https://doi.org/10.48048/tis.2022.4204

Scheduling in IaaS Cloud Computing Environment using Sailfish Optimization Algorithm

Manoj Kumar* and Suman

Department of Computer Science and Engineering, DCR University of Science and Technology, Murthal, Sonepat 131039, India

(*Corresponding author's e-mail: manojsarasyiya@gmail.com)

Received: 21 December 2020, Revised: 31 May 2021, Accepted: 21 June 2021

Abstract

Due to the exceptional benefits of cloud computing, it has magnetized IT leaders and entrepreneurs at all levels. The cloud's popularity is attributed to various technologies like the Internet of Things (IoT), mobile computing, Fog, etc. Scheduling in cloud computing is still a challenging issue due to its NP-Hard nature. In recent years, many techniques have been proposed for optimal scheduling that can subsequently improve efficient Quality of Service (QoS). This paper has developed and analysed a novel Sailfish Optimization-based Scheduling Algorithm. SOSA is implemented on 2 data-sets; a real-world data-set from NASA workload, and a randomly generated data-set. SOSA exhibited 13.71 and 7.81 % average performance improvement in makespan compared to GA and PSO and 11.30 and 30.78 % average improvement in execution cost compared to GA and PSO.

Keywords: Cloud, Quality of service, Sailfish optimization, Scheduling

Introduction

Distributed computing and virtualization technology sordid the evolvement of cloud computing in 21st century. The pay-per-utilize model of cloud computing has increased the demand of cloud computing. The different types of services like infrastructure, platform, and software are provided by a cloud service provider (CSP) to their users as a demand [1]. The ability of virtualization over the internet provides scalable resources and software services to the customers [2].

The expansion of Internet of Things (IoT) [3], Fog Computing[4], Edge Computing , Mobile Cloud [5], along with cloud computing, increased the demand for cloud services. The Virtual Machines (VM) process the tasks of cloud users and CSP charges from users as per the pay-per-use model and service level agreement (SLA). Each user has its own criteria for pricing and deadline. There is an inverse relation between makespan and cost of execution. Customers always search for best service with minimum cost, and CSP tries to increase their profit by providing best services to their customers. This increases a trade-off between cost and makespan in cloud computing.

Scheduling of resources in cloud computing has played a vital role in minimization of trade-offs. It plays an important role in minimizing execution cost, makespan, efficient resource utilization, and energy consumption in the cloud [6,7]. It aligns all tasks to be executed on time with minimum cost and maximum resource utilization. Scheduling in cloud computing is based on 2 types of tasks. Firstly, scientific workflow scheduling [8], in which every task is executed by each virtual machine, and each task is dependent on the other. Secondly, independent task scheduling, in which every task is associated with an independent virtual machine and executed on it. There is no dependency on the order of execution of tasks.

Heuristic and meta-heuristic techniques have played a vital role in solving various engineering problems. They have also been applied in cloud computing for scheduling, load balancing, VM allocation and placement, migration, and VM consolidation. This paper presents a novel Sailfish Optimization Algorithm based scheduling technique named (SOSA) for scheduling of independent task to the corresponding virtual machines as SOSA provides better QoS to users in terms of cost of execution and makespan and better resource utilization.

Related work

Researchers and academicians to find the global optima in scheduling using heuristics and meta-heuristics techniques have carried out a lot of research. Due to limitations of trapping at local minima, researchers had shifted towards nature-inspired algorithms to get high convergence rate. These algorithms were also used to optimize various problems of cloud computing. In this section, few existing works are reviewed and analyzed.

Genetic Algorithm (GA) are stochastic techniques based on natural selection and genetics mechanism. They are widely used in scheduling and other domains of cloud computing. It starts with some initial solutions, where each solution is called chromosome. The crossover, mutation, and selections procedure in each iteration moves it towards better solution [9,10]. Swarm intelligence-based algorithms had also played a vital role in scheduling problems. The particle moves through search space and each particle represents a candidate solution. The performance of each particle is represented using its fitness function. The other 2 variable velocity helps in updating the position of each particle. The best particle gives final solution [11-13].

Aravind Rajagopalan *et al.* [14] proposed hybrid task scheduling algorithm using firefly and genetic algorithm to minimize the execution time. The powerful combination of firefly and evolutionary genetic algorithm had great convergence to their optimal solution. The implementation was carried out in CloudSim [15] simulator. Cost, resource utilization, and QoS factor can also be optimized. Wu Daqin *et al.* [16] enhanced the particle swarm optimization technique by avoiding its fall in local optimum. It effectively reduced the execution time for jobs but failed to maintain a balanced distribution of load to each virtual machine. Zhao *et al.* [17] presented a novel scheduling algorithm using an artificial algorithm and deep learning to minimize makespan and maintain load balancing in a cloud environment for heterogeneous cloud environments.

Sanaj *et al.* [18] developed a chaotic squirrel search algorithm for multi-objective scheduling in cloud environment. The algorithm is implemented on the CloudSim simulator to minimize cost, energy, makespan, and efficient resource utilization. In recent years, many researchers have combined features for various meta-heuristics techniques to increase efficiency of scheduler. Safari *el al.* [19] developed a hybrid algorithm by merging features of PSO and simulated annealing to schedule tasks in a cloud computing environment to minimize execution time and response time.

Gobalakrishnan *et al.* [20] improved the Grey Wolf Optimization algorithm to optimize processing cost and time to increase the Quality of Service(QoS). Further, it is improved by Xuan Chen [21] and added the capability of Whale Optimization to achieve better convergence speed and accuracy for better optimal solution. The technique exhibited better resource utilization on small- and large-scale tasks.

Xiujin Shi *et al.* [22] applied self-adaptive preferred Differential Evolution (DE) algorithm for scheduling to achieve load balancing, minimize makespan and stability for the system. Laith Abualigah *et al.* [23] proposed a novel multi objective algorithm by combining features of Ant Lion Optimization (ALO) with elite-based differential evolution that avoid it getting trapped in local search. The proposed algorithm was tested on synthetic and real-world traces dataset and found more efficient in terms of minimizing makespan and degree of imbalance in cloud environment.

To overcome the various limitation of existing techniques, a novel Sailfish optimization-based scheduling algorithm has been proposed to achieve better Quality of Service (QoS) in terms of makespan and cost.

Scheduling problem in cloud computing

Scheduling problem comprises of assignment of resources of cloud to customers in efficient manner. For some customers price is important while for others time is a big factor. There are many customers who want better performance irrespective of price. So, cloud service provider has to manage and utilize their resources in such a way that it can fulfill customer's requirement with profit. So, let us assume that, in cloud computing environment, there are *N* resources and *J* jobs submitted by customers at given time *t*. So, resource scheduling problem can be stated by Eq. (1) containing *R* physical resources, *S* virtual resources for *n* users *U*.

$$RS = \sum_{x=0}^{mn}(R_x + S_x \dots\dots\dots\dots + N_x) \times J_x \rightarrow \mathrm{U}_x^z \quad (1)$$

where mapping of *m* tasks with *n* available physical resources to the virtual resources is done for users so that fitness of z particular objective $F = (F_1, F_2, F_3 \dots . . F_z)$ is achieved. Suppose, there are p number of jobs $(J_1, J_2, J_3 \dots\dots\dots . . J_p)$. Cloud broker has to allocate different virtual machines $(V_1, V_2, V_3 \dots\dots\dots\dots . V_q)$

in order to complete their demand with minimum time and cost. The Expected Execution Time (EET) of all jobs to the virtual machines is presented using Eq. (2) as EET matrix where each element is recognized by $EET\ (J_p, V_q)$.

$$EET(J_pV_q) = \begin{bmatrix} J_1V_1 & J_1V_2 & J_1V_3 \dots\dots\dots J_1V_q \\ J_2V_1 & J_2V_2 & J_2V_3 \dots\dots\dots J_2V_q \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ J_pV_1 & J_pV_2 & J_pV_3 \dots\dots\dots J_pV_q \end{bmatrix} \quad (2)$$

If $C = (C_1, C_2, C_3 \dots\dots\dots\dots C_p)$ is cost set of each job then, Expected Cost (EC) of all jobs to be paid by all users to the service providers is presented using Eq. (3) as EC matrix where each cost of each job to each virtual machine is recognized by $EC(C_p, V_q)$.

$$EC(C_p, V_q) = \begin{bmatrix} C_1V_1 & C_1V_2 & C_1V_3 \dots\dots\dots C_1V_q \\ C_2V_1 & C_2V_2 & C_2V_3 \dots\dots\dots C_2V_q \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ C_pV_1 & C_pV_2 & C_pV_3 \dots\dots\dots C_pV_q \end{bmatrix} \quad (3)$$

The makespan of all the cloudlets is calculated by using Eq. (4)

$$\text{Makespan, } F(x) = \bigcup_{i=1}^{p} Ti\ \forall i \in N, i = 1, 2, 3 \dots\dots p. \quad (4)$$

where T_i is the completion time of each cloudlet.
The execution cost of all the cloudlets is calculated by using Eq. (5)

$$\text{Execution Cost, } H(x) = \sum_{i=1}^{p} Cq * Ti\ \forall\ \text{i} \in \text{N}, i = 1, 2, 3 \dots p. \quad (5)$$

Where C_q is cost of VM_q per unit time used and T_i is duration of using that resource. In this paper, our objective is to minimize makespan and cost, so our objective function is:

$$\text{Objective Function} = \boldsymbol{min(G(x))}\ where\ G(x) = F(x) + H(x) \quad (6)$$

To make a balance between cost and makespan function, we used 1 variable alpha α, so Eq. (6) becomes:

$$G(x) = \alpha \times F(x) + (1 - \alpha) \times H(x) \quad (7)$$

where $\alpha \in (0, 1)$.

Proposed sailfish optimization-based scheduling algorithm (SOSA)

Sailfish optimization algorithm is the powerful optimization algorithm developed by S. Shadravan *et al.* [24]. It used the attacking behavior of Sailfish that mimics sailfish group hunting. Secondly, it uses the alteration of attacks to break defense of grouping prey. Thirdly, prey movements can be updated in search space and hunter is allowed to catch to the appropriate prey to become fittest. Sailfish optimizer can be used in multi-dimensional search space. In our problem, we use 2-dimensional search space for attacking the prey.

Initialization of sailfish

Sailfish is a population-based meta-heuristic algorithm, so we assume that sailfish is the candidate solution and the problem variable is the sailfish's position. So, initially a random solution is generated using Eqs. (2) and (3) which can be presented in 2- dimensional array in Eq. (8).

$$SF_{position} = \begin{bmatrix} SF_1, 1 & SF_1, 2 & SF_1, 3 \dots\dots\dots SF_1, d \\ SF_2, 1 & SF_2, 2 & SF_2, 3 \dots\dots\dots SF_2, d \\ SF_3, 1 & SF_3, 2 & SF_3, 3 \dots\dots\dots SF_3, d \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ SF_m, 1 & SF_m, 2 & SF_m, 3 & \dots\dots & SF_m, d\# \end{bmatrix} \quad (8)$$

where m gives the number of sailfish, d is the number of variables and SF_{ij} shows the j^{th} dimension of ith sailfish. So, fitness of each sailfish is computed by calculation of fitness function using Eqs. (4) to (6) as follows:

$$Fitness\ f = f(sailfish) = f(SF_1, SF_2, \ldots \ldots SF_m). \quad (9)$$

The fitness value of each position of sailfish is calculated using the Eq. (7).

Elitism

It is a process in which fitness values are copied to next generation without any change. The best position sailfish is saved in each iteration and it is called as an elite. It may be possible that some sardines are injured during attack during hunting, so position of sailfish and injured sardine at their best fitness is saved in $X^i_{elite\_SF}$ and $X^i_{injured\_S}$.

Updating rule

In Sailfish optimization, at the ith iteration, new position of sailfish $X^i_{new\_SF}$ gets updated using Eq. (10).

$$X^i_{SF\_New} = X^i_{elite\_SF} - \lambda^i \times (rand(0,1) \times (\frac{X^iSF + X^iinjured_S}{2}) - X^i_{old}) \quad (10)$$

where $X^i_{elite\_SF}$ is the position of elite formed till now, *rand (0,1)* is the random number generated between 0 and 1. λ_i is a coefficient at the ith iteration that generated as follows:

$$\lambda_i = 2 \times rand\ (0, 1) \times PD - PD \quad (11)$$

Where *PD* is the prey density that shows number of preys at each iteration. *PD* is calculated by using following formula in Eq. (12):

$$PD = 1 - (\frac{N_{SF}}{N_{SF} + N_S}) \quad (12)$$

where N_{SF} and N_S denotes the number of sailfish and number of sardines, respectively.

Hunting and Prey

At the beginning, sailfish have more power to attack on prey and sardines are not injured. So, sardines sustain high speed. With the passage of time, their hunting power will decrease, and they need to update their position. In this SFO algorithm, the position of sardine $X^i_{new\_S}$ is updated by Eq. (13).

$$X^i_{new\_SF} = r \times \left(X^i_{elite\_SF} - X^i_{old_S} + AP\right) \quad (13)$$

where *AP* is calculated by using the Eq. (14).

$$AP = A \times (1 - (2 * itr * \varepsilon)) \quad (14)$$

where *AP* is attacking power.

The number of sardines updating their positions and amount of their displacement depends on the power of sailfish attack (AP). Using the Eq. (13), the number of sardines that update their position (N_U) and number of variables (N_V) from them can be calculated as follows:

$$N_U = (N_S \times AP) \quad (15)$$

$$N_V = (d_i \times AP) \quad (16)$$

The flow diagram of Sailfish optimization-based scheduling algorithm is depicted in **Figure 1**.

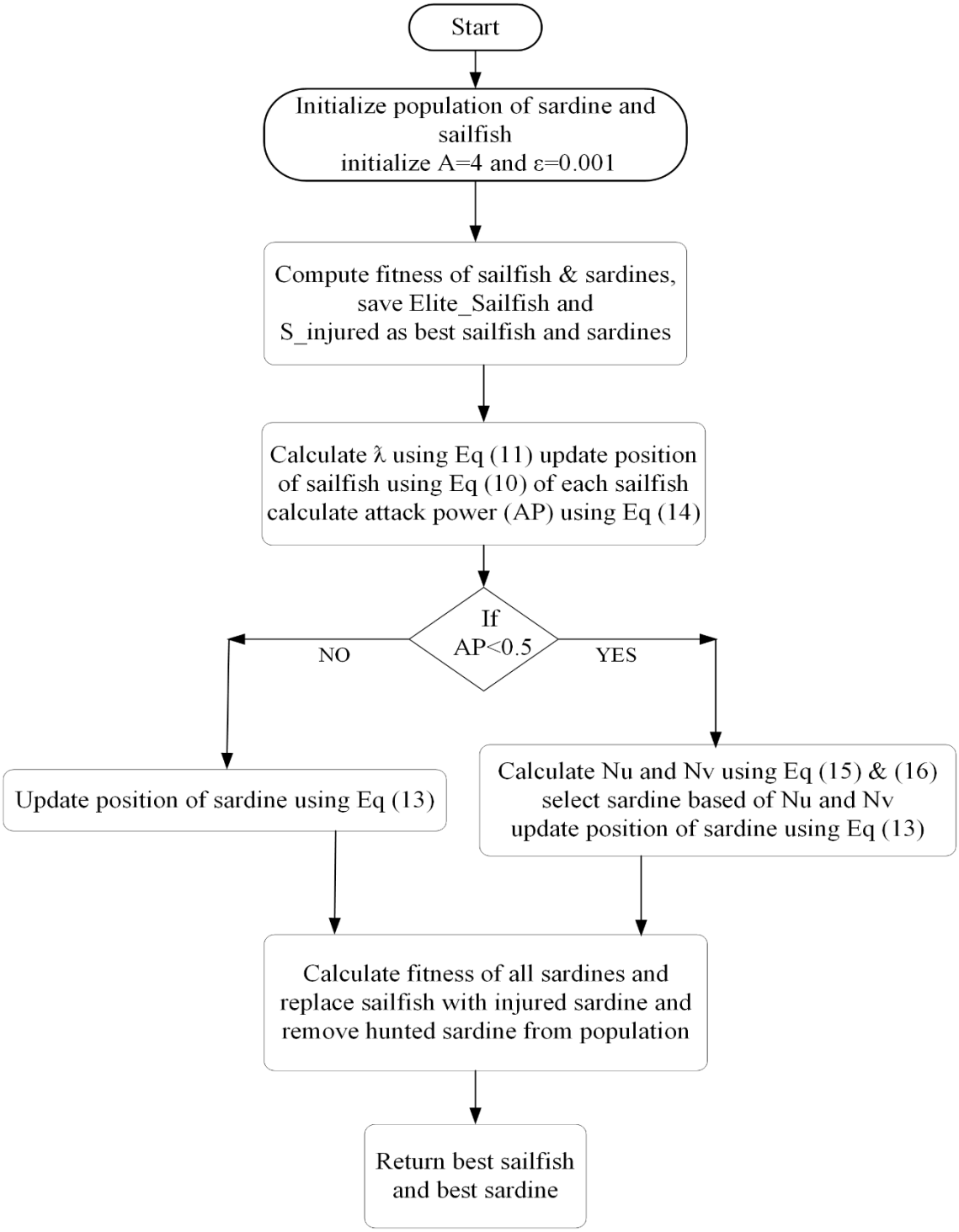

Figure 1 Flowchart of sailfish optimization-based scheduling algorithm.

Simulation set-up

Simulation was carried with Apache NetBeans IDE 11.3 configured with CloudSim Toolkit 3.03 [15], which is java based simulator, installed on HP Prodesk desktop PC having 64 bit windows 10, 16 GB RAM, 3.20 GHz CPU, with 8 cores, from 15 min to 2 h. All simulation parameters for settings of CloudSim are presented in table 1. The proposed SOSA was tested on NASA (named as DS1) real world workload taken from [25] recognized by CloudSim and randomly generated data-set (named as DS2) each having cloudlet size ranging from 1,000 - 10,000 in length. The data-set DS2 was generated as mean of 50 runs of cloudlet generator file to remove any type of biasness. The pricing model of all virtual machines was taken from Google App Engine. Proposed SOSA algorithm is compared with other heuristic techniques like GA and PSO to exhibit its performance in terms of execution cost and makespan. All these algorithms were developed in java using MOEA framework as used in [14]. The parameter values of all algorithms are depicted in **Table 2**.

Table 1 Parameter settings in CloudSim.

S No	Entities	Parameters	Values
1	User	No. of user	10 - 100
		Broker	1
2	Cloudlet	No. of Cloudlet	100 - 1,000
		Length	1,000 - 5,000
		File Size	600 MB
3	VM	No. of VM	50
		Policy	Space Shared
		Bandwidth	10,000 bps
		CPU Speed	600 - 2,400 MIPS
		VMM	Xen
		OS	Linux
		No. of CPU	1 each
4	Datacenter	No. of Datacenter	1

Table 2 Parameter values used in meta-heuristic algorithms.

Algorithms	Parameters	Values
GA	Population Size	1,000
	Max Iteration	1,000
	Crossover rate	0.5
	Mutation Rate	0.1
PSO	Particle size	100
	Self-coefficient, c1, c2	2
	Max Iteration	1,000
	Inertia Weight	0.9 - 0.4
SOSA	Population Size	30
	Inertial A	4
	Max Iteration	1,000
	Initial ε	0.001

Results and discussion

This section discusses the performance of proposed SOSA in terms of Quality of Service it can delivers to the users. The default scheduler of CloudSim is not considered for the comparison. SOSA is population based meta-heuristic optimization method and is compared with other meta-heuristic techniques i.e., Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) is considered to evaluate the performance of SOSA. Since makespan (sec) and execution cost ($) are 2 important

parameters of Quality of Service from user's point of view. So, we considered these factors for the comparison.

Case I (Data-set: DS1)

Figure 2 shows the makespan of SOSA algorithm computed with data-set DS1. Simulation has been performed on various number of cloudlets range of 100 - 1,000. The makespan is increasing with increase in number of cloudlets. The makespan of SOSA is lower than GA and PSO. Performance improvement ratio (PIR) of SOSA for makespan is in range of 7.58 to 28.26 % as compared to GA, with an average of 13.71 % and in range of 1.32 to 22.35 % as compared to PSO, with an average of 7.87 %.

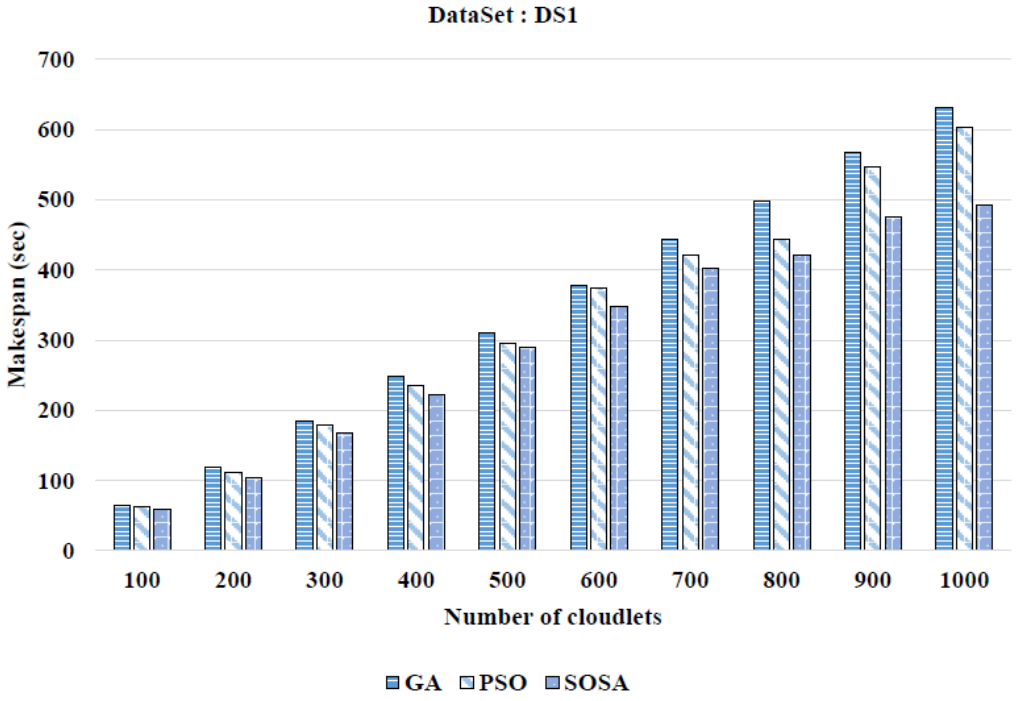

Figure 2 Makespan calculated for scheduling on DS1.

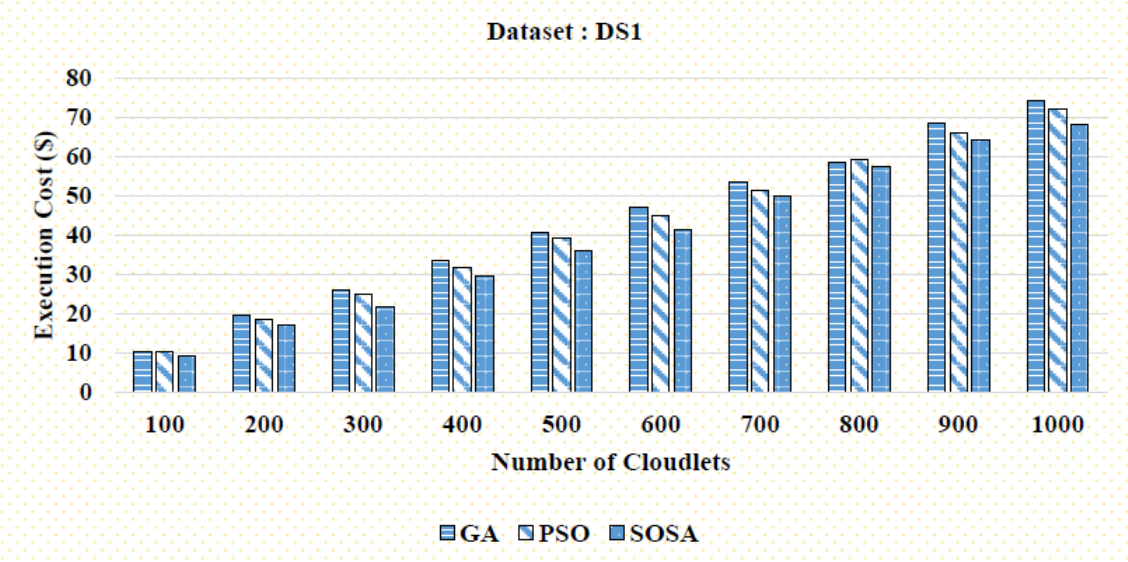

Figure 3 Execution cost calculated for scheduling on DS1.

Figure 3 shows the cost of execution computed with data-set DS1. Simulation has been performed on different number of cloudlets ranging from 100 to 1,000. The cost of execution increases with increase in the number of cloudlets. The execution cost of SOSA is lower than GA and PSO. Performance improvement ratio (PIR) of SOSA in terms of cost is in range of 1.67 to 19.28 % in comparison to GA, with an average of 11.30 % and in range of 11 to 45.19 % in comparison to PSO, with an average of 12.15 %.

Case II (Dataset: DS2)

Figure 4 shows the makespan of SOSA algorithm on data-set DS2. Simulation was performed by keeping same parameters values for GA and PSO in Cloudsim. The makespan of SOSA is lower than GA and PSO. Performance improvement ratio (PIR) of SOSA for makespan is in range 11.59 to 22.98 % in comparison to GA, with an average of 17.06 % and in range of 3.25 to 12.12 % in comparison to PSO, with an average of 8.94 %.

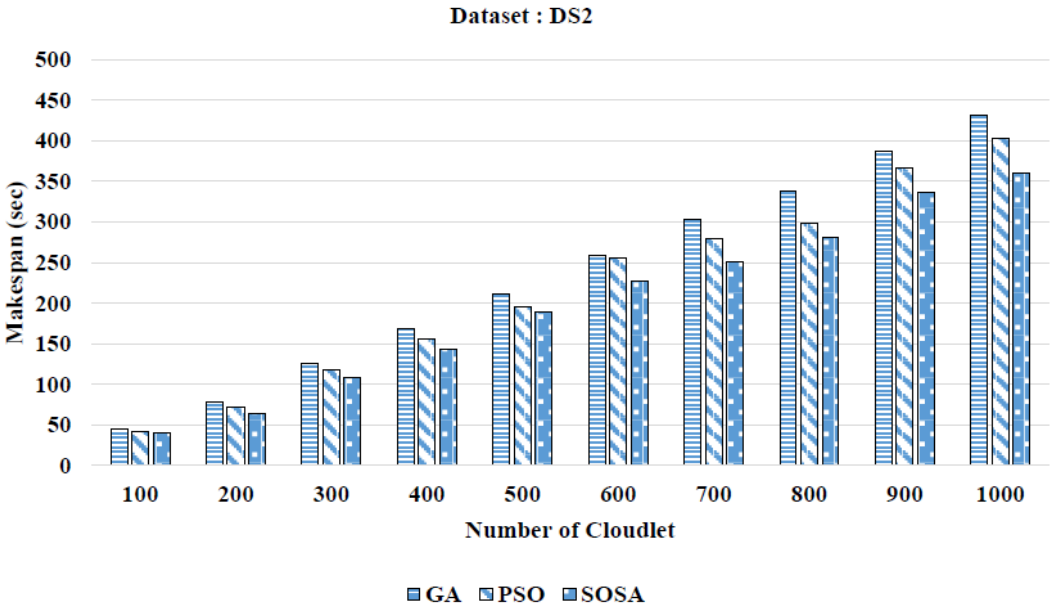

Figure 4 Makespan calculated for scheduling on DS2.

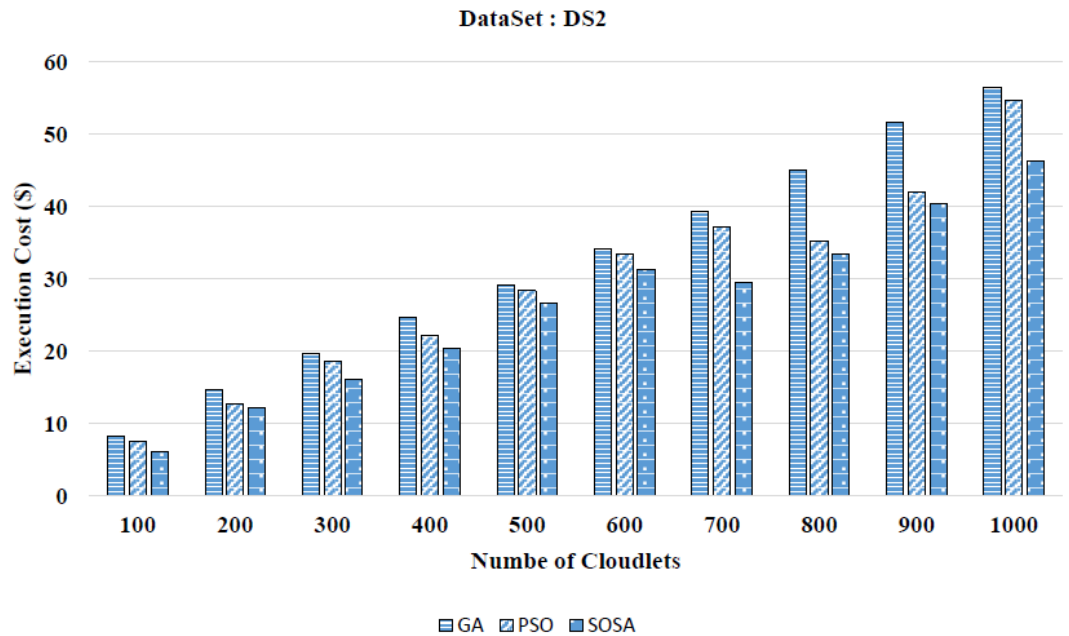

Figure 5 Execution cost calculated for scheduling on DS2.

Figure 5 shows the variation in execution cost with data-set DS2. Simulation has been performed on different number of cloudlets ranging from 100 to 1,000. The cost of execution increases with increasing in number of cloudlets. The execution cost of SOSA is lower than GA and PSO. Performance improvement ratio (PIR) of SOSA in terms of cost is in range of 9.18 to 36.98 % in comparison to GA, with an average of 23.72 % and in range of 4.05 to 26.49 % in comparison to PSO, with an average of 12.15 %.

Statistical significance on DS1

The statistical significance is carried on the results obtained from various data-sets DS1 and DS2. However, only real workload data-set i.e., DS1 was significantly tested with its correlation and regression values in this paper. It is found from **Table 3** that makespan has significant correlation with execution cost at R = 1 and confidence level was 99 %. The relation of number of cloudlets with its makespan and execution cost is depicted in Eq. (17).

$$N = 1.164 * M + 4.714 * C - 3.469 \quad (17)$$

where N is number of cloudlets, M is makespan (sec) and C is execution cost ($) calculated for proposed SOSA algorithm.

Table 3 Statistical significance on DS1.

		Cloudlets	**Makespan**	**Cost**
Cloudlet	Pearson Correlation	1	1.000**	0.999**
	Sig.(2 tailed)	-	0.000	0.000
Makespan	Pearson Correlation	1.000**	1	0.998**
	Sig.(2 tailed)	0.000	-	0.000
Cost	Pearson Correlation	0.999**	0.998**	1
	Sig.(2 tailed)	0.000	0.000	-

Conclusions

Efficient scheduling in IaaS cloud computing environment is still a big challenge for researchers in the world. Due to its NP hard nature, scientists and researchers are continuously trying to provide optimal solutions to provide better quality of service at minimum cost and in less time. In this article, a newly developed population based meta-heuristic technique namely, Sailfish Optimization based Scheduling Algorithm (SOSA) has been proposed. Simulation is performed on real workload traces from NASA (DS1) and randomly generated dataset (DS2). It exhibited significant results that SOSA provides better Quality of Service with less cost in less time to user. The exploration and exploitation capability of Sailfish optimization can be applied on various problems of cloud computing like load balancing, VM migration, VM allocation etc. The other parameters of QoS like response time, penalty cost, utilization of resources etc. may be considered in future. Sailfish optimization is a promising area for future research in VM allocation and placement, and can be applied on scientific workflow in cloud.

Acknowledgements

This research work is supported by UGC under National Fellowship program of Ministry of Social Justice and Empowerment, Govt. of India, Grant No. 2017-18/29146. Authors also acknowledge Department of Computer Science and Engineering of DCR University and Technology, Murthal, Sonepat for providing various facilities in research lab.

References

[1] K Sreenu and M Sreelatha. W-scheduler: whale optimization for task scheduling in cloud computing. *Cluster Comput.* 2019; **22**, 1087-98.

[2] DD Bhatt. A revolution in information technology: Cloud computing. *Walailak J. Sci. Tech.* 2011; **9**, 108-13.

[3] J Gubbi, R Buyya, S Marusic and M Palaniswami. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* 2013; **29**, 1645-60.

[4] H Gupta, AV Dastjerdi, SK Ghosh and R Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Softw. Pract. Exper.* 2017; **47**, 1275-96.

[5] J Kumar, A Malik, SK Dhurandher and P Nicopolitidis. Demand-based computation offloading framework for mobile devices. *IEEE Syst. J.* 2018; **12**, 3693-702.

[6] M Kumar, Suman and S Sangwan. A survey on virtual machine scheduling algorithms in cloud computing. *Int. J. Comput. Sci. Eng.* 2018; **6**, 485-90.

[7] MK Suman. Priority-based virtual machine selection algorithm in cloud computing. *Int. J. Recent Technol. Eng.* 2019; **8**, 1457-62.

[8] S Smanchat and K Viriyapant. Scheduling dynamic parallel loop workflow in cloud environment. *Walailak J. Sci. Tech.* 2016; **15**, 19-27.

[9] Y Ge and G Wei. GA-based task scheduler for the cloud computing systems. *In*: Proceedings of the 2010 IEEE International Conference on Web Information Systems and Mining, Sanya, China. 2010, p. 181-6.

[10] MM Rashidi, OA Bég, AB Parsa and FF Nazari. Analysis and optimization of a transcritical power cycle with regenerator using artificial neural networks and genetic algorithms. *Proc. Inst. Mech. Eng. A: J. Power Energ.* 2011; **225**, 701-17.

[11] A Mousapour, A Hajipour, MM Rashidi and N Freidoonimehr. Performance evaluation of an irreversible miller cycle comparing FTT (finite-time thermodynamics) analysis and ann (artificial neural network) prediction. *Energy* 2016; **94**, 100-9.

[12] J Chang, Z Hu, Y Tao and Z Zhou. Task scheduling based on dynamic non-linear PSO in cloud environment. *In*: Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS) IEEE, Beijing, China. 2018, p. 877-80.

[13] MM Rashidi, M Ali, N Freidoonimehr and F Nazari. Parametric analysis and optimization of entropy generation in unsteady MHD flow over a stretching rotating disk using artificial neural network and particle swarm optimization algorithm. *Energy* 2013; **55**, 497-510.

[14] A Rajagopalan, DR Modale and R Senthilkumar. *Optimal scheduling of tasks in cloud computing using hybrid firefly-genetic algorithm. In:* SC Satapathy, KS Raju, K Shyamala and DR Krishna (Eds.). Advances in decision sciences, image processing, security and computer vision, Springer, Cham, 2020, p. 678-87.

[15] RN Calheiros, R Ranjan, A Beloglazov, CAFD Rose and R Buyya. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* 2009; **39**, 701-36.

[16] D Wu. Cloud computing task scheduling policy based on improved particle swarm optimization. *In:* Proceedings of the 2019 International Conference on Virtual Reality and Intelligent Systems, Jishou, China. 2019, p. 99-101.

[17] Z Tong, H Chen, X Deng and K Li. A Scheduling scheme in the cloud computing environment using deep Q-learning. *Inf. Sci.* 2020; **512**, 1170-91.

[18] M S Sanaj and PMJ Prathap. Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere. *Eng. Sci. Technol. Int. J.* 2020; **23**, 891-902.

[19] E Safari, SO Pourhashemi and M Gharahkhani. A hybrid swarm particle optimization algorithm for task scheduling in cloud computing. *J. Comput. Decis. Support Syst*. 2020; **7**, 18-26.

[20] G Natesan and A Chokkalingam. An improved grey wolf optimization algorithm based task scheduling in cloud computing environment. *Int. Arab J. Inf. Technol.* 2020; **17**, 73-81.

[21] X Chen, L Cheng, C Liu, Q Liu and J Liu. A WOA-based optimization approach for task scheduling in cloud computing systems. *IEEE Syst. J.* 2020; **14**, 3117-28.

[22] X Shi, X, X Zhang, M Xu. A Self-Adaptive Preferred Learning Differential Evolution Algorithm for Task Scheduling in Cloud Computing. *In:* Proceedings of the IEEE International Conference on

Advances in Electrical Engineering and Computer Applications, Dalian, LiaoNing, China. 2020, p. 145-8.

[23] L Abualigah, A Diabat. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Clust. Comput.* 2020; **5**, 205-23.

[24] S Shadravan, HR Naji and VK Bardsiri. The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Eng. Appl. Artif. Intell.* 2019; **80**, 20-34.

[25] SHH Madni, MSA Latiff, J Ali and SM Abdulhamid. Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds. *Arab. J. Sci. Eng.* 2019; **44**, 3585-602.