Task Scheduling in Cloud Computing using Lion Optimization Algorithm

Nora Almezeini and Prof. Alaaeldin Hafez
College of Computer and Information Sciences
King Saud University
Riyadh, Saudi Arabia

***Abstract*—Cloud computing has spread fast because of its high performance distributed computing. It offers services and access to shared resources to internet users through service providers. Efficient performance of task scheduling in clouds is one of the most important research issues which needs to be focused on. Various task scheduling algorithms for cloud based on metaheuristic techniques have been examined and showed high performance in reasonable time such as scheduling algorithms based on Ant Colony Optimization (ACO), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). In this paper, we propose a new task-scheduling algorithm based on Lion Optimization Algorithm (LOA), for cloud computing. LOA is a nature-inspired population-based algorithm for obtaining global optimization over a search space. It was proposed by Maziar Yazdani and Fariborz Jolai in 2015. It is a metaheuristic algorithm inspired by the special lifestyle of lions and their cooperative characteristics. The proposed task scheduling algorithm is compared with scheduling algorithms based on Genetic Algorithm and Particle Swarm Optimization. The results demonstrate the high performance of the proposed algorithm, when compared with the other algorithms.**

Keywords*—Cloud computing; task scheduling algorithm; cloud scheduling; lion optimization algorithm; optimization algorithm

I. Introduction

Cloud computing is considered to be a distributed system that offers services to internet users through service providers such as Amazon, Google, Apple, and Microsoft. Cloud computing uses internet technologies to offer elastic services that support variable workloads and dynamic access to computing resources.

Many of the scientific researches on cloud computing had focused on the performance efficiency of task scheduling. Task scheduling focuses on mapping tasks to appropriate resources, efficiently. Finding an optimal solution in cloud computing is considered an NP-complete problem. Each scheduling algorithm is based on one or more strategy. The most important strategies or objectives commonly used are time, cost, energy, quality of service (QoS), and fault tolerance [1], [2]. Several scheduling algorithms based on heuristic algorithms, such as Min-Min, Max-Min, and Heterogeneous Earliest Finish Time (HEFT) algorithms have been developed for cloud systems [3], [4]. In addition, different metaheuristic task scheduling algorithms that generate optimal schedules, such as the scheduling algorithm based on Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) [3], [5] have also been developed.

In this study, a new task scheduling algorithm is proposed for cloud environment using the concept of lion optimization algorithm (LOA), which was proposed in [6]. LOA is a nature-inspired algorithm based on the special lifestyle of lions and their cooperative behaviors. To evaluate the performance of the proposed algorithm, a comparative study is done among the proposed algorithm, task scheduling based on PSO algorithm, and task scheduling using the GA.

The main objective of this research paper is to propose a task-scheduling technique for cloud computing using the LOA to minimize the total execution time of the task on the cloud resources (makespan). Section 2 reviews some literature on LOA and some metaheuristic algorithms. Section 3 describes the proposed algorithm. Section 4 presents the experimental methodology and simulation parameters, followed by metrics used in experiment in Section 5. Section 6 presents the results of simulations and comparisons. Finally, conclusion and the future work are discussed in Section 7.

II. Related Work

Many metaheuristic algorithms have been proposed and applied for task scheduling in the area of cloud computing. Metaheuristic algorithms depend on two techniques to be effective. The first technique is “exploitation”, which exploits the best solution from among the previous results. The second technique is “exploration”, which explores new areas of the solution space. Most of these algorithms are distinguished and remarkable, such as the Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and League Championship Algorithm (LCA), and many more algorithms [5], [7].

The GA is a metaheuristic technique that was introduced by Holland in 1975 [8]. It provides useful solutions to optimization problems by applying the principles of evolution. The GA begins by initializing a population with random candidate solutions called individuals. Each individual is evaluated by a fitness function, which can be different according to the given optimization objective. Then, a proportion of the population is selected to reproduce a new generation. After that two main genetic operators are used to generate the new-generation population. These two operators are: crossover and mutation [9].

Using the GA in cloud task scheduling is a powerful approach as it provides better solutions with increase in the population size and number of generations. However, the

random generation of the initial population leads to schedules that are not very fit. Therefore, when these schedules are mutated with each other, there is a very low probability of producing a child better than the parents. Therefore, many researches have been conducted on improving the GA, especially, the initial steps, in order to improve the performance. For example, the authors in [10] improved the GA by using the Min-Min and Max-Min algorithms for generating the initial population. This provided a better initial population and better solutions than the standard GA, which initialized the population randomly.

The PSO metaheuristic algorithm was proposed by Kennedy and Eberhart in 1995 [11]. It was derived from the social behavior of particles. Each particle has position and velocity, which are initialized randomly. Moreover, each particle has a fitness value, and knows its personal best value (pbest) and the global best value (gbest). In each iteration, the particle improves its position based on its velocity using the gbest and pbest values [5], [12].

Task scheduling in clouds using PSO algorithm was found to be faster than that using the GA. It spent shorter time to complete the different scheduling tasks. In addition, the PSO provided better results for large-scale optimization problems, than the GA. However, many techniques and strategies were developed to improve the PSO for task scheduling. For example, [13] proposed an algorithm that combined the ACO and PSO algorithms in order to improve the performance. This combination improved the convergence speed and the resource utilization ratio.

In this paper, a new task-scheduling algorithm has been proposed using the concept of a new optimization algorithm called Lion Optimization Algorithm. It is based on the lifestyle and social organization of lions. In 2012, Wang [14] proposed an algorithm inspired by a few characteristics of lions, named the "Lion Pride Optimizer". It was based on the fighting and mating between lions. Rajakumar [15] proposed an algorithm named "The Lion's Algorithm", which was based on the mating, territorial defense, and territorial takeover. In 2015, Yazdani and Jolai [6] proposed the Lion Optimization Algorithm (LOA), which was different from the previous algorithms. It was inspired by simulating the isolated life style and cooperative behaviors of lions, such as hunting, territory marking, migration, and the different life styles of the nomad and resident lions, in addition to mating and fighting.

III. Task Scheduling based on Lion Optimization Algorithm

The LOA was developed based on the simulations of the behaviors of lions, such as hunting, mating, and defense. Lions have two organizational behaviors: resident behavior and nomadic behavior. Residents live in groups called prides. A resident lion may become a nomad, and vice versa. In the LOA, the initial population is generated randomly over the solution space where every single solution is called a "Lion". (%N) of lions in the population are selected randomly as nomad lions and the rest of the population are residents. Residents are divided randomly into (P) prides. (%S) of the lions in each pride are considered female and the rest are male. However, this proportion is reversed for nomad lions. The parameters and their meanings are shown in Table 1.

The proposed task-scheduling algorithm based on the LOA is detailed in the following steps:

Step 1: Initialize population

In this study, our target is to resolve the task scheduling for cloud computing and minimize the makespan of the solution, which is the maximum completion time for all tasks. Therefore, we should map each underlying solution to a lion. A lion represents a task scheduling solution, which is initialized randomly by mapping cloud tasks (cloudlets) to cloud resources (virtual machines (VMs)). For example, Fig. 1 shows a lion represents a schedule of five tasks that have been assigned randomly to three VMs. Such that, each lion will represent a random schedule solution, so the initial lion population is constructed randomly over the solution space for the LOA algorithm.

The goal of the proposed algorithm is to find the best lion (solution) that has the best fitness value. The fitness value is the makespan of that solution, which is the maximum completion time for the tasks. Moreover, each lion knows its own best solution (schedule of tasks), as well as the global best solution, which are updated progressively during optimization.

TABLE I. Parameters of LOA

Parameter	Definition of the parameters
%*N*	Percent of nomad lions
P	Number of prides
%*S*	Percent of female lions in each pride
%*R*	Roaming percent
%*Ma*	Mating percent of female lions
%*I*	Immigrate rate of female lions in each pride

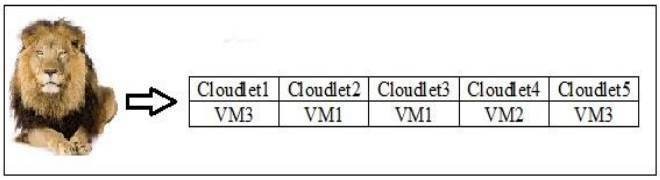

Fig. 1. A lion representation.

TABLE II. Values of LOA Parameters

Parameter	Value
%*N*	20
P	4
%*S*	80
%*R*	20
%*Ma*	30
%*I*	40

In our proposed algorithm, each lion has the following parameters:

- ***vmPositions List***: initially contains random schedule of VMs
- ***vmBestPositions List***: in order to save the best schedule for that lion.
- ***Fitness***: represents the makespan of current *vmPositions*
- ***Best Fitness***: represents the makespan of *vmBestPositions*

As mentioned previously, the initial population will be classified into residents and nomads. Table 2 shows the value of parameters used in the experiment. %20 of lions are nomads. Residents will be divided into 4 prides randomly, and in each pride, %80 of lions are females, whereas the rest are males. However, it is the reverse for nomad lions: %(100−80) of the nomads are females and the rest are males.

Moreover, each pride has its own territory. ***Territory*** is a collection of the best visited positions of the pride members. In our study, the territory of the pride is formed by the best solution (task scheduling) of each lion in the pride. This will help to save the best positions or solutions obtained over each iteration. In our proposed algorithm, the territory of the pride will consist of *vmBestPositions* of each lion in the pride.

Step 2: Each pride will do the following:

Step 2.1: Hunting

Based on LOA, some females of the pride are selected randomly for hunting. These hunters move toward the prey and encircle it, to catch it. In our proposed algorithm, this strategy will help achieve a better solution, as each hunter will update its best visited position (task schedule solution) and the global best position (solution) by moving toward the prey.

First, the selected female hunters are partitioned into three groups randomly: left wing, center, and right wing. The group with highest cumulative fitness is considered as the center while the other two groups as the two wings.

Next, a prey is generated at the center of the hunters, as follows [6]:

$$PREY = \frac{\sum hunters\ positions}{number\ of\ hunters}$$

During hunting, each hunter moves toward the prey according to its group, as follows [6]:

If a hunter belongs to the center group:

$$Hunter' = \begin{cases} rand\big((Hunter)\ PREY\big) . & (Hunter) < PREY \\ rand\big(PREY.(Hunter)\big) . & (Hunter) > PREY \end{cases}$$

If it belongs to the left or right wings:

$$Hunter' = \begin{cases} rand\big((2 \times PREY - Hunter)\ PREY\big) . \\ \quad (2 \times PREY - Hunter) < PREY \\ rand\big(PREY.(2 \times PREY - Hunter)\big) . \\ \quad (2 \times PREY - Hunter) > PREY \end{cases}$$

Where, *PREY* and *Hunter* are their current positions, and *Hunter'* is the new position of the hunter.

Throughout hunting, if the new position of the hunter is better than its previous position, which means that it has improved its own fitness, the prey will escape from the hunter, and the new position of the prey will be [6]:

$$PREY' = PREY + rand(0.1) \times PI \times (PREY - Hunter)$$

Where, *PI* is the percentage of improvement in the hunter's fitness.

Step 2.2: Remaining Females

As some females in each pride go hunting, remained females in the pride will move toward one of the positions of territory. The first step based on LOA is calculating the tournament size of the pride. This is done by first calculating the number of lions in the pride, who improved their fitness in the last iteration (Success value). Then, the tournament size for this pride is calculated as follows [6]:

$$T_j^{Size} = max\left(2.\ ceil\left(\frac{K_j\ (S)}{2}\right)\right)$$

Where, $K_j\ (S)$ is the number of lions in pride *j*, who improved their fitness in the last iteration.

Tournament size of a pride is varied in every iteration based on success value. If success value decreased, the tournament size increases and this will enhance diversity.

After that, for each remaining female in the pride, a place is selected from the pride's territory by tournament selection to move the female toward the selected place. As mentioned before, the lion's personal best visited position is updated as well as the global best position.

Step 2.3: Roaming

Roaming strategy is a strong local search and it helps our proposed algorithm to find a good solution and improve it. Based on LOA, each resident male in the pride roams within the pride's territory.

First, %*R* of the territory positions is selected randomly so that the lion will visit these selected positions. During roaming, if the new place of that male lion is better than its personal best visited position, its best visited position is updated, and that place is marked as territory. Finally, the best visited position for that lion is set as its current position and the global best position is updated if necessary.

Step 2.4: Mating

Based on LOA, %*Ma* of the female lions in the pride mate with one or more resident males, where all of them are selected randomly, to produce offspring. This will share information between genders and new offspring will inherit characteristics from both genders.

Each mating operation will produce two new offspring, according to the following equations:

$$Offspring1 = \beta \times Female\ Lion + \sum \frac{(1-\beta)}{\sum_{i=1}^{NR} S_i} \times Male\ Lion \times S_i,$$

$$Offspring2=(1-\beta)\times Female\ Lion+\sum \frac{\beta}{\sum_{i=1}^{NR} S_i}\times Male\ Lion\times S_i$$

Where, β is a randomly generated number with normal distribution with mean 0.5 and standard deviation 0.1. S_i = 1 if male i is selected for mating, otherwise it is 0. NR is the number of resident males in the pride.

One of the two offspring is considered as male and the other as female, randomly. Our proposed algorithm by mating strategy will share solutions between each other and produce new solutions that may have better fitness value.

Step 2.5: Defense

New mature resident males will fight other males in the pride. The weakest males will leave their pride and become nomads. This behavior can be simulated by merging new mature males and old males. Then, all males are sorted according to their fitness values. The weakest males are driven out of the pride and become nomads, whereas the remaining males become resident males. This strategy assists our proposed algorithm to retain powerful male lions as solutions that play an important role in LOA.

Step 3: Each lion of Nomads will do the following:

Step 3.1: Roaming

All nomad male and female lions roam and move randomly in the search space. The new positions of the nomad lions are determined as follows:

$$Lion' = \begin{cases} Lion & if \quad rand>pr \\ RAND & otherwise \end{cases},$$

$$pr=0{\cdot}1+min\left(0{\cdot}5.\frac{Nomad-BestNomad}{BestNomad}\right),$$

where, *rand* is a random number between 0 and 1, *pr* is a probability, *Nomad* is the fitness value of the current nomad, and *BestNomad* is the best fitness value of the nomad lions.

If the new place of a nomad lion is better than its personal best visited position, the best visited position of that lion is updated. The global best position is also updated, if necessary.

Step 3.2: Mating

%*Ma* of female nomads are selected randomly. Every female mates with only one male nomad, who is also selected randomly, and produces two offspring according to the equations mentioned in the previous part; one of them is male and the other is female.

Step 3.3: Defense

Nomad males attack prides randomly to try to take over a pride by fighting the male lions in the pride. If the nomad lion is strong enough, the weak male lion will be driven out of the pride and will become a nomad.

Step 4: Migration

For each pride, the maximum number of females is determined by %S of the population. For migration, %*I* of the maximum number of female lions plus the surplus females (number of female offspring) are selected randomly to migrate from their pride and become nomads. This mechanism will preserve the diversity of the population and share information among prides.

Step 5: Equilibrium

At the end of each iteration, the number of live lions must be controlled. Therefore, the female nomad lions are sorted based on their fitness values. The best females are selected and distributed to prides to fill the empty places of the migrated females.

The weakest females are removed with respect to the maximum number of female nomads %(*1-S*). Male nomads are also sorted based on their fitness values, and the lions with the least fitness will be removed with respect to the maximum number of male nomads %*S*.

Step 6: Steps 2, 3, 4, and 5 are repeated till the last iteration.

The previous steps of the LOA are summarized in the pseudo code presented below:

Pseudo code : LOA-based Task Scheduling

Input: *List of Cloudlets (Tasks), List of VMs*

Output: *the best solution for tasks allocation on VMs*

Steps:

1. ***Initialize***
   Set value of parameters Number of Lions, VMs, Iterations
   Generate random solution for each Lion
   Initiate Prides and Nomad lions
2. ***For each Pride***
   Some females are selected randomly for hunting.
   Remained females move toward best selected positions of territory.
   Each male roams in %R of territory.
   %M of females mate with one or more resident males.
   Weakest male drive out from pride and become nomad.
3. ***For each Nomad lion***
   Both male and female move randomly in the search space
   %M of females mate with only one male
   Nomad males attack prides .
4. ***For each pride***
   %I of females Immigrate from pride and become nomad.
5. ***Do***
   Each gender of nomad lion are sorted based on their fitness value
   Best females are selected and distributed to prides filling empty places
   Nomad lions with least fitness value will be removed based on the max permitted number of each gender.
6. ***If (t< Iterations)***
   Go to step 2

Return best solution.

IV. Experimental Methodology

The proposed algorithm was implemented using CloudSim. CloudSim is a simulation framework that enables us to simulate, model, and experience the cloud system [16]. The main nodes of ClouSim are Datacenters, hosts, VMs, cloudlets and brokers [17].

The Datacenter is responsible for creating the core infrastructure services that are required for the cloud. It acts as the cloud service provider and it consists of same or different configuration hosts (servers).

A host in a datacenter represents the characteristics of the physical resources such as storage server or compute server. It is characterized by host id, RAM, storage, bandwidth, processing power (MIPS) and number of processing elements (PE). Hosts are responsible for creating VMs and managing the VM migration, VM destruction, and VM provisioning. The VMs created on a host are characterized by VM id, image size, RAM, bandwidth, processing power (MIPS) and number of processing elements (PE).

Cloudlets in CloudSim represent the tasks that should be uploaded to the cloud for processing. Each cloudlet has a pre-defined length, file input size, and file output size. The broker is a mediator between the users and cloud service providers. It maps the requests of users to the appropriate provider such that it insures the achievement of the Quality of Service (QoS) requirements [18].

Scheduling of CPU resources (Processor elements PE) in CloudSim is modeled at two levels: Hos and VM. At Host level, fractions of each PE are shared among VMs running on the host. This scheduler is called *VmScheduler* and it is a parameter of the Host constructor. At VM level, each VM divides the resources received from the host and shares them to each cloudlet running on that VM. This scheduler is called *CloudletScheduler* and it is a parameter of VM constructor.

There are two default policies in both levels: *SpaceShared* and *TimeShared*. This means that *VmScheduler* and *CloudletScheduler* can be in any combinations of these two policies. For example, it is possible to use *VmSchedulerTimeShared* and *CloudletSchedulerSpaceShared* or vice versa. Also, it is possible to use the same policy for both schedulers. In the *SpaceShared* scheduling policy, only one VM/cloudlet is allowed to be executed at a given instance of a time. In *TimeShared* scheduling policy, it allows multiple VMs/cloudlets to multitask and run simultaneously within a host/VM [19].

The proposed algorithm was written in the Java programing language. It has been simulated on Intel Core i5 Processor, 2.3 GHz machine having 3 MB of L3 Cache and 4 GB of RAM running Mac OS, Eclipse IDE 4.4 and CloudSim Toolkit 3.0.3.

The cloud is simulated in CloudSim with 1 datacenter. Two hosts are created in the datacenter where each host has the following configuration: RAM = 2048 MB, storage = 1 GB, and bandwidth = 10 Gbps. Each VM has the following characteristics: RAM = 512 MB, processing power is varied between 100-1000 MIPS, bandwidth = 1 Gbps, and image size = 10 GB. The cloudlets have the characteristics: file size = 300 MB, output file = 300 MB, and the length is varied between 1000 – 2000 MI. In our experiment, both *VmScheduler* and *CloudletScheduler* utilized the *TimeShared* policy.

The testing dataset is produced randomly. Tasks (cloudlets) are generated randomly with different lengths between 1000 and 2000 million instructions (MI). VMs are also generated randomly with different capacities between 100 and 900 million instructions per second (MIPS). The cost of resources for calculating cost based on VM's specification is as follow: \$0.12, \$0.13, \$0.17, \$0.48, \$0.52, and \$0.96 per hour. The parameter set for CloudSim is shown in Table 3 [20].

TABLE III. CLOUDSIM PARAMETERS

Entity	**Parameter**	**Values**
Cloudlet	No of cloudlets	50 – 500
	length	1000-2000
Virtual Machine	No of VMs	15
	RAM	512 MB
	MIPS	100-1000
	Size	10000
	bandwidth	1000
	Policy type	Time Shared
	VMM	Xen
	Operating System	Linux
	No of CPUs	1
Host	No of Hosts	2
	RAM	2048MB
	Storage	1000000
	Bandwidth	10000
	Policy type	Time Shared
Data Center	No of Data Center	2

V. EXPERIMENTAL METRICS

The performance metrics for the proposed task scheduling is based on makespan, cost, average utilization, and degree of imbalance. The following describes these performance metrics [20], [21]:

A. Makespan

Makespan determines the maximum completion time by indicating the finishing time of last task. Minimizing the makespan is the most popular optimization criteria for task scheduling. It can be calculated using the following equation:

$$Makespan = max_{taski\ i}\ (Fn_{Time})$$

Where, Fn_{Time} shows the finishing time of task i.

B. Cost

Cost means the total amount of payment to cloud provider against the resource utilization. The main purpose for cloud providers is to increase revenue and profit while cloud users aim to reduce the cost with efficient utilization. Cost is measured as follows:

$$Cost = \sum_{resource\ i} (C_i * T_i)$$

Where, C_i represents the cost of VM $_i$ per time unit and T_i represents the time for which VM $_i$ is utilized.

C. Average Utilization

Maximizing resource utilization is another important criteria for cloud providers by keeping resources as busy as possible to earn maximum profit. The following equation is used to measure the average utilization:

$$Average\ Utilization = \frac{\sum_{i=1}^{n} Execution\ time\ of\ resource\ i}{Makespan * n}$$

Where, n is the number of resources.

D. Degree of Imbalance

Degree of imbalance (DI) means the amount load distribution among the VMs regarding to their execution capacity. The small value of DI shows that the load of the system is more balanced. It is computed by:

$$DI = \frac{T_{max} - T_{min}}{T_{avg}}$$

Where, T_{max}, T_{min}, and T_{avg} are the maximum, minimum, and average execution time of all VMs.

VI. Results and Evaluation

The results of the proposed algorithm are compared with scheduling algorithms that based on two popular metaheuristic algorithms: PSO and GA [22], [23]. In all cases, the population size is set to 100 and the number of iterations is 100. These algorithms are compared with each other based on makespan, cost, average resource utilization, and degree of imbalance.

Fig. 2 shows the comparison of makespan between LOA, PSO, and GA. The x-axis denotes the number of cloudlets and the y-axis denotes the makespan. When the numbers of cloudlets are less, the makespan of the three algorithms are convergent. However, LOA produces much better makespan time when the number of cloudlets increases.

In Fig. 3, the comparison of cost is shown between LOA, PSO, and GA. The x-axis indicates the number of tasks and the y-axis indicates the cost per hour of the execution of tasks. The outcomes show the cost of LOA is between PSO and GA although the difference is not great.

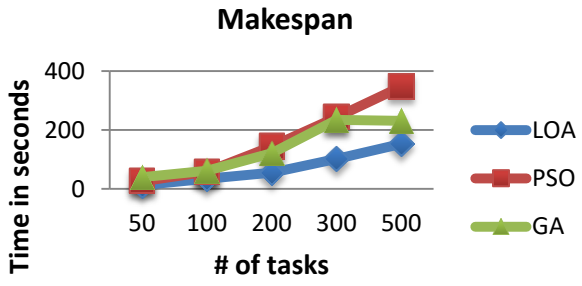

Fig. 2. Makespan results.

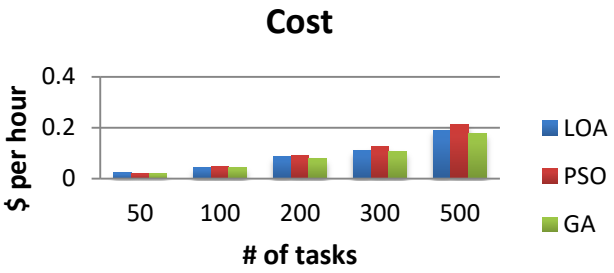

Fig. 3. Cost results.

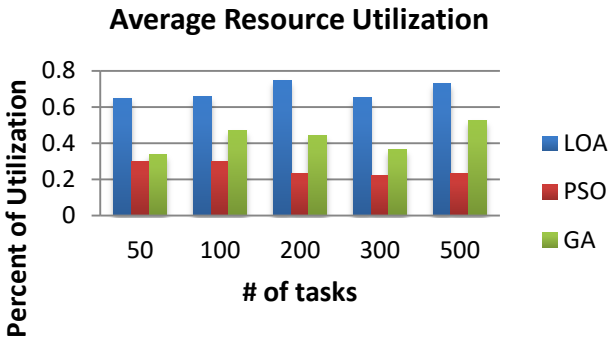

Fig. 4. Average resource utilization.

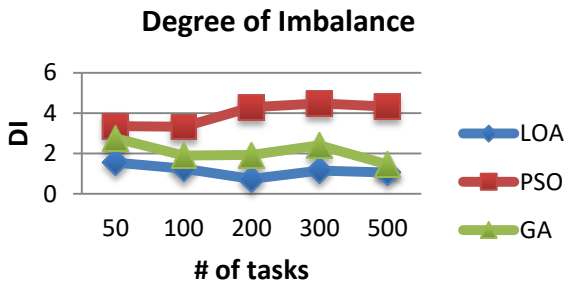

Fig. 5. Degree of imbalance.

Fig. 4 shows the comparison of average resource utilization between LOA, PSO, and GA. It is obvious that LOA provides a very high utilization of resources when compared with PSO and GA.

Fig. 5 explains the comparison of degree of imbalance between LOA, PSO, and GA. The x-axis signifies the number of cloudlets and the y-axis signifies the degree of imbalance. The comparison result tells that LOA produces much better degree of imbalance than PSO and GA.

It is obvious that the proposed task scheduling algorithm based on LOA provides a high performance and much better results than the other two algorithms. It can solve the optimization problems in task scheduling with high performance because it searches for the optimal solution using different strategies. Each solution "lion" has a specific gender and is classified as a resident or nomad, and all of them have their own strategies to search for the optimal solution, as explained previously.

VII. Conclusion

Various metaheuristic optimization algorithms have been used to develop task scheduling techniques for cloud computing. In this paper, a new cloud task-scheduling algorithm was proposed, based on the concept of LOA, which is a newly constructed algorithm based on the lifestyle of lions. The performance of the proposed algorithm was compared with that of the PSO and GA metaheuristic algorithms. It provided an outstanding result in minimizing the makespan and degree of imbalance. Also, it produced high utilization of resources.

In future work, we aim to enhance the proposed algorithm to decrease the cost of executing the tasks on cloud resources, using cloud pricing models.

Acknowledgment

This research project was supported by a grant from the Deanship of Graduate Studies, King Saud University, Saudi Arabia.

References

[1] D. P. Chandrashekar, "Robust and fault-tolerant scheduling for scientific workflows in cloud computing environments." University of Melbourne, Australia, 2015.

[2] N. Almezeini and A. Hafez, "An enhanced workflow scheduling algorithm in cloud computing," in CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, 2016, vol. 2.

[3] M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: a comprehensive analysis," J. Netw. Comput. Appl., vol. 66, pp. 64–82, 2016.

[4] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," PLoS One, vol. 12, no. 5, p. e0176321, May 2017.

[5] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," Egypt. informatics J., vol. 16, no. 3, pp. 275–295, 2015.

[6] M. Yazdani and F. Jolai, "Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm," J. Comput. Des. Eng., vol. 3, no. 1, pp. 24–36, 2016.

[7] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "An Appraisal of Meta-Heuristic Resource Allocation Techniques for IaaS Cloud," Indian J. Sci. Technol. Vol. 9, Issue 4, January 2016, Jan. 2016.

[8] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," Mach. Learn., vol. 3, no. 2, pp. 95–99, 1988.

[9] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing," Procedia Technol., vol. 10, pp. 340–347, 2013.

[10] P. Kumar and A. Verma, "Scheduling Using Improved Genetic Algorithm in Cloud Computing for Independent Tasks," in Proceedings of the International Conference on Advances in Computing, Communications and Informatics, 2012, pp. 137–142.

[11] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on, 1995, pp. 39–43.

[12] A. I. Awad, N. A. El-Hefnawy, and H. M. Abdel_kader, "Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments," Procedia Comput. Sci., vol. 65, pp. 920–929, 2015.

[13] X. Wen, M. Huang, and J. Shi, "Study on Resources Scheduling Based on ACO Allgorithm and PSO Algorithm in Cloud Computing," 2012 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science. pp. 219–222, 2012.

[14] B. Wang, X. Jin, and B. Cheng, "Lion pride optimizer: An optimization algorithm inspired by lion pride behavior," Sci. China Inf. Sci., vol. 55, no. 10, pp. 2369–2389, 2012.

[15] B. R. Rajakumar, "The Lion's Algorithm: A New Nature-Inspired Search Algorithm," Procedia Technol., vol. 6, pp. 126–135, 2012.

[16] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," 2009 International Conference on High Performance Computing & Simulation. pp. 1–11, 2009.

[17] T. Goyal, A. Singh, and A. Agrawal, "Cloudsim: simulator for cloud computing infrastructure and modeling," Procedia Eng., vol. 38, no. Supplement C, pp. 3566–3572, 2012.

[18] S. Mehmi, H. K. Verma, and A. L. Sangal, "Simulation modeling of cloud computing for smart grid using CloudSim," J. Electr. Syst. Inf. Technol., vol. 4, no. 1, pp. 159–172, 2017.

[19] H. S. Sidhu, "Comparative analysis of scheduling algorithms of Cloudsim in cloud computing," 2014.

[20] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," PLoS One, vol. 12, no. 5, p. e0176321, 2017.

[21] A. Keshk, "Cloud Computing Online Scheduling," IOSR J. Eng., vol. 4, no. 3, pp. 07–17, 2014.

[22] E. Ibrahim, N. A. El-Bahnasawy, and F. A. Omara, "Task Scheduling Algorithm in Cloud Computing Environment Based on Cloud Pricing Models," in Computer Applications & Research (WSCAR), 2016 World Symposium on, 2016, pp. 65–71.

[23] E. Ibrahim, N. A. El-Bahnasawy, and F. A. Omara, "Load Balancing Scheduling Algorithm in Cloud Computing System with Cloud Pricing Comparative Study."