An upgraded fruit fly optimisation algorithm for solving task scheduling and resource management problem in cloud infrastructure

K. Loheswaran

Abstract

In this manuscript, an upgraded Fruit Fly Optimization Algorithm (FFOA) is proposed for optimising task scheduling and resource management processes. The proposed FOA algorithm is utilized to address the issues. In the proposed algorithm, every solution is represented by fruit fly. Every fruit fly upgrades their status through well-organized smell search process. First, a basic approach is put forward to allocate every task for numerous resources and execution time is measured for every task. Second, the overloaded virtual machines (VMs) are identified and the load is balanced to obtain optimal system resource utilisation. The ability of Fly Task Scheduling Algorithm is to schedule the VMs execution time of tasks is minimal. The results of the Fruit Fly-based algorithms such as Fruit Fly Task Scheduling Algorithm, Modified Fruit Fly Task Scheduling Algorithm, Improved Fruit Fly Task Scheduling Algorithm and Multi-Swarm Fruit Fly Task Scheduling Algorithm (MSFFTSA) are proposed and analysed. The upgraded Fruit Fly Optimization algorithm of MSFFTSA is compared with different Fruit Fly algorithms. Finally, the proposed algorithm is compared with other algorithms and the experimental results shows that the proposed MSFFTSA technique is better than the other algorithms of the fruit fly algorithms.

1 | INTRODUCTION

Cloud computing is the delivery of different services through the Internet; it is used to implement commercial applications. The different computing techniques are distributed computing, parallel and grid computing. Cloud computing cannot involve new multi-technologies; it represents a new way of managing IT. In several cases, it is not only change the workflow within the IT organisation; it mostly resulting in complete reorganisation of the IT industry. Cost savings including scaling can be extremely attained from cloud computing. The Cloud metaphor refers to the ubiquitous availability as well as approachability of computing resources through Internet technique [1]. The most vital service in the Internet computing is to make use of cloud infrastructure. Infrastructure as a service is a rapidly developing field at the cloud computing. It is a kind of servicing model, which offers hardware and software resources such as computer as virtual machines (VMs), storage space, network devices and load balance to the users’ on-demand basis. Cloud computing is an upcoming distributed system that follows pay and use model [2].

The basic conception of cloud computing is to access the resources available in the remote server instead of accessing in the local computer. Sharing of resources in the remote server is done with the help of distributed competing technique. A basic problem of cloud computing is task scheduling and resource management, which is an NP-complete problem. An eminent task scheduler uses scheduling strategies for all types of tasks and handles workload variations based on any given time. Task scheduling and resource allocation is a significant aspect of cloud computing. The VMs run in parallel while scheduling the tasks. It shows the way of problems in task scheduling within the available resources. One virtual machine is assigned more than one task. The scheduler is responsible for tasks running simultaneously and the scheduler makes sure that the VMs are not under loaded or overloaded. The load-balancing
algorithms aim to minimize the processing time including maximize the resource application [3]. Cloud task scheduling is an NP-complete optimisation issue also multi-heuristic algorithms is proposed to address it.

The objective of the task scheduling algorithms is to accelerate the execution process, which assigns resources to workload having different execution time in a volatile way. Allocating resources to balance the workload can be classified into static and dynamic algorithms. The static algorithm does not appropriate to cloud infrastructure due to fewer variations in load. In dynamic algorithms, the workload is balanced among heterogeneous resources. These algorithms are extremely successful because of the task prioritising operations. VMs are always loaded with multiple tasks, so the dynamic nature of the algorithm removes the tasks and assigns to under-loaded VMs of the similar data centre. Pan introduced the fruit fly optimisation is a new evolutionary meta-heuristic algorithm. As its name implies, it reflects the intelligent behaviour of the fruit fly swarm in search of food. Fruit fly optimisation contains several benefits like less parameter, simple structure etc. Therefore, it is extensively utilized to deal different types of issues at real world [4]. In this manuscript, an upgraded Fruit Fly Optimization Algorithm (FFOA) is proposed for optimising task scheduling and resource management processes, which is utilized Gaussian mutation operator with differently scaled variances from local optimum. Meng et al. [5] have suggested IFFOA, in which multi-swarms formed independently simultaneous so that it could have diversity between swarms and achieve the broader explorer active scope without getting caught up in local optima.

This study proposes that task scheduling issue can be addressed by relating food foraging activities of fruit fly. A detailed analysis of the foraging behaviour of fruit flies to find the food source is derived in this algorithm. The flies locate the food source with the help of smell and vision. Fruit fly identifying the food source with the aid of smell and vision then it flies toward the exact location. In a similar way, the overloaded VMs are identified and the tasks are allocated efficiently with the aid of proposed algorithm. The task prioritisation is performed to all the overloaded VMs using fruit fly algorithm. VMs running with a smaller number of high priority tasks are assigned tasks with high priority. This kind of scheduling makes the process to complete earlier. Task allocation for the under-loaded VMs is performed in the ascending order of a particular data centre which is based on load. The tasks are denoted as fruit flies and the food sources as VMs. Assigning tasks to VMs is similar to a fruit fly searching for food source. The VMs that is overloaded is similar to the fruit fly that is exhausted in the food source. Scheduling tasks to the under loaded VMs is related as fruit fly searching another food source. The tasks that are eliminated updates the leftover tasks regarding the status of the VMs and the fruit fly performs its search based on smell and vision. The tasks are updated based on high priority tasks, which are presently processed in the VMs. This updating process is similar to the fruit fly locating the food source and revising the other fruit flies position in the group. Thus, tasks assignment is carried out based on VMs availability. This proposed algorithm works well in the cloud infrastructure.

1.1 | Objective and contribution of the work

The objectives of this manuscript are given below:

- A procedure for scheduling tasks and allocating resources in the cloud infrastructure inspired by fruit fly behaviour.
- Correlation of the task scheduling process with the standard Fruit fly algorithm and its variants.
- Mathematical analysis of the task scheduling using VMs in the cloud computing infrastructure.
- Performance analysis of the proposed algorithm and comparison with respect to its other variants of the proposed algorithm.

In this manuscript, fruit fly optimisation based algorithm has been proposed to upgrade the task scheduling problem in the cloud infrastructure. The proposed Fruit Fly-based algorithm is executed utilising open-source simulator named CLOUDSIM [6,7].

The remainder of this manuscript is designed as follows: Section 2 deals with the related works in task scheduling methods. Section 3 explains the problem description. Section 4 delineates the solution representation. Section 5 discusses the foraging behaviour of fruit fly and its variants of the proposed technique with detailed algorithm. Section 6 discusses the experimental results of Fruit Fly-based algorithm and evaluates the performance and compares with its variants. Finally, Section 7 concludes the manuscript.

2 | LITERATURE REVIEW

Task scheduling with resource management at the cloud infrastructure is NP-Complete problem [8]. Highly efficient scheduling techniques such as deterministic methods, heuristic algorithms and nature-inspired optimisation algorithms were utilised to address the problem. Scheduling tasks to process with uniform weights and scheduling tasks to one or two processors with weights equal [9]. Guo et al. [10] have suggested deterministic methods such as the algorithm of branch as well as bound to the resource-scheduling issue and heuristic algorithms are suggested with jobs based on higher processing time were given higher priority than jobs with less processing time [11]. Heuristic algorithms like Particle Swarm Optimization (PSO) was applied in cloud computing environment for scheduling intensive tasks in order to minimize the processing cost [12].

Another heuristic approach using PSO was suggested to task scheduling at cloud infrastructure to optimize the computation and transmission cost [13]. This study shows that PSO approach optimizes the workload on the recourses. A study in the cloud environment was to analyse the problems with load-balancing techniques. To handle critical situations a
new algorithm was incorporated to proactively influence the load-balancing decisions [14]. An improved algorithm was suggested for processing power in dynamic load-balancing strategy, which helps to improve the client service quality [15].

Now a days, with the fast development of efficient hybrid optimisation algorithms, such as adaptive hybrid heuristic [16], fuzzy GA optimisation [17], Ant Colony Optimization and PSO hybrid [18], PSO and Cuckoo Search [19], Firefly algorithm [20], Honey bee algorithm [21], improved GA [22] hybrid meta heuristics such as gravitational emulation local search (GELS) algorithm [23] and so on, were also applied to address the task scheduling including resource management issues.

The main objective of this research was to minimize the execution cost in the process of task scheduling and resource management in VMs of cloud infrastructure. The cost factor of cloud infrastructure was a measure to be considered. The various optimisation algorithms optimize the tasks by removing the tasks from overloaded VMs. The workload distribution among the VMs was based on dynamic load balance strategy. This dynamic strategy ensures that every computing resource was distributed uniformly. In the present study, a novel kind of optimisation algorithm was needed to minimize the execution cost and distribute the resources efficiently.

Lawanyashri et al. [24] suggested an energy-aware hybrid FFO to load balance the cloud environments to electronic health records applications. In this manuscript, a hybrid FFOA along a simulated annealing method to optimize the speed of convergence and quality of the solution has been suggested.

Shridhar Domana et al. [12] suggested a hybrid bioinspired algorithm for task scheduling including resource management at cloud infrastructure. It have been allocated the tasks for VMs in a proficient way utilising modified PSO, then the allocation or resource management as required by the tasks, was manipulated with proposed Hybrid Bio-Inspired algorithm (Modified PSO + Modified CSO). Experimental results shows that the proposed HYBRID algorithm outperforms peer research and other algorithms (ACO, MPSO, CSO, RR and Exact algorithm based on branch-and-bound technique) depending upon proficient use of cloud resources, optimized reliability and minimized average response time.

Wu Deng et al. [25] have introduced an enhanced Ant Colony Optimization Algorithm depending upon Hybrid techniques for scheduling problem. Here, the optimisation issue was segmented as many sub-problems also the ants in the population were segmented as elite ants including common ants to upgrade the rate of convergence, avert falling into the local optimum value. The pheromone updating technique was applied to update optimisation capability. The pheromone diffusion mechanism was applied to produce the pheromone that ants release in a specific point, which gradually affects nearby areas to a specific range. The co-evolution strategy was utilized to exchange information amid various sub-populations in order to implement information sharing. Table 1 tabulates the overall comparison of literature survey.

### 2.1 Problem statement and motivation of the work

The related work section indicates that various techniques were used in task scheduling together with resource management problem in cloud infrastructure. However, there are some drawbacks existed. Existing works does not considered dynamic resources requirement in efficient manner, also take long time for scheduling. The above mentioned drawbacks in existing systems motivated us to propose this technique.

### 3 PROBLEM DESCRIPTION

This study states that a set of cloud tasks with a set of cloud resources, like CPU, storage, network and memory that has to schedule. The VMs are the computing resources; the tasks are processed with the help of VMs, which are assigned to one or
more tasks. In cloud environment, the task scheduling as well as resource allocation process are actually NP complete problems [26]. Task scheduling problem aims to decrease the execution time also optimize the utilisation resources. At cloud computing environment, the task scheduling is declared as below: The overall implementation time of the task is called make span [27]. The Equation (1) defines the make span as follows:

\[
\text{Makespan} = \max\{CT_{ij}; i \in T, i = 1, 2, \ldots n \text{ and } j \in VM, j = 1, 2, \ldots m}\tag{1}
\]

Here, \(T_i\) denotes the task completion time in \(VM_j\) as \(CT_{ij}\) [2]. To formulate the problem, a set of \(m\) VMs is represented as \(VM = \{VM_1, VM_2, \ldots VM_m\}\) is a set of \(m\) VMs, \(T = \{T_1, T_2, \ldots, T_n\}\) represents set of \(n\) tasks. In the model \(R\) denotes that all the VMs are independent and parallel. Scheduling of these independent tasks to these VMs cannot be interrupted. \(CT_i\) denotes the completing time of task as \(T_i\). \(CT_{\text{max}}\) denotes the make span. The model \(R\) is represented as \(R|\text{npmt}|CT_{\text{max}}\). The task \(T_i\)'s execution time is represented as \(P_{ij}\) on virtual machine \(VM_j\). All tasks execution time in a \(VM_j\) is:

\[
P_j = \sum_{i=1}^{n} P_{ij} \quad j = 1, \ldots, m \tag{2}
\]

Equation (3) is obtained by minimising \(CT_{\text{max}}\).

\[
\sum_{i=1}^{n} P_{ij} \leq CT_{\text{max}} \quad j = 1, \ldots, m \tag{3}
\]

Equation (4) is obtained from Equations (2) and (3).

\[
P_j \leq CT_{\text{max}} \quad j = 1, \ldots, m \tag{4}
\]

Based on VM's capacity, the task's processing time changes from one VM to other. \(CT_{\text{max}}\) representation is Equation (5):

\[
CT_{\text{max}} = \max_{i=1}^{n} CT_i, \max_{j=1}^{m} \sum_{i=1}^{n} P_{ij} \tag{5}
\]

Equation (6) shows the capacity of a VM

\[
C_j = P_{\text{eumj}} \times P_{\text{einputj}} + v_{\text{mbw}} \tag{6}
\]

where, \(P_{\text{eumj}}\) is the processing element, \(VM_j\) denotes processors number for all processors, \(P_{\text{einputj}}\) represents millions of instructions per second and \(v_{\text{mbw}}\) represents communication bandwidth.

Equation (7) represents all VMs capacity.

\[
C = \sum_{j=1}^{m} C_j \tag{7}
\]

The data centre capacity is the summation of all VMs. Equation (8) represents the load on VM.

\[
L_{\text{eVM, t}} = \frac{N(T, t)}{S(VM_t)} \tag{8}
\]

On a service queue of \(VM_i\), \(N\) denotes the number of tasks and \(t\) denotes time.

Equation (9) implies the load of all VMs.

\[
L = \sum_{i=1}^{m} L_{\text{eVM, t}} \tag{9}
\]

Equation (10) shows VM's processing time.

\[
PT_i = \frac{L_{\text{eVM, t}}}{C_t} \tag{10}
\]

Equation (11) determines all VM's processing time.

\[
PT = \frac{L}{C} \tag{11}
\]

Equation (12) shows the standard deviation of load [3].

\[
\sigma = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (PT_i - PT)^2} \tag{12}
\]

4 | SOLUTION REPRESENTATION

Heuristic algorithms are appropriate to solve the task scheduling together with resource allocation issue. Heuristic algorithms help to identify the optimal solution to this problem in rapid time [28]. In this study, Fruit fly-based optimisation is used to task scheduling and allocating resources. To address the task-scheduling issue, a set of tasks are given as input. The aim is to assign one or more tasks to one VM [29].

A permutation of integers represents a task. An exclusive integer ID has been assigned to each task represents a solution. For example, if there are 10 tasks their IDs indicated as T1, T2, T3, T4, T5, T6, T7, T8, T9 and T10. In the proposed algorithm, each fruit fly denotes a solution for task allocated with \(n\) elements in an array. All the elements in an array is a random integer from 1 to \(n\).

The solution representation of assigning 10 tasks to 5VMs is shown in Table 2. For example, Fruit Fly 1, tasks T1, T4 and
| Task IDs | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ | $T_9$ | $T_{10}$ |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Fruit fly 1 | VM1 | VM2 | VM3 | VM1 | VM2 | VM3 | VM1 | VM4 | VM5 | VM4 |
| Fruit fly 2 | VM2 | VM4 | VM1 | VM2 | VM3 | VM5 | VM1 | VM4 | VM4 | VM3 |
| Fruit fly 3 | VM1 | VM5 | VM2 | VM4 | VM3 | VM2 | VM5 | VM3 | VM2 | VM1 |

**TABLE 2** Solution representation

![Figure 1: Foraging process of the fruit flies](image)

T7 are allocated to VM1, tasks T2 and T5 are allocated to VM2, tasks T3 and T6 are allocated to VM3, tasks T8 and T10 are allocated to VM4, tasks T9 is allocated to VM5 [30].

5 | FRUIT FLY-BASED ALGORITHM TO SOLVE TASK SCHEDULING

This segment illustrates the standard fruit fly algorithm. The different variants of fruit fly algorithm such as modified FFO approach, upgraded FFO approach, multi-swarm FFO approach and their fitness function for task scheduling problem is discussed [31].

5.1 | Fruit Fly Task Scheduling Algorithm

Moderate and humid regions are suitable for fruit flies to live. These flies are better than other species in terms of smell and vision. During hunting, it flies in random direction guided by some odour to locate the position. During the food hunting process, the fruit flies interacts with the neighbours and compares with best solution so far and its fitness. FFOA is a global optimisation technique in terms of food foraging behaviour of fruit fly. It is an interactive evolutionary method with two-search process namely smell and vision.

It locates the food source using smell to fly toward the exact place in the smell search process. With the help of sensitive vision, it flies toward the food source in vision search process. Figure 1 depicts the foraging process of fruit fly [32]. The fruit fly searches for the under-utilized VM (food) to allocate the removed task. If more than one under-utilized VM is available in the same data centre based on task priority with load on the virtual machine, the decision is carried out, that is, finding an optimal solution. The task list and load of all VM in VM list is updated for waiting tasks after submitting the under-loaded VM. When updating the status of the VM will be helpful for the other fruit flies to search their food.

The fruit fly's food for ageing process is classified as parameter settings and initial population location, smell search part and vision search part. Figure 2 portrays the basic fruit fly algorithm. The parameters in fruit fly population size represented as $NP$, maximum iteration number represented as $T$, random flight distance $rand$ value. The random initialisation of the search space is described in the Equations (13) and (14).
where, \( \text{rand} \) denotes random function, the other parameters are upper and lower bound in two-dimensional search space. Based on smell search part, population size \( NP \) is generated in random. The individual fruit fly’s distance and direction of flight is represented below,

\[ X_i = X\text{ axis} + r\text{Value} ; \]
\[ Y_i = Y\text{ axis} + r\text{Value} ; \]  

where, \( i = 1, 2, \ldots, NP \) and \( r\text{Value} \) indicates random number from \(-1 \) to \(1\). \( \text{Dist} \) represents distance of food location and the smell concentration denoted as \( (S_i) \).

\[ \text{Dist} = \sqrt{x_i^2 + y_i^2} \]
\[ S_i = 1/\text{Dist} ; \]

For each fruit fly (Smell) denotes smell concentration and the fitness function is calculated.

\[ \text{Smell}_i = \text{fitness} (S_i) ; \]

The vision search part uses greedy selection method. The individual with maximum smell value is identified in the swarm.

Next, \( (\text{Bsmell}) \) denoted as current maximum smell value is compared with \( (\text{Sbest}) \) denoted as previous value. If \( \text{Bsmell} \) is lesser than \( \text{Sbest} \) then \( \text{Sbest} \) is updated with \( \text{Bsmell} \) and using it, it flies to a new location with maximum smell value.

\[ [\text{Bsmell}, \text{bestIndex}] = \text{min} (\text{Smell}) ; \]
\[ \text{Sbest} = \text{Bsmell} ; \]
\[ X\text{ axis} = X (\text{Bindex}) ; \]
\[ Y\text{ axis} = Y (\text{Bindex}) ; \]

The smell and vision search process are repeated until \( \text{Smell}_i \) is not better than the previous iterative \( \text{Smell}_i \) or when it reaches maximum number of iterations [32].

\[ 1. \text{Initialize NP, } T, \text{rand value} \]
\[ 2. \text{Random initiation the position of population } x\text{-axis, } y\text{-axis} \]
\[ 3. X_i = X\text{-axis} + \text{randValue} \]
\[ 4. Y_i = Y\text{-axis} + \text{randValue} ; \]
\[ 5. \text{Dist} = \sqrt{x_i^2 + y_i^2} ; \]
\[ 6. S_i = 1/\text{Dist} ; \]
\[ 7. \text{Smell}_i = \text{fitness}(S_i) ; \]
\[ 8. [\text{Bsmell}, \text{bestIndex}] = \text{min}(\text{Smell}) ; \]
\[ 9. \text{Sbest} = \text{Bsmell} ; \]
\[ 10. X\text{ axis} = X(\text{bestIndex}) ; \]
\[ 11. Y\text{ axis} = Y(\text{bestIndex}) ; \]
\[ 12. \text{While } i<T \]
\[ 13. X_i = X\text{-axis} + \text{randValue} ; \]
\[ 14. Y_i = Y\text{-axis} + \text{randValue} ; \]
\[ 15. \text{Dist} = \sqrt{x_i^2 + y_i^2} ; \]
\[ 16. S_i = 1/\text{Dist} ; \]
\[ 17. \text{Smell}_i = \text{fitness} (S_i) ; \]
\[ 18. [\text{bestsmell}, \text{bestIndex}] = \text{min} (\text{Smell}) ; \]
\[ 19. \text{if best smell } < \text{Smell best} \]
\[ 20. \text{smellbest } = \text{bestSmell} \]
\[ 21. X\text{ axis} = X(\text{bestIndex}) ; \]
\[ 22. Y\text{ axis} = Y(\text{bestIndex}) ; \]
\[ 23. \text{end if} \]
\[ 24. i = i+1 \]
\[ 25. \text{end while} \]
\[ S_i = 1/\text{Dist}_i + \Delta, \Delta = \text{Dist}_i \times (0.5 - \delta), \ 0 \leq \delta \leq 1 \quad (24) \]

To enhance the overall search ability and produce negative fitness, the escape parameter \( \delta \) is used.

In standard FFOA, the fruit flies moving in two-dimensional search spaces for global optimal solution. The fruit flies moving in three-dimensional space to identify the food position. Hence, the food distance is extended in three-dimensional search spaces using the Equation (25).

\[ \text{Dist} = \sqrt{x_i^2 + y_i^2 + z_i^2} \quad (25) \]

### 5.3 Improved Fruit Fly Task Scheduling Algorithm

In smell search part, the standard fruit fly algorithm uses a fixed search radius [35]. An enhanced version of the algorithm has been introduced to eliminate the fixed values in the search radius. In order to optimise the efficiency of the fruit fly algorithm and remove the disadvantages with search radius fixed values, also modifying the search radius dynamically with the count of iteration, so the search radius can be tuned to various evolution phases. Unlike the basic FO that modifies the entire decision variant of the swarm location to create a new solution, we randomly select one decision variant along uniform distribution to enhance the intensive search. The comparative studies are conducted in terms of 30 different test operations. The experimental outcomes show that the proposed improved fruit fly algorithm is an efficient search algorithm; moreover, it performs accurately than five state-of-art harmony search algorithms on solving higher-dimensional functions. The Equation (26) shows how the search radius is changed dynamically.

\[ \lambda = \lambda_{\text{max}} \times \exp \left( \ln \left( \frac{\lambda_{\text{min}}}{\lambda_{\text{max}}} \right) \times \frac{t}{T} \right) \quad (26) \]

where, \( \lambda \) denotes search radius, \( \lambda_{\text{min}} \) denotes minimum search radius and \( \lambda_{\text{max}} \) denotes maximum search radius. The decision variables are used to evaluate the fitness by eliminating the parameters \( \text{Dist}_i \) and \( S_i \). To increase the search intensively, a random decision variable is selected to produce current solutions as shown in Equation (27).

\[
X_{ij} = \begin{cases} 
X_{-\text{axis}_j} \pm \lambda \times \text{rand}, & \text{if } j = d \\
X_{-\text{axis}_j}, & \text{otherwise}
\end{cases} \quad (27)
\]

where, \text{rand} denotes an uniform distribution that returns a value \([0,1]\), random integer \(d[1,n]\) and the number of decision variables \(n\).

### 5.4 Multi-Swarm Fruit Fly Task Scheduling Algorithm

To improve the solution variety, to acquire effectual exploration and to avoid local optima, a multiple-swarm fruit fly Optimization algorithm (MSFOA) [36] is proposed. MSFOA is proved to upgrade better and the rate of convergence speed of original algorithm. When it comes to Multi-Swarm Fruit Fly Task Scheduling Algorithm (MSFFTSA) depending upon FFOA. MSFFTSA compute convergence rate and allocate the task more accurately than other FTAs. Many sub-swarms are produced from fruit fly swarms and these sub-swarms move autonomously at search space to obtain global optimal solution. The exploring radius for smell is modified to every fruit fly in the group and its current location is generated.

\[ X_i = X_{-\text{axis}} + R(t) \times \text{Random Value} , \quad (28) \]

\[ R(t) = \left( \frac{UB_t - LB_t}{2} \right) \times \left( \frac{T - t}{T} \right)^{\varrho} \quad (29) \]

Here, \( R(t) \) denotes exploring radius. \( R(t) \) variable is assigned as multi-scale iteration \( t \) in the smell search process and \( \varrho \) equal to 2 ~ 6. The upgraded Fruit Fly Task Scheduling Algorithm (FTSA) of MSFTSA is compared with different Fruit Fly Task Scheduling Algorithms and experiment results show MSFTSA is better when compared with FTSA, MSFFTSA and IFTSA algorithms.

### 6 EXPERIMENTAL RESULTS

#### 6.1 Test platform and parameter settings

The test platforms for the experiments are Processor: CPU Intel Core I7-2670, Frequency: 2.20 GHz, Memory: 8 GB, Operating System: Windows 7, Simulation software: CLOUDSIM 3.0.3.

#### 6.1.1 CLOUDSIM parameter setting

An open source simulator called Cloudsim 3.0.3, capable of running in Linux and Windows operating systems is used in this study [6].

The simulator is used to implement the task scheduling process based on the standard fruit fly swarm technique and compared with its variants. The CLOUDSIM toolkit supports cloud system modelling components like data centres, host, VMs, scheduling with resource provisioning policies. A toolkit is an application that opens up the possibility of evaluating hypotheses before software development in an environment where experiments can be developed. Here, 50...
VMs utilising VM component, set the RAM property as 256–2084 MB for all VMs, MIPS as 500–1000. It generates 12 tasks using Cloudlet component and set the Cloudlet length property into 1000–2000. Table 3 shows the parameter settings of CLOUDSIM simulator.

### 6.1.2. Fruit fly algorithm parameters

In this study, the algorithms is assigned population size as $NP$ of 10–50, the maximum iterations used 1000. Rand value = 1, $M = 5$, $\phi = 6$, $\lambda_{\text{max}} = \lambda_{\text{max}} - (\text{upperbound} - \text{lowerbound})/2$ and $\lambda_{\text{min}} = 1 \times 10^{-5}$.

### 6.2 Simulation results and comparisons

This segment demonstrates the performance analysis of the proposed algorithm is compared with its variants. Every algorithm is run 10 times independently and the average value is taken for evaluation.

To assess the performance of the Fruit Fly Task Scheduling Algorithm (FFTSA), Modified Fruit Fly Task Scheduling Algorithm (MFFTSA), Improved Fruit Fly Task Scheduling Algorithm (IFFTSA) and Multi-Swarm Fruit Fly Task Scheduling Algorithm (MSFFTSA) algorithm, parameters such as VMs, population size are used for comparison. Figure 3 shows VMs response time in seconds for FFTSA, MSFFTSA, IFFTSA and MSFFTSA. The X-axis denotes...
Algorithm is efficient in terms of task allocation and resource using VMs efficiently without overload. Thus, the MSFFTSA is better than other algorithms depending upon the number of tasks. The experimental results illustrate there was a major load was balanced to obtain maximum resource utilisation.

The experimental results show MSFFTSA is better when compared with FFTSA, MSFFTSA and IFFTSA algorithms. Figure 4 compare time span and number of tasks for FFTSA, MSFFTSA, IFFTSA. This graph illustrates that MSFFTSA is capable when assessed with other variants of the algorithms. Around 250 tasks were used for our comparisons.

Figure 5 represents the comparison of population size and the make span for FFTSA, MSFFTSA, IFFTSA and MSFFTSA. X-axis represents population size and Y-axis represents make span in seconds. The population size is varied and make span is analysed. The graph clearly shows that the make span duration is minimal for IFFTSA and MSFFTSA algorithms when likened with other algorithms. Figure 6 depicts the comparison of VMs and number of tasks in the proposed algorithm.

The X-axis implicates VMs, whereas Y-axis implicates number of tasks. The results are analysed, and it is found that MSFFTSA is better than other algorithms depending upon utilising the maximum amount of resources for given tasks by using VMs efficiently without overload. Thus, the MSFFTSA algorithm is efficient in terms of task allocation and resource utilisation.

7 CONCLUSIONS

In this manuscript, an upgraded FFOA was proposed for optimising task scheduling and resource management processes with a measure to decrease the execution time and increase the system resource utilisation. Task prioritisation was also considered in this algorithm. First, a basic approach was put forward to allocate the tasks to multiple computing resources also the implementation time of the task was calculated. Second, the overloaded VMs were identified and the load was balanced to obtain maximum resource utilisation. The experimental results illustrates there was a major improvement in average implementation time and also optimum resource utilisation of VMs in proposed algorithm. Finally, the implemented algorithm was likened to the other algorithms then the experimental results demonstrate the MSFFTSA technique is better than the other algorithms of fruit fly algorithm.

REFERENCES

1. Marvi, S.S. Shyam, G.K.: Resource management for Infrastructure as a Service (IaaS) in cloud computing: a survey. J. Netw. Comput. Appl. 41, 424–440 (2014)
2. Navinipour, N., Milani, F.S.: Task scheduling in the cloud computing based on the cuckoo search algorithm. Int. J. Model. Optim. 5(1), 44 (2015)
3. Krishna, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. 13(5), 2292–2303 (2013)
4. Meng, T., et al.: An enhanced fruit fly optimization for the flexible shop scheduling problem with lot streaming. 2018 37th Chinese Control Conference (CCC), IEEE (2018)
5. Tan, Y., et al.: A self-adaptive modified fruit fly optimization algorithm. 2017 36th Chinese Control Conference (CCC), IEEE (2017)
6. Calheiros, R.N., et al.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exper. 41, 23–50 (2011)
7. Transpire Online.: Behavior of FFO using meta-heuristics method, Transpire Online (2019). https://transpireonline.blog/tag/optimization-algorithm/
8. Guddeti, R.M., Buyya, R.: A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment IEEE Trans. Serv. Comput. 7, 23–50 (2014)
9. John Yue, J., et al.: Enhanced cloud sim. U.S. Patent Application No. 15/085,689.
10. Guo, L., et al.: Task scheduling optimization in cloud computing based on heuristic algorithm. J. Netw. 7(3), 547 (2012)
11. Ponz-Tienda, J.L., Salcedo-Bernal, A., Pellicer, E.: A parallel branch and bound algorithm for the resource leveling problem with minimal lags. Comput. Aided Civ. Infrastruct. Eng. 32(6), 474–498 (2017)
12. Guddeti, R.M., Buyya, R.: ‘A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment’. IEEE Trans. Serv. Comput. 13(1), 2017
13. Pandey, S., et al.: A particle swarm optimization (PSO)-based heuristic for scheduling workflow applications in cloud computing environments. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 400–407, Perth (2010)
14. Radojevic, B., Zagari, M.: Analysis of issues with load balancing algorithms in hosted (cloud) environments. MIPRO, 2011 Proceedings of the 34th International Convention Opatija, Croatia (2011)
15. Ren, X., Lin, R., Zou, H.: A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast. Cloud computing and intelligence systems, CCIS, Beijing (2011)
16. Rahman, M., Li, X., Palit, H.R.: Hybrid heuristic for scheduling data Analytics workflow applications in hybrid cloud environment. Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW). Shanghai (2011)
17. Tayal, S.: Tasks scheduling optimization for the cloud computing systems. Int. J. Adv. Eng. Sci. Technol. (IJAEIST), 5, 111–115 (2011)
18. Wen, X.T., Huang, M.H., Shi, J.H.: Study on resources scheduling based on ACO algorithm and PSO algorithm in cloud computing. 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science. Guilin, China (2012)
19. Al-maamari, A., Omara, F.: Task scheduling using hybrid algorithm in cloud computing environments. IOSR J. Comput. Eng. 17, 96–106 (2015)
20. Esa, D.I., Younis, A.: Scheduling jobs on cloud computing using firefly algorithm. Int. J. Grid Distrib. Comput. 9(7), 149–158 (2016)
21. Krishna, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments, Appl. Soft Comput. 13(5), 2292-2303 (2013)
22. Zhong, H., Tao, K., Zhang, X.: An approach to optimized resource scheduling algorithm for open-source cloud systems. ChinaGrid Conference (ChinaGrid). Guangzhou, China (2010)

23. Pooranian, Z., et al.: Hybrid metaheuristic algorithm for job scheduling on computational grids. Informatica, 37(2), 157 (2013)

24. Lawanyashi, M., Balasamy, B., Subha, S.: Energy-aware hybrid fruit fly optimization for load balancing in cloud environments for EHR applications. Inform. Med. Unlocked, 8, 42–50 (2017)

25. Deng, W., Xu, J., Zhao, H.: An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. IEEE Access, 7, 20281–20292 (2019)

26. Jiang, H., et al.: A multi-objective algorithm for task scheduling and resource allocation in cloud-based disassembly. J. Manuf. Syst. 41, 239–255 (2016)

27. Mythili, S., et al.: Ideal position and size selection of unified power flow controllers (UPFCs) to upgrade the dynamic stability of systems: an antlion optimiser and invasive weed optimisation algorithm. Trans. Hong Kong Inst. Eng. 27, 25–37 (2020)

28. Xia, W., et al.: Programmable hierarchical C-RAN: from task scheduling to resource allocation. IEEE Trans. Wireless Commun. 18(3), 2003–2016 (2019)

29. Pan, Y., Shi, Y.: A grey neural network model optimized by fruit fly optimization algorithm for short-term traffic forecasting. Eng. Lett. 25(2), 198–204 (2017)

30. Sharma, V., et al.: OFFRP: optimised fruit fly based routing protocol with congestion control for UAVs guided ad hoc networks. IJAHUC, 27(4), 233–255 (2018)

31. Liu, Q., et al.: A hybrid fruit fly algorithm for solving flexible job-shop scheduling to reduce manufacturing carbon footprint. J. Clean Prod. 168, 668–678 (2017)

32. Wu, L., Zuo, C., Zhang, H.: A cloud model-based fruit fly optimization algorithm. Knowl. Base Syst. 89, 603–617 (2015)

33. Pan, W.T.: Using modified fruit fly optimization algorithm to perform the function test and case studies. Connection Sci. 25, 151–160 (2013)

34. Han, Y., et al.: Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. Int. J. Prod. Res. 54(22), 6782–6797 (2016)

35. Pan, Q.K., et al.: An improved fruit fly optimization algorithm for continuous function optimization problems. Knowl.-Based Syst. 62, 69–83 (2014)

36. Yuan, X., et al.: On a novel multi-swarm fruit fly optimization algorithm and its application. Appl. Math Comput. 233, 260–271 (2014)

How to cite this article: Loheswaran K. An upgraded fruit fly optimisation algorithm for solving task scheduling and resource management problem in cloud infrastructure. IET Netw. 2021;10:24–33. https://doi.org/10.1049/ntw.12001