ADV MATH SCI JOURNAL
Advances in Mathematics: Scientific Journal **10** (2021), no.1, 423–432
ISSN: 1857-8365 (printed); 1857-8438 (electronic)
https://doi.org/10.37418/amsj.10.1.42

AN EFFICIENT LOAD SCHEDULING TECHNIQUE USING OPPOSITIONAL SPARROW SEARCH ALGORITHM FOR CLOUD COMPUTING ENVIRONMENT

J. Robert Adaikalaraj[1] and T. Vengattaraman

ABSTRACT. Cloud computing (CC) environment is commonly applied for provisioning effectual services and offers diverse benefits, along with some challenges like load balancing and load scheduling. The load scheduling can be defined as the procedure of assigning and balancing the load like tasks or cloudlets to virtual machines (VMs) in the cloud platform. A main goal of the load scheduling algorithm is to lessen the transfer time and total cost incurred in the system. This paper presents a new oppositional sparrow search algorithm (OSSA) for efficient load scheduling in CC environment. The classical sparrow search algorithm (SSA) is based on the group wisdom, foraging and anti-predation behaviour of sparrows. To improve the convergence speed of SSA, Opposition based learning (OBL) concept is incorporated to it. The proposed OSSA based load scheduler has the ability of scheduling the load effectively in the CC platform. An extensive set of simulations were carried out to ensure the goodness of the OSSA and the results are investigated under several aspects. The experimental results indicated the superior nature of the OSSA over the compared methods on all the applied scenarios.

[1]*corresponding author*

2020 *Mathematics Subject Classification.* 68T20.

Key words and phrases. Cloud computing, Load scheduling, Sparrow Search, Oppositional based learning, Energy efficiency.

Submitted: 20.12.2020; *Accepted:* 06.01.2021; *Published:* 21.01.2021.

1. INTRODUCTION

Generally, the cloud platforms are created on the basis of distributed computing, grid computing and virtualization. The cloud platform is mostly comprised with maximum benefits like storage area with global access platforms and minimal requirement for hardware in the end user scenario. Even though cloud platforms are beneficial, it is constrained with some limitations. Few of the major cloud challenges are scalability, higher accessibility, load balancing, cost, and performance [1]. Also, it reallocates the entire load to specific nodes which leads to efficient resource consumption and also maximizes the response time of allocated operations. The promising issue in cloud environment is to maintain the best performance while the data access point is requested [2]. The significant objectives of load balancing are load evaluation, load compression, reliable function, effective performance, node interaction, node election and several other actions should be considered in deploying the method In CC, load balancing depends upon the effective load balancing which is modified from conventional thought of load balancing structure and achieved with the help of commodity servers to perform the load balancing. Mostly, load balancers stimulate the accessibility of cloud resources and enhance the function.

In [3] the authors defined the importance and models of load balancing for resolving the problems of node imbalancement. Load scheduling is said to be the task of assigning and computing the cloudlets on VMs in CC optimally which limits the processing cost. The key objective of load scheduling is to accomplish better performance by minimizing the computational time and overall processing cost. The dynamic methods apply bio-centric swarm or gravity oriented approaches for scheduling the overheads. The swarm relied techniques depends upon the particles which are applied to schedule the workload in CC defined by [4] for placing the upcoming particle which depends the position and velocity of a particle. The features and demerits of diverse load scheduling models are explained in [5].

2. THE PROPOSED OSSA BASED LOAD SCHEDULER

The proposed OSSA based cloud scheduler is based on SSA and OBL concepts. The OSSA intends to lessen the total computation cost and schedules the load on the VMs effectively. The computation cost comprises the transfer as well as

execution cost in the cloudlets. It offers maximum search space exploitation and user fulfilment over the compared methods. The data center has $(VMs)^{Cloudlets}$ possible ways to execute cloudlets on the corresponding VMs. When 3 cloudlets are implemented on 2 VMs, then the probability is 8. The sparrows S are initialized in CloudSim simulator, as defined below:

$$S_i = \left(s_i^1,\, s_i^2,\, \ldots,\, s_i^n,\, \ldots,\, s_i^d\right)$$

$$\forall i = 1 \text{ to } 25 \text{ and } n = 1\ to\ 10.$$

The fitness function validates the fitness value of the sparrows in the search space. The first sparrow is initialized using OBL followed by the selection of next sparrow using optimal fitness value. It is based on bandwidth, MIPS, execution cost and transfer cost of the cloudlets and the VMs. Consider $\mathrm{Ct}_{exec}(M)_j$ is the total cost of execution of every sparrow assigned for the calculation of the VM resources PCj. It is determined by totalling every weight allocated to the nodes in mapping of the sparrows S of every cloudlet assigned to every resource. Assume $\mathrm{Ct}_{\mathrm{frons}}(M)_j$ is the total transfer cost among the cloudlets assigned for calculating the VM resource PCj. The outcome is the multiplication of the output file size and communication cost. The average cost of data between two resources of communication is provided as $dS\left(k1\right), S\left(\mathrm{k2}\right)$ and the sparrows are not relying on one another. The total cost is added for every sparrow S by execution and transfer cost and afterwards the cost is additionally reduced for the evaluation of the fitness value. The fitness function can be represented as follows:

$$\mathrm{Ct}_{exec}(S)_j = \sum_k \omega_{kj}, \qquad \forall S\left(\mathrm{k}\right) = j$$

$$\mathrm{Ct}_{trans}(S)_j = \sum_{\mathrm{k1}\in T} \sum_{\mathrm{k2}\in T} d_{S(\mathrm{k1}),S(\mathrm{k2})} * e_{\mathrm{k1,k2}}$$

$$\forall S\left(k1\right) = j \quad \text{and} \quad S\left(1 < 2\right)$$

$$\mathrm{Ct}_{\mathrm{total}}(S)_j = \mathrm{Ct}_{exec}(S)_j + \mathrm{Ct}_{trans}(S)_j$$

$$Cost_{Total}\left(S\right) = \max\left(\mathrm{Ct}_{\mathrm{total}}(S)_j\right), \qquad \forall j \in S$$

$$Minimize\ \left(\mathrm{Cost}_{Total}\left(S\right), \forall S\right)$$

2.1. **Sparrow Search Algorithm (SSA).** Generally, sparrow is the species of bird that is most familiar one because it survives more near to us and tends to associate with their groups. For simulation purposes, virtual sparrows are used for searching food sources. The positions of the sparrows are defined in Eq. (2.1), as given below:

$$X=\begin{bmatrix} \chi_{1,1} & \chi_{1,2} & \cdots & \chi_{1,d} \\ \chi_{2,1} & \chi_{2,2} & \cdots & \chi_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ \chi_{n,1} & \chi_{n,2} & \cdots & \chi_{n,d} \end{bmatrix},$$

where n indicates the number of sparrows and *d* refers the dimensions of the parameters which needs to be tuned. Next, the fitness value of every sparrow can be determined using the following equation:

$$F_X = \begin{bmatrix} f([x_{1,1} & x_{1,2} & \cdots & \cdots & x_{1,d}]) \\ f([x_{2,1} & x_{2,2} & \cdots & \cdots & x_{2,d}]) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f([x_{n,1} & X_{n,2} & \cdots & \cdots & x_{n,d}]) \end{bmatrix},$$

where the value exists in the every row of F_x defines the fitness va1ue of all the individuals. In SSA, the producer with optimal fitness value holds the priority of obtaining food in the searching process. Besides, the producer sparrow is accountable to search food and guide the movement of the whole population. Based on the rules (2.1) and (2.1), during each iteration, the location of the producer is updated as be1ow:

$$x_{i,j}^{\mathrm{t}+1}=\begin{cases} X_{i,j}^{t} \cdot \exp\left(\frac{-i}{\alpha \cdot iter\_\max}\right) & \text{if } R_2 < ST \\ X_{i,j}^{t} + Q.L & \text{if } R_2 \geq ST \end{cases},$$

where t represents the present round, $j = 1, 2, \ldots, d$. $\chi_{i,j}^{\mathrm{t}}$ defines the va1ue of the jth dimension of the $i\mathrm{th}$ sparrow at iteration t. Besides, $iter\_\max$ is a constant with several iterations. $\alpha \in (0, 1]$ is a random number. $R_2\,(R_2 \in [0, 1])$ and $ST\,(ST \in [0.5, 1.0])$ defines the alarm value and the safety threshold correspondingly. Besides, O is an arbitrary number that follows the normal distribution and L depicts a matrix of $1 \times d$ for which every element inside is 1.

(1) If R_2<ST, it indicates the absence of predators and the producers enter into a searching model

(2) If $R_2 = ST$, it indicates the few sparrows have discovered the predator, and every sparrow needs to fly to a safer area at a faster rate.

In case of scroungers, it is needed to enforce the rules (1) and (2). When they won the battle, they will attain the producer's food instantly; else they persist to accomplish the rules (1) and (2). The position update equation of the scrounger can be represented by:

$$X_{i,j}^{\mathrm{t}+1}=\begin{cases} \mathrm{Q}\cdot \exp\left(\frac{x^{\mathrm{r}}_{\mathrm{worst}-X^{\mathrm{t}}_{i,j}}}{i^2}\right) & if\ i>n/2 \\ X_P^{\mathrm{t}+1}+\left|X_{i,j}^{\mathrm{t}}-X_P^{\mathrm{t}+1}\right| A^{+}\cdot L & \text{otherwise} \end{cases}$$

where X_P is the optimum position attained by the producer, $\mathrm{X}_{\mathrm{worst}}$ represents the present global worst location, A defines a matrix of 1 $\times d$ for which every element inside is arbitrarily allocated to 1 or -1, and $A^{+}{=}A^{T}\left(AA^{T}\right)^{-1}$. If $i{>}n/2$, it is recommended that the $i\mathrm{th}$ scrounger with the worst fitness value is most probable to be hungry. During simulations, the sparrows are assumed as the ones which are danger aware in 10-20 % of the total population. The initialized positions of the sparrows are arbitrarily created in the population. Based on the rules, it can be mathematically defined as follows:

$$X_{ij}^{t+1}=\begin{cases} X_{\mathrm{best}}^{t}+\beta\cdot\left|X_{ij}^{t}-X_{\mathrm{best}}^{t}\right| & \text{if } f_i{>}f_g \\ \chi_{ij}^{t}+K\cdot\left(\frac{|X_{ij}^{t}-X_{\mathrm{worst}}^{t}|}{(f_i-f_w)+\varepsilon}\right) & \text{if } f_i{=}f_g \end{cases},$$

where X_{best} is the present global optimal location, β is the step size control variable, is a normal distributionof random numbers with a mean value of 0 and a variance of 1. $K\in[-1,\ 1]$ is a random number. Here f_i is thefitness value of the present sparrow. f_g and f_w are the present global best and worst fitness values, correspondingly, ε is the small constant used to eliminate the zero-division-error. In case of simplicity, if $f_i{>}f_g$, it is denoted that the sparrow exist at the edge of the swarm, X_{best} defines the position of the middle of the population and is secure around it. $f_i{=}f_g$ indicates that the sparrows, which are in the center of the population, are conscious of the risk and requires moving nearer to other sparrows and K defines the direction of the sparrow movement.

2.2. **Oppositional Sparrow Search Algorithm (OSSA).** OBL is an effective tool used for optimization to enhance the convergence rate of different meta-heuristic algorithms. The effective design of the OBL comprises the validation

of the present population in the identical round to determine the optimal candidate for a provided problem. The concept of OBL has been effectively employed in and the idea of opposite number is required to be defined for explaining OBL.

2.3. **OSSA Enabled Load Scheduling Process.** As discussed in previous sections, the OSSA are an integration of SSA and OBL concepts to attain optimal population initialization and faster convergence. The evaluation function can be represented as follow.

$$\text{Evaluation function } = 1 - \left(\alpha_{ti} \times \frac{\mathrm{t}_i - \mathrm{t}_{\min}}{t_{\max} - \mathrm{t}_{\min}} + \alpha_{ci} \times \frac{c_i - c_{\min}}{c_{\max} - C_{\min}} \right).$$

Here, every executable process has been validated for selecting the optimal execution order. To schedule the load effectively, the OSSA has been employed. Besides, different task as input comprises two variables namely arrival and run time. The evaluation function can be determined as:

$$1 - \left(\alpha_{ti} \times \frac{t_i - \mathrm{t}_{\min}}{\mathrm{t}_{max} - t_{\min}} + \alpha_{ci} \times \frac{c_i - c_{\min}}{c_{max} - c_{\min}} \right),$$

where $\mathrm{f}_{\min}$ is minimum run time, $\mathrm{f}_{\max}$ is maximum run time, $c_{\min}$ is minimum input time and $c_{mox}$is maximum input time.

3. Performance Validation

The projected OSSA method was evolved with the help of MATLAB tool. In addition, each operation has the corresponding login time and implementation time that has been initiated randomly and provided the application of Job shop software. A group of methods like improved PSO (IPSO), IPSO with firefly, Firefly with multi-objective IPSO (FIMPSO), GA, RR, first come first serve (FCFS), SJF and firefly algorithms were applied for comparison. In addition, collection of metrics is applied for examining the execution time, resource application, scalability, make span as well as throughput.

3.1. **Results analysis interms of Reliability.** Fig.1 defines the reliability analysis of the projected OSSA on the under various task sizes. This model provides higher reliability is assumed to be efficient load scheduler. From the figure, it is shown that the OSSA have exhibited maximum reliability than the compared methods under all types of tasks. Simultaneously, the RD model has defined ineffective performance when compared with other technologies.

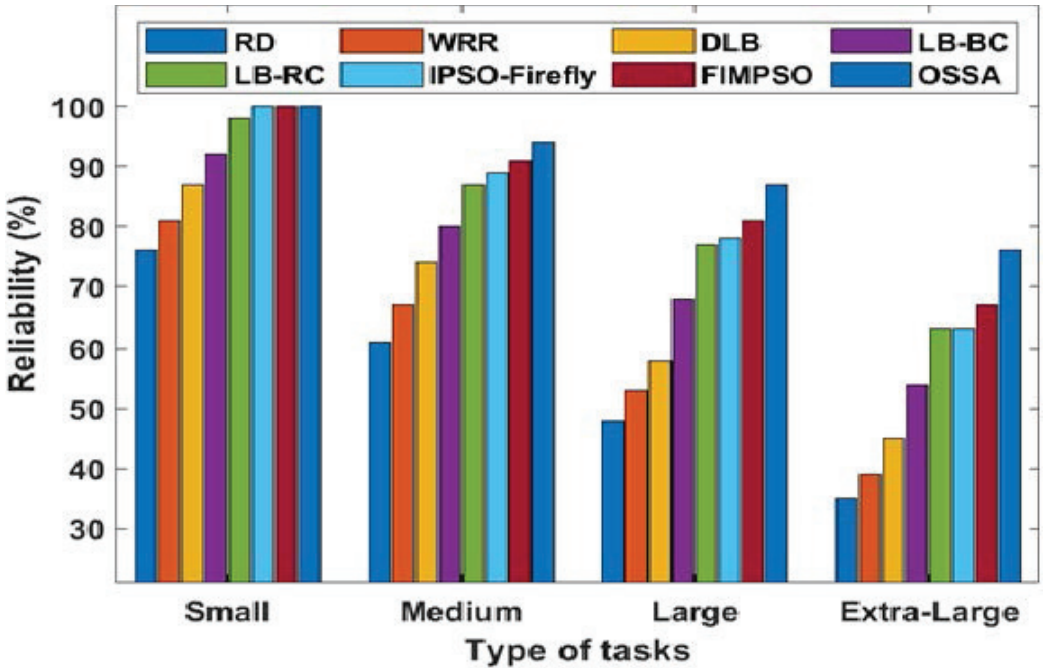

FIGURE 1. Reliability Analysis of OSSA with other models

3.2. **Results analysis interms of Average Execution Time.** Fig.2 illustrates the average execution time analysis of the proposed OSSA on the under different task sizes. The method has provided lower average execution time has been assumed as a productive load scheduler. From the figure, it is clear that the OSSA have showcased least average execution time than the compared methods under every task. Concurrently, the RD approach has represented poor performance when compared with all other techniques. For sample, under the existence of supplementary tasks, the existing RD is treated as an ineffective method and attained a higher average execution time of 110ms. Meantime, the WRR model has obtained moderate average execution time of 103ms. In the same way, the LB-BC method has attempted to imply reasonable performance with the average execution time of 88ms. Similarly, the LB-RC and IPSO-Firefly methodologies have exhibited superior performance to the previous models by achieving least average execution time of 82ms and 79ms correspondingly. Furthermore, the FIMPSO method has implemented competing performance with the less average execution time of 76ms. Thus, the projected OSSA has outperformed the previous technologies and reaching lower average execution time of 73ms.

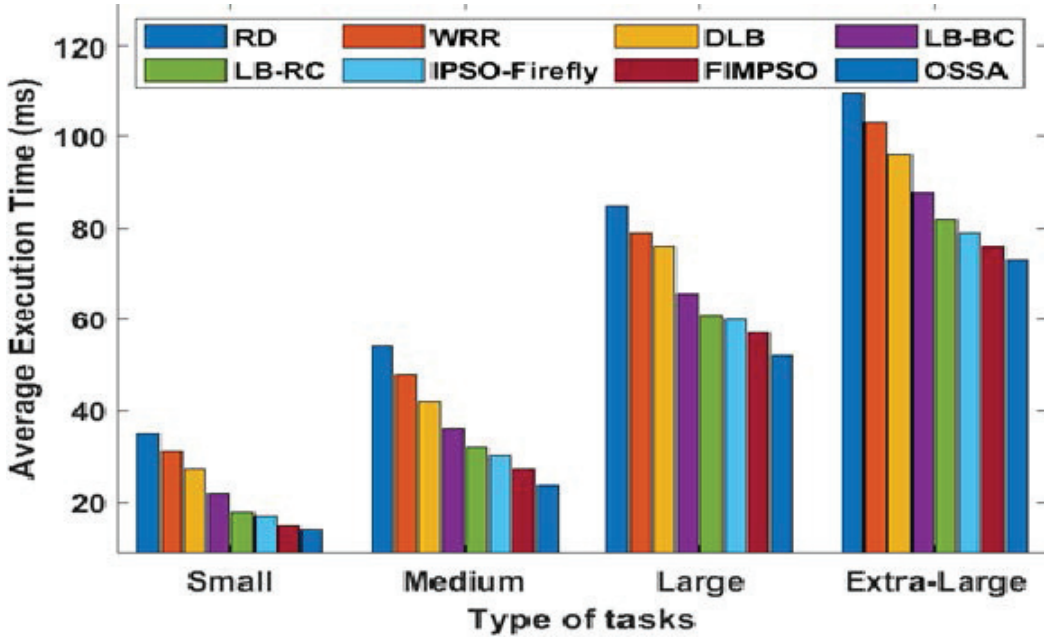

FIGURE 2. Average Execution time Analysis of OSSA with other models

3.3. **Results analysis interms of Average Throughput.** Fig. 3 defines the average throughput analysis of the deployed OSSA on the under several task sizes. The model which has generated greater average throughput has been assumed as a productive load scheduler. From the figure, it is noticeable that the OSSA has showcased higher average throughput over the related methods under all types of tasks. The overall experimental analysis confirmed the superior nature of the OSSA under varying number of tasks, memory and CPU capacities.

4. CONCLUSION

In this paper, an efficient load scheduler using OSSA has been presented for CC environment. The classical SSA is based on the group wisdom, foraging and anti-predation behaviour of sparrows. To improve the convergence speed of SSA, OBL concept is incorporated to it. The OSSA includes the OBL mechanism in the SSA to improve the convergence rate and thereby enhances the overall system performance. The presented OSSA based load scheduler has the capability of scheduling the load effectively in the CC platform. The proficient performance of the OSSA has been validated and the results are examined under

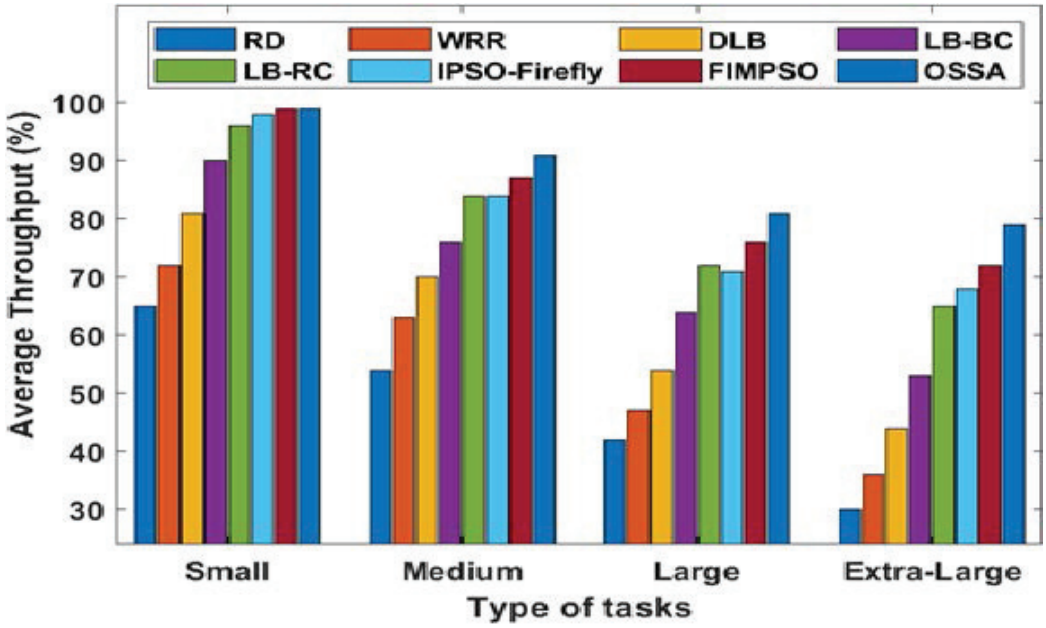

FIGURE 3. Average Throughput Analysis of OSSA with other models

diverse aspects. The experimental outcome indicated the superior characteristics of the OSSA over the compared methods on all the applied scenarios. The proposed OSSA can be extended to the use of machine learning and deep learning models.

REFERENCES

[1] A. ADITYA, U. CHATTERJEE, S. GUPTA: *A comparative study of different static and dynamic load balancing algorithm in cloud computing with special emphasis on time factor*, Int. J. Curr. Eng. Technol., **5**(3) (2015), 1898–1907.

[2] M.M. GOLCHI, S. SARAEIAN, M. HEYDARI: *A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments:Performance evaluation*, Computer Networks, **162** (2019), art.id.106860.

[3] A. KHIYAITA, BAKKALI EL, M. ZBAKH, D.E. KETTANI: *Load balancing cloud computing: State of art*, IEEE National Days of Network Security and Systems, 2012, 106–109.

[4] E. PACINI, C. MATEOS, C.G. GARINO: *Distributed job scheduling based on swarm intelligence: A survey*, Comput. Electr. Eng., **40** (2014), 252–269.

[5] D. CHAUDHARY, B. KUMAR: *Analytical study of load scheduling algorithms in cloud computing*, IEEE International Conference on Parallel, Distributed and Grid Computing, 2014, 7–12.

PG Department of Computer Applications,
Joseph's College of Arts and Science (Autonomous), Cuddalore.
Email address: j.rubertraj@gmail.com

Department of Computer Science, Pondicherry University, Puducherry, India
Joseph's College of Arts and Science (Autonomous), Cuddalore.
Email address: vengattaramant@gmail.com