Vol. 2, No. 1, pp. 1-16, (June 2024)

ASWAN SCIENCE AND TECHNOLOGY BULLETIN (ASTB)

Online ISSN: 3009-7916, Print ISSN: 1110-0184

Journal homepage: https://astb.journals.ekb.eg/ E-mail: essamshaalan@sci.aswu.edu.eg

Original Article

Optimization of Task Scheduling in Cloud Computing Using the Sine Cosine Algorithm

Ahmed Y. Hamed[a*], Moatamad R. Hassan[b], and M. Kh. Elnahary[a]

[a]**Faculty of Computers and Artificial Intelligence, Department of Computer Science, Sohag University, Sohag, 82524, Egypt.**

[b]**Department of Computer Science and Mathematics, Faculty of Science, Aswan University, Aswan, Egypt**

Received: 16/05/2024 Accepted: 05/06/2024

Abstract

Cloud computing has revolutionized extensive parallel processing and distributed computation, offering computer resources through a usage-based payment model that significantly enhances accessibility and scalability. However, the effectiveness and speed of cloud services heavily depend on how tasks are scheduled and executed. Current task scheduling methods often struggle to balance performance metrics such as throughput, makespan, efficiency, and speedup, leading to suboptimal utilization of cloud resources. Addressing this critical gap, our study introduces a novel task-scheduling algorithm specifically designed for cloud computing environments. Rooted in the sine cosine algorithm, our approach is tailored to meet the unique demands of cloud setups, optimizing resource allocation and task execution. Rigorous testing across three distinct scenarios demonstrates that our algorithm outperforms existing methods in terms of throughput, makespan, efficiency, and speedup. These results highlight the practical effectiveness and efficiency of our algorithm, offering a significant advancement in optimizing task scheduling within cloud computing systems. Our work thus contributes to enhancing the performance and reliability of cloud services, supporting better resource management and user satisfaction.

Keywords: Sine Cosine Algorithm; Cloud Computing; Heterogeneous Virtual Machines; Task Scheduling;

1. Introduction

Cloud technology represents a revolutionary paradigm aimed at delivering universal, user-friendly, readily accessible network connectivity. It encompasses a reservoir of customizable computing assets that can be swiftly acquired and relinquished, demanding minimal administrative involvement or interaction on the part of service facilitators.

Corresponding author*: E-mail address: ayhamedd@gamil.com

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

Cloud computing now delivers dynamic services such as memory, bandwidth, applications, data, and services of information technology through the internet. Various factors, including work schedules, determine the dependability and cloud services' performance. Scheduling strategies extend to resource, task, or workflow tiers interchangeably. End-users submit requests to the data center for computational tasks, referred to as tasks. A task refers to a small unit of work that must be finished within a specified timeframe. Scheduling tasks involves allocating tasks from cloud customers to cloud providers based on resource availability. Scheduling is done based on many characteristics to improve cloud performance holistically. A task might encompass activities like processing, data input, program retrieval, or operations of storage. Data centers are classified according to the agreed-upon the requested services and service level. Each work is subsequently allocated to an accessible server. The servers complete the operation that is requested and return a result, or response, to the user. The scheduling of cloud tasks is a Non-deterministic Polynomial (NP) complete issue. Users send their jobs to the cloud scheduler during the task-scheduling procedure. The scheduler interacts with the information system hosted on the cloud to assess the current availability status of resources and their characteristics before assigning tasks to diverse resources according to their specific needs. The scheduler will allocate multiple tasks of the user to several Virtual Machines (VMS). A well-planned schedule always distributes virtual machines most efficiently. A solid scheduling method always increases Central Processing Unit (CPU) usage, time to completion, and overall throughput. Scheduling tasks may be done in various ways based on multiple criteria. Tasks can be statically allocated to diverse resources during the build phase or dynamically allocated during runtime (Mathew et al., 2014).

To effectively address the scheduling tasks challenge, our work introduces an innovative solution using the efficient sine cosine algorithm, referred to as Efficient Sine Cosine (ESC). This novel algorithm aims at minimizing makespan while optimizing throughput, efficiency, and speedup. Unlike traditional scheduling methods, the ESC algorithm leverages the unique characteristics of the sine cosine algorithm to allocate tasks more efficiently and effectively in cloud computing environments. Furthermore, our study contributes to the existing literature by rigorously testing the ESC algorithm across diverse scenarios represented by directed acyclic graphs (DAGs), showcasing its superior performance compared to other scheduling algorithms. The practical implications of our findings are substantial, as they pave the way for improved resource utilization, enhanced service performance, and better user satisfaction in cloud computing systems. By introducing the ESC algorithm and demonstrating its effectiveness through rigorous testing, our work makes a significant contribution to advancing task scheduling techniques in cloud computing, addressing a critical gap in current methodologies, and providing a valuable tool for optimizing cloud service performance.

The document is structured as follows: Section 2 delves into the relevant literature. Section 3 outlines the problem. In Section 4, the algorithm based on sine and cosine principles is expounded. The ESC approach is detailed in Section 5. Section 6 scrutinizes the performance of the proposed algorithm. Lastly, Section 7 wraps up with conclusions and prospects for future research.

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

2. Related Work

Cloud technology delivers computational elements like hardware and software to users via a network, presenting a core concept focused on dispersing extensive storage, computational power, and data access for scientific purposes. Within cloud computing, tasks from users undergo strategic organization and execution, ensuring services are delivered effectively through optimal resource allocation. Job scheduling relies on diverse strategies for task assignment, enhancing efficiency in service provision. This work (Senthil Kumar & Venkatesan, 2019) presents a streamlined approach to scheduling tasks that enhances the methodology of scheduling tasks. Frequently, optimization methods are employed to address nondeterministic hard scenarios. User tasks are saved in the queue management using this mechanism. If the job is repeated, the priority is determined, and appropriate resources are allocated. New jobs are assessed and queued for execution instantaneous. The result from the real-time queue undergoes processing through the haybird technique that uses genetic and particle swarms. This method integrates genetic and particle swarm optimization algorithms for implementation. By employing the hybrid algorithm, appropriate resources are selected for user tasks within the real-time queue.

Cloud represents the commercialization and evolution of parallel computing, grid computing, and distributed computing. An inherent challenge in this domain is task scheduling, a complex NP-hard optimization problem that has spurred the development of various meta-heuristic techniques. A proficient scheduler of tasks must modify its scheduling approach to dynamic environments and diverse workloads. This research introduces a cloud scheduling tasks strategy utilizing the ant colony method (Tawfeek et al., 2015). The method is a stochastic search strategy designed for workload allocation to virtual machines. The primary objective of these algorithms is to minimise the completion time for a given set of jobs.

The emerging technology of cloud computing enables pay-as-you-go models for consumers, delivering high-performance capabilities. Additionally, cloud computing encompasses a heterogeneous system housing diverse application data. Optimizing transfer and processing times becomes pivotal for applications handling intensive data or computations. The authors have crafted a scheduling task model aimed at minimizing the costs of processing. They introduce a method based on particle swarm optimization (PSO), drawing from their study (Guo et al., 2012) and emphasizing a small position value criterion. This study contrasts the PSO algorithm with a variant incorporating crossover and mutation techniques.

Leveraging cloud resources for extensive programs can be cost-effective. These programs can be broken into task sequences, represented as a Directed Acyclic Graph (DAG). Nodes denote tasks, and edges show task dependencies. Cloud users are billed based on resource usage, but early scheduling algorithms focused only on reducing task completion time (makespan). To address this, we propose a cost-effective scheduling system using two heuristic techniques. The first dynamically assigns tasks to affordable VMs using Pareto dominance. The second minimizes costs for non-critical tasks, ensuring efficient cloud task scheduling (Su et al., 2013).

This research (Alsubaei et al., 2024) introduces a two-machine learning approach employing K-means clustering to enhance performance and assist in the selection of cloud scheduling technologies. The first method, dubbed Efficient K-means (Ekmeans), is complemented by

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

the second technique known as K-means HEFT (KmeanH), with HEFT denoting Heterogeneous Earliest End Time.

3. Problem Description

Within cloud computing frameworks, task organization is depicted through a visual schema involving a set of NTSK tasks, denoted as TSK_1, TSK_2, TSK_3, and so forth. Each task is interconnected through GRA and E-bound edges, signifying different aspects of their functional demands (Hamed & Alkinani, 2021). The nodes within this representation stand for step-by-step operations executed within a virtual environment, with each node containing one or multiple data inputs. The commencement or conclusion of a task is triggered by the availability of input data. A hierarchical relationship denoted as $TSK_i \rightarrow TSK_j$ implies precedence where TSK_i must be executed before TSK_j. The duration of task execution denoted as TSK_i, is characterized by its weight. Communication expenses between tasks, labeled as $COM\_COST(TSK_i, TSK_j)$, are non-existent if both tasks are executed on the same virtual platform. Time-related parameters such as task initiation and completion are denoted by $Start\_Time(TSK_i, VLM_j)$ and $Finish\_Time(TSK_i, VLM_j)$, respectively (Hamed & Alkinani, 2021). The timing of data arrival for TSK_i on virtual machine VLM_j is also part of this organizational framework.

$$Data\_Arrival(TSK_i.VLM_j)=\max\{Finish\_Time(TSK_k,VLM_j)+COM\_COST(TSK_i,TSK_k)\} \quad (1)$$

In the realm of cloud computing, the task scheduling challenge revolves around determining the optimal assignment or schedule for starting tasks on virtual machines. The objective is to reduce the overall task completion duration and execution costs while adhering to precedence constraints. The time taken to finish a task, also referred to as the completion time, schedule length, or finish time, is determined by the following calculation:

$$\text{Completion Time} = \max(Finish\_Time(TSK_i, VLM_j)) \quad (2)$$

$$Start\_Time(TSK_i, VLM_j) = \max\{Ready\_Time(VLM_j), Data\_Arrival(TSK_i, VLM_j)\} \quad (3)$$

$$Finish\_Time(TSK_i, VLM_j) = Start\_Time(TSK_i, VLM_j) + WEIT(TSK_i, VLM_j) \quad (4)$$

$$Ready\_Time(VLM_j) = Finish\_Time(TSK_i, VLM_j) \quad (5)$$

$$\text{Speedup} = \min_{VLM_j} \left(\sum_{TSK_i} \frac{WEIT_{i,j}}{\text{Completion Time}} \right) \quad (6)$$

$$\text{Efficiency} = \frac{\text{Speedup}}{NVLM} \quad (7)$$

$$\text{Throughput} = \frac{NTSK}{\text{Completion Time}} \quad (8)$$

Algorithm 1: Completion Time Calculation (Hamed & Alkinani, 2021)

$Ready\_Time[VLM_j] = 0$
For each task TSK_i
{

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

Remove the first task, denoted as TSKi, from DLST and proceed with its execution.
For each VLM_j
{
If TSK_i is allocated on VLM_j
Calculate the Start_Time, Finish_Time, and Ready_Time
End If
}
}
Calculate Completion Time

4. Sine Cosine Algorithm

The Sine-Cosine algorithm (SCA) (Abdel-Basset et al., 2021; Mirjalili, 2016) It's a metaheuristic method that solves optimization problems inspired by the sine and cosine mathematical forms. SCA begins By dispersing its solutions across the search space of the problem and then When computing the fitness value for each solution, Z* represents the solution with the best fitness and is utilized throughout the optimization process to search for improved solutions. Within the optimization process, the SCA will update the position of each solution mathematically using the following formula:

$$Z_l^{u+1} = \begin{cases} Z_l^u + rand_1 * \sin(2\pi rand_2) * |rand_3 Z_l^* - Z_l^u| & rand_4 < \frac{1}{2} \\ Z_l^u + rand_1 * \cos(2\pi rand_2) * |rand_3 Z_l^* - Z_l^u| & rand_4 \geq \frac{1}{2} \end{cases} \quad (9)$$

Where Z_l^u denotes the position of the current at dimension lth, u represents the current iteration, and $rand_1$, $rand_2$, and $rand_3$ are random integers $rand_1$ is used in the optimization process To maintain a balance between exploration and exploitation, the exploitation operator is computed using the following formula:

$$rand_1 = q - q\left(\frac{u}{u_{max}}\right) \quad (10)$$

where q denotes a constant value and u_{max} denotes the maximum of iterations that may be performed. The method starts with a strong exploration capability at the beginning of the optimization process, which steadily decreases with each iteration until it fades away after the optimization process. In contrast, exploration and exploitation capability grows with each iteration until it reaches a plateau after the optimization phase.

Algorithm 2: Sine Cosine

Create a population of solutions Z_i where i = 1, 2…, N
Evaluate Z_i
Determine the best solution Z_i
u = 1
While ($u < u_{max}$) do
Update $rand_1$
For each Z_i do
Update the position

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

EndFor
Examine each solution's fitness value Z_{u+1}
if better, change the optimal solution Z* by Zu+1
u=u+1
EndWhile

5. The Proposed Algorithm

Given that the sine cosine algorithm operates on continuous value vectors, we'll explore five techniques for converting these continuous values into discrete values. The initial method involves the Smallest Position Value (SPV) principle, as proposed by (Dubey & Gupta, 2017). Another approach is the Largest Position Value (LPV) function, suggested by (Wang et al., 2011). Furthermore, we'll utilize the round nearest, floor nearest, and ceil nearest functions. Referencing Table 1, we'll leverage the modulus function along with the number of virtual machines in the SPV and LPV strategies to adjust the result accordingly.

Table 1 Change continuous to discrete

Pop	2.4	3.0	1.0	1.2	1.3	1.5	2.2
SPV rule	3	4	5	6	7	1	2
modulus with SPV and NVLM=3	1	2	3	1	2	2	3
LPV rule	2	1	7	6	5	4	3
modulus with LPV and NVLM=3	3	2	2	1	3	2	1
round nearest function	2	3	1	1	1	2	2
floor nearest function	2	3	1	1	1	1	2
ceil nearest function	3	3	1	2	2	2	3

Algorithm 3: changes a continuous to discrete

Functionconvert
Rando ∈ [1:5]
If (Rando == 1)
    Converts by SPV
Elseif (Rando == 2)
    Converts by LPV

Elseif (Rando == 3)
    Converts by round
Elseif (Rando == 4)
    Converts by floor
Else
    Converts by ceil
Endif

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

EndFunction

Algorithm 4: ESC

Input dag, including communication and computation costs.
Create a pop of solutions Z_i where i = 1, 2…, N
Change the initial pop Z_i by **Algorithm 3**
compute the completion time after converting by **Algorithm 1**
Determine the best solution (the best schedule length)
u = 1
while ($u < u_{max}$) do
  By using Eq. 10, update $rand_1$
  For each Z_i do
    By using Eq. 9, update the position
    Change the solution obtained by **Algorithm 3**
    Compute the completion time after converting by **Algorithm 1**
  End for
  Examine each solution's fitness value Z_{u+1}
  if better, change the optimal solution Z* by Zu+1
  u=u+1
End while

6. Evaluation of the ESC

In this study, we demonstrate the performance of the Efficient Sine Cosine (ESC) algorithm through its application in three distinct simulation scenarios, leveraging MATLAB as our simulation platform. These scenarios encompass various task and virtual machine configurations commonly encountered in cloud computing environments. Each scenario, including one with eleven tasks and three heterogeneous virtual machines, another with eleven tasks and three different heterogeneous virtual machines, and a third with eleven tasks distributed across two heterogeneous virtual machines, was initialized with specific parameters such as a population size of 100, maximum number of iterations of 100, and a q value of 2 for the ESC algorithm. We developed and implemented the ESC algorithm using custom scripts and functions within MATLAB's simulation environment, which facilitated the modeling and simulation of cloud computing environments, encompassing task scheduling, resource allocation, and performance evaluation. Subsequently, key performance metrics including throughput, makespan, efficiency, and speedup were comprehensively analyzed to assess the effectiveness and efficiency of the ESC algorithm in optimizing task scheduling within diverse cloud computing scenarios.

6.1 Case Study 1

We examine a scenario involving eleven tasks scheduled across three heterogeneous virtual machines. The cost associated with executing each task on different virtual machines is outlined in Table 2 (Keshanchi & Navimipour, 2016). Additionally, Table 3 details the start and finish times of each task on various virtual machines, along with the schedule generated by ESC. We compare the results achieved by ESC with those of three heterogeneous earliest finish time HEFT algorithms: HEFT-T, HEFT-B, and HEFT-L (Keshanchi & Navimipour, 2016), as well as the multiple priority queues with memetic algorithm (MPQMA) (Keshanchi & Navimipour, 2016). The outcomes from ESC, HEFT-T, HEFT-B, HEFT-L, and MPQMA are summarized in Table 4. Furthermore, Figs. 1- 4 provides graphical representations of the results

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

obtained by ESC, HEFT-T, HEFT-B, HEFT-L, and MPQMA in terms of throughput, efficiency, speedup, and makespan.

Table 2 The cost of computation for the first case

TSk/VLM	VLM_1	VLM_2	VLM_3
TSk_0	6	7	5
TSk_1	10	9	8
TSk_2	8	9	7
TSk_3	11	13	15
TSk_4	13	7	10
TSk_5	6	10	8
TSk_6	18	12	15
TSk_7	17	10	12
TSk_8	14	20	11
TSk_9	12	8	10
TSk_{10}	9	13	17

Table 3 Allocation that obtained by ESC for the first case

	VLM_1		VLM_2		VLM_3	
	Start_Time	Finish_Time	Start_Time	Finish_Time	Start_Time	Finish_Time
TSk_0	-	-	0	7	-	-
TSk_1	-	-	-	-	25	33
TSk_2	21	29	-	-	-	-
TSk_3	-	-	7	20	-	-
TSk_4	-	-	20	27	-	-
TSk_5	47	53	-	-	-	-
TSk_6	29	47	-	-	-	-
TSk_7	-	-	27	37	-	-
TSk_8	53	67	-	-	-	-
TSk_9	-	-	37	45	-	-
TSk_{10}	67	76	-	-	-	-

Table 4 The comparative outcomes for the first case

Algorithm	HEFT-T	HEFT-B	HEFT-L	MPQMA	ESC
Makespan	94	89	89	80	76

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

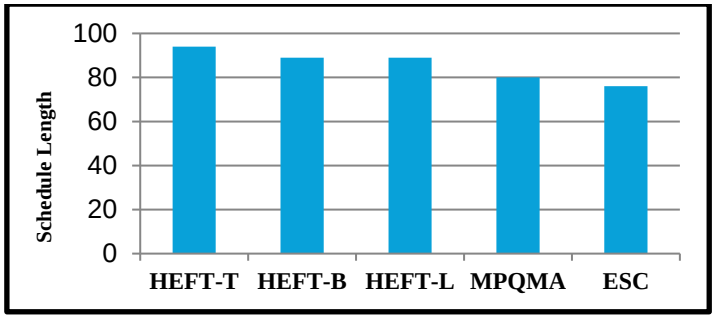

Fig. 1 Makespan comparison for the first case

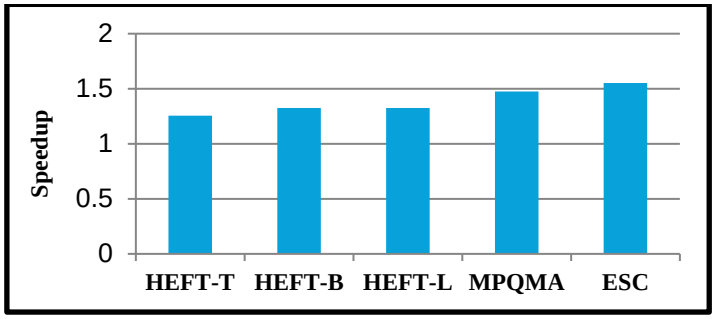

Fig. 2 Speedup comparison for the first case

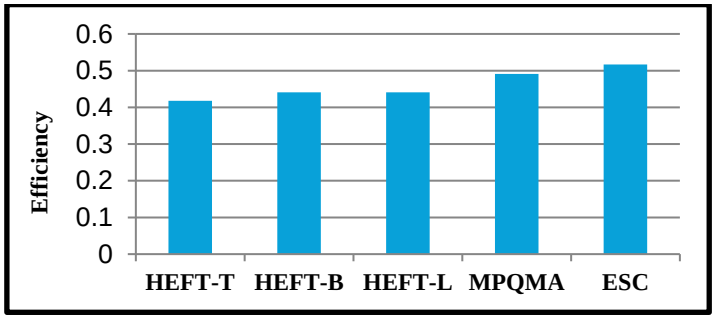

Fig. 3 Efficiency comparison for the first case

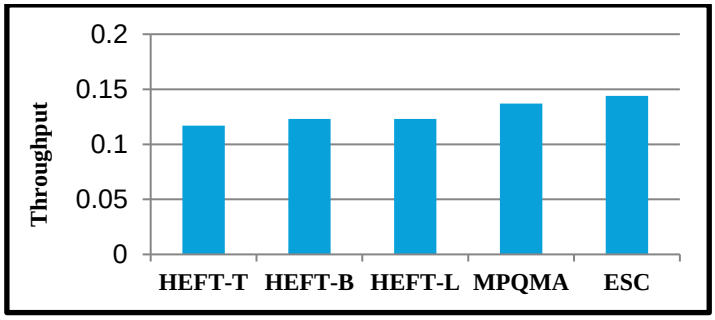

Fig. 4 Throughput comparison for the first case

6.2 Case Study 2

We examine a scenario involving eleven tasks scheduled across three heterogeneous virtual machines. The cost associated with executing each task on different virtual machines is outlined in Table 5 (Hosseini Shirvani, 2018). Additionally, Table 6 details the start and finish

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

times of each task on various virtual machines, along with the schedule generated by ESC. We compare the results achieved by ESC with those obtained by the Genetic Algorithm (GA) (Hamed & Alkinani, 2021) and the Quantum Genetic Algorithm with Rotation Angle Refinement (QGARAR) (Gandhi et al., 2018). The outcomes from ESC, GA, and QGARAR are summarized in Table 7. Furthermore, Figs. 5- 8 provides graphical representations of the results obtained by ESC, GA, and QGARAR in terms of throughput, efficiency, speedup, and makespan.

Table 5 The cost of computation for the second case

TSk/VLM	VLM_1	VLM_2	VLM_3
TSk_0	7	9	8
TSk_1	10	9	14
TSk_2	5	7	6
TSk_3	6	8	7
TSk_4	10	8	6
TSk_5	11	13	15
TSk_6	12	15	18
TSk_7	10	13	7
TSk_8	8	9	10
TSk_9	15	11	13
TSk_{10}	8	9	10

Table 6 Allocation that obtained by ESC for the second case

	VLM_1		VLM_2		VLM_3	
	Start_Time	Finish_Time	Start_Time	Finish_Time	Start_Time	Finish_Time
TSk_0	0	7	-	-	-	-
TSk_1	7	17	-	-	-	-
TSk_2	17	22	-	-	-	-
TSk_3	-	-	-	-	25	32
TSk_4	-	-	-	-	32	38
TSk_5	22	33	-	-	-	-
TSk_6	33	45	-	-	-	-
TSk_7	-	-	-	-	38	45
TSk_8	-	-	40	49	-	-
TSk_9	45	60	-	-	-	-
TSk_{10}	60	68	-	-	-	-

Table 7 The comparative outcomes for the second case

Algorithm	GA	QGARAR	ESC
Makespan	71	70	68

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

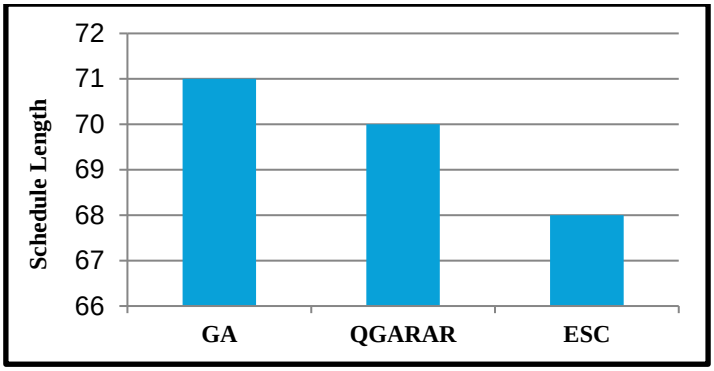

Fig. 5 Makespan comparison for the second case

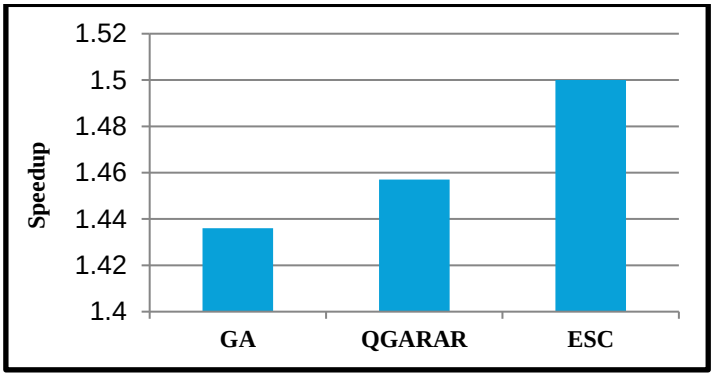

Fig. 6 Speedup comparison for the second case

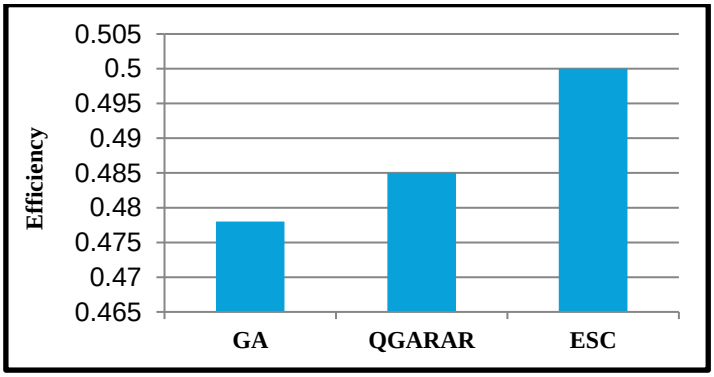

Fig. 7 Efficiency comparison for the second case

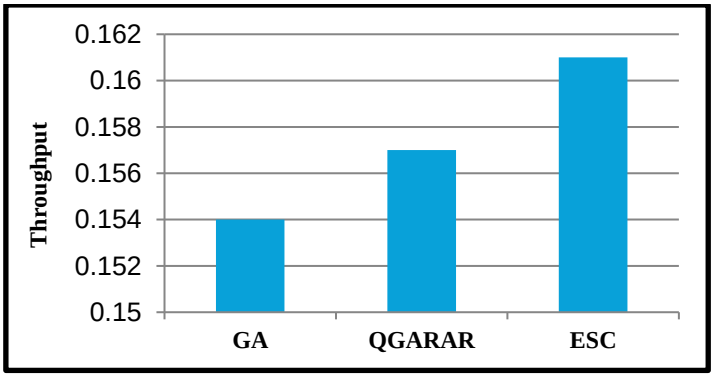

Fig. 8 Throughput comparison for the second case

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

6.3 Case Study 3

We examine a scenario involving eleven tasks scheduled across two heterogeneous virtual machines. The cost associated with executing each task on different virtual machines is outlined in Table 8 (Hosseini Shirvani, 2020). Additionally, Table 9 details the start and finish times of each task on various virtual machines, along with the schedule generated by ESC. We compare the results achieved by ESC with those obtained by GA (Xu et al., 2014) and particle Swarm Optimization (PSO) (Al Badawi & Shatnawi, 2013). The outcomes from ESC, GA, and PSO are summarized in Table 10. Furthermore, Figs. 9- 12 provides graphical representations of the results obtained by ESC, GA, and PSO in terms of throughput, efficiency, speedup, and makespan.

Table 8 The cost of computation for the third case

TSk/VLM	VLM_1	VLM_2
TSk_1	7	9
TSk_2	10	9
TSk_3	5	7
TSk_4	6	8
TSk_5	10	8
TSk_6	11	13
TSk_7	12	15
TSk_8	10	13
TSk_9	8	9
TSk_{10}	15	11
TSk_{11}	8	9

Table 9 Allocation that obtained by ESC for the third case

	VLM_1		VLM_2	
	Start_Time	Finish_Time	Start_Time	Finish_Time
TSk_1	0	7	-	-
TSk_2	7	17	-	-
TSk_3	-	-	21	28
TSk_4	17	23	-	-
TSk_5	23	33	-	-
TSk_6	33	44	-	-
TSk_7	-	-	28	43
TSk_8	-	-	43	56
TSk_9	44	52	-	-
TSk_{10}	-	-	56	67
TSk_{11}	-	-	67	76

Table 10 The comparative outcomes for the third case

Algorithm	GA	PSO	ESC
Makespan	78	77	76

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

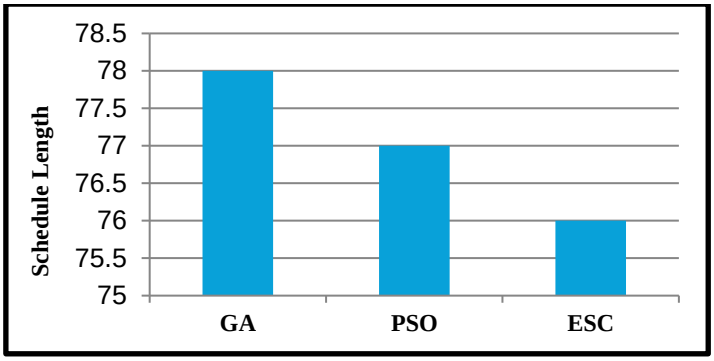

Fig. 9 Makespan comparison for the third case

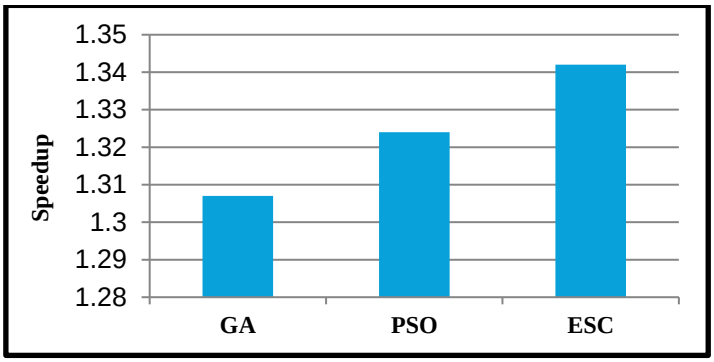

Fig. 10 Speedup comparison for the third case

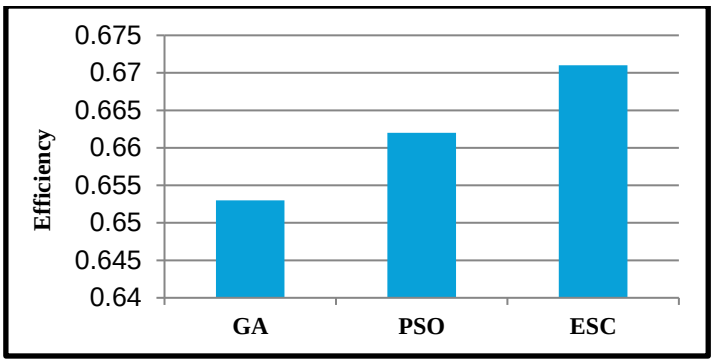

Fig. 11 Efficiency comparison for the third case

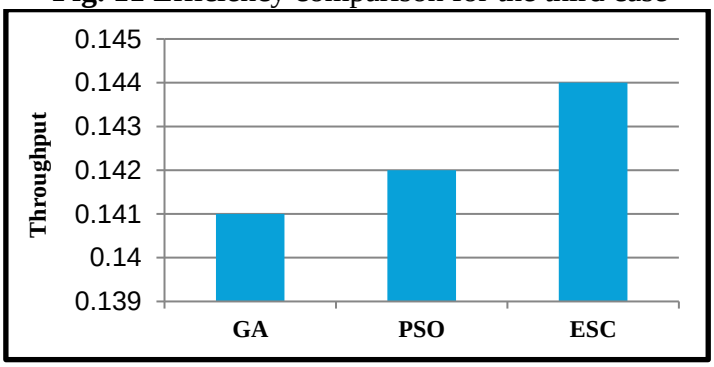

Fig. 12 Throughput comparison for the third case

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

7. Conclusion and Future Work

In this study, we presented an efficient sine cosine algorithm specifically designed for task scheduling in cloud computing environments. Our algorithm effectively allocates tasks to available virtual machines (VMs), optimizing key performance metrics. Through extensive testing using directed acyclic graphs (DAGs) representing various scenarios, our algorithm consistently outperformed existing methods in terms of throughput, makespan, efficiency, and speedup. These results highlight the algorithm's potential to enhance the performance and efficiency of cloud services. However, it is important to note the main limitation of our proposed method. The efficiency and accuracy of our algorithm are highly dependent on having precise and up-to-date information regarding resource availability and characteristics within the cloud environment. Inaccurate or outdated data can lead to suboptimal task allocations and performance degradation. Addressing this limitation will be a focus of our future work. Looking ahead, future work will focus on developing an advanced algorithm that further incorporates resource load balancing based on DAGs. This enhancement aims to improve resource utilization and service reliability, addressing the dynamic and heterogeneous nature of cloud environments. By continuing to refine and expand upon our current approach, we aim to contribute to the ongoing advancement of task-scheduling techniques in cloud computing.

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

References

Abdel-Basset, M., Mohamed, R., Abouhawwash, M., Chakrabortty, R. K., & Ryan, M. J. (2021). EA-MSCA: An effective energy-aware multi-objective modified sine-cosine algorithm for real-time task scheduling in multiprocessor systems: Methods and analysis. *Expert Systems with Applications*, *173*(February), 114699. https://doi.org/10.1016/j.eswa.2021.114699

Al Badawi, A., & Shatnawi, A. (2013). Static scheduling of directed acyclic data flow graphs onto multiprocessors using particle swarm optimization. *Computers and Operations Research*, *40*(10), 2322–2328. https://doi.org/10.1016/j.cor.2013.03.015

Alsubaei, F. S., Hamed, A. Y., Hassan, M. R., Mohery, M., & Elnahary, M. K. (2024). Machine learning approach to optimal task scheduling in cloud communication. *Alexandria Engineering Journal*, *89*(January), 1–30. https://doi.org/10.1016/j.aej.2024.01.040

Dubey, I., & Gupta, M. (2017). Uniform mutation and SPV rule based optimized PSO algorithm for TSP problem. *Proceedings of 2017 4th International Conference on Electronics and Communication Systems, ICECS 2017*, *17*, 168–172. https://doi.org/10.1109/ECS.2017.8067862

Gandhi, T., Nitin, & Alam, T. (2018). Quantum genetic algorithm with rotation angle refinement for dependent task scheduling on distributed systems. *2017 10th International Conference on Contemporary Computing, IC3 2017*, *2018-Janua*(August), 1–5. https://doi.org/10.1109/IC3.2017.8284295

Guo, L., Zhao, S., Shen, S., & Jiang, C. (2012). Task scheduling optimization in cloud computing based on heuristic Algorithm. *Journal of Networks*, *7*(3), 547–553. https://doi.org/10.4304/jnw.7.3.547-553

Hamed, A. Y., & Alkinani, M. H. (2021). Task scheduling optimization in cloud computing based on genetic algorithms. *Computers, Materials and Continua*, *69*(3), 3289–3301. https://doi.org/10.32604/cmc.2021.018658

Hosseini Shirvani, M. (2018). A new Shuffled Genetic-based Task Scheduling Algorithm in Heterogeneous Distributed Systems. *Heterogeneous Distributed Systems. J. Adv. Comput. Res*, *9*(4), 19–36. http://jacr.iausari.ac.ir/article_660143.html

Hosseini Shirvani, M. (2020). A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems. *Engineering Applications of Artificial Intelligence*, *90*(September 2019), 103501. https://doi.org/10.1016/j.engappai.2020.103501

Keshanchi, B., & Navimipour, N. J. (2016). Priority-based task scheduling in the cloud systems using a memetic algorithm. *Journal of Circuits, Systems and Computers*, *25*(10), 1–33. https://doi.org/10.1142/S021812661650119X

Mathew, T., Sekaran, K. C., & Jose, J. (2014). Study and analysis of various task scheduling algorithms in the cloud computing environment. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, 658–664. https://doi.org/10.1109/ICACCI.2014.6968517

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems*, *96*, 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

Senthil Kumar, A. M., & Venkatesan, M. (2019). Task scheduling in a cloud computing environment using HGPSO algorithm. *Cluster Computing*, *22*(s1), 2179–2185. https://doi.org/10.1007/s10586-018-2515-2

Su, S., Li, J., Huang, Q., Huang, X., Shuang, K., & Wang, J. (2013). Cost-efficient task

Online ISSN: 3009-7916, Print ISSN: 1110-0184. https://astb.journals.ekb.eg/

scheduling for executing large programs in the cloud. *Parallel Computing*, *39*(4–5), 177–188. https://doi.org/10.1016/j.parco.2013.03.002

Tawfeek, M., El-Sisi, A., Keshk, A., & Torkey, F. (2015). Cloud task scheduling based on ant colony optimization. *International Arab Journal of Information Technology*, *12*(2), 129–137.

Wang, L., Pan, Q. K., & Tasgetiren, M. F. (2011). A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Computers and Industrial Engineering*, *61*(1), 76–83. https://doi.org/10.1016/j.cie.2011.02.013

Xu, Y., Li, K., Hu, J., & Li, K. (2014). A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. *Information Sciences*, *270*, 255–287. https://doi.org/10.1016/j.ins.2014.02.122