

Abstract: Efficient task scheduling plays a critical role in cloud computing environment. In this paper, we proposed a novel trust-based fruit fly optimisation algorithm (TFOA) for task scheduling. Even though traditional scheduling algorithms, namely first come first serve, round robin, ant colony optimisation and so on are used broadly in cloud computing process but still efficient scheduling is not achieved. In general cloud service provider desires to receive the customer task in a faster rate allocated on the most trustworthy resource by using TFOA. Simulation outcomes show that the proposed algorithm performs better than the existing transitional algorithms like round robin and particle swarm optimisation (PSO) in terms of reduced makespan and turnaround time and efficient resource utilisation.

Keywords: task scheduling; fruit fly optimisation; trust; makespan; turnaround time.

Reference to this paper should be made as follows: Govindaraj, P. and Natarajan, J. (2020) 'Trust-based fruit fly optimisation algorithm for task scheduling in a cloud environment', *Int. J. Internet Manufacturing and Services*, Vol. 7, Nos. 1/2, pp.97–114.

Biographical notes: Priya Govindaraj completed her BTech in Computer Science in the Madras University and MTech in Computer Science and Engineering in the Vellore Institute of Technology. She is working as an Assistant Professor (Senior) in the School of Computer Science and Engineering, VIT. She is having 14 years of teaching experience and currently she is pursuing her PhD in Trust Management. She published the papers in more than 30 journals. Her areas of interest are cloud computing, virtualisation techniques and internet of things.

Jaisankar Natarajan is working as a Professor in the School of Computer Science and Engineering, VIT. He is having more than 20 years of teaching experience. He has done his PhD on wireless senor networks in VIT. He published the papers in more than 40 journals. His area of interest are computer networks, data mining, wireless sensor networks, cloud computing and internet of things.

1 Introduction

Cloud computing is an evolving technology among several traditional computing paradigms such as grid computing, cluster computing and ubiquitous computing. It is mainly used in a distributed environment which is made up of numerous data centres, load balancers, servers and so on. It provides the services with high availability and the resources are scalable according to the customer need. Though there is a huge development in the cloud computing, but still efficient task scheduling is the challenging problem in a distributed environment.

Scheduling allows optimum allocation of resources among several given tasks in a finite time to attain preferred quality of service scheduling problems involves tasks that should be reserved on resources focus to some restrictions to enhance some objective function. The main objective is to construct a schedule that specifies the resource on which the task execution should take place (Karger et al., 1998). According to the deployment model of a cloud, scheduling procedures in a cloud computing system will be different.

Kumari and Monika (2015) have stated that scheduling is a procedure which is used to improve the job execution time. Scheduling is in charge for selecting the best appropriate resources by considering some parameters to execute the task. In order to gain profit for both cloud users and providers, an efficient scheduling should be done. Błażewicz et al. (2013) and Leung (2004) have defined the "scheduling as finding an optimum solution to schedule the set of tasks T = {T_1, T_2, .., T_n} on set of m machines M = {M_1, M_2, ..., M_m} with respect to set of predefined constraints and dimensions".

Optimisation problem are widely used in the field of science, engineering and finance. Over a decade to solve those problems, stochastic algorithms namely GA, ant colony optimisation, simulated annealing, and particle swarm optimisation (PSO) have been developed as they are flexible to find the solution. These algorithms are difficult to understand and have complex computational processes. To overcome this, new stochastic algorithm fruit fly optimisation algorithm (FOA) is developed by wan Tsao Pan in the year 2011. It is developed from the food searching behaviour of drosophila fruit fly. Compared to other algorithms FOA is easy to understand and requires simple parameters for computational process.

This paper mainly focused on tasks scheduling which is done by using TFOA. Our proposed TFOA is more efficient than other algorithms with respect to task scheduling. TFOA outperforms when compared to benchmark algorithms by means of reduced and makespan, turnaround time (TAT) with efficient utilisation of cloud resources. Henceforth, the main contributions in this paper are mentioned below:

1 A trust management model is designed for evaluating the trust value of a resource.

2 A novel TFOA is considered for effective task scheduling and schedule the tasks on the most reliable resource in the cloud.

3 Performance evaluation of proposed algorithms with traditional algorithms in terms of makespan, TAT and response time.

2 Related work

Rule-based scheduling procedure is popularly used on various cloud computing systems as they are very simple and ease implementation. Tsai et al. (2014) have presented a new scheduling algorithm called hyper heuristic (HHSA) to find the better scheduling solutions in cloud computing. This algorithm is used to find out the better candidate solutions dynamically using detection operators. The tasks are scheduled effectively in order to diminish the TAT and maximise resource utilisation. They have considered computational complexity and computing capacity to schedule the tasks (Sindhu and Mukherjee, 2011).

Yang et al. (2011) have introduced the failure identification and recovery situation in cloud computing objects and proposed a reinforcement learning (RL) algorithm to create fault tolerant scheduling. The efficacy of the cloud services are depending upon the performance of the cloud customer tasks given to cloud computing system. Task scheduling which is a NP complete problem plays substantial role in improving performance of the cloud services. The cloud users have submitted the jobs to the job scheduler during the scheduling process. The scheduler investigates the information repository for checking the resources availability and henceforth assigning the tasks on various resources according to the task requirements. Job scheduler allocates various customer tasks to several virtual machines (VM). Task scheduling can be performed either statically or dynamically (Mathew et al., 2014).

Habibi and Navimipour (2016) have used one of the most dominant meta-heuristic methods called an imperialist competitive algorithm (ICA) to improve the scheduling problems in cloud computing. It is proven that 0.7% enhancement in execution time when comparing GA. Six rule-based heuristic algorithms first come first serve (FCFS), minimum execution time (MET), Maxmin, Min-min, minimum completion time (MCT), and sufferage are executed and useful for scheduling an independent tasks in similar and dissimilar environments and compared their performance by means of makespan, cost, throughput and degree of imbalance. Performance analysis of job scheduling is done on those heuristic algorithms.

Choudhary and Peddoju (2012) have defined an efficient scheduling from the users' perspective is based on completion time or execution cost of a task, etc. and from the cloud service providers perspective is to guarantee that the resources are used efficiently to their maximum capability and no resource is left idle. The incoming tasks are clustered based on reduced cost or execution time and prioritised. According to the task constraints resources can be chosen using greedy approach. Dehkordi and Bardsiri (2015) have offered a new method for task scheduling by learning automata (LA). This method is trained by task execution historical data on the cloud, then split task to several classes and assess them and maintain virtual machine to capture physical resources at any period based on the task classes in order to improve efficiency of cloud network.

Shan et al. (2013) have studied the performance of FOA and tested with six different nonlinear functions. The results have been proven that FOA could not resolve multifaceted optimisation problems effectually. To enrich the FOA performance, an improved FOA is implemented to enhance the searching efficiency and quality. Allah (2016) has proposed a hybrid algorithm which combines both fruit fly and fire fly

(FOA-FA) algorithms to resolve the nonlinear programming problem (NLPP). This hybrid algorithm gives better results by controlling uninterrupted optimisation and accomplishing robust assessment. The hybrid algorithm swifts up the convergence and improve its performance. The FOA-FA is verified on various benchmark problems and shown its efficiency.

Tawfeek et al. (2013) have investigated that the best task scheduler would adjust the scheduling policy according to dynamic environment and the type of the task. Task scheduling procedure used in ant colony optimisation algorithm is compared with FCFS and round robin (RR). The main aim of ACO is to decrease the makespan of a set of tasks and it is arbitrary optimisation search technique which is used to allocate the incoming jobs to the VM. Zhang et al. (2016) have proposed a novel multi scale cooperative mutation fruit fly optimisation (MSFOA) procedure to overcome the limitation on local search space. The effectiveness is compared with benchmark functions and they proved that MSFOA gives better results than other enhanced versions of FOA.

Pathania and Rasool (2017) have taken AHP technique to assess the quality of website service among the e-commerce websites. The proposed work experiments to compute the client decision on the basis of various criteria which will create the impact on website service quality. The proposed work uses the decision to detect the client choices with respect to various existing e-commerce websites.

Petruni et al. (2017) have introduced a method for the evaluation and human reliability analysis process for the automotive industries. The authors have used AHP method to assist safety supervisors and risk inspectors in the selection process. The selected technique to be assessed based on the appropriate criteria in an automotive environment.

2.1 FOA algorithm

This section explains about the original FOA. FOA was developed by searching actions of fruit flies in the environment. FOA is a swarm intelligent search algorithm which performs the food search procedure of the drosophila's fruit fly. The fruit fly searching for food process consists of two steps. In the first step, it uses Osphresis organ to smell food sources and start to fly in that direction. In second step, fruit fly uses sensitive vision for food finding and flocking location to move towards the food location. The original FOA is given in eight steps.

Algorithm 1 Algorithm for FOA

Step 1 Initialization of fruit fly swarm location. LP is the location parameter of initial swarm

$$\begin{cases} \text{x-axis} = \text{rand(LP)} \\ \text{y-axis} = \text{rand(LP)} \end{cases} \tag{1}$$

Step 2 Generation of each and every fruit fly location in the swarm. V is the range parameter and x, y denotes coordinates

$$\begin{cases} a_j = \text{x-axis} + \text{rand(V)} \\ b_j = \text{y-axis} + \text{rand(V)} \end{cases} \tag{2}$$

Step 3 Computing the distance between the origin and every individual fruit fly.

$$\mathrm{Dist}_j = \sqrt{a_j^2 + b_j^2} \tag{3}$$

Step 4 Computation of smell concentration judgement value of each and every individual fruit fly and origin

$$S_j = \frac{1}{\mathrm{Dist}_j} \tag{4}$$

Step 5 Calculation of each individual fruit fly smell concentration value

$$\mathrm{smell}_j = \mathrm{smell\_fun}(S_j) \tag{5}$$

Step 6 Identify the optimum smell concentration value in the swarm, represented by the largest value

$$[\mathrm{bestSmell} \quad \mathrm{bestIndex}] = \mathrm{maximum}(\mathrm{Spell}_j) \tag{6}$$

Step 7 Reservation of the known optimal smell concentration value and update the location of the swarm.

$$\begin{cases} \mathrm{optimalsmell} = \mathrm{bestSmell} \\ \text{x-axis} = x_{\mathrm{bestindex}} \\ \text{y-axis} = y_{\mathrm{bestindex}} \end{cases} \tag{7}$$

Step 8 If the maximum number of iteration is got terminate the algorithm else go to step 2.

3 The proposed work

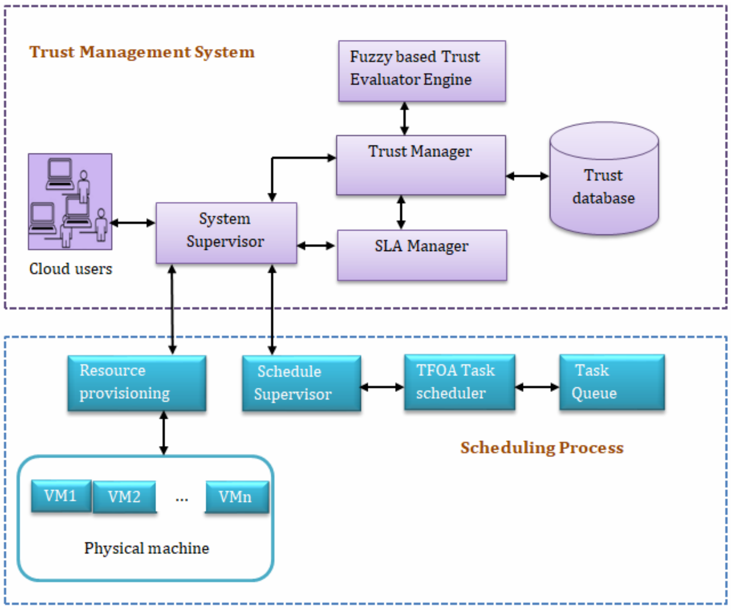
3.1 Trust management scheduling model

The architecture of the trust management scheduling model is given in Figure 1. Cloud customers can communicate with system supervisor and cloud service discovery via the interface. Cloud service discovery offers the list of resources. The service level agreement manager is an in charge for retaining the agreement between cloud customer and cloud service provider It acquires the resources trust values from the trust manager and supports the system supervisor to choose the resource based on the trust value. System supervisor connects to the VM through the middleware that takes care of VM creation, configuration and deployment of virtual machine in an infrastructure. Trust database is storage for keeping the trust values of cloud services.

In fuzzy evaluation engine, the cloud user select the trustworthy cloud service among various services by using the trust value that are calculated by considering the quality of service parameters namely availability, success rate, turnaround efficiency and feedback. There are three processes namely fuzzification, rule implication and defuzzification primarily used in fuzzy inference system. The crisp data is given as input to the fuzzifier then it is converted in to fuzzy input by using linguistic variables, membership functions. Having done the fuzzification, the rules are applied to fuzzy input to get the fuzzified output. Finally the trust value is taken from the fuzzified output by using defuzzification process.

The TFOA is proposed for task scheduling. Initially tasks are sent to the task queue then task scheduler procures the task from the queue.

Figure 1 Trust-based scheduling model (see online version for colours)



The schedule supervisor talks with the fuzzy-based trust evaluator engine through system supervisor. This trust model analyses the transactions and acquires the detailed trusted resource information of task. It handovers to task scheduler that executes the high priority task on the most trustworthy resource in the cloud using analytical hierarchy process (AHP) in TFOA optimisation algorithm.

3.2 Problem formulation

A set of p tasks $T = \{t_1, t_2, t_3, \ldots. t_p\}$ should be scheduled on n VM $Vm = \{vm_1, vm_2, vm_3, \ldots. vm_n\}$ with the following parameter id, length, completion time of the task, priority of the task and deadline of the task and focus to some constraints to optimise some objective functions in a cloud environment. The optimisation problem is mathematically formulated as follows

- Multi objective function

$$F_1(x) = \text{Minimise}\{C_{\max}\} \tag{8}$$

$$F_2(x) = \text{Minimise}\{Td_j\} \tag{9}$$

- Constraints

$$C_{\max} \geq S_j + P_j \qquad \forall_j \in J \tag{10}$$

$$Td_j \leq 0 \tag{11}$$

Objective function $F_1(x)$ aims to minimise the completion time of the last job that is called as makespan in our proposed work is given in equation (8). The second objective function is in equation (9) reduces the tardiness. In order to achieve optimal solution, proposed work should satisfy the both the constraints. Constraint in equation (10) specifies that the makespan should be greater than or equal to the sum of the starting and processing time of the job. Condition in equation (11) should defined that the tardiness value could be lesser than or equal to 0.

3.3 Objective functions

In the proposed TFOA algorithm, we are considering the following optimisation criteria such as makespan, tardiness, TAT and average resource utilisation for scheduling the tasks on the most trust worthy resource.

3.3.1 Objective 1: reducing makespan

Makespan defines the completion time of the last task. The most popularly used optimisation criterion during task scheduling is minimisation of makespan since the cloud users wants the faster execution of job. Makespan is given in equation (12).

$$\text{Makespan} = \max_{j \in \text{tasks}} \{c_j\} \tag{12}$$

where c_j = completion time of task j.

3.3.2 Objective 2: reducing tardiness

Tardiness defines the time difference between the completion time and deadline of the task. It denoted as the delay in task execution. C_j and D_j are the completion time and deadline of the task j respectively. If the tardiness value is zero then it is an optimal schedule.

$$Td_j = C_j - D_j \tag{13}$$

3.3.3 Objective 3: reducing TAT

TAT defines how much time is taken for a task to complete the execution from the submission. It is the sum of waiting and execution time which is given in equation (14). W_j and E_j are the waiting and execution time of task j.

$$TAT = W_j + E_j \tag{14}$$

3.3.4 Objective 4: maximise resource utilisation

The popular provider desired optimisation criteria is to maximise the resource utilisation is given in equation (15). In order to maximise the resource utilisation, the resources has to be kept as much by busy as possible. The cloud service provider wants to earn more profit by hiring limited number of resources.

$$A_{ru} = \frac{\sum_{1}^{n} T_j}{\text{makespan} \times n} \tag{15}$$

A_{ru}	average resource utilisation
T_j	time taken by the resource j to complete all jobs
n	total number of resources.

3.4 Smell and vision-based searching process

A smell-based searching process is primary searching procedure in which K fruit flies are generated. In TFOA algorithm, the smell-based approach sort the tasks and resources based on the priority and trust value respectively.

Vision-based searching process smell concentration value is computed for the fruit flies. In TFOA most trustworthy resource is found and the tasks are allocated to the optimal resource.

3.5 Task priority

The priority among the task is computed by AHP method. Each task is compared with other tasks separately. The ratio of priority of t_i to t_j for using the resource is computed by using comparison matrix which is given in equation (16).

$$T_g^{ij} = \begin{cases} \dfrac{1}{T_g^{ji}} & i \neq j \\ 1 & i = j \end{cases} \tag{16}$$

T_g represents the a matrix with n rows and n columns. Assume that U^1, U^2, …., U^r are r comparison matrixes of the tasks. A priority vector should be calculated for each and every comparison matrixes that is given in equation (17). Assume that ω^1, ω^2, …, ω^r are the priority vectors of U^1, U2, …., U^r respectively.

$$\omega T_p^{ij} = \lambda_{\max} \omega \tag{17}$$

The normal matrix of the tasks is given in equation (18).

$$\Delta = \left[\omega^1 \;\; \omega^2 \ldots \omega^r\right] \tag{18}$$

Here Δ is a matrix with p rows (no of tasks) and n columns (no of resources). The next step to find the comparison matrix for the resources according to priorities. This matrix is used to identify the high priority resource in the resource list based on the trust value. In this case the matrix is having r rows and r columns. Assume that V is the comparison matrix for resources and γ be the priority vector of V. Finally priority vector of scheduling (P_v) of task is calculated by using equation (19).

$$P_v = \Delta\gamma \tag{19}$$

From this find the maximum value and then select the corresponding task from T to which trustworthy resource is allocated.

3.6 *Load balancing decision*

Let execution period of task is $\{E_1, E_2, \ldots. E_n\}$ and maximum computation time of task is $\{C_1, C_2, \ldots C_n\}$.

The virtual machine utilisation is calculated as the proportion between computation time of task and execution period of task which is given in equation (20).

$$U_i = \frac{C_i}{E_i} \tag{20}$$

In this research work, resource utilisation is determined. If the resource utilisation is less than or equal to 1, the load is balanced among the VM. The scheduling process is done, only if Ui <= 1, i.e., Ci <= Ei.

3.7 *Smell and vision-based searching process*

A smell-based approach is the principal searching procedure in which K fruit flies are generated. In TFOA algorithm, the smell-based approach sorts out the task on the basis of the priority which is computed using the analytic hierarchy process (AHP) and resources are sorted based on the trust value.

In the vision based searching process, smell absorption value is computed for the fruit flies. The most trustworthy resource is found in TFOA and the highest priority tasks are allocated to the optimal trustworthy resource.

3.8 *Identifying optimal resource*

Resources (VM) are grouped and evaluated by using trust values which is computed from fuzzy-based trust evaluator engine. Then the resources are sorted based on the trust value. In order to allocate the most optimal resource to the high priority task, determine the resource that reduces the TAT, zero tardiness and increases the resource utilisation value.

4 Trust-based fruit fly optimisation algorithm

The proposed TFOA algorithm has the following steps. The first step is to initialise the maximum iteration count, population size and the trust value of the resources which is computed from the fuzzy-based trust model. This is referred to as the initialisation phase. Then the tasks and resources in the TaskList and ResourceList are listed out respectively.

The second step in the proposed algorithm is a smell-based searching phase. In smell-based searching, the fruit flies are generated and list the available resources in ResourceList and tasks in the TaskList.

The next step is a vision-based searching phase in which the generated fruit flies are evaluated. Then the tasks in taskList based on the priority are sorted and evaluated using AHP method. This is explained in subsection and sort the resources based on the trust values. Then resource utilisation is computed using the equation (20).

Steps to check whether the virtual machine is balanced or not.

```
if (resource utilisation (Ui <= 1) then
    Calculate makespan, $Td_i$, $TAT_i$ and $A_{ru}$
    Find the best resource which minimises the TAT, zero tardiness.
    Assign high priority task to trustworthy resource.
else
    Find the next best resource and repeat the steps.
```

In the vision-based approach, the best optimal resource is identified only if resource utilisation is less than or equal to 1. Then the high priority task is assigned to the best resource and removed from the TaskList. TaskList and ResourceList are then updated. If the resource utilisation is greater than 1, the virtual machine is not balanced. The next best resource is found. The steps are repeated. The proposed TFOA performed well. Time complexity for the proposed TFOA is $O(n^2)$.

4.1 TFOA algorithm

An optimum task scheduling is done in cloud environment with respect to some constraints to optimise some objective functions. Our proposed work use multi objective function with two constraints. According to the trust model architecture, the proposed work extends the basic fruit fly optimisation algorithms by taking the trust value of the resource. The algorithm for TFOA is given in Algorithm 2. A TFOA algorithm schedules the high priority task on most trustworthy optimal resource.

In TFOA algorithm, first initialise the maximum iteration count, population size and trust value of the resources which is computed from the fuzzy-based trust model. Then list out the tasks and resources in tasklist and resourcelist respectively. In the next step evaluate the generated fruit fly and sort the task in the tasklist based on the priority and the resources based on the trust value. Then calculate the tardiness, TAT and average resource utilisation and finally find out the best resource which minimises the TAT, zero tardiness and maximum trust value and assign the high priority task to that optimal resource.

Algorithm 2 Trust-based fruit fly optimisation algorithm

```
Input: Swarm_size, Max_iter, Trust_value
Output: An optimal resource
1   Begin
2       For all t in num_swarms
3           Generate k fruit flies $F_k$ {k = 1, 2 ,……. t}
4           List the tasks in TaskList
5           List the resources in the resource list
6       End for
7       For all t in num_swarms
8           Evaluate the generated fruit flies $F_t$
9           Sort the tasks in the task list based on the priority.
10          Sort the resources in resource list based on trust value.
11      End For
```

```
12          For all task i in the TaskList
13              For all resource j in the Resource List
14        Compute Ui from equation (20).
15          If (Ui <= 1) then
16                  Compute $T_{di}$, $TAT_i$ and $A_{ru}$
17              Find the best resource which minimizes $TAT_i$, zero tardiness and having
                maximum trust value.
18          Assign task $T_i$ to Resource $R_j$ that gives maximum trust value.
19          Remove task $T_i$ from the TaskList
20          Update the TaskList and ResourceList of each k
21          Update $TAT_i$ of all tasks in TaskList
22          Else
23              Find the next best Resource $R_j$ and go to step 16.
24          End For
25            End For
26    End
```

5 Results and discussion

5.1 Case study

In this section, an example for finding the priority of a task is provided. The trustworthy resource among several resources is identified using the TFOA algorithm. AHP is used for allocating the resources for the task given the highest priority. In this case, five criteria and four tasks have been considered. The comparison matrix is formed by comparing the pair of elements, i.e., to what extent is the element in the matrix on the left is better than that on the right. This is evaluated by the relative scale and the description provided by Saaty (2008) in Table 1. Comparison matrices for all the criteria which are given in Table 2.

Table 1 Relative scale defined in AHP

AHP scale	*Description*
1	Equally important
2	Weakly more important
3	Moderately important
4	Moderate plus important
5	Essentially important
6	Essentially plus important
7	Very essential important
8	Very essential plus important
9	Extremely important
Reciprocals	If an activity m is assigned to any one of the above said value when compared to activity n then the activity n is assigned to reciprocal with respect to m.

Table 2 Comparison matrix for criteria

Criteria	*Tasklength*	*Deadline*	*Cost*	*Waiting time*	*Makespan*
Tasklength	1	1/5	3	1/2	5
Deadline	5	1	7	1	7
Cost	1/3	1/7	1	1/4	3
Waiting time	2	1	4	1	7
Makespan	1/5	1/7	1/3	1/7	1

Table 3 Priority vector of criteria

Criteria	*Tasklength*	*Deadline*	*Cost*	*Waiting time*	*Makespan*	*Priority vector*
Tasklength	0.117	0.08	0.196	0.173	0.217	0.156
Deadline	0.586	0.401	0.456	0.346	0.3043	0.418
Cost	0.039	0.057	0.065	0.086	0.13	0.075
Waiting time	0.234	0.401	0.261	0.346	0.3043	0.309
Makespan	0.023	0.057	0.021	0.049	0.043	0.038

The priority vector of criteria is shown in Table 3. First sum of each column is computed and then divide each entity by the sum. The priority vector is computed by dividing the total row value by total number of criterion, i.e., 5.

Compute the values for max, consistency ratio (CR), consistency index (CI) by using equation (17), equation (19) and equation (20) respectively for checking the consistency of VM. V is consistent because max = 5.286, CI = 0.07. Random index for this scenario is 1.12 where n value is 5. Then CR is calculated as 0.06. In this case, CR is less than 0.1 and then it becomes consistent. In the same way, consistency is checked for all cases. Saaty (2005) has computed random indices for RI for some values is given in Table 4.

Table 4 Random index

n	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
RI	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

Table 5 Comparison matrix-based on the task length

Task length	*Task1*	*Task2*	*Task3*	*Task4*
Task1	1	1/2	4	1/6
Task2	2	1	3	1/5
Task3	1/4	1/3	1	7
Task4	6	5	1/7	1

Comparison matrix and priority vector for task length, deadline, cost, waiting time and makespan are given in Table 5, Table 6, Table 7, Table 8, Table 9, Table 10, Table 11, Table 12 and Table 13.

Table 6 Priority vector based on task length

Tasklength	*Task1*	*Task2*	*Task3*	*Task4*	*Priority vector*
Task1	0.108	0.073	0.359	0.02	0.14
Task2	0.216	0.146	0.269	0.024	0.16
Task3	0.027	0.048	0.089	0.837	0.25
Task4	0.648	0.732	0.013	0.12	0.37

Table 7 Comparison matrix based on the deadline

Deadline	*Task1*	*Task2*	*Task3*	*Task4*
Task1	1	1/2	1/3	1/6
Task2	2	1	3	1/5
Task3	3	1/3	1	4
Task4	6	5	1/4	1

Table 8 Priority vector based on deadline

Deadline	*Task1*	*Task2*	*Task3*	*Task4*	*Priority vector*
Task1	0.083	0.073	0.073	0.031	0.065
Task2	0.167	0.1464	0.655	0.037	0.251
Task3	0.250	0.049	0.218	0.745	0.315
Task4	0.5	0.732	0.054	0.186	0.368

Table 9 Comparison matrix based on the cost

Cost	*Task1*	*Task2*	*Task3*	*Task4*
Task1	1	1/5	1/3	2
Task2	5	1	1/2	2
Task3	3	2	1	1/4
Task4	1/2	2	4	1

Table 10 Priority vector based on cost

Cost	*Task1*	*Task2*	*Task3*	*Task4*	*Priority vector*
Task1	0.105	0.038	0.057	0.0381	0.145
Task2	0.526	0.192	0.086	0.381	0.296
Task3	0.316	0.385	0.172	0.047	0.23
Task4	0.053	0.385	0.686	0.190	0.32

Table 11 Comparison matrix based on the waiting time

Waiting time	*Task1*	*Task2*	*Task3*	*Task4*
Task1	1	1/5	4	1/2
Task2	5	1	1/3	1/2
Task3	1/4	3	1	6
Task4	2	2	1/6	1

Table 12 Priority vector based on waiting time

Waiting time	*Task1*	*Task2*	*Task3*	*Task4*	*Priority vector*
Task1	0.121	0.032	0.727	0.062	0.236
Task2	0.606	0.161	0.061	0.062	0.222
Task3	0.030	0.484	0.1818	0.75	0.36
Task4	0.242	0.322	0.030	0.125	0.180

Table 13 Comparison matrix based on the makespan

Makespan	*Task1*	*Task2*	*Task3*	*Task4*
Task1	1	7	1/3	2
Task2	1/7	1	5	3
Task3	3	1/5	1	1/6
Task4	1/2	1/3	6	1

Table 14 Priority vector based on makespan

Makespan	*Task1*	*Task2*	*Task3*	*Task4*	*Priority vector*
Task1	0.2155	0.821	0.027	0.325	0.347
Task2	0.031	0.117	0.406	0.487	0.260
Task3	0.647	0.023	0.081	0.027	0.195
Task4	0.108	0.039	0.487	0.162	0.199

The task's priority vector Δ and criterion priority vector γ is given below.

$$\Delta = \begin{bmatrix} 0.14 & 0.065 & 0.145 & 0.236 & 0.347 \\ 0.16 & 0.25 & 0.406 & 0.487 & 0.260 \\ 0.25 & 0.023 & 0.081 & 0.027 & 0.195 \\ 0.108 & 0.039 & 0.487 & 0.162 & 0.199 \end{bmatrix}$$

$$\gamma = \begin{bmatrix} 0.156 \\ 0.418 \\ 0.075 \\ 0.309 \\ 0.038 \end{bmatrix}$$

Priority vector (P_v) is computed. From this result, the task Task4 has the highest priority to access the trustworthy VM.

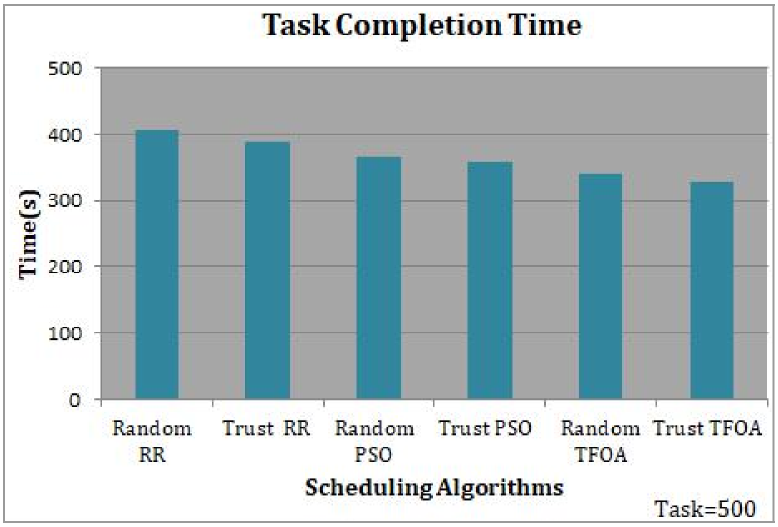
$$P_v = \begin{bmatrix} 0.146 \\ 0.230 \\ 0.160 \\ 0.299 \end{bmatrix}$$

The TFOA applied for optimisation of the task scheduling in a cloud environment. In this section, the experimental study of the proposed scheduling algorithm using Cloudsim simulator is shown. Various classes, namely, task class, brokers class, VM class and datacentre class have been used in CloudSim for the implementation of the TFOA. Task behaviour is defined by task class. Brokers are represented by Datacentrebroker class. VM is represented by VM class. It runs on a host and shares the host list with other VM's. Datacentre class is responsible for Virtual Machine management. In the proposed work, the maximum number of iteration(100), swarm size (50), 25 VM's with 1,000 MIPS to 2,000 MIPS computing power and bandwidth availability of 1,000 Mbps to 4,000 Mbps have been considered for simulation.

5.2 The performance evaluation

TFOA is evaluated with resources of different numbers. The time required for completing the task by RR, PSO and TFOA are evaluated. Figure 2 shows the time taken by TFOA is comparatively lower than other two algorithms. The performance of the proposed TFOA algorithm outperforming other algorithms such as RR and PSO with respect to makespan, TAT and resource utilisation.

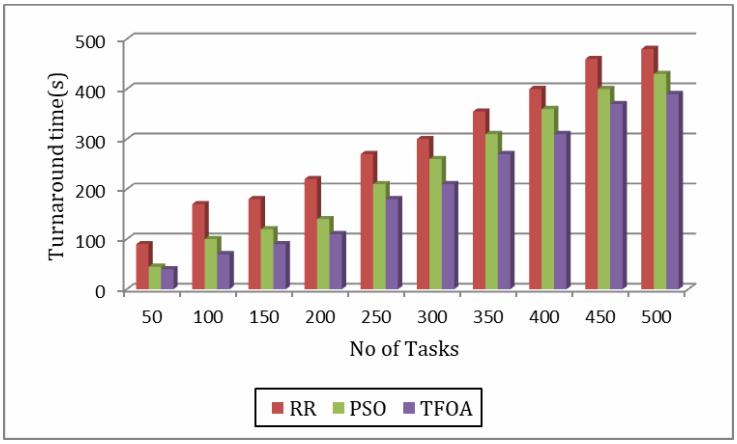
Figure 2 Task completion time comparison (see online version for colours)



5.3 Turnaround time

Figure 3 illustrates the TAT comparisons of the proposed algorithm (TFOA), RR and PSO. It is clearly shown that TFOA significantly reduces the TAT compared to other algorithms. When the number of tasks increases, the TAT also increases but it is comparatively lower than the other algorithms. The X-axis in the graph denotes that the number of tasks and Y-axis specifies the TAT in seconds.

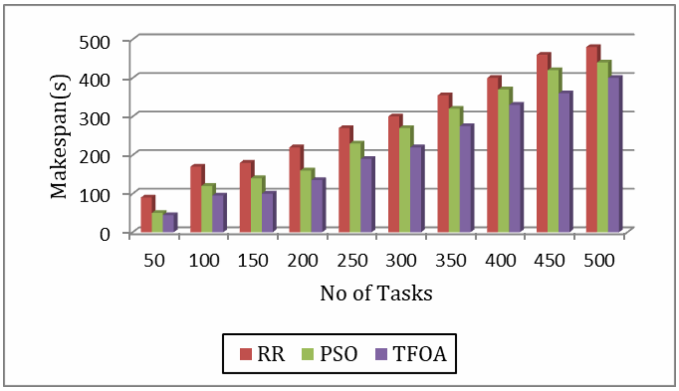
Figure 3 TAT evaluation (see online version for colours)



5.4 *Makespan*

Figure 4 indicates the makespan comparison of TFOA, PSO and RR. It is proven that our proposed algorithm gives the better result compared to other algorithms. The X-axis and Y-axis in the graph represents number of tasks and makespan in seconds respectively.

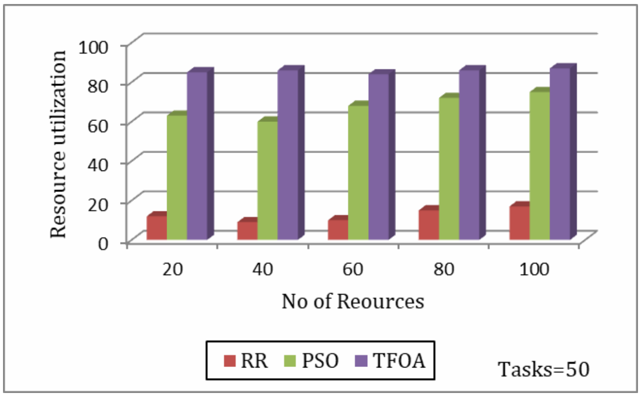
Figure 4 Makespan comparison (see online version for colours)



5.5 *Average resource utilisation*

Figure 5 illustrates the resource utilisation comparison of RR, PSO and our proposed algorithm TFOA. In X-axis denotes the number of resources and Y-axis represents resource utilisation factor. This graph depicts the resource utilisation with 500 tasks. It is clearly shown that RR algorithm uses maximum cloud resources; PSO provides better resource utilisation than RR. On the other hand, our proposed algorithm TFOA is more efficient in using the unused resources from the virtual machine.

Figure 5 Average resource utilisation (see online version for colours)



6 Conclusions

Task scheduling has been considered as the most critical problem in cloud computing. An optimised scheduling increases the performance of task scheduling in a cloud environment. A novel, TFOA is implemented for scheduling the tasks on the most trustworthy resources. AHP method is used for finding the priority of tasks. The proposed TFOA outperforms the existing algorithms such as RR, PSO. Performance evaluation is done on makespan, TAT and resource utilisation with existing algorithms and it is shown that the TFOA provides the better outputs in terms of minimising the makespan, TAT and maximising the resource utilisation.

References

Allah, R.M.R. (2016) 'Hybridization of fruit fly optimization algorithm and firefly algorithm for solving nonlinear programming problems', *Int. J. Swarm Intel. Evol. Comput.*, Vol. 5, p.134, DOI: 10.4172/2090-4908.1000134.

Błażewicz, J., Ecker, K.H., Pesch, E., Schmidt, G. and Weglarz, J. (2013) *Scheduling Computer and Manufacturing Processes*, Springer Science & Business Media.

Choudhary, M. and Peddoju, S.K. (2012) 'A dynamic optimization algorithm for task scheduling in cloud environment', *International Journal of Engineering Research and Applications (IJERA)*, Vol. 2, No. 3, pp.2564–2568.

Dehkordi, S.T. and Bardsiri, V.K. (2015) 'Optimization task scheduling algorithm in cloud computing', *Journal of Advances in Computer Engineering and Technology*, Vol. 1, No. 3, p.17–22.

Habibi, M. and Navimipour, N.J. (2016) 'Multi-objective task scheduling in cloud computing using an imperialist competitive algorithm', *International Journal of Advanced Computer Science & Applications*, Vol. 1, No. 7, pp.289–293.

Karger, D., Stein, C. and Wein, J. (1998) 'Scheduling algorithms', in *Algorithms and Theory of Computation Handbook*, 9pp, Article ID 108768 [online] http://dx.doi.org/10.1155/2013/108768.

Kumari, E. and Monika, P. (2015) 'Review on task scheduling algorithms in cloud computing', *International Journal of Science, Environment, and Technology*, Vol. 4, No. 2, pp.433–439.

Leung, J.Y. (Ed.) (2004) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press.

Mathew, T., Sekaran, K.C. and Jose, J. (2014) 'Study and analysis of various task scheduling algorithms in the cloud computing environment', in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, September, pp.658–664, IEEE.

Pathania, A. and Rasool, G. (2017) 'Investigating e tailer's perceived website quality using analytical hierarchy process technique', *Procedia Computer Science*, Vol. 122, pp.1016–1023.

Petruni, A., Giagloglou, E., Douglas, E., Geng, J., Leva, M.C. and Demichela, M. (2017) 'Applying analytic hierarchy process (AHP) to choose a human factors technique: choosing the suitable human reliability analysis technique for the automotive industry', *Safety Science*.

Saaty, T.L. (2005) *Theory and Applications of the Analytic Network Process: Decision Making with Benefits, Opportunities, Costs, and Risks*, RWS Publications.

Saaty, T.L. (2008) 'Decision making with the analytic hierarchy process', *International Journal of Services Sciences*, Vol. 1, No. 1, pp.83–98.

Shan, D., Cao, G. and Dong, H. (2013) 'LGMS-FOA: an improved fruit fly optimization algorithm for solving optimization problems', *Mathematical Problems in Engineering*, 9pp, Article ID 108768 [online] http://dx.doi.org/10.1155/2013/108768.

Sindhu, S. and Mukherjee, S. (2011) 'Efficient task scheduling algorithms for cloud computing environment', in *High-Performance Architecture and Grid Computing*, pp.79–83, Springer, Berlin, Heidelberg.

Tawfeek, M.A., El-Sisi, A., Keshk, A.E. and Torkey, F.A. (2013) 'Cloud task scheduling based on ant colony optimization, in *2013 8th International Conference on Computer Engineering & Systems (ICCES)*, November, pp.64–69, IEEE.

Tsai, C.W., Huang, W.C., Chiang, M.H., Chiang, M.C. and Yang, C.S. (2014) 'A hyper-heuristic scheduling algorithm for the cloud', *IEEE Transactions on Cloud Computing*, Vol. 2, No. 2, pp.236–250.

Yang, B., Xu, X., Tan, F. and Park, D.H. (2011) 'An utility-based job scheduling algorithm for cloud computing considering reliability factor', in *2011 International Conference on Cloud and Service Computing (CSC)*, December, pp.95–102, IEEE.

Zhang, Y., Cui, G., Wu, J., Pan, W.T. and He, Q. (2016) 'A novel multi-scale cooperative mutation fruit fly optimization algorithm', *Knowledge-Based Systems*, Vol. 114, pp.24–35.