Received: October 2, 2023. Revised: December 19, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Multi-objective Task Scheduling Algorithm in Cloud Computing Using Improved Squirrel Search Algorithm

Henning Titi Ciptaningtyas[1] Ary Mazharuddin Shiddiqi[1] Diana Purwitasari[1]
Fuad Dary Rosyadi[1] Muhammad Nur Fauzan[1]

[1]*Faculty of Electrical and Intelligent Information Technology (FTEIC),*
Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia
* Corresponding author's Email: ary.shiddiqi@if.its.ac.id

Abstract: The optimization of resource allocation methods in cloud computing environments is essential for enhancing system performance and efficiency. The squirrel search algorithm (SSA) is a metaheuristic algorithm based on swarm intelligence, designed to address optimization challenges. One issue that might arises in the SSA is premature convergence. To address this concern, we propose improving the performance of SSA by integrating it with the opposition based learning (OBL) method. The primary objectives of this method encompass the optimization of makespan, throughput, and resource utilization. The improved SSA algorithm was subjected to a comparative analysis with the genetic algorithm (GA), particle swarm optimization (PSO), and the original SSA. The experiment was performed using the CloudSIM simulator, utilizing three different datasets: the SDSC dataset, a simple random dataset, and a stratified random dataset. The factors under consideration for evaluation encompass makespan, average start time, average finish time, average execution time, total wait time, total scheduling length, throughput, resource utilization, total energy consumption, and imbalance degree. From the experimental results, the improved SSA exhibits superiority over other algorithms in the optimization of makespan on a simple random dataset with an average value of 8.333, as well as minimizing overall energy consumption on the san diego super computer (SDSC) blue horizon dataset which has an average value of 449 kWH. The improved SSA exhibits a gradual increase in the experimental results, rendering the outcomes more foreseeable for a greater number of tasks.

Keywords: Cloud computing, Opposition based learning, Optimization, Squirrel search algorithm, Task scheduling.

1. Introduction

Optimizing cloud provisioning and task scheduling is a complex problem that classical methods struggle to solve with the required accuracy and speed [1]. Metaheuristic algorithms provide near-optimal solutions in a reasonable timeframe [2, 3]. Nature-inspired optimization algorithms imitate biological or physical mechanisms and principles and can be grouped into three categories: evolutionary algorithms (EA), swarm intelligence (SI), and physics-based (PB) algorithms [4]. Swarm intelligence algorithms, which collect and utilize complete information from the search space during optimization, are particularly effective.

The squirrel search algorithm (SSA) is a swarm intelligence algorithm introduced by Jain in 2019 to solve optimization problems, including real-time heat flow experiment case studies. The SSA method shows promising results in accuracy and efficiency for optimizing cloud provisioning and task scheduling problems [5, 6].

Comparative statistical analysis has shown that SSA outperforms other methods to reach optimal global solutions with better convergence behavior. However, premature convergence can be a problem for optimization problems when a population converges too early, leading to suboptimal results [7, 8]. To address this issue, the opposition based learning (OBL) method replaces the candidate solution with the lowest fitness with its opposite.

Despite the advanced research in cloud provisioning, further investigation remains open on how optimization techniques impact the performance and convergence of cloud provisioning algorithms. Additionally, the scalability and computational efficiency of the optimization algorithms to handle increasing workloads in dynamic cloud environments is another issue to consider.

In order to enhance the efficiency of cloud task scheduling, we elaborate the opposition-based learning (OBL) technique with the nature-based optimization method SSA (squirrel search algorithm) to optimize multiple objectives. We use a simulation environment called CloudSIM 4.0 and experimental data from synthetic datasets (simple random dataset and stratified random dataset) and real-world dataset (san diego super computer (SDSC) blue horizon logs [9]). The cloud tasks used in this experiment were independent tasks that must be completed within deadline constraints. We focus on optimizing makespan, throughput, and resource utilization, which are important objectives for cloud consumers and providers. We prioritize makespan as the optimization metric because speed is critical to meeting service level agreements (SLAs) for cloud consumers. Additionally, we consider throughput, which measures the number of job completions per unit of time, as an important criterion for cloud consumers. Finally, we also aim to optimize resource utilization, as many data centers suffer from low usage levels of cloud resources, resulting in energy and effort waste.

The improved SSA underwent a comparative examination alongside the genetic algorithm (GA), particle swarm optimization (PSO), and the original SSA. The criteria being considered for evaluation are makespan, average start time, average end time, average execution time, total wait time, total scheduling length, throughput, resource utilization, total energy consumption, and imbalance degree.

The rest of the paper presents the literature review in section related work, the proposed method in section proposed methodology, the experimental results in section experimental result, and finally, concludes our study and future research in section conclusion.

2. Related work

The process of scheduling task inside a cloud platform is a multifaceted issue that entails the fulfillment of several objectives to meet the needs of both cloud consumers and providers. Objective functions are typically categorized into groups based on time efficiency, resource efficiency, and cost efficiency. Time efficiency encompasses several factors such as Makespan, wait time, scheduling duration, start time, finish time, and execution time. Resource efficiency encompasses metrics such as throughput, resource utilization, energy consumption, and imbalance degree. Cost efficiency pertains to the expenses borne by cloud service providers for operating their cloud services or the expenses paid by cloud service customers for utilizing cloud services.

Researchers often employ a hybrid algorithm to obtain optimal solutions, combining two or more task-scheduling algorithms. The hybrid algorithm typically consists of several phases, each with a specific objective. During the initial phase, the algorithm employs specific criteria to identify and discard suboptimal solutions. The subsequent phase involves the identification of solutions that are close to optimal, as determined by various performance metrics. Swarm optimization is particularly well-suited for this phase due to its ability to efficiently locate the global minima of the fitness function.

In recent research, a range of optimization methods have been employed using the CloudSIM framework to enhance the effectiveness of cloud task scheduling. Table 1 displays the findings and drawbacks of some cloud task scheduling algorithms.

Ciptaningtyas conducted a recent literature study that examines hybrid implementations of GA combined with other algorithms for cloud task scheduling [10]. However, these studies can become trapped in local optima. PSO is a widely used method in swarm intelligence. Singh conducted a literature study that highlights various instances of hybrid PSO methods employed in the context of cloud task scheduling [11]. PSO may also become trapped in local optima and fail to investigate the solution space sufficiently. PSO may also be computationally unfeasible when applied to large-scale cloud scheduling problems, which could restrict its application in real-world cloud environments. Swarm intelligence algorithms may encounter challenges like as local minima and early convergence, which can impede their overall efficacy. To address this issue, researchers have incorporated OBL into these algorithms [12-14].

The method employed in the study survey fails to account for variations in cloud task duration, hence diminishing its effectiveness in predicting fluctuations in cloud work length. Predicting the performance of the algorithm for cloud task scheduling becomes challenging when dealing with a substantial number of tasks and varied task lengths.

This study combines the nature-based optimization method SSA (squirrel search algorithm) with opposition-based learning (OBL) in order to optimize three objective functions (makespan, resource utilization, and throughput) while preventing premature convergence. Prior research

Table 1. Comparison of task scheduling algorithms

Algorithm	Objective Function	Methods and Finding	Drawbacks
Non-dominated Sorting Genetic Algorithm (NSGA) III [15]	Finished time and energy consumption	Select efficient resources that can lower the response time, cost, and power consumption	The computational requirements of NSGA-III can increase rapidly, particularly as the complexity of the optimization problem rises.
PSO [16]	makespan, resource utilization, and task response time	Resource and deadline-aware load balancing algorithm	In large-scale cloud systems with many tasks and resources, the search space is vast and the algorithm may struggle to explore it.
CR-PSO [17]	Makespan and cost	Chemical reaction optimization in the first stage and PSO in the second phase to optimize time complexity, makespan, execution time, and cost	The optimal settings may vary across different stages and cloud task scheduling scenarios
ANN-PSO [18]	Makespan, energy consumption, and the average start time	Provisioning technique using artificial neural network (ANN) and PSO	The efficacy of the system is significantly dependent on the proper tuning of parameters; therefore, this may restrict its applicability in practical cloud task scheduling scenarios.
Cuckoo Search Algorithm + OBL [12]	Makespan and cost	Combined the brood parasitic behavior of cuckoos with OBL. It has good improvement compared to GA, PSO, and Improved Differential Evolution Algorithm (IDEA).	The quantity of cloud tasks is limited and evenly distributed. Therefore, the effects of implementing this technique on a large-scale cloud task with varied lengths are unknown.
Lion + OBL [13]	Makespan and cost	Combined wild lions' behavior and social organization with OBL. It has good improvement compared to GA, PSO, and GWO.	The heightened complexity can hinder the scalability and efficiency of the algorithm, particularly in large-scale cloud environments where timely task scheduling is crucial.
Sunflower optimization + OBL [14]	Throughput, cost, energy and makespan	Combined OBL and sunflower optimization, which mimics the movement of sunflowers toward the sun	It has limited versatility and adaptability in accommodating diverse cloud environments with various characteristics.

has overlooked examining the impact of the method on the other metric, instead concentrating solely on the enhancement of their objective functions. Furthermore, most of the research failed to consider variations in the duration and quantity of cloud tasks. In order to fill these shortcomings, we have put forth a hybrid squirrel search algorithm and opposition based learning (SSO-OBL) that considers not only the objective functions, but also evaluates the impact on other metrics related to time efficiency and resource efficiency. In addition, our model is applied to three different datasets (simple random dataset, stratified random dataset, and SDSC dataset) that differ in terms of cloud task count and cloud task length. By considering these factors, our proposed model could enhance time efficiency, ensuring resource efficiency and improving the overall performance of the system in the different length of incoming cloud task.

3. Proposed methodology

The squirrel search algorithm (SSA) is a nature-inspired optimization algorithm. It is inspired by the southern flying squirrel (*Glaucomys volans*) and northern flying squirrel (*Glaucomys sabrinus*), an arboreal and nocturnal type of rodent that remarkably adapted for gliding locomotion. The flying squirrel (FS) does not fly. FS locomotion involves glides, which allow small mammals to cover vast distances swiftly and efficiently with minimal energy expenditure. It has patagia (plural form of patagium), a parachute-like membrane that stretches from the wrist to the ankle made of fur. Flying squirrels use patagia to glide and their tail to steer aerodynamically among the trees and can modify lift and drag. The recorded flights of flying squirrels are up to 300 feet [5].

FS uses dynamic foraging behavior to exploit food resources optimally. In autumn, they glide and explore forest areas to eat acorns on the spot, which are available in abundance, while storing other nuts (hickory nuts). During the winter season, FS engage in the consumption of hickory nuts in order to meet their increased nutritional requirements resulting from the colder temperatures. Hickory nuts are eaten instantly after encountering during foraging and are also taken out from reserve food stores to reduce the costly foraging trips. The reduction of greenery in forested areas during the winter season heightens the vulnerability of FS to predation by owls, raccoons, and snakes. Consequently, FS exhibit decreased levels of activity during this period, yet FS does not enter a state of hibernation. Based on dietary needs, the FS selectively eats some nuts and stores others to utilize both available mast nuts optimally [5].

The simplification of the mathematical model is based on the following pre-suppositions: The forest contains n number of FS, and each FS is presumed to be on a single tree; Each FS exhibits dynamic foraging behavior individually by searching for food and utilizing available food resources optimally; The forest contains three distinct varieties of trees: normal tree (NT), oak tree (acorn nuts food source / AT), and hickory tree (hickory nuts food source / HT).

We use squirrel search algorithm to optimize makespan, throughput, and resource utilization within deadline constraints and use opposition-based learning (OBL) to avoid premature convergence. Then, we compare the the result with GA, PSO and original SSA. The proposed algorithm can be seen in Fig. 1 while the notation definition can be seen in Table 2.

Table 2. Notation in the SSA-OBL

Notation	Description
FS	Flying squirrel
FS_L	Lower bound of flying squirrel
FS_U	Upper bound of flying squirrel
$FS_{i,j}$	The j th dimension of i th flying squirrel
U(0, 1)	A uniformly distributed random number in the range [0, 1]
P_{dp}	Predator presence probability, considered to be 0.1 in all cases.
FS_{at}	Flying squirrel at acorn tree
FS_{ht}	Flying squirrel at hickory tree
FS_{nt}	Flying squirrel at normal tree
d_g	Random gliding distance
G_c	Gliding constant = 1.9, the balance between exploration and exploitation, which is obtained after rigorous analysis.
R_1, R_2, R_3	Random number in the range [0, 1]
n_1	The number of FS that inhabit AT and are in the process of relocating towards a HT
n_2	The number of FS observed on NT and their subsequent movement towards AT
n_3	The number of FS observed on NT and their subsequent movement towards HT
Sc	Seasonal constant
S_{min}	Minimum value of Sc

4. Simulation setup

We conducted the experiments using CloudSim, an Open-Source framework specifically designed for simulating cloud computing services. The experiments involved 54 Virtual Machines distributed across 18 Hosts in 6 data centers. To manage task scheduling and virtual machine (VM) allocation, we connected each data center to a data center broker.

This entity acts as the central coordinator, responsible for managing the VM list (which is a list of existing virtual machines and their status), the task list (which is a list of existing tasks and their status), and the end-users who assign tasks and receive the processed output. The data center broker establishes connections with these components to ensure the seamless execution of the experiment. The infrastructure of the data center is illustrated in Fig. 2, and Fig. 3.

Proposed algorithm: Modified squirrel search algorithm + opposition based learning

```
Begin:
Specify input variables.
Produce locations at random for n number of FS
    using $FS_i = FS_L + U(0,1) \times (FS_U) - FS_L$
Generate the opposite location for each flying
    squirrel using $\breve{x}_i = a_i + b_i - x_i$
Get VM resource utilization.
Mark low, medium, and high resource utilization
    rate VM as a normal tree, acorn tree, and hickory
    tree, respectively
Evaluate the fitness of every FS location.
Arrange the locations of FS in ascending order
    based on their fitness value.
Declares the type of tree the FS is on
A random selection will be made among the FS
    located on regular trees, with some being
    directed towards hickory nut trees and the others
    will be directed into acorn nut trees.

while (iteration < 10)
    Replace the lowest fitness with the opposite
    location
    For t = 1 to n1
        if $R_1 \geq P_{dp}$
            $FS_{at}^{t+1} = FS_{at}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{at}^{t})$
        else
            $FS_{at}^{t+1}$ = a random search space position
        end
    end

    For t = 1 to n2
        if $R_2 \geq P_{dp}$
            $FS_{nt}^{t+1} = FS_{nt}^{t} + d_g \times G_c \times (FS_{at}^{t} - FS_{nt}^{t})$
        else
            $FS_{nt}^{t+1}$ = a random search space position
        end
    end

    For t = 1 to n3
        if $R_3 \geq P_{dp}$
            $FS_{nt}^{t+1} = FS_{nt}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{nt}^{t})$
        else
            $FS_{nt}^{t+1}$ = a random search space position
        end
    end

    $S_c^t = \sqrt{\sum_{k=1}^{d} \left(FS_{at,k}^{t} - FS_{ht,k}^{t}\right)^2}$

    Update $S_{min}$ value using $S_{min} = \frac{10E^{-6}}{360^{t/\left(\frac{tm}{2.5}\right)}}$
    if $(S_c^t < S_{min})$
        Relocate FS at random using $FS_{nt}^{new} = FS_L +$
            $Lévy(n) \times (FS_U - FS_L)$
    end
end
The placement of FS on HT represents the ultimate
    optimal solution
End
```

Figure. 1 Pseudocode of SSA-OBL

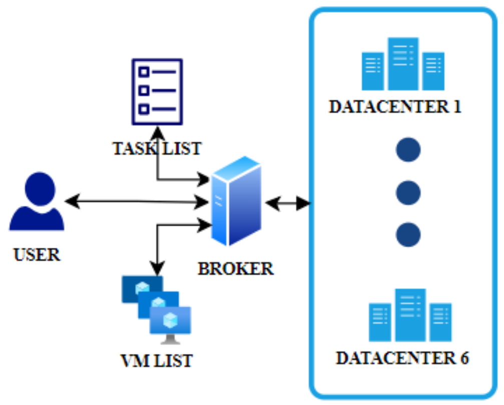

Figure. 2 Network diagram of cloud simulation

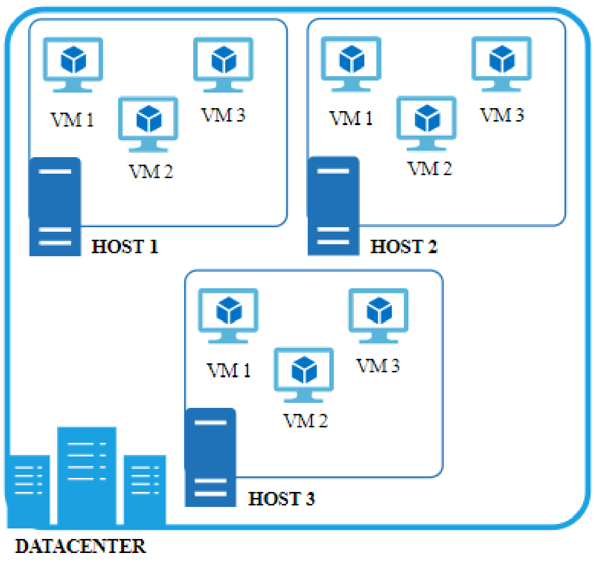

Figure. 3 Datacenter diagram

There are three datasets used in this research: simple random dataset, stratified random dataset, and SDSC dataset. The process of simple random data sampling entails the production of a set of 10,000 random integers that fall within a predetermined range. The aforementioned outcome can be attained by employing the *randbetween* function within a spreadsheet, wherein the lower boundary is set at 10,000 and the upper boundary is set at 40,000. In order to construct a stratified random data sampling, a total of 100 classes were utilized, taking into consideration the data distribution within the SDSC dataset. The range of random numbers generated spans from 0 to 8,800,000. The cleaned version of the SDSC dataset consists of 7,395 data.

5. Experimental results and discussion

The cloud tasks number on simple random and stratified random datasets ranges from 1,000 to 10,000, increasing by increments of 1,000 for each experiment. The research was conducted ten times on each dataset to obtain the average data for each dataset. After collecting all the data and calculating the parameters, the data was used to generate a graphical representation. The graphical representations provide a comparative analysis of the results achieved through the utilization of the genetic technique (GA), particle swarm optimization (PSO), squirrel search algorithm (SSA), and our proposed algorithm (SSA-OBL).

5.1 Makespan

Makespan is used to denote the time interval between the commencement and the conclusion of a task. A lower makespan is indicative of superior performance. The Makespan formula can be seen in (1), where i is the task number, and F_i shows the Finishing time of task I [10].

$$Makespan = \max_{i \in tasks}\{F_i\} \quad (1)$$

The makespan values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 127,357; 127,357; 101,699; and 104,852 respectively. Makespan for the simple random dataset and stratified random dataset is depicted in Fig. 4 and Fig. 5. The best makespan on simple and stratified random dataset are using SSA-OBL, while the best one in SDSC dataset is using SSA. The SSA-OBL algorithm demonstrates a high level of effectiveness in the efficient execution of cloud activities, as seen by its ability to achieve minimal makespan and maintain a consistent rate of increase.

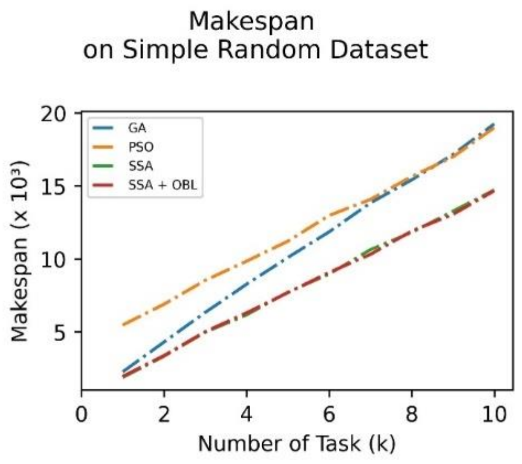

Figure. 4 Makespan (simple random dataset)

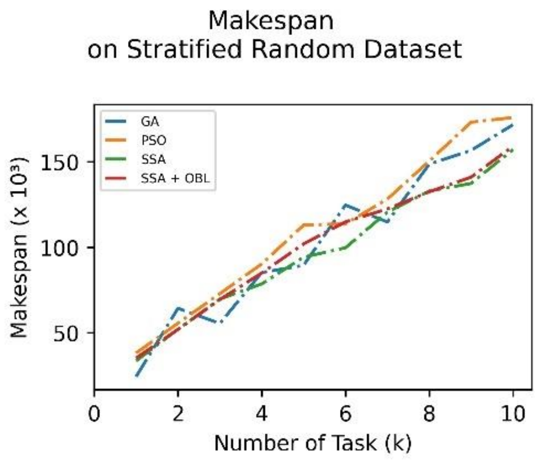

Figure. 5 Makespan (stratified random dataset)

5.2 Average start time

The average start time refers to the mean duration required to initiate the processing of all tasks. It is advantageous to act or initiate actions at an earlier point in time. The average start time formula can be seen in (2), where R_i is ith resource and nR is the number of resources [6].

$$Avg\ Start\ Time = \frac{\sum_{i=1}^{n} Start\ Time\ of\ Ri}{nR} \quad (2)$$

The average start time values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 28,987; 26,175; 25,210; and 25,554 respectively. Fig. 6 and Fig. 7 illustrate the makespan observed in both simple and stratified random dataset.

The best average start time on simple random dataset, stratified random dataset, and SDSC dataset are using SSA, GA, and SSA respectively. The performance of SSA-OBL is suboptimal in relation to this parameter due to its higher complexity compared to the other three algorithms.

5.3 Average finish time

The average finish time refers to the mean duration needed to successfully complete an entire task. A more favorable outcome is achieved when the finish time is earlier. The average finish time formula can be seen in Eq. (3), where R_i is ith resource and nR is the number of resources [6].

$$Avg\ Finish\ Time = \frac{\sum_{i=1}^{n} Finish\ Time\ of\ Ri}{nR} \quad (3)$$

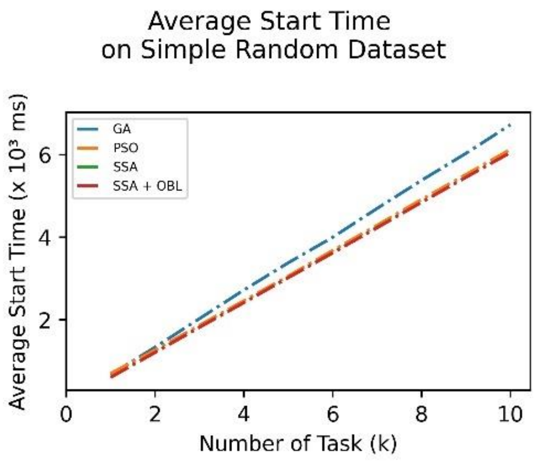

Figure. 6 Average start time (simple random dataset)

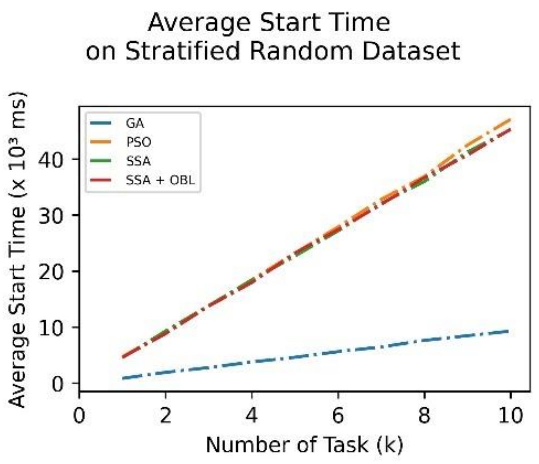

Figure. 7 Average start time (stratified random dataset)

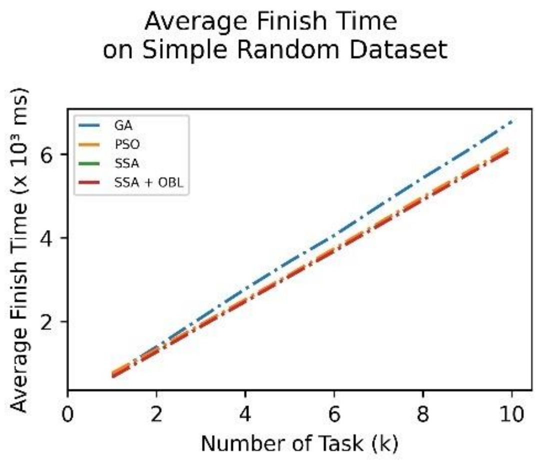

Figure. 8 Average finish time (simple random dataset)

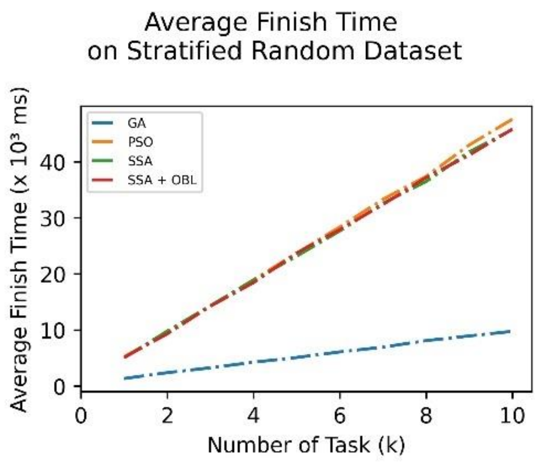

Figure. 9 Average finish time (stratified random dataset)

The average finish time values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 29,325; 26,555; 25,580; and 25,917 respectively. Average finish time for the simple random dataset and stratified random dataset is depicted in Fig. 8 and Fig. 9.

The best average finish time on simple random dataset, stratified random dataset, and SDSC dataset are using SSA, GA, and SSA respectively. It exhibits unsatisfactory performance in the case of SSA-OBL, mostly due to its comparatively larger complexity when compared to the other three algorithms.

5.4 Average execution time

The average execution time refers to the mean duration required for the completion of a certain task. The average execution time should be as short as possible. The average execution time formula can be seen in (4), where T_i is the time completion of ith task and nT is the number of tasks [6].

$$Average\ Execution\ Time = \frac{\sum_{i=1}^{n} Ti}{nT} \quad (4)$$

The average execution time values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 337; 379; 369 and 372 respectively. Figure 10 and Figure 11 illustrate the average execution time observed in both simple random dataset and stratified random dataset.

The best average execution time on simple random dataset, stratified random dataset, and SDSC dataset is using GA. The reason for the low execution time of the GA is attributed to its minimal complexity.

5.5 Total wait time

The total wait time refers to the duration required for the initiation of processing the initial task, also known as the delay. The optimal outcome is achieved by minimizing the total wait time. If there is no delay, then the total wait time is zero.

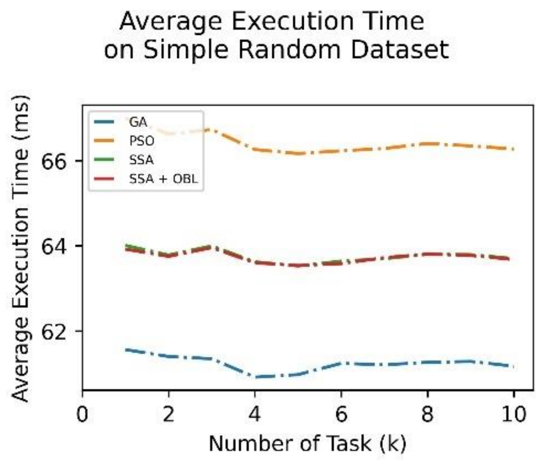

Figure. 10 Average execution time (simple random dataset)

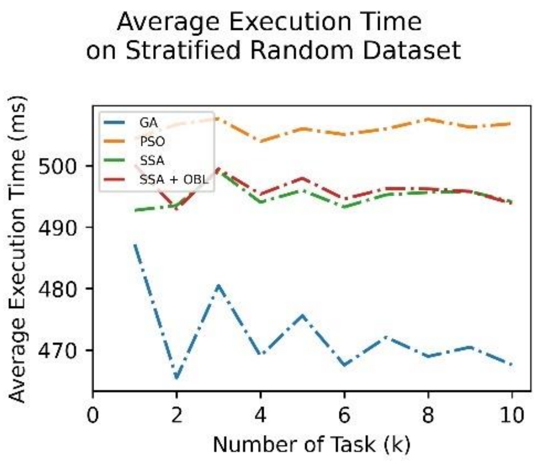

Figure. 11 Average execution time (stratified random dataset)

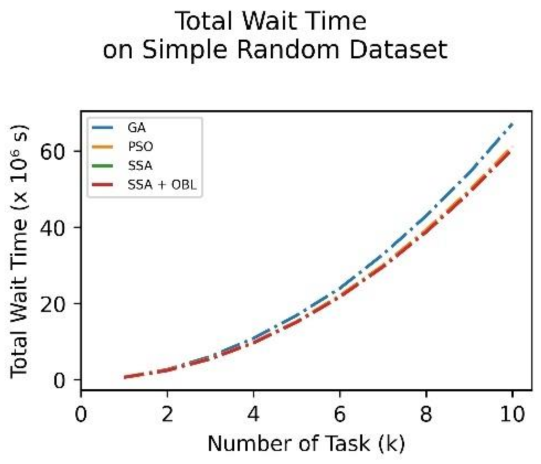

Figure. 12 Total wait time (simple random dataset)

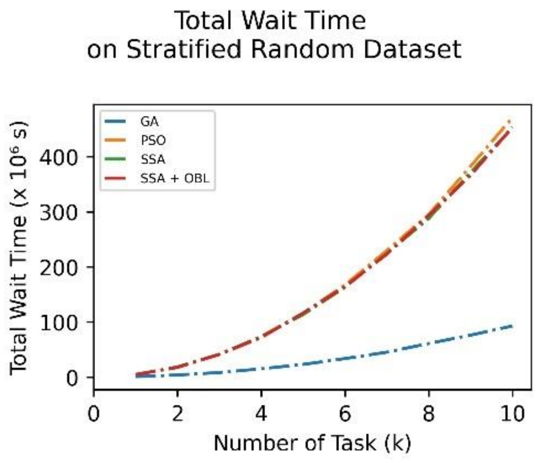

Figure. 13 Total wait time (stratified random dataset)

The total wait time values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 214,359 s; 193,565 s; 186,425 s; and 188,896 s respectively. Total wait time for the simple random dataset and stratified random dataset is depicted in Fig. 12 and Fig. 13.

The smallest total wait time on simple random dataset, stratified random dataset, and SDSC dataset are using SSA, GA, and SSA respectively. The parameter performs poorly in SSA-OBL due to its higher complexity than the other three algorithms.

5.6 Total scheduling length

The scheduling length refers to the duration required to complete the simulation from its initiation to its conclusion. A shorter scheduling duration is preferable. Total scheduling length is the the sum of scheduling time and makespan [19].

The scheduling length values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 214,487 s; 193.699 s; 186.527 s; and 189.000 s respectively. Fig. 14 and Fig. 15 illustrate the scheduling length observed in both simple random dataset and stratified random dataset.

The best total scheduling length on simple random dataset is using SSA, on stratified random dataset is using GA, while the best one in SDSC dataset is using SSA. Because SSA-OBL's parameter is more complex than the other three algorithms, it has low performance compared to them.

5.7 Throughput

Throughput refers to the quantification of the number of tasks that are successfully accomplished within a given timeframe. A larger throughput is indicative of superior performance. The throughput formula can be seen in (5), where nT is the number of tasks [10].

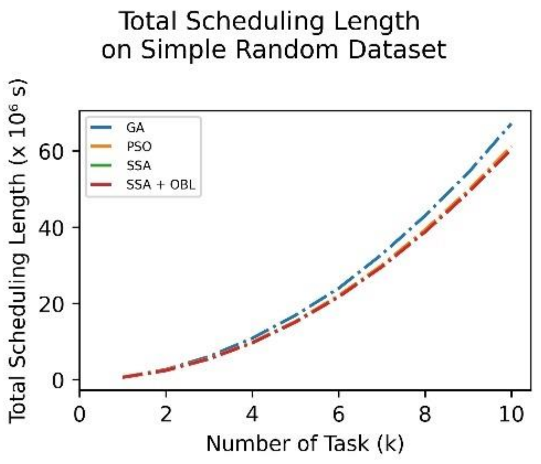

Figure. 14 Total scheduling length (simple random dataset)

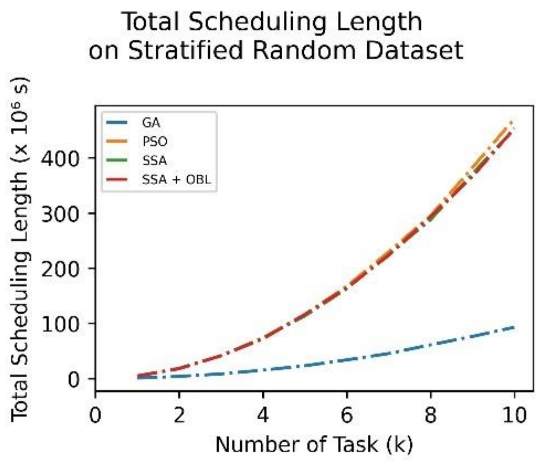

Figure. 15 Total scheduling length (stratified random dataset)

$$Throughput = \frac{nT}{Makespan} \quad (5)$$

The throughput values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 0,06; 0,06; 0,07; and 0,07 respectively. Throughput for the simple random dataset and stratified random dataset is depicted in Fig. 16 and Fig. 17.

SSA provides the highest throughput on all three datasets tested, while SSA-OBL has a slightly lower throughput than SSA. But SSA-OBL has a steady increase compared to the other three algorithms.

5.8 Total energy consumption

Total energy consumption refers to the aggregate amount of energy required to complete all assigned tasks. Minimizing energy use is preferable.

The total energy consumption values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are

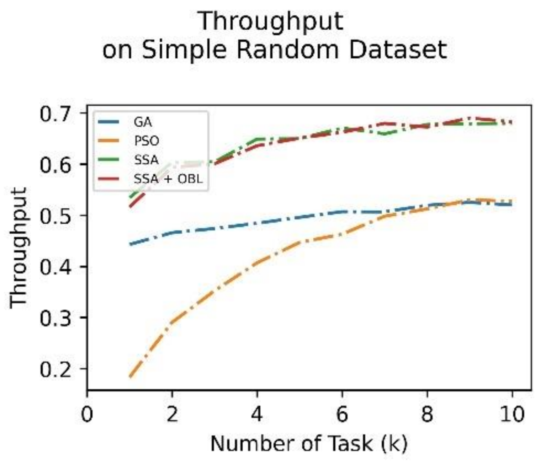

Figure. 16 Throughput (simple random dataset)

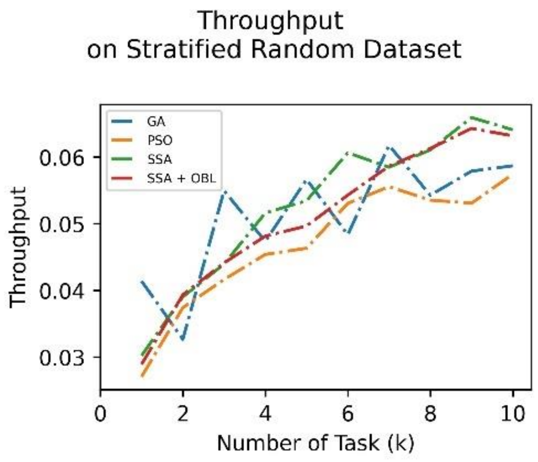

Figure. 17 Throughput (stratified random dataset)

584,43; 505,07; 452,46; and 449,99 respectively. Total energy consumption for the simple random dataset and stratified random dataset is depicted in Fig. 20 and Fig. 21.

The SSA-OBL exhibits the least energy consumption when used to the SDSC dataset, but the SSA demonstrates the lowest energy consumption when employed on the remaining two datasets. The resource utilization for SSA-OBL exhibits a marginal increase compared to that of SSA.

5.9 Resource utilization

Resource utilization refers to the measurement of the rate at which resources are utilized in order to complete all assigned tasks. The bigger resource utilization is the better. Resource utilization is calculated as in (8) where RU is resource utilization.

$$RU = \frac{\sum_{i=1}^{n} Resource\ i\ finishing\ time}{Makespan*n} \quad (8)$$

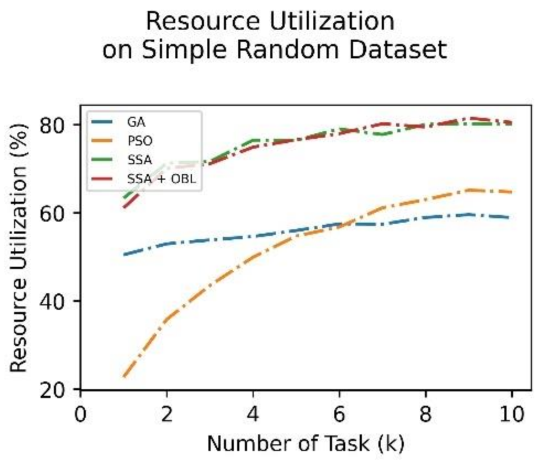

Figure. 18 Resource utilization (simple random dataset)

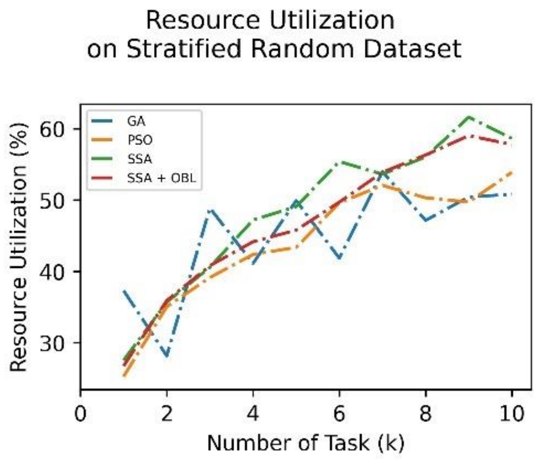

Figure. 19 Resource utilization (stratified random dataset)

The resource utilization values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 36,38; 39,97; 50,96; and 49,20 respectively. Figure 18 and Figure 19 illustrate the resource utilization observed in both simple random dataset and stratified random dataset.

SSA demonstrated the best level of resource utilization across all three tested datasets. The SSA-OBL algorithm exhibits marginally reduced utilization of resources in comparison to SSA, while presenting a consistent upward trend relative to the remaining three methods.

5.10 Total energy consumption

Total energy consumption refers to the aggregate amount of energy required to complete all assigned tasks. Minimizing energy use is preferable. The total energy consumption values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 584,43; 505,07; 452,46; and 449,99 respectively. Total energy consumption for the simple random dataset and stratified random dataset is depicted in Fig. 20 and Fig. 21.

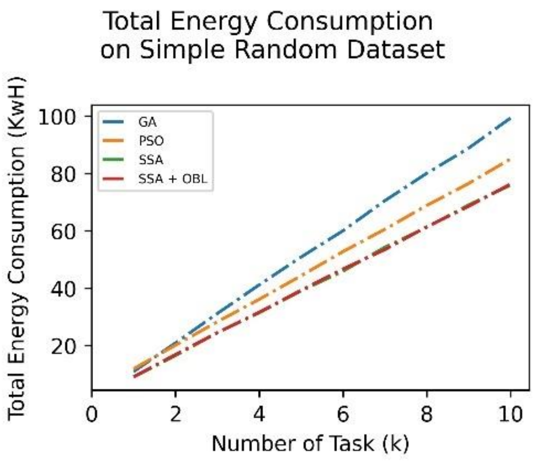

Figure. 20 Total energy consumption (simple random dataset)

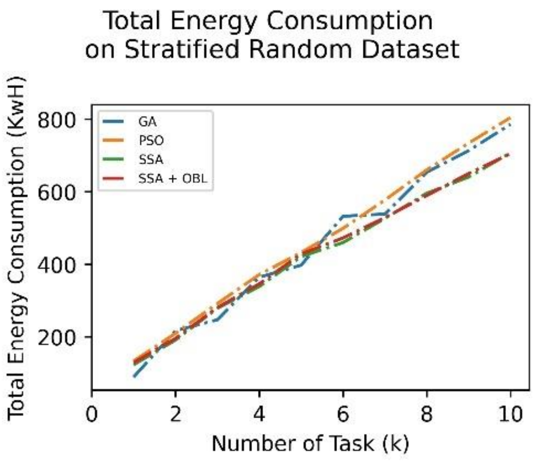

Figure. 21 Total energy consumption (stratified random dataset)

The SSA-OBL exhibits the least energy consumption when used to the SDSC dataset, but the SSA demonstrates the lowest energy consumption when employed on the remaining two datasets. The resource utilization for SSA-OBL exhibits a marginal increase compared to that of SSA.

5.11 Imbalance degree

The imbalance degree refers to the extent of imbalance displayed between the task with the highest length and the task with the lowest length within the dataset under consideration. A smaller imbalance degree is preferable. It is calculated as in (10), where T_{max}, T_{min}, and T_{avg} represent the maximum, minimum, and average total execution time, respectively.

$$Imbalance\_Degree = \frac{T_{max}-T_{min}}{T_{avg}} \qquad (10)$$

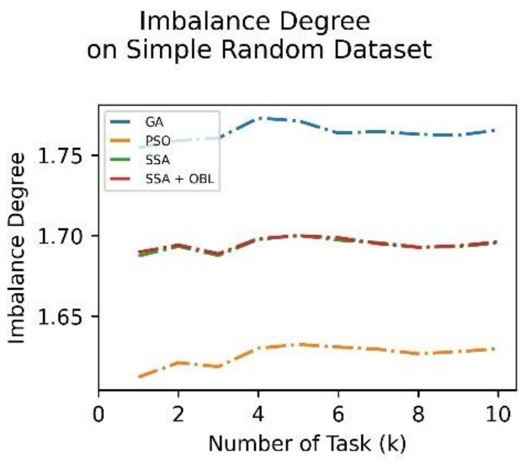

Figure. 22 Imbalance degree (simple random dataset)

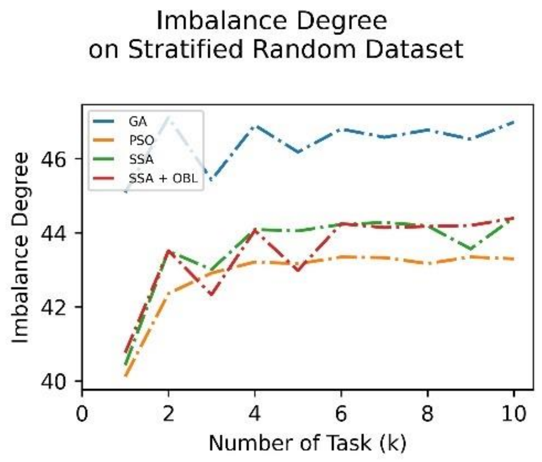

Figure. 23 Imbalance degree (stratified random dataset)

The imbalance degree values on the SDSC dataset for GA, PSO, SSA, and SSA-OBL are 57,08; 55,39; 52,77 and 53,24 respectively. Fig. 22 and Fig. 23 illustrate the imbalance degree observed in both simple random dataset and stratified random dataset.

The optimal degree of imbalance for a simple random dataset and a stratified random dataset is achieved through the use of PSO, but for the SDSC dataset, the optimal degree of imbalance is attained utilizing the SSA. The SSA-OBL method may not be optimal for datasets that exhibit significant fluctuation in the number and duration of tasks.

The optimization of resource allocation strategies in cloud computing environments is crucial for improving system performance and efficiency. An optimization algorithm may encounter issues with premature convergence. The issue was resolved by combining the squirrel search algorithm (SSA) with opposition-based learning (OBL). The primary objectives of SSA-OBL incorporate the optimization of makespan, throughput, and resource utilization. A comparative analysis was performed to evaluate the performance of the SSA-OBL algorithm in comparison to the GA, PSO, and SSA algorithms. Using the CloudSIM simulator, the experiment was conducted using three datasets: the SDSC, a simple random, and a stratified random dataset. The criteria for evaluation are makespan, average start time, average end time, average execution time, total wait time, total scheduling length, throughput, resource utilization, total energy consumption, and imbalance degree.

The SSA-OBL algorithm has been shown to be highly effective in facilitating the efficient execution of cloud activities. This is seen in its capability to accomplish a low makespan and sustain a steady rate of increase. The SSA-OBL algorithm exhibits superior energy consumption performance on the SDSC dataset, but it demonstrates less effective performance compared to the SSA algorithm on the other dataset. However, it exhibits a better incremental growth in value.

The SA-OBL algorithm is suboptimal in terms of average start time, average finish time, average execution time, and total scheduling length when compared to the PSO, GA, and SSA algorithms, mostly due to its higher level of complexity. The SSA-OBL algorithm has slightly lower performance in terms of throughput and resource utilization when compared to SSA. However, it demonstrates a more consistent and gradual rise in performance compared to other algorithms. The SSA-OBL method may not be the most optimal approach for datasets that demonstrate substantial variability in the quantity and duration of cloud tasks.

6. Conclusion

We employ the squirrel search algorithm to optimize the efficiency of makespan, throughput, and resource utilization while ensuring compliance with deadline constraints. In addition, we utilize opposition-based learning (OBL) to avoid early convergence while dealing with cloud task scheduling challenges. Subsequently, we assess the result by juxtaposing it with the outcomes achieved using genetic algorithm (GA), particle swarm optimization (PSO), and the original SSA. We performed a comparative analysis of four algorithms using three datasets with distinct characteristics: a simple random dataset (with tasks evenly distributed and minimal variation in task length), a stratified random dataset (with tasks unevenly distributed and significant variation in task length), and an SDSC dataset (a dataset derived from real-world data).

The improved SSA exhibits superiority over other algorithms in the optimization of makespan on a simple random dataset with an average value of 8.333, as well as minimizing overall energy consumption on the san diego super computer (SDSC) Blue Horizon dataset which has an average value of 449 kWH. In conclusion, the SSA-OBL algorithm demonstrates a higher level of effectiveness compared to GA, PSO, and SSA algorithms in optimizing the makespan. Additionally, it proves to be successful in decreasing the energy consumption. The SSA-OBL demonstrates a progressive rise, making the results more predictable for a wider range of cloud task.

For future research, the outcomes of this study can be enhanced at both the data and algorithm levels. SSA-OBL can be observed in enormous data volume real-world cloud environments. It is also possible to enhance the algorithm by combining SSA with other algorithms.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, Ary Mazharuddin Shiddiqi and Henning Titi Ciptaningtyas; methodology, Henning Titi Ciptaningtyas; software, Muhammad Nur Fauzan; validation, Fuad Dary Rosyadi, Ary Mazharuddin Shiddiqi, and Henning Titi Ciptaningtyas; formal analysis, Ary Mazharuddin Shiddiqi; investigation, Ary Mazharuddin Shiddiqi and Henning Titi Ciptaningtyas; resources, Fuad Dary Rosyadi; data curation, Fuad Dary Rosyadi; writing—original draft preparation, Henning Titi Ciptaningtyas; writing—review and editing, Ary Mazharuddin Shiddiqi and Diana Purwitasari; visualization, Muhammad Nur Fauzan; supervision, Ary Mazharuddin Shiddiqi; project administration, Diana Purwitasari; funding acquisition, Fuad Dary Rosyadi and Henning Titi Ciptaningtyas.

Acknowledgments

This work was supported by Institut Teknologi Sepuluh Nopember (ITS) under Scientific Research Scheme Research Grant Batch 1, Number 1706/PKS/ITS/2023.

References

[1] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing", *Egyptian Informatics Journal*, Vol. 16, pp. 275-295, 2015.

[2] H. Singh, S. Tyagi, P. Kumar, S. S. Gill, and R. Buyya, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions", *Simulation Modelling Practice and Theory*, Vol. 111, p. 102353, 2021.

[3] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends", *Swarm and Evolutionary Computation*, Vol. 62, p. 100841, 2021.

[4] X. S. Yang, "Nature-Inspired Optimization Algorithms", *Nature-Inspired Optimization Algorithms*, pp. 1-263, 2014.

[5] M. Jain, V. Singh, and A. Rani, "A novel nature-inspired algorithm for optimization: Squirrel search algorithm", *Swarm and Evolutionary Computation*, Vol. 44, pp. 148-175, 2019.

[6] P. S. Rawat, P. Gupta, P. Dimri and G. P. Saroha, "Power efficient resource provisioning for cloud infrastructure using bio-inspired artificial neural network model", *Sustainable Computing: Informatics and Systems*, Vol. 28, p. 100431, 2020.

[7] H. R. Tizhoosh, "Opposition-Based Learning: A New Scheme for Machine Intelligence", In: *Proc. of International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, 2005.

[8] S. Mahdavi, S. Rahnamayan, and K. Deb, "Opposition based learning: A literature review", *Swarm and Evolutionary Computation*, Vol. 39, pp. 1-23, 2018.

[9] M. O. Ahmad and R. Z. Khan, "Cloud computing modeling and simulation using cloudsim environment", *International Journal of Recent Technology and Engineering*, Vol. 8, No. 2, pp. 5439-5445, 2019.

[10] H. T. Ciptaningtyas, A. M. Shiddiqi, and D. Purwitasari, "A Systematic Literature Review of Genetic Algorithm-Based Approaches for Cloud-Based Task Scheduling", In: *Proc. of 14th International Conference on Information, Communication Technology and System (ICTS)*, Surabaya, Indonesia, 2023.

[11] G. Singh and A. K. Chaturvedi, "Particle Swarm Optimization-Based approaches for Cloud-Based Task and Workflow scheduling: A Systematic Literature Review", In: *Proc. of 2021 Second International Conference on Secure Cyber Computing and Communication (ICSCCC)*, Jalandhar, India, 2021.

[12] P. Krishnadoss and P. Jacob, "OCSA: Task

Scheduling Algorithm in Cloud Computing Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 11, No. 3, pp. 271-279, 2018, doi: 10.22266/ijies2018.0630.29.

[13] P. Krishnadoss and P. Jacob, "OLOA: Based Task Scheduling in Heterogeneous Clouds", *International Journal of Intelligent Engineering and Systems*, Vol. 12, No. 1, pp. 114-122, 2019, doi: 10.22266/ijies2019.0228.12.

[14] C. Chandrashekar and P. Krishnadoss, "Opposition based sunflower optimization algorithm using cloud computing environment", *Materials Today: Proceeding*, Vol. 62, No. 7, pp. 4896-4902, 2022.

[15] L. Imene, S. Sihem, K. Okba and B. Mohamed, "A Third Generation Genetic Algorithm NSGA-III for Task Scheduling in Cloud Computing", *Journal of King Saud University - Computer and Information Sciences*, pp. 1-15, 2022.

[16] S. Nabi and M. Ahmed, "PSO-RDAL: particle swarm optimization-based resource- and deadline-aware dynamic load balancer for deadline constrained cloud tasks", *Journal of Supercomputing*, Vol. 78, No. 4, pp. 4624-4654, 2022.

[17] K. Dubey and S. C. Sharma, "A Novel Multi-Objective CR-PSO Task Scheduling Algorithm with Deadline Constraint in Cloud Computing", *Sustainable Computing: Informatics and Systems*, Vol. 32, pp. 1-20, 2021.

[18] A. Srivastava and N. Kumar, "An energy efficient robust resource provisioning based on improved PSO-ANN", *International Journal of Information Technology*, Vol. 15, No. 1, pp. 107-117, 2023.

[19] F. A. Emara, A. A. G. Elrab, A. Sobhi, and K. R. Raslan, "Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 5, pp. 571-582, 2021, doi: 10.22266/ijies2021.1031.50.

[20] P. S. Rawat, P. Dimri, P. Gupta and G. P. Saroha, "Resource provisioning in scalable cloud using bio-inspired artificial neural network model", *Applied Soft Computing*, Vol. 99, 2021.

APPENDIX

SDSC DATASET

Table 3. Experimental result of each scoring parameter on SDSC dataset

Scoring Parameter	GA	PSO	SSA	SSA + OBL
Makespan	127.357,80	127.357,80	101.699,70	104.852,40
Average Start Time	28.987,74	26.175,75	25.210,30	25.544,35
Average Finish Time	29.325,54	26.555,37	25.580,00	25.917,32
Average Execution Time	337,80	379,61	369,70	372,97
Total Wait Time	214.359.872,40	193.565.257,20	186.425.740,80	188.896.001,40
Total Scheduling Length	214.487.230,20	193.699.071,60	186.527.440,50	189.000.853,80
Throughput	0,06	0,06	0,07	0,07
Resource Utilization	36,38	39,97	50,96	49,20
Total Energy Consumption	584,43	505,07	452,46	449,99
Imbalance Degree	57,08	55,39	52,77	53,24

SIMPLE RANDOM DATASET

Table 4. Experimental result of makespan scoring parameter on simple random dataset as in Figure. .

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	2.262,30	5.468,10	1.880,70	1.951,80
2K	4.303,50	6.894,60	3.330,60	3.378,30
3K	6.334,80	8.531,70	4.974,00	5.014,50
4K	8.263,50	9.843,00	6.167,40	6.304,20
5K	10.094,10	11.209,20	7.716,30	7.686,60
6K	11.850,90	12.973,20	8.955,60	9.071,70
7K	13.830,00	14.068,50	10.629,60	10.308,30

8K	15.412,20	15.640,80	11.815,80	11.903,10
9K	17.151,00	16.988,10	13.263,00	13.049,70
10K	19.248,90	18.984,30	14.721,90	14.664,30

Table 5. Experimental result of average start time scoring parameter on simple random dataset as in Figure.

Total Cloudlets Finished	**GA**	**PSO**	**SSA**	**SSA + OBL**
1K	666,02	696,69	600,83	603,84
2K	1.332,68	1.263,26	1.202,79	1.208,64
3K	2.023,86	1.862,40	1.811,93	1.817,27
4K	2.717,18	2.457,87	2.411,19	2.412,56
5K	3.376,20	3.053,93	3.014,63	3.019,56
6K	3.994,00	3.675,27	3.611,55	3.619,28
7K	4.692,31	4.283,58	4.234,42	4.224,34
8K	5.371,16	4.908,94	4.839,75	4.841,26
9K	6.024,99	5.512,12	5.446,49	5.446,30
10K	6.718,76	6.119,18	6.042,63	6.044,84

Table 6. Experimental result of average finish time scoring parameter on simple random dataset as in Figure.

Total Cloudlets Finished	**GA**	**PSO**	**SSA**	**SSA + OBL**
1K	727,58	763,69	664,84	667,76
2K	1.394,08	1.329,88	1.266,57	1.272,39
3K	2.085,20	1.929,13	1.875,92	1.881,23
4K	2.778,09	2.524,13	2.474,81	2.476,16
5K	3.437,17	3.120,10	3.078,16	3.083,10
6K	4.055,24	3.741,50	3.675,19	3.682,85
7K	4.753,51	4.349,87	4.298,11	4.288,06
8K	5.432,42	4.975,35	4.903,55	4.905,07
9K	6.086,27	5.578,47	5.510,29	5.510,07
10K	6.779,94	6.185,45	6.106,33	6.108,51

Table 7. Experimental result of average execution time scoring parameter on simple random dataset as in Figure.

Total Cloudlets Finished	**GA**	**PSO**	**SSA**	**SSA + OBL**
1K	61,56	67,00	64,01	63,92
2K	61,40	66,63	63,78	63,75
3K	61,34	66,73	64,00	63,95
4K	60,91	66,26	63,62	63,60
5K	60,97	66,16	63,52	63,53
6K	61,24	66,23	63,64	63,58
7K	61,20	66,28	63,69	63,72
8K	61,26	66,40	63,80	63,80
9K	61,28	66,34	63,79	63,77
10K	61,17	66,27	63,70	63,68

Table 8. Experimental result of total wait time scoring parameter on simple random dataset as in Figure.

Total Cloudlets Finished	**GA**	**PSO**	**SSA**	**SSA + OBL**
1K	665.416,80	696.094,20	600.232,50	603.243,90
2K	2.664.161,10	2.525.311,80	2.404.385,10	2.416.071,60
3K	6.069.771,00	5.585.400,00	5.433.985,80	5.450.012,10

4K	10.866.312,90	9.829.074,60	9.642.352,50	9.647.856,90
5K	16.877.987,10	15.266.657,70	15.070.170,60	15.094.812,60
6K	23.960.410,20	22.048.012,80	21.665.696,40	21.712.054,50
7K	32.841.938,70	29.980.876,50	29.636.715,60	29.566.160,10
8K	42.964.488,00	39.266.757,00	38.713.210,20	38.725.297,20
9K	54.219.517,20	49.603.710,60	49.013.054,10	49.011.281,10
10K	67.181.649,30	61.185.838,50	60.420.308,40	60.442.374,60

Table 9. Experimental result of task scheduling length scoring parameter on simple random dataset as in Figure

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	667.679,10	701.562,30	602.113,20	605.195,70
2K	2.668.464,60	2.532.206,40	2.407.715,70	2.419.449,90
3K	6.076.105,80	5.593.931,70	5.438.959,80	5.455.026,60
4K	10.874.576,40	9.838.917,60	9.648.519,90	9.654.161,10
5K	16.888.081,20	15.277.866,90	15.077.886,90	15.102.499,20
6K	23.972.261,10	22.060.986,00	21.674.652,00	21.721.126,20
7K	32.855.768,70	29.994.945,00	29.647.345,20	29.576.468,40
8K	42.979.900,20	39.282.397,80	38.725.026,00	38.737.200,30
9K	54.236.668,20	49.620.698,70	49.026.317,10	49.024.330,80
10K	67.200.898,20	61.204.822,80	60.435.030,30	60.457.038,90

Table 10. Experimental result of throughput scoring parameter on simple random dataset Figure. 16

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	0,44	0,18	0,53	0,52
2K	0,47	0,29	0,60	0,59
3K	0,47	0,35	0,60	0,60
4K	0,48	0,41	0,65	0,64
5K	0,50	0,45	0,65	0,65
6K	0,51	0,46	0,67	0,66
7K	0,51	0,50	0,66	0,68
8K	0,52	0,51	0,68	0,67
9K	0,53	0,53	0,68	0,69
10K	0,52	0,53	0,68	0,68

Table 11. Experimental result of resource utilization scoring parameter on simple random dataset as in Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	50,48	22,73	63,32	61,10
2K	52,96	35,82	71,24	70,05
3K	53,83	43,52	71,62	71,12
4K	54,62	49,92	76,46	74,88
5K	55,98	54,72	76,43	76,56
6K	57,49	56,77	79,02	77,95
7K	57,42	61,13	77,74	80,19
8K	58,93	62,99	80,05	79,44
9K	59,59	65,12	80,22	81,48
10K	58,91	64,72	80,15	80,49

Table 12. Experimental result of total energy consumption scoring parameter on simple random dataset as in Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	10,92	11,87	8,99	9,12
2K	20,90	20,14	16,52	16,86
3K	31,19	28,28	24,46	24,51
4K	41,16	36,12	31,43	31,62
5K	50,86	44,33	39,07	39,28
6K	60,02	52,76	45,93	46,69
7K	70,49	60,53	54,27	53,44
8K	80,04	68,83	61,21	61,37
9K	88,81	76,50	69,01	68,54
10K	99,30	85,03	75,89	76,30

Table 13. Experimental result of imbalance degree scoring parameter on simple random dataset as in Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	1,75	1,61	1,69	1,69
2K	1,76	1,62	1,69	1,69
3K	1,76	1,62	1,69	1,69
4K	1,77	1,63	1,70	1,70
5K	1,77	1,63	1,70	1,70
6K	1,76	1,63	1,70	1,70
7K	1,76	1,63	1,70	1,69
8K	1,76	1,63	1,69	1,69
9K	1,76	1,63	1,69	1,69
10K	1,77	1,63	1,70	1,70

STRATIFIED RANDOM DATASET

Table 14. Experimental result of makespan scoring parameter on stratified random dataset as in Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	24.376,20	38.160,60	33.544,50	35.235,60
2K	64.306,50	55.589,10	52.050,30	51.948,60
3K	55.450,50	72.849,30	69.680,40	69.405,90
4K	85.202,70	90.219,30	78.550,80	84.920,10
5K	89.418,30	112.952,40	94.179,30	101.937,30
6K	124.604,70	113.685,00	99.671,10	114.793,80
7K	114.748,80	128.116,50	120.544,80	122.353,80
8K	148.514,10	150.314,10	132.783,90	132.357,30
9K	156.586,20	173.052,60	137.270,40	140.928,00
10K	171.572,10	175.754,40	157.053,30	159.035,10

Table 15. Experimental result of average start time scoring parameter on stratified random dataset as in Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	838,65	4.689,58	4.521,38	4.592,96
2K	1.910,88	9.039,00	9.293,27	8.813,23
3K	2.766,87	13.753,35	13.829,91	13.734,38
4K	3.769,20	18.450,41	18.406,03	17.952,64
5K	4.608,97	23.185,06	22.671,36	23.223,09
6K	5.610,48	27.861,28	27.167,63	27.408,42
7K	6.456,61	32.806,05	32.100,25	31.884,73
8K	7.625,49	36.871,22	35.981,22	36.689,30
9K	8.460,36	42.493,28	41.280,90	40.698,00
10K	9.284,49	47.092,59	45.260,74	45.346,01

Table 16. Experimental result of average finish time scoring parameter on stratified random dataset as in Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	1.325,83	5.193,93	5.014,10	5.093,08
2K	2.376,32	9.545,69	9.786,77	9.306,14
3K	3.247,37	14.260,97	14.328,98	14.233,83
4K	4.238,23	18.954,33	18.900,05	18.447,99
5K	5.084,57	23.691,05	23.167,33	23.721,02
6K	6.078,03	28.366,32	27.660,88	27.902,98
7K	6.928,69	33.312,02	32.595,50	32.381,00
8K	8.094,44	37.378,75	36.476,85	37.185,49
9K	8.930,82	42.999,51	41.776,68	41.193,79
10K	9.752,07	47.599,39	45.754,84	45.839,82

Table 17. Experimental result of average execution time scoring parameter on stratified random dataset Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	487,18	504,35	492,72	500,12
2K	465,44	506,68	493,50	492,91
3K	480,49	507,62	499,08	499,45
4K	469,03	503,92	494,02	495,35
5K	475,60	505,99	495,97	497,93
6K	467,54	505,04	493,26	494,56
7K	472,07	505,96	495,25	496,27
8K	468,95	507,53	495,63	496,20
9K	470,46	506,23	495,78	495,78
10K	467,59	506,80	494,11	493,82

Table 18. Experimental result of total wait time scoring parameter on stratified random dataset Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	838.053,90	4.688.979,30	4.520.779,20	4.592.356,20
2K	3.820.557,60	18.076.808,70	18.585.334,80	17.625.258,00
3K	8.298.824,40	41.258.262,60	41.487.916,50	41.201.351,10
4K	15.074.415,00	73.799.243,10	73.621.731,60	71.808.147,00
5K	23.041.872,00	115.922.301,30	113.353.779,60	116.112.439,80
6K	33.659.296,20	167.164.074,90	163.002.165,30	164.446.938,90
7K	45.192.099,60	229.638.170,70	224.697.564,00	223.188.916,50
8K	60.999.116,40	294.964.987,50	287.844.948,90	293.509.584,00
9K	76.137.869,70	382.434.092,10	371.522.739,60	366.276.612,60
10K	92.838.858,30	470.919.872,70	452.601.353,70	453.454.059,60

Table 19. Experimental result of task scheduling length scoring parameter on stratified random dataset Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	862.430,10	4.727.139,90	4.554.323,70	4.627.591,80
2K	3.884.864,10	18.132.397,80	18.637.385,10	17.677.206,60
3K	8.354.274,90	41.331.111,90	41.557.596,90	41.270.757,00
4K	15.159.617,70	73.889.462,40	73.700.282,40	71.893.067,10
5K	23.131.290,30	116.035.253,70	113.447.958,90	116.214.377,10
6K	33.783.900,90	167.277.759,90	163.101.836,40	164.561.732,70
7K	45.306.848,40	229.766.287,20	224.818.108,80	223.311.270,30
8K	61.147.630,50	295.115.301,60	287.977.732,80	293.641.941,30
9K	76.294.455,90	382.607.144,70	371.660.010,00	366.417.540,60
10K	93.010.430,40	471.095.627,10	452.758.407,00	453.613.094,70

Table 20. Experimental result of throughput scoring parameter on stratified random dataset Figure

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	0,04	0,03	0,03	0,03
2K	0,03	0,04	0,04	0,04
3K	0,05	0,04	0,04	0,04
4K	0,05	0,05	0,05	0,05
5K	0,06	0,05	0,05	0,05
6K	0,05	0,05	0,06	0,05
7K	0,06	0,06	0,06	0,06
8K	0,05	0,05	0,06	0,06
9K	0,06	0,05	0,07	0,06
10K	0,06	0,06	0,06	0,06

Table 21. Experimental result of resource utilization scoring parameter on stratified random dataset as in Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	37,31	25,26	27,55	26,75
2K	28,17	35,04	35,68	35,90
3K	48,87	39,14	40,57	40,79
4K	41,12	42,37	47,19	44,15
5K	49,91	43,36	49,07	45,78
6K	41,85	49,61	55,39	49,69
7K	53,97	52,05	53,61	53,88
8K	47,14	50,30	56,03	56,30
9K	50,43	49,72	61,60	59,01
10K	50,80	53,87	58,62	57,77

Table 22. Experimental result of total energy consumption scoring parameter on stratified random dataset Figure. 21

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	89,02	132,99	122,70	128,73
2K	216,80	209,39	191,01	195,25
3K	246,88	291,67	278,50	280,67
4K	364,14	371,33	338,48	345,83
5K	397,35	432,33	421,03	428,22
6K	532,31	498,79	459,65	472,41
7K	538,30	576,94	526,17	528,50
8K	653,67	659,97	595,72	589,35
9K	713,03	733,89	640,29	649,94
10K	785,98	804,02	707,08	704,96

Table 23. Experimental result of imbalance degree scoring parameter on stratified random dataset as in Figure.

Total Cloudlets Finished	GA	PSO	SSA	SSA + OBL
1K	45,08	40,11	40,43	40,76
2K	47,11	42,36	43,51	43,51
3K	45,43	42,91	43,01	42,33
4K	46,91	43,21	44,09	44,06
5K	46,18	43,16	44,05	42,98
6K	46,80	43,34	44,22	44,24
7K	46,58	43,33	44,28	44,14
8K	46,77	43,17	44,19	44,17
9K	46,53	43,34	43,56	44,20
10K	46,99	43,30	44,41	44,39