Received: February 12, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

OCSA: Task Scheduling Algorithm in Cloud Computing Environment

Pradeep Krishnadoss[1]* **Prem Jacob[2]**

[1]*St. Joseph's College of Engineering, Chennai, Tamilnadu, India*
[1, 2]*Sathyabama Institute of Science and Technology, Chennai, Tamilnadu, India*
*Corresponding author's Email: pradeepkrishnadoss@gmail.com

Abstract: Cloud computing offers dynamic allocation of resources on demand, the feature which makes it to stand apart providing great performance, scalability, cost efficient and less maintenance, thus making it an apt choice. Task scheduling becomes the essential factor in increasing the performance for the dynamic allocation of resources which is most essential in the cloud environment to increase performance and decrease the cost. In this work, a solution is proposed using makespan and cost, taking them as important constraints for the optimization problem. We have merged two algorithms namely, cuckoo search algorithm (CSA) and oppositional based learning (OBL) and created a new hybrid algorithm called oppositional cuckoo search algorithm (OCSA) to provide solution to the above stated issue. Our proposed OCSA algorithm showed noticeable improvement over the other task scheduling algorithms. The proposed work is simulated in cloudsim programming environment and the simulation results show the effectiveness of the proposed work by minimizing cost and makespan parameters. The obtained results are better in comparison to other existing algorithms like particle swarm optimization (PSO), Improved Differential Evolution Algorithm (IDEA) and genetic algorithm (GA).

Keywords: Task scheduling, Cuckoo search algorithm, Oppositional based learning, Resource, Makespan, Cost.

1. Introduction

In the fast growing global business environment, maintaining rapid application development in the information technology sector has been cumbersome. Expediting the software deployment strategies by reducing the time and effort requires the application of a recent trend called Cloud computing. Cloud computing is everywhere, in the simplest terms it can be defined as storing and accessing data and programs over the Internet instead of your computer's hard-drive. Cloud is just a metaphor of the Internet. All the applications in cloud computing are provided as services each time and every time it is demanded. Thus the services including application storage, network, server and other services can be utilized effectively and efficiently. This results in enormous savings with respect to time and cost.

Most of the quality of services parameters including execution time, cost, scalability, reliability, energy and load balancing have been achieved to a remarkably satisfying level with the help of cloud computing. In cloud computing, customer may be provided with numerous virtualized resources to utilize; it is not possible for anyone to allocate the jobs manually. Hence, to allocate the resources to the virtual machine layer, the load balancing algorithm becomes essential.

1.1 Cloud models

The models in cloud computing are classified as follows:

Public cloud model: This model is defined as a cloud computing infrastructure owned and operated by the third party service provider to provide services to various types of users, and these services are provided through the internet. An advantage of this model is that they are typically larger in scale. The customers on public cloud share the same infrastructure pool with limited configurations and

security protection as they are wholly managed by the service provider.

Private cloud model: It is constructed exclusively for each project and it is also known as cloud computing infrastructure. While giving the concerns regarding security and control, it also gives permission to host applications in the cloud. It also overcomes some of the difficulties posed by a public cloud because the model can also be hosted by an outside cloud supplier.

Hybrid cloud model: This model is defined as a cloud computing infrastructure that combines the advantages of both public and private cloud model using special methods that allow data and application transfer between them.

1.2 Cloud computing services

Infrastructure as a service (IaaS) enables users to remotely access a network, storage or computing infrastructure. In other words it delivers cloud computing infrastructure-as an on-demand service, thus outsourcing the vital resources at a lower cost. E.g. Amazon Web Service, Microsoft Azure.

Platform as a service (PaaS) is a cloud computing platform that gives permission to create web applications in a very fast and simple way, thereby providing a relief from buying and maintaining the software and infrastructure beneath it. Google App Engine is the best example for this.

Software as a service (SaaS) enables a provider to license an application to customers either as a service on demand or through subscription at no charge through the Internet. In other words rather than buying the software, it can be rented. It is a pay-as-you-go model. E.g. Salesforce, Cisco WebEx.

1.3 Cloud computing tools

Cloud tools and technologies are used to sort out the business solutions based on the organizational requirements. There are numerous cloud computing tools namely Eucalyptus, Open Nebula, Nimbus, and Openstack, each having a variety of strategies for deployment.

Load balancing in cloud computing: It is the process of distributing workload and computing resources in a cloud computing environment. Load balancing allows enterprise to manage application or workload demands by allocating resources among multiple computers, networks or servers, thus accounting for substantial savings in terms of time and cost.

1.4 Types of load balancing algorithms

Load balancing algorithms are of two types namely static and dynamic. The static scheduling technique is of non pre-emptive type. In this technique, tasks are assigned to the processor during compile time i.e. before the program execution. The main objective of static scheduling algorithm is to reduce the execution time. However, the disadvantage is that it is not feasible to change the load during execution time, as once the load is allocated to the node it cannot be transferred.

Dynamic scheduling is based on monitoring changes on the system workload and redistributing the work accordingly. Redistribution is carried out by transferring the jobs between the processors based on workload, thus improving the efficiency of the system. This technique has one drawback where it incurs a run-time overhead among processors.

Cloud computing comprises of a data centre, which are distributed geographically. There exists a data centre controller, which handles jobs, as and when it is submitted by a customer. The controller uses a VM to process an incoming request.

Round Robin algorithm involves the requests to be provided in a circular basis with specific time quantum or in other words a VM is assigned the first request, after which, the remaining are assigned in a circular fashion. Although the workload distribution processors are similar, the job processing time is not the same. The limitation of this algorithm is that at any point of time some system may be loaded heavily, while others may be idle. Another version of the round robin is the weighted round robin.

In this method, each VM is assigned a weight. If one VM can handle twice as much load as another VM, the power has a weight of two. In which case, for every request assigned to a weaker VM, two requests will be assigned to the powerful VM. The disadvantage of this algorithm is that the advanced load balancing requirements like processing time is not considered. In [1] the authors provide a comparative analysis on the basis of Quality of Service (QoS) to project clear ideas on scheduling algorithms**.**

In [2] a new technique called Optimal Time Algorithm (OTA) is presented to develop productiveness of tasks performance remarkably. It does not take into consideration the difference between implementation times of tasks on same processor; clusters of tasks that are dissimilar may

be formed. This method efficiently makes use of the features and qualities of MapReduce tasks for building a best task scheduling. It gets rid of the short comings involving the minimization of makespan of workloads with the help of the workflow of MapReduce jobs.

In [3] a hybrid algorithm is proposed to consolidate the advantages of Ant Colony Optimization Algorithm (ACO) and Cuckoo search. The reduction of makespan time is possible using this algorithm. Jobs are executed in specific time intervals by allocating the required resources. The result obtained proves that hybrid algorithm performs better than ACO algorithm based on the performance and makespan.

The migration of multiple co-located and live virtual machines from one node to another is required for power saving on cloud platforms. This migration is vital for running services, and due to this reason it needs to be completed quickly. These live-migration strategies optimize the migration performance of one or more VMs by focusing on Virtual Machine Monitor (VMM) and pay less attention towards the cloud platforms which controls and schedules multiple migrations. This paper considers the drawbacks of scheduling operations to reduce makespan [4].

For most favourable use of cloud's power, we require effective and efficient scheduling algorithms that in turn choose the most optimal resources for task execution. It involves the assignment of tasks to the existing resources in such a way that helps in maximizing utilization and reduces makespan. The complete time required for all tasks to finish is known as makespan. Utilization can be said to be the overall extent to which network resources are used. The task scheduling is categorized under NP-complete problem and meta-heuristics, due to its heterogeneous nature and dynamic resource requirement. The aim of this paper is to optimize task scheduling which uses Particle Swarm Optimization (PSO) algorithm to reduce the makespan. Inertia weights that have been used are different. The Linear Descending Inertia Weight (LDIW) with a mean of 22.7% reduction in makespan shows that the performance is best [5].

The important topic which deals in cloud computing is task scheduling and energy usage. There are two steps in scheduling process, one is job prioritization and the other is processor choosing. Various priorities may go to different manufacturing time and for every processor the usage of energy may differ. Thus an optimal scheduling algorithm must involve job prioritization and choose the optimal processor in such a manner that manufacturing time and energy usage is decreased. This paper [6] explains two steps in process algorithm for scheduling named as TETS in which the starting step includes prioritizing job and second includes processor allocation. There are three prioritization procedures to prioritize jobs to produce best starting chromosomes and allocating jobs to processors. The outputs after simulation of algorithm showed that this algorithm is much better than the previous algorithms on the basis of energy usage and manufacturing time. It is capable of improve the energy usage by 20% and manufacturing time by 4%.

In [7] a solution is proposed using multiple techniques and algorithms to fulfil the requirements of users by making sure that the tasks are completed on time with the given cost. This paper puts forward a job scheduling algorithm named as Time Based Ant Colony Optimization Algorithm (TBACO) which is inspired from basic ACO. The aim is to decrease the manufacturing time of given tasks once they are scheduled on the different datacenters. These datacenters are known as the resource providers in cloud. This is simulated in CloudSim and the results clearly showed that TBACO algorithm performed much better than basic ACO with a savings of 59% in makespan.

In this paper a convex optimization model is proposed which maximizes power savings while satisfying the job completion deadline and also maintaining the temperature within admissible bounds. By varying the frequency of operation of core, optimization is ensured. The main characteristics of this approach are the bounds of operation. The allowed operation limit enables the elasticity of computing resources in an autonomic manner to achieve a pareto front for the solutions. For validation purpose simulations can be done on a variable workload [8].

Many computing resources allow the implementation of jobs in cloud surroundings. Thus, it becomes essential to find the suitable node for finishing a job; this improves the process of large-scale cloud computing circumstances. The collection of required information and analysis of service providers to make decisions is quite time-consuming for consumers. This is a demanding job from computer usage perspective since the same computation may be processed repeatedly by different customers of the same needs. The performance is increased by distributing load among more than one node of the distributing system for effective resource usage and task response time. It also makes sure that all processors do close to the same task at any instance of time. The tool which

helps users to simulate large-scale cloud applications with the aim of understanding the process under various deployment configurations is Cloud Analyst. The simulated results are based on the algorithm [9].

This paper illustrates MMSF which is minimum makespan task scheduling framework and MMA which is minimum manufacture time of the job scheduling algorithm. This algorithm is made with two goals. That includes reducing the total manufacturing time and increasing the virtual machine usage. The job scheduling problem is said to be a multi-objective optimization problem. Optimization techniques are used to solve it. The experiment proves that MMA performs better than traditional task algorithms in terms of total makespan and virtual machine utilization [10].

This paper presents two SLA-based job scheduling algorithms; the following are SLA-MCT and SLA-Min-Min for heterogeneous multi-cloud surrounding. The first one is single-phased and the next one is two-phased in scheduling. Three levels of SLA identified by customer are supported in this algorithm. The algorithms also incorporate the SLA gain cost for successful completion of the services and SLA violation cost for the unsuccessful end of services. Benchmarks and synthetic datasets are used for simulation. The outputs of the explained SLA-MCT are compared with three single-phase job scheduling algorithms and the following are CLS, Execution-MCT, and Profit-MCT, and the results of SLA-Min-Min are compared with two-phase scheduling algorithms, which are Execution-Min-Min and Profit-Min-Min in terms of four process metrics, namely manufacturing time, mean cloud usage, profit, and penalty cost of the services. The output proves that this algorithm effectively maintains a balance between manufacturing time and profit cost of the services [11].

This paper presents a novel cloud-based workflow scheduling (CWSA) policy for executing-intensive jobs applications in multi-tenant cloud computing surroundings, which assists in decreasing all job completion time, tardiness, cost of implementation of the jobs, and make use of unique resources of cloud effectively. The presented algorithm is compared with the traditional algorithms, i.e., First Come First Served (FCFS), EASY Backfilling, and Minimum Completion Time (MCT) scheduling policies to determine the process. A proof-of-concept experiment of real-world scientific jobs applications is done to explain the scalability of the CWSA, in which the effectiveness of the proposed solution is verified. The simulation outputs prove that this scheduling policy improves the jobs process and performs better than the above mentioned alternative scheduling policies under typical deployment scenarios [12].

The process time of every task is in terms of general distribution in user module. The task scheduling module involves the process of taking a weighted sum of makespan and flow time as the objective function and utilizes an Ant Colony Optimization (ACO) and a Genetic Algorithm (GA) to solve the drawback of cloud task scheduling. Simulation results prove that the convergence speed and output performance of Genetic Algorithm-Chaos Ant Colony Optimization (GA-CACO) are optimal [13].

A Virtual Machine (VM) scheduling policy has a very important role in the life cycle of cloud computing. There are various steps which can be used for allocation and scheduling performances, which could affect the process and working of the cloud surroundings. In this paper [14], an analysis and execution of the existing scheduling methods are processed using CloudSim. Also an approach for process tuning of the light load cloud surrounding is done by arranging jobs and VMs based on heuristics function. This has shown process tuning on basis of parameters like implementation time, manufacturing and throughput of scheduled tasks.

In [15] author was the first person to propose the Opposition Based Learning. The optimization methods' first step is to initialize the population to improve the optimization solution. If the criterion is satisfied, the search terminates and haphazard guesses are implemented in the absence of prior information. Utilization of the opposite random numbers is the main principle of OBL. Thus the opposite random number might provide another chance for finding the solution.

In [16] author was used new hybrid techniques called Opposition Learning-Based Grey Wolf Optimizer Algorithm. Main contribution of this paper is to minimize the makespan and cost in cloud computing environment.

The above described survey did not provide near optimum result when performance and cost are considered together. In this proposed method we utilize makespan and cost among VMs as performance metrics to optimize task and resource, using an Oppositional Cuckoo Search Algorithm (OCSA) algorithm based on the proposed models in cloud. The proposed scheduling hybridizes two algorithms namely cuckoo search algorithm (CSA) and oppositional based learning (OBL). This new hybrid algorithm optimizes the task and resources more efficiently. The organization of this paper is as follows: Section 2 presents proposed scheduling

using OCSA algorithm. Section 3present the Result and discussion part. The conclusion part is given in section 4.

2. Problem definition with solution framework

The main objective of proposed methodology is to schedule the task with minimum makespan and cost. In this paper, the scheduling is done parallel i.e. all the tasks are processed simultaneously. Scheduling performs an important role in directing tasks within the cloud. Scheduling process first analyzes, how much of resources are needed to complete the task and which task should be allocated to which computing component.

Basically, a large application can be split into smaller sub-tasks prior to parallel processing. The total gain of the computation can potentially be higher by breaking down a computation into smaller jobs and executing the jobs on more than one processor. Scheduling all the jobs on a given number of processors which is already there in order to increase the gain without violating precedence constraints is the ultimate goal of a task scheduling algorithm. It is very challenging task for the task scheduling system. Therefore our optimization approach based scheduling has been proposed to overcome the difficulty in the present scheduling approaches to minimize cost and execution time.

In this, each task which is already submitted consist of a number of embarrassingly simultaneous and autonomous jobs. Each job needs to be executed in a single VM instance type. Consider that $PM = \{PM_1, PM_2, ..., PM_n\}$ is a set of cloud physical Machine. $VM_i = \{VM_1, VM_2, ..., VM_I\}$ is a set of virtual machines (VM) types and $T = \{T_1, T_2, ..., T_m\}$ is a set of task.

$$Objectivefunction = \sum_{i=1}^{m} T_i (\alpha \cdot cost + \beta \cdot Executiontimes) \quad (1)$$

Cost: The cost of a task is the cost required to create an optimal scheduling based on the number of virtual machine movements divided by the total number of virtual machines on a particular physical machine. The cost function is expressed by the following equation:

$$C_i = \frac{1}{PM} \sum_{i=1}^{m} \left(\frac{No.of\ movement\ in\ VM}{Total\ VMs} \right) \quad (2)$$

Table 1. Notation used in OCSA algorithm

Notations	Description
PM_i	Physical machine i $1 \le i \le n$
VM_i	Virtual machine i, $1 \le i \le I$
T_i	Task i $1 \le i \le m$
C_i	Cost
E_i	Execution Time
α, β	Control parameters
CPU_i	Number of CPU
TL	Task Length
PC	Processor Capacity

Where C_i denotes cost, PM denotes Physical machine.

Execution time: The execution time of a complete task in the task scheduling algorithm depends on the task Length and processing capacity. This function is expressed in the following equation:

$$E_i = TL/PC \quad (3)$$

Where E_i denotes Execution Time, *TL* represents Task Length, *PC* denotes Processor Capacity

2.1 Proposed OCSA based scheduling approach

The goal of this research is to schedule the task based on the OCSA algorithm. Our hybrid optimization algorithm is composed of the cuckoo search algorithm and Opposition based learning algorithm.

Step 1: Solution encoding

$$S_i = \begin{Bmatrix} t_1 cpu_1 & t_2 cpu_2 \cdots\cdots & t_n cpu_n \\ t_2 cpu_5 & t_1 cpu_3 \cdots\cdots & t_n cpu_n \\ t_3 cpu_1 & t_1 cpu_4 \cdots\cdots & t_n cpu_n \end{Bmatrix} \quad (4)$$

Step 2: Opposition based learning solutions

The opposite solution op ($z_1^*, z_2^*, z_3^* \ldots\ldots, z_n^*$).

$$Z_{ij} = a_i + b_i - z_i \quad (5)$$

i€1,2,3…n.

Step 3: Fitness calculation

The fitness function is utilized to assess every task in view of the cost, makespan and resource. In this paper, the minimization capacity is taken as the fitness. For minimization issues, the fitness evolution is performed by assessing the best and

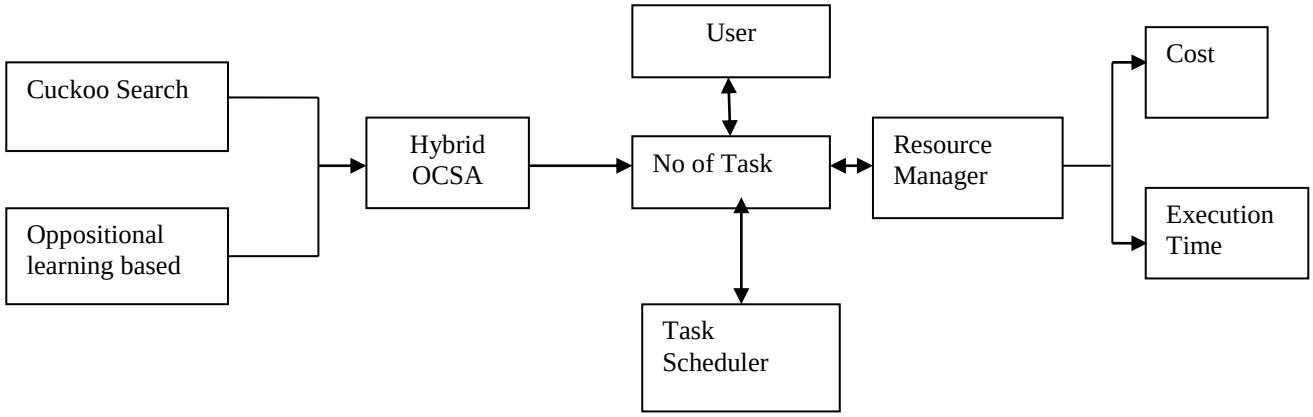

Figure.1 Architecture diagram of task scheduling

most exceedingly awful wellness for all specialists at a number of iteration.

$$FF_i = \sum_{i=1}^{m} T_i \, (\alpha \cdot cost + \beta \cdot Execution\ time) \quad (6)$$

Step 4: Update based on cuckoo search algorithm

The calculated fitness is then updated based on cuckoo search, which is shown below:

$$S_i(k+1) = x_i(k) + \alpha \oplus Levy \quad (\lambda) \quad (7)$$

Step 5: Stop criteria when it is satisfied

2.2 Proposed flow chart of OCSA

The step by step decision process of proposed OCSA algorithm is illustrated in Fig. 2. The overall architecture of proposed task scheduling is given in Fig. 1. User specifies the task and hands it over to the task scheduler. Task scheduler schedules the task based on fitness function of each task. Resource manager monitors the virtual machine usage. Hybrid OCSA algorithm is effectively used for reducing the cost and execution time.

2.3 Proposed algorithm of OCSA

Input:

Number of task, Number of Host machines, Number of virtual machine.

Output:

Minimize cost and makespan.

OCSA Algorithm

Step 1: Randomize Initialization of solution as population and generate opposite learning based algorithm.

Step 2: Evaluate the fitness function of the task till the optimal solution is obtained.

Step 3: Fitness function of resources is randomly selected.

Step 4: If the fitness function of the task is greater than the resources select another random resources.

Step 5: The worst resources are eliminated and the best solution is retained.

Step 6: Rank the solution to find the current best and stop.

3. Result and discussion

This section presents the output obtained from the proposed OCSA scheduling based task scheduling method. Here the input task is 100 to 1000. In our experiment 100 VMs and 200 VMs were considered. We have implemented our proposed task scheduling, using Java (jdk 1.6) with Cloudsim tool. A series of experiments were conducted on a PC with Windows 7 OS at 2 GHz dual core with 4 GB main memory, running a 64-bit version of Windows 2007. The results of proposed Hybrid OCSA are compared with PSO, GA and with the result that is obtained in [17], where the author had used an Improved Differential Evolution Algorithm (IDEA). The results obtained shows that our proposed technique produces better performance and cost when compared to other techniques.

3.1 Comparison of makespan

One of the main aspects of proposed system is to reduce the execution time of the task. We have compared different algorithms and their performance with varied makespan values from the simulation, the algorithm considered here are OCSA, PSO, GA and IDEA. Fig. 3 compares the performance of the tasks execution time using 100 VMs. For the given tasks of 250, the values obtained are 180.35, 188.3, 194.2 and 190.78 for OCSA, PSO, GA and IDEA respectively. When the task is increased to 500, the values obtained are 291.2, 299,

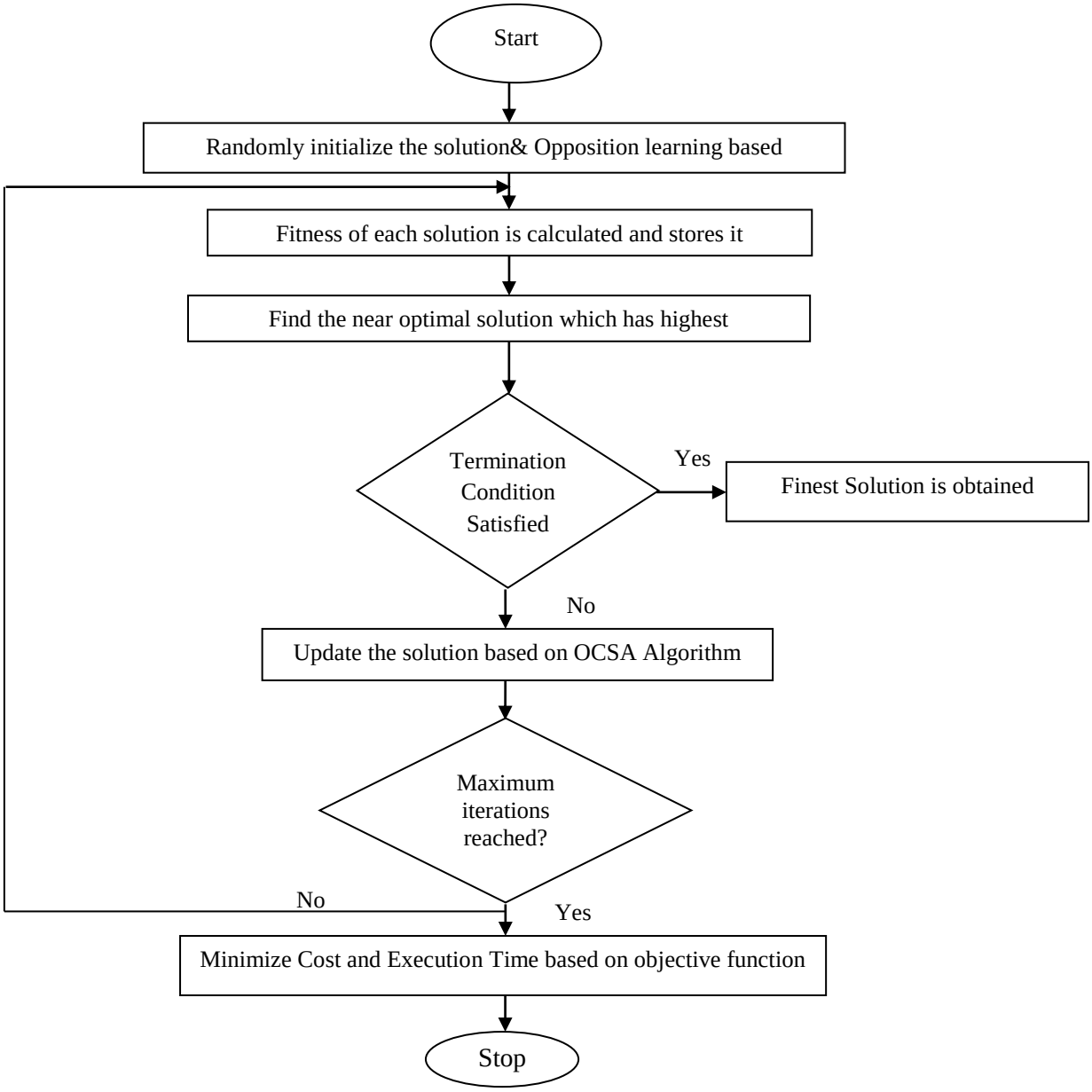

Figure.2 Flow Chart of proposed algorithm

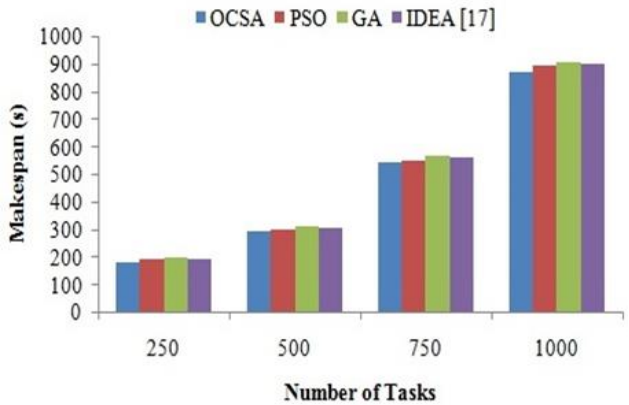

Figure. 3 Makespan of 100 VMs

312.4 and 305.23 for OCSA, PSO, GA and IDEA respectively. For 750 tasks, the values obtained are 539.4, 548, 566 and 557.6 for OCSA, PSO, GA and IDEA respectively. For 1000 tasks, the values obtained are 870.3, 891.1, 907.6 and 900.43 for OCSA, PSO, GA and IDEA respectively.

From Fig. 4 we infer that for the task 250, the corresponding makespan values of OCSA, PSO, GA and IDEA are 96.34, 101, 108.5 and 104.12 respectively. These results show the performance of OCSA is better considering double the number of VMs 200. When the task is increased to 500, the corresponding makespan values of OCSA, PSO, GA

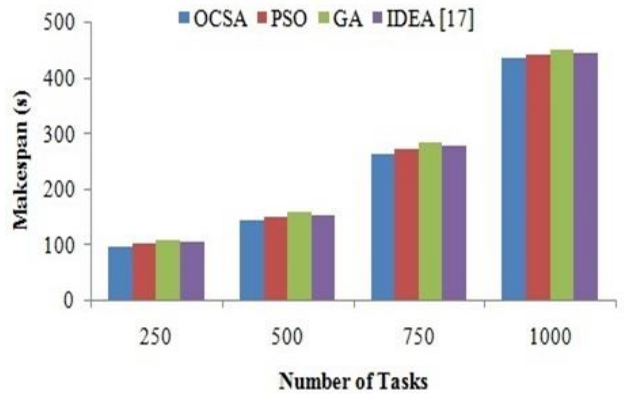

Figure.4 Makespan of 200 VMs

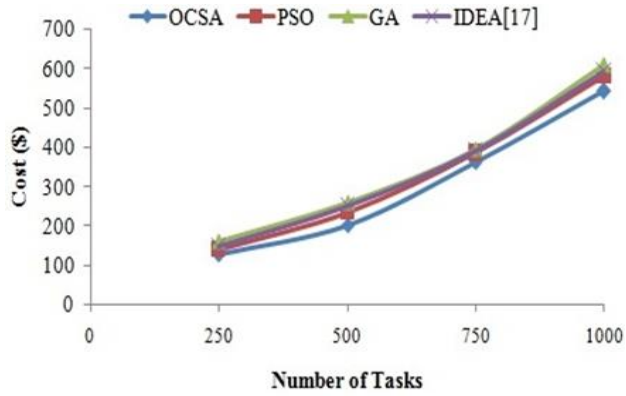

Figure. 5 cost of 100 VMs

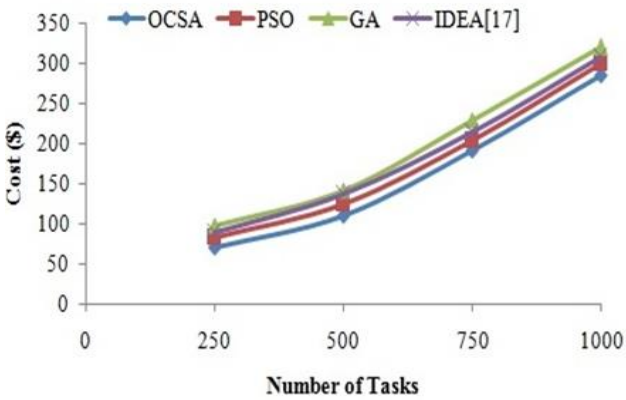

Figure. 6 cost of 200 VMs

and IDEA are 141.5, 148.5, 156.7 and 151.67 respectively. For 750 tasks, the corresponding makespan values of OCSA, PSO, GA and IDEA are 262.5, 271.5, 283.4 and 277.8 respectively. For 1000 tasks, the corresponding makespan values of OCSA, PSO, GA and IDEA are 435,439.5, 449.5 and 443.1 respectively. Thus the results obtained using OCSA make it optimal for usage due to the significant difference it provides when compared to rest. Also this difference increases steadily as makespan increases to 250, 500, 750 and 1000 indicating that OCSA yields better performance.

3.2 Comparison of cost

In Fig. 5 the performance metric is computed for analyzing the maximum cost of the task per schedule. The maximum cost of the task for 250,500, 750 and 1000 of the given four algorithms are calculated. For the first one done with 100VMs, the cost of 250 tasks is 126.73, 138.7, 160 and 145.53 for OCSA, PSO, GA and IDEA respectively, in which the OCSA is ahead of the rest. In the second one, the cost of 500 tasks are 202.4, 233, 260 and 249.7 for OCSA, PSO, GA and IDEA respectively in which the OCSA is ahead of the rest. In third one the cost of 750 tasks are 360.45, 384.4, 395 and 388.3 for OCSA, PSO, GA and IDEA respectively, in which the OCSA is ahead of the rest. In fourth one the cost of 1000tasks are 540.3, 580, 610 and 592.1 for OCSA, PSO, GA and IDEA respectively, in which the OCSA is ahead of the rest.

Fig. 6 presents the last comparison for cost using 200 VMs that is carried using the same set of algorithms.

It could be inferred from Figs. 5 and 6 in which our OCSA approach shows an overall improvement with respect to cost and time. For 250 tasks the cost of OCSA, PSO, GA and IDEA are 70.2, 83.4, 97.8 and 89.56 respectively.

When jobs are 500 the cost of OCSA, PSO, GA and IDEA are 110.3, 124.8, 141.4 and 136.9 respectively. For 750 tasks the cost of OCSA, PSO, GA and IDEA are 191.34, 203.8, 228.8 and 214.32 respectively. For 1000 tasks the cost of OCSA, PSO, GA and IDEA are 285.4, 299, 319.4 and 307.32 respectively.

This clearly shows that the performance of OCSA increases even further when the numbers of tasks are increased to 500, 750 and 1000. The cost of OCSA is much lesser when compared to PSO and IDEA. Also there is a drastic difference in cost when OSCA is compared to GA, thus proving the cost efficiency of OCSA.

From Figs. 3 to 6, we conclude that OSCA provides enhanced performance and is cost efficient when compared to the rest of the algorithms.

4. Conclusion

In this paper, hybridization of cuckoo search and opposition learning algorithm is done to propose oppositional cuckoo search algorithm (OCSA). The main objective of the scheduling technique is to assign users task and minimize cost and makespan of the system. The multi-objective optimization approach is used to improve the scheduling performance compared to single function. The experimental outputs were taken based on 100 to

1000 tasks and 100 VM and 200 VM. By performing hybridization of cuckoo search and oppositional based learning, a highly efficient solution for the scheduling mechanism can be achieved. The results produced clearly show that the OCSA out performs IDEA, GA and PSO in terms of performance and cost. In future, more Quality of service (QoS) parameters can be integrated with our OSCA approach to extend it to support real time operations.

References

[1] K. Pradeep and T. P. Jacob, "Comparative analysis of scheduling and load balancing algorithms in cloud environment", In: *Proc. of International Conf. on Control, Instrumentation, Communication and Computational Technologies*, pp. 526-531, 2016.

[2] R. Raju, J. Amudhavel, M. Pavithra, S. Anuja, and B. Abinaya, "A heuristic fault tolerant Map Reduce framework for minimizing makespan in Hybrid Cloud Environment", In: *Proc. of International Conf. on Green Computing Communication and Electrical Engineering*, pp. 1-4 , 2014.

[3] R. Raju, R. Babukarthik, P. Chandramohan, P. Dhavachelvan, and T. Vengattaraman, "Minimizing the makespan using Hybrid algorithm for cloud computing", In: *Proc. of International Conf. on Advance Computing Conference*, pp. 957-962, 2013.

[4] X. Yuan, Y. Li, Y. Wang, and K. Sun, "Scheduling cloud platform managed live-migration operations to minimize the makespan", In: *Proc. of IFIP International Conf. on Network and Parallel Computing*, pp. 595-599, 2014.

[5] A. Khaliliand and M. Seyed, "Makespan improvement of PSO-based dynamic scheduling in cloud environment", In: *Proc. of Iranian Conf. on Electrical Engineering*, pp. 613-618, 2015.

[6] M. Shojafar, M. Kardgar, A. Hosseinabadi, S. Shamshirb, and A. Abraham, "TETS: A Genetic-Based Scheduler in Cloud Computing to Decrease Energy and Makespan", In: *Proc. of International Conf. on Hybrid Intelligent Systems*, pp.103-115, 2016.

[7] H. Sachdeva, K. Sakshi, and V. Amandeep, "An Enhanced Strategy to Minimize Makespan in Cloud Environment to Accelerate the Performance", In*: Proc. of International Conference on ICT for Sustainable Development*, pp. 179-191, 2016.

[8] S. Usman, K. Bilal, N. Ghani, S. Khan, and L. Yang, "Thermal-aware, power efficient, and makespan realized Pareto front for cloud scheduler", In*: Proc. of International conf. on Local Computer Networks*, pp. 769-775, 2015.

[9] K. Garala, and H. Dobariya, "Effective selection of node for cloud environment using makespan", In*: Proc. of International conf. on Communication Networks*, pp.138-141, 2015.

[10] N.Sasikaladevi, "Minimum Makespan Task Scheduling Algorithm in Cloud Computing", *International Journal of Grid and Distributed Computing*, Vol. 9, No. 11, pp.61-70, 2016.

[11] K. Panda and K. Jana, "SLA-based task scheduling algorithms for heterogeneous multi-cloud environment", *The Journal of Supercomputing*, Vol. 73, No.6, pp.2730-2762, 2017.

[12] P. Rimal and M. Maier, "Workflow Scheduling in Multi-Tenant Cloud Computing Environments", *IEEE Transactions on Parallel and Distributed Systems*, Vol.28, No.1, pp.290-304, 2017.

[13] H. Cui, X. Li, T. Yu, H. Zhang, Y. Fang, and Z. Xia, "Cloud Service Scheduling Algorithm Research and Optimization", *Security and Communication Networks*, Vol. 2017, 2017.

[14] G. Lal, T. Goel, V. Tanwar, and R. Tiwari, "Performance Tuning Approach for Cloud Environment", *The International Symposium on Intelligent Systems Technologies and Applications*, pp. 317-326, 2016.

[15] H. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence", In: *Proc. of International Conf. on Computational Intelligence for Modelling Control and Automation*, Vol. 1, pp. 695–701, 2005.

[16] N. Gobalakrishnan and C. Arun, "Opposition Learning-Based Grey Wolf Optimizer Algorithm for Parallel Machine Scheduling in Cloud Environment", *International Journal of Intelligent Engineering and Systems*, Vol.10, No.1, pp. 186-195, 2017.

[17] D. Zou, H. Liu, L. Gao, and S. Li, "An improved differential evolution algorithm for the task assignment problem", *Engineering Applications of Artificial Intelligence*, Vol.24, No.4, pp. 616-624, 2011.