Cloudlet Scheduling using Merged CSO algorithm

Swati Gupta
Computer Science and Engineering
IIT(BHU)
Varanasi,India
swatigupta.rs.cse17@itbhu.ac.in

Ravi Shankar Singh
Computer Science and Engineering
IIT(BHU)
Varanasi,India
ravi.cse@iitbhu.ac.in

Aniket Anand
Ceramic Engineering
IIT(BHU)
Varanasi, India
aniket.anand.cer16@iitbhu.ac.in

***Abstract*— Cloud computing is used to provide information technology services to users over the internet. Scheduling is a very important aspect of cloud computing to manage resources and user demands. A numerous number of users are requesting the Service Providers for various requests which needs to be scheduled for an efficient optimization of various constraints. There exist a number of algorithms for Cloud Task Scheduling based on Swarm Intelligence. Two of them are Particle Swarm Optimization (PSO) algorithm and Cat Swarm Optimization (CSO) algorithm. Both the algorithms have various merits and demerits. In this paper, a merged CSO(MCSO) is presented with the aim to combine the merits of CSO and PSO to obtain better results. The basic purpose of developing this algorithm is to minimise the execution cost and time for running the algorithm to reach that optimised cost mapping. The algorithm steps are described, and it is implemented in CloudSim simulator and much-improved results are obtained. The algorithm performs comparatively well as compared to both PSO and CSO in terms of execution time and convergence.**

Keywords— ***swarm intelligence; Cloudlet scheduling; execution cost; makespan; PSO; CSO; task scheduling; cloud computing;***

I. INTRODUCTION

Cloud computing is a service that enables users to access any amount of resources and other services anytime, anywhere over the internet. It is on-demand computing resources. Cloud computing [1] includes both the application which the users use over a cloud and the data centers which provide software and different services. Thus, in cloud computing, the services are sold which is known as utility computing. Since the meaning of cloud computing means providing the service, SaaS (Software as a Service) is a basic service of cloud computing. Similarly, when this service is available for only a specified group of people it is called a private cloud. In recent days it has been widely used due to its various features. One of its features includes on-demand service which means the user must pay only for the time when the service is used, unlike past where to run applications or programs people required to download software on a physical computer. This makes service economical and keeps the load balanced on service providers because different users have different requirement of service and different amount of usage.

There are various services which are offered by cloud-computing providers such as a) PaaS: Platform as a Service, b) IaaS: Infrastructure as a Service and c) SaaS: Software as a Service [2]. The hardware layer and infrastructure layer collectively provides IaaS. It forms the resources for technologies like VMware. The platform layer provides PaaS and technologies like Microsoft Azure, Google AppEngine, Amazon SimpleDB/S3 use this layer to develop applications. On the application layer, SaaS is provided. Various applications run on this layer because of advantages like lower operating cost, automatic scaling, etc. The goal of cloud computing is to allow users to take benefit of different technologies and on pay on demand basis. The main technology which makes this possible is virtualization. A Virtual Machine (VM) is a software program that runs the operating system and other applications. In computing, efficient scheduling helps to minimize money cost and the execution time for whole application and helps in load balancing over the VMs in the cloud environment. Scheduling is most important for improving the efficiency of all cloud-based services. Various tasks are assigned to most appropriate VM processor and the order of their execution is very important. Therefore, a proper scheduling is required to manage these resources and maintain system performance. Task scheduling plays a vital part in cloud computing systems where the aim is to reduce turnaround time and improve resource utilization by allocating a task to best suitable re-source for execution. Efficient task scheduling is required for reducing the latency, the most important quality of service. There can be two types of scheduling methods- scheduling the VM and scheduling host. In this paper, an efficient algorithm for Cloudlet scheduling based on Swarm Intelligence (SI) is presented which is inspired by the interaction of animals with their environment. This paper presents an algorithm that can be used for efficient task scheduling. The proposed algorithm reduces the overall makespan and the execution cost of the schedule based on SI is presented which is inspired by the interaction of animals with their environment. This paper presents an algorithm that can be used for efficient task scheduling. The proposed algorithm reduces the overall running time and the execution cost of the schedule.

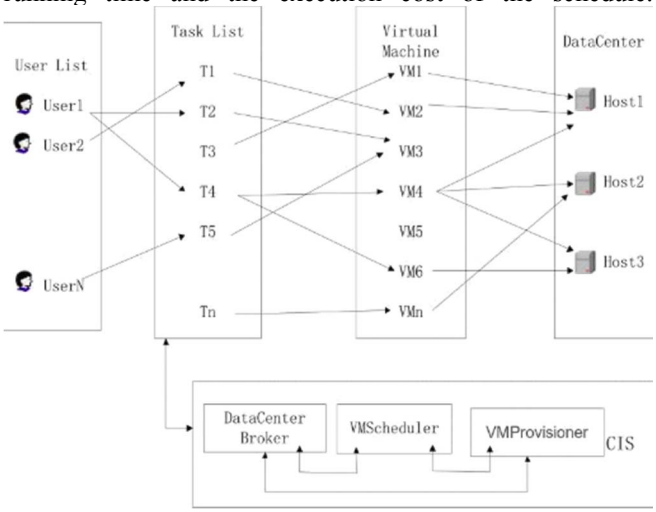

Figure 1: Scheduling in Cloud Computing

II. Related Work

CSO is an algorithm of swarm intelligence [3]. It was developed by Chu and Tsai in 2007. In CSO, initially, the population of cats is randomly distributed into seeking mode and tracing mode. There are more cats in seeking mode than tracing mode due to their natural behaviour to spend more time resting, which is evident from the small value of mixing ratio (MR). After the distribution of all the cats into the above modes, positions are updated, and fitness function is calculated again. The cat with the best solution is maintained in memory to keep a record of them.

For the cats which are in seeking mode, based on self-position consideration (SPC) value, it is determined whether a cat is eligible to move in the next step. If the SPC value of a cat is true, then it moves to a new position where its fitness values are again calculated. If the calculated fitness values are same for all the cats, then all these candidate cats move to their new position. Otherwise only those cats selected randomly using roulette wheel are moved to new positions. The cats which are in tracing mode are chasing their prey in M-dimensional decision space based on prey's position and velocity. Particle Swarm Optimization (PSO) is an optimisation algorithm to obtain optimised mapping in the cloud. PSO was developed by Kennedy and Eberhart [4]. This algorithm follows many iterations updating the position and velocity of all the particles in each iteration. The positions and velocities of each particle are initialized randomly. The movement of particles in PSO are based on their own best position until the current iteration and the best position among all the particles. Ahmad M. and others have proposed a Hybrid GA-PSO algorithm [5] combining both the algorithms to allocate tasks to resources efficiently in cloud computing environment. The GA-PSO algorithm reduces the execution time and total execution cost of the workflow task. It also works well in balancing the load on dependent tasks efficiently and the results converge to an optimal solution faster than other algorithms. If there are total n iterations, then for n/2 iterations GA algorithm is executed and its results are used as input for PSO which executes for other n/2 iterations. Yiqiu Fang and others proposed a task scheduling algorithm[6] whose main objective is to maintain load on various Virtual Machines and efficient resource utilization based on virtualization and dynamic demand of resources that is, flexibility feature of cloud computing. Of the two levels proposed, the first level which is from application layer to infrastructure layer specifies the demands for the task while the second level which is from infrastructure layer to platform layer finds the appropriate resources to be allocated for the task to be done. Xing Jia Wei, Wang Bei, and Li Jun proposed the Simulated Annealing Multi-Population Genetic Algorithm (SAMPGA) [7] which is a combination of simulated annealing algorithm (SA) and multi-population genetic algorithm (MPGA). SA avoids local optimum and improves the performance of global optimum while MGPA finds a better solution using min-max algorithm to improve search efficiency and helps to improve convergence speed. This algorithm also reduces execution time, total cost and manages efficient load balancing. In [8], task allocation has been done considering service reliability in grid computing environment based on social spider optimization technique. The authors also formulated formulae for transactions in the environment keeping in mind the resource availability. In [9], the different cloudlets in ADSF algorithm are as-signed according to the arrival time keeping in mind the deadline constraint of the tasks. They are allocated to VMs according to the mapping obtained by ADSF algorithm. In paper entitled," Balanced task allocation in the on-demand computing-based transaction processing system using social spider optimization" [10], the authors have also applied a swarm intelligence algorithm to schedule transaction-based processing system. In some another papers by D P Mahato and R S Singh [11], [12], they have developed an algorithm based on Ant Colony Optimisation and Honey Bee behaviour for maximising availability and load balancing of tasks respectively.

III. Proposed Algorithm

In cloud, a large number of users are requesting for a huge amount of services at a time, so scheduling of these request is a major concern which in turn not only beneficial for Cloud Service Providers (CSPs) but also for the users. There are various objectives for mapping these tasks to the processors, some of which are execution cost, deadline, makespan, energy based etc. We have worked to provide the mapping that has least execution cost and makespan. All the works have mostly worked on makespan and none has considered the actual running time to reach that optimised schedule.

The problem can be stated as:" To *find a mapping, where the total execution cost of the schedule is minimum considering the communication and execution cost. Also, the time taken to reach that mapping is to be minimised."*

The proposed algorithm is a merged approach of CSO-PSO algorithms. The purpose of choosing CSO algorithm is an experimental way to check whether the approach will provide with better results or not. CSO has two modes and the way in which these two modes merge in accordance with a user-defined proportion, CSO can present better performance.

CSO: is an optimization algorithm based on swarm intelligence (SI) used in cloud computing. SI is an artificial intelligence approach in which many self-organized individuals form a society and coordinate with each other. It is mainly inspired by the biological system such as a swarm of bees, colony of ants, and school of fish. The agents of swarm interact for a common goal: performance optimization and ability to withstand even difficult problems. The task is divided among all the agents of the swarm and thus they all interact indirectly with each other. CSO is a gradually developing method based on the interaction of a group of cats with their environment for solving optimization problems. From the behaviour of cats, according to various surveys it has been observed that various cats spend most of the time in a sleeping or a resting position, but they have very high alertness and curiosity about their environment while resting. This helps them in the excessive utilization of resources with minimum energy usage. At a given point of time, some cats are moving around chasing their prey while the others are resting keeping an eye on surroundings. Therefore, the population of cats is divided into two subgroups - seeking mode and tracing mode are named. A large population of cats is in

seeking mode with only a small population are in tracing mode. This is evident from the value of mixing ratio (MR) which is small. The population of cats is sorted into these two modes randomly.

PSO: is an optimization inspired by the how a swarm of fish or birds would behave while searching for food. Since proposed in 1995 by Kennedy and Eberhart many variations of this algorithm have been published. It is a clustering algorithm having position and velocity as the major features. The basic principle of PSO is self-organization of particles to solve optimization problem through cooperation.

The following are the steps to be followed in the algorithm:

Algorithm 1 Merged Cat Swarm Optimization (MCSO)
1. Initially assign the processor to tasks according to the round-robin algorithm.
2. Initially, assign pbest of each particle as its current allocation.
3. Calculate the execution cost (using fitness function) of the current mapping.
4. If the execution cost is less than the previous one, then update pbest to current execution cost.
5. Update gbest as the best particle from pbest i.e. having minimum execution cost.
6. Update the velocity vector and then the position vector according to Eq. (1) and (2) respectively.
7. If iteration=2, go to step 8 else go to step 3.
8. CSO: Initialise the population with mapping obtained in step 5.
9. Randomly assign MR values to set cats into tracing and seeking mode according to MR i.e. Mixture ratio of cats.
10. Calculate the fitness value of population and memorize the best fitness to the best cat.
11. Update position of cats according to the mode it is in.
12. Repeat steps 8 to 11 until stopping criteria are met.

$$v_i^{k+1} = \omega v_i^{\,k} + c \times rand \times (pbest_i - x_i^{\,k}) + c \times rand \times (gbest - x_i^{\,k}) \quad (1)$$

$$x_i^{\,k+1} = x_i^{\,k} + v_i^{k+1} \quad (2)$$

where,

v_i^{k+1}	particle's velocity at k+1th iteration
ω	inertia coefficient
v_i^{k}	particle's velocity at the kth iteration
c	acceleration coefficient
$rand$	random value ($0 \le rand \le 1$)
$pbest_i$	best position of ith particle
x_i^{k}	particle's position at iteration k
$gbest$	swarm's best position
x_i^{k+1}	particle's position at k+1th iteration

A. Seeking Mode

All the cats in this mode are in resting mode and are just watching their surroundings. The cats are keeping a track of all the situations and available opportunities it can make in an environment. On sensing danger or based on the position of its prey decides its next move.

There are mainly four parameters in this mode namely:

CDC: counts of dimension to change

SMP: Seeking memory pool

SRD: seeking range of the selected dimension

SPC: self-position consideration

SPC value is a Boolean which determines whether the cat is a current candidate for movement for each iteration. Its value can be true if a particular is a candidate for movement otherwise false. SRD is taken to be 2. SMP values are different for each cat. If the SPC value is true, then we increment SMP value by 1 for that cat. If SPC is false then, we don't change the SMP value for that cat. CDC value is taken to be 1 as we changed only 1 dimension(X).

The following are the algorithm steps in seeking mode:

Algorithm 2 Seeking Mode
1. Built j duplicates of the ith cat as prescribed by SMP.
2. Randomly update CDC.
3. Calculate execution cost i.e. fitness function of every cat.
4. Record the best mapping having minimum fitness function (cost).

B. Tracing Mode

This is another mode of the algorithm. The cat is in motion and its movement is determined by position and velocity of its prey.

$$v_i^{\,t+1} = \omega \times v_i^{\,t} + r \times c \times (x_{best} x_i^{\,t}) \quad (3)$$

$$x_i^{\,t+1} = x_i^{\,t} + v_i^{\,t+1} \quad (4)$$

where,

ω	inertia coefficient
r	random value ($0 \le r \le 1$)
c	acceleration coefficient
v_i^t	previous velocity
x_{best}	best location

The following are the steps of tracing mode of the algorithm:

Algorithm 3 Tracing Mode Algorithm
1) Find velocity for the ith cat using eq 3.
2) Update position vector using eq 4.
3) Calculate the execution cost i.e. fitness value of the mappings obtained.
4) Update the solution with best fitness value of the mapping obtained.

The main objective of merging these two algorithms is to reduce the overall makespan execution time and reduce total cost. Merged CSO algorithm combines the strength of both the algorithms for cloudlet scheduling i.e. Cloud Task Scheduling in CloudSim simulation tool.

There are various merits and demerits of both PSO and CSO. But the proposed merged algorithm is such that it takes the advantage of merits of both the algorithms. The following are the merits [13] of PSO:

- Equations in PSO are very simple leading to easy calculation and implementation.
- PSO makes very few assumptions about the optimization problem and thus can be applied to a very large number of particles.
- As the new position of particles is updated on the basis of the best solution among the swarm, each particle and hence the entire swarm would be guided toward the best solution.

- Since PSO is based on SI, it can be applied to both engineering and scientific research applications.

But there are also some demerits [14] of PSO:

- PSO can be a victim of local optima when there is high dimensional problem.
- It fails on the problem when the particles are largely scattered and are to be optimized.

To overcome these demerits CSO is executed for another half of iterations. Following are the advantages of CSO:

- CSO shows much better convergence time than other SI algorithms like PSO or PSO with a weighting factor (PSO-WF).
- CSO also outperforms other algorithms in finding global best solutions.
- CSO works well when there are large numbers of agents scattered.

While following are the disadvantages of CSO:

- Because of the complexity, pure CSO cannot achieve an accurate solution.

CSO requires many iterations to find optimal solutions. Since PSO is comparatively independent of the initial population as compared to other evolutionary methods, the convergence is very favourable thus improving the efficiency of algorithm by lowering the time. Also, in PSO, the impact of parameters to the solution is less effective as compared to other algorithms. Therefore, we implement PSO initially and then feed its results as an input for CSO. Since the initial population is not allotted randomly, but by round-robin method and the optimized results of PSO are fed to CSO, this will help the particles to move in a specified direction rather than moving randomly. Thus, this will help in faster convergence toward a global minimum. Real life challenges encountered during implementation can be stated as the simulation environment sometimes took more time then supposed and thus the algorithm needs to be rechecked for more optimised time. Also, when the number of tasks will be increased there will be a problem of more running time which needs Graphic Processing Units (GPUs) for their implementation to reduce the time taken for proper scheduling of requests in least time.

C. Fitness Function

A fitness function is a criterion which is optimized to get the desired result. We have used the same fitness functions for all the algorithms so that they can be compared on the same basis. The algorithm assigns the processors to the tasks. The fitness function calculates the execution cost of the mapping. The algorithm minimizes the execution cost (fitness function) by changing the mapping iteratively. In every subsequent iteration, a mapping of lesser execution cost is produced. The fitness function uses parameters which are communication cost, execution cost of every task on each processor and file size of each task to be communicated. The fitness function as defined in is used. Let M be a mapping assigned in an iteration. P is the set of processors and T is the set of tasks. Cexe(Mp) is the execution cost of running the tasks on the allotted processors according to M. u_{tp} represents the running cost of task t on processor p. C_{ac}(Mp) is the access cost of communicating data between the processors on which the task t1 and t2 are assigned. $comm_{M(t1),\ M(t2)}$is the communication cost between the processors on which they are mapped and $data_{t1t2}$ is the file size to be communicated. C_{total}(Mp) is the sum of access cost and execution cost. The maximum cost of each mapping is minimized which is the fitness function. The equations are as follows:

$$C_{exe}(M_p) = \sum_{t} u_{t_p} \qquad \forall M(t) = p \qquad (5)$$

$$C_{ac}(M_p) = \sum_{t1 \in T} \sum_{t2 \in T} comm_{M(t1),\ M(t2)} data_{t1t2} \qquad (6)$$

$$\forall M(t1) = p \ M(t2) \neq p$$

$$C_{total}(M_p) = C_{exe}(M_p) + C_{ac}(M_p) \qquad (7)$$

$$FitnessFunction: Minimize(Cost(M)) + Minimise\ T(run) \qquad (8)$$

D. Sample Data Used

The cost which contributes to total cost while scheduling includes execution of cost of the task in processors (denoted by matrix TP), communication cost between processors (denoted by matrix PP) and file size of the task to be communicated (denoted by matrix data) as shown:

TABLE I. PROCESSOR-PROCESSOR COST MATRIX

Processor	*Processor 1*	*Processor 2*	*Processor 3*
Processor 1	0.00	0.17	0.21
Processor 2	0.17	0.00	0.22
Processor 3	0.21	0.22	0.00

TABLE II. TASK-PROCESSOR COST MATRIX

Tasks	Processors		
	Processor 1	*Processor 2*	*Processor 2*
Task 1	1.23	1.12	1.25
Task 2	1.27	0.17	1.28
Task 3	0.13	1.11	2.11
Task 4	1.26	1.12	0.14
Task 5	1.89	1.14	1.22
Task 6	1.27	0.47	1.28
Task 7	0.13	1.11	1.11
Task 8	1.26	1.62	0.14
Task 9	1.13	1.12	1.25
Task 10	1.89	1.14	0.42

TABLE III. FILE SIZE OF TASK MATRIX

Task	Input Data	Output Data
Task 1	30	30
Task 2	10	10
Task 3	10	10
Task 4	10	10
Task 5	30	60
Task 6	30	50
Task 7	30	20

Task 8	70	60
Task 9	30	40
Task 10	30	60

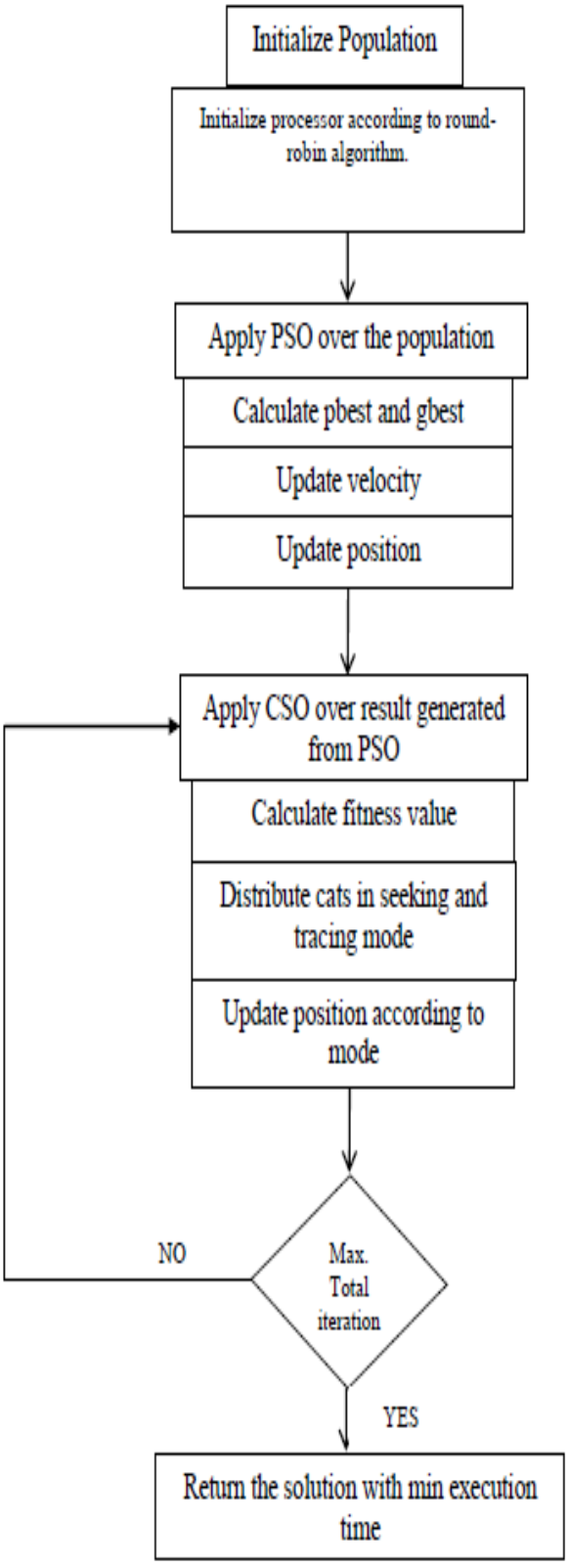

Figure 2 Flowchart of Algorithm

IV. RESULTS

The algorithm stated above is implemented in the cloud environment using CloudSim simulator where cloudlets are the tasks and a virtual machine is a processor on which the task is to be scheduled.

A. *Execution Cost*

Execution cost is the total cost for scheduling tasks on the corresponding processor calculated using a specified formula. It depends on the cost of interaction of processors with each other, cost of interaction of task with each processor and cost due to file of size which is to be communicated. It is obtained by calculating the value of fitness function using the value of TP, PP and data matrices. The results are plotted in Fig 3 shows the execution cost of various algorithms for 100 iterations. The algorithms are run 10 times to eradicate any randomisation error as much as possible and the graph of all the runs is plotted. It can be seen from the graph that MCSO gives lesser total cost than CSO and comparable cost with PSO. The population is initialized by round-robin algorithm and is passed to PSO algorithm. The solution which is obtained from PSO is fed as input for CSO. The results which are obtained after n iterations of MCSO are much better than either PSO or CSO because MCSO is not initialized by random population in this case, as in the case of normal CSO.

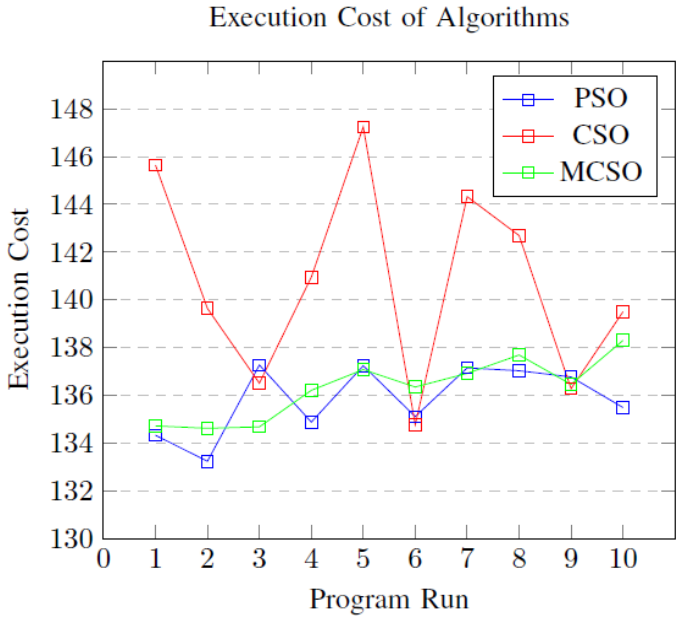

Figure 3 Execution Cost in different Runs

B. *Makespan*

Makespan in task scheduling is the total time taken for a set of tasks to completely execute. Figure 2 compares the makespan for different scheduling algorithms. The proposed algorithm gives us the least makespan among compared algorithms thus showing that it is much efficient than previous algorithms for task scheduling. It takes less time to converge and finds the best solution in a lesser number of iterations.

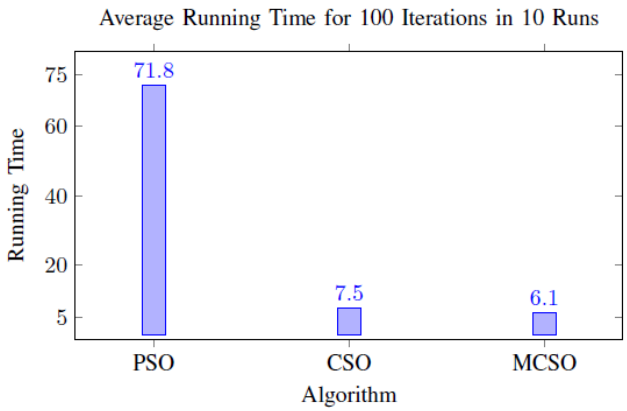

Figure 4 Average Running Time of Algorithms

V. CONCLUSION AND FUTURE SCOPE

In this paper, three task scheduling algorithms are implemented, and results are compared based on execution cost and makespan. The execution cost of Merged CSO algorithm iscomparable to PSO but smaller than CSO. The Merged CSO takes the least makespan and thus is most efficient because it takes the results of PSO as its input position and velocity. Moreover, the position for PSO is initialized by round-robin method instead of random initialization as in normal PSO or normal CSO. So, MCSO performs much better than normal CSO or PSO by giving somewhat optimal solution in least amount of time.In future work, improvements can be done to further optimize task scheduling algorithms to improve efficiency in scheduling. Optimization can be done to reduce the average execution cost along with other constraints too like deadline, energy, load balancing etc.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," Journal of Internet Services and Applications, vol. 1, no. 1, pp. 7–18, May 2010. [Online]. Available: https://doi.org/10.1007/s13174-010-0007-6

[3] M. Bahrami, O. Bozorg-Haddad, and X. Chu, "Cat swarm optimization (cso) algorithm," in Advanced Optimization by Nature-Inspired Algo-rithms. Springer, 2018, pp. 9–18.

[4] L. Guo, S. Zhao, S. Shen, and C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm," Journal of networks, vol. 7, no. 3, p. 547, 2012.

[5] M. Manasrah and H. Ba Ali, "Workflow scheduling using hybrid ga-pso algorithm in cloud computing," Wireless Communications and Mobile Computing, vol. 2018, 2018.

[6] Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in International Conference on Web Information Systems and Mining. Springer, 2010, pp. 271–277.

[7] X. J. Wei, W. Bei, and L. Jun, "Sampga task scheduling algorithm in cloud computing," in Control Conference (CCC), 2017 36th Chinese. IEEE, 2017, pp. 5633–5637.

[8] D. P. Mahato and R. S. Singh, "On maximizing reliability of grid transaction processing system considering balanced task allocation using social spider optimization," Swarm and Evolutionary Computation, vol. 38, pp. 202–217, 2018.

[9] R. Kumar and S. Gupta, "Arrival based deadline aware job scheduling algorithm in cloud," in 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC), Sept 2017, pp. 176–180.

[10] D. P. Mahato and R. S. Singh, "Balanced task allocation in the on-demand computing-based transaction processing system using social spider optimization," Concurrency and Computation: Practice and Ex-perience, vol. 29, no. 18, p. e4214, 2017.

[11] D. P. Mahato and R. S. Singh, "Maximizing availability for task schedul-ing in on-demand computing–based transaction processing system using ant colony optimization," Concurrency and Computation: Practice and Experience, vol. 30, no. 11, p. e4405, 2018.

[12] D. P. Mahato and R. S. Singh, "Load balanced transaction scheduling using honey bee optimization considering performability in on-demand computing system," Concurrency and Computation: Practice and Expe-rience, vol. 29, no. 21, p. e4253, 2017.

[13] Q. Bai, "Analysis of particle swarm optimization algorithm," Computer and information science, vol. 3, no. 1, p. 180, 2010.

[14] M. Li, W. Du, and F. Nian, "An adaptive particle swarm optimization algorithm based on directed weighted complex network," Mathematical problems in engineering, vol.2014,2014