Applied Science and Innovative Research
ISSN 2474-4972 (Print) ISSN 2474-4980 (Online)
Vol. 7, No. 2, 2023
www.scholink.org/ojs/index.php/asir

Original Paper

Task Scheduling Optimization in Cloud Computing by Jaya Algorithm

Ahmed Y. Hamed[1], M. Kh. Elnahary[1] & Hamdy H. El-Sayed[1]

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

Received: March 2, 2023 Accepted: March 16, 2023 Online Published: March 20, 2023

doi:10.22158/asir.v7n2p30 URL: http://doi.org/10.22158/asir.v7n2p30

Abstract

Cloud computing provides resources to its consumers as a service. The cloud computing paradigm offers dynamic services by providing virtualized resources via the internet for enabling applications, and these services are provided by large-scale data centers known as clouds. Cloud computing is entirely reliant on the internet to provide its services to consumers. Cloud computing offers several advantages, including the fact that users only pay for what they use weekly, monthly, or yearly, that anybody with an internet connection may use the cloud, and that there is no need to purchase resources, hardware, or software on their own. This paper proposes an efficient task scheduling algorithm based on the Jaya algorithm for the cloud computing environment. We evaluate the performance of our method by applying it to three instances. The recommended technique produced the optimal solution in makespan, speedup, efficiency, and throughput, according to the findings.

Keywords

Heterogeneous resources, Jaya algorithm, Task scheduling, Cloud Computing

1. Introduction

Task scheduling in heterogeneous computing systems linked by high-speed networks has received a lot of attention. Such approaches promise the quick processing of computationally heavy applications with a wide range of calculation requirements. A significant application can be divided into several smaller subtasks before parallel processing. These smaller subtasks usually involve dependencies that indicate precedence restrictions, in which the outcomes of other subtasks are necessary before a particular subtask may be done. Decomposing a computation into smaller subtasks and performing the subtasks on several processors might lower the computation's overall execution time, i.e., the makespan. As a result, a task scheduling algorithm usually schedules all subtasks on a given number of available

processors to minimize makespan while respecting precedence restrictions. The development of task scheduling algorithms that allocate subtasks of an application to processors is a difficulty in heterogeneous computing systems. As a result, various techniques for minimizing makespan for parallelizing subtasks with precedence connections have been presented. The precedence connections are represented as a directed acyclic graph (DAG) with vertices representing computations and directed edges describing dependencies between those vertices. DAGs have been demonstrated to be expressive for various applications (Xu, Li, Hu, & Li, 2014). This paper introduced an efficient algorithm based on the Jaya algorithm called the efficient Jaya algorithm (EJA) to lower the makespan and optimize the speedup, efficiency, and throughput to handle the task scheduling problem successfully.

The paper is organized as follows: The notations are presented in section 2. Related work is presented in Section 3. The problem description is given in Section 4. The Jaya algorithm is given in Section 5. Section 6 describes the EJA approach. The evaluation of the proposed algorithm is presented in section 7. Section 8 concludes and offers future work.

2. Notation

GRT	It refers to the graph of tasks
TAS_i	It refers to the task i
VTM_i	It refers to the virtual machine i
NVTM	It refers to a virtual machine's number
NTAS	It refers to the number of tasks
$COM\_CO(TAS_i, TAS_j)$	It refers to the communication cost between TAS_i and TAS_j
$Sta\_Time(TAS_i, VTM_j)$	It refers to the start time of task i on a VTM_j
$Fts\_Time(TAS_i, VTM_j)$	It refers to the finish time of task i on a VTM_j
$Rey\_Time(VTM_i)$	It refers to the VTM's ready time i
DALT	It refers to a list of tasks arranged in topological order of DAG
$Data\_Arr(TS_i, VM_j)$	It refers to the time of task's i data arrival to VTM_j

3. Related Work

A heuristic-based task scheduling approach in parallel and distributed heterogeneous computing systems generally consists of two phases: job prioritization and processor selection. Varying priority produces different makespan on a heterogeneous computing system in a heuristic-based job scheduling method. As a result, an intelligent scheduling algorithm should be able to give priority to each subtask based on the resources required to reduce makespan. This work (Xu, Li, Hu, & Li, 2014) proposes a job scheduling approach for heterogeneous computing systems based on a multiple priority queues genetic algorithm (MPQGA). The primary concept behind our method is to use the benefits of both evolutionary and heuristic algorithms while avoiding their downsides. The suggested technique uses a

genetic algorithm (GA) approach to prioritize each subtask while searching for a solution for the task-to-processor mapping using a heuristic-based earliest completion time (EFT). The MPQGA approach also creates crossover, mutation, and fitness functions appropriate for directed acyclic graph (DAG) scheduling.

Cloud computing provides resources to its consumers as a service. The cloud computing paradigm offers dynamic services by providing virtualized resources via the internet for enabling applications, and these services are provided by large-scale data centers known as clouds. Cloud computing is entirely reliant on the internet to provide its services to consumers. Cloud computing offers several advantages, including the fact that users only pay for what they use (weekly, monthly, or yearly), that anybody with an internet connection may use the cloud, and that there is no need to purchase resources (hardware, software) on their own. This study (June, 2014) introduces a novel approach, Hybrid enhanced particle swarm optimization with mutation crossover, to get the most out of resources. Optimized resource use is vital, and scheduling plays a significant role.

Cloud computing has lately experienced rapid growth and has emerged as a commercial reality in information technology. Cloud computing is a supplement, consumption, and delivery model for internet-based Information Technology services charged per usage. The scheduling of cloud services influences the cost-benefit of this computing paradigm by service providers to users. Tasks should be scheduled efficiently in such a circumstance to decrease execution cost and time. In this research (Kaur, & Verma, 2012), the authors suggested a meta-heuristic-based scheduling method that reduces execution time and cost. An enhanced genetic algorithm is created by combining two existing scheduling methods for scheduling activities while considering their computational complexity and computing capability of processing elements.

Due to the expansion of data centers' size, complexity, and performance, client needs in execution time and throughput has become increasingly complicated. Against this backdrop, this paper introduces a new resource allocation model that improves task scheduling by combining a multi-objective optimization (MOO) and particle swarm optimization (PSO) technique. The authors create a novel multi-objective PSO (MOPSO) algorithm based on a unique ranking technique. This algorithm's fundamental idea is that jobs are assigned to virtual machines to reduce waiting time and maximize system throughput (Alkayal, Jennings, & Abulkhair, 2016).

When excellent efficiency is required, task scheduling is one of the essential concerns in heterogeneous contexts. Because task scheduling is a Nondeterministic Polynomial (NP)-hard issue, various evolutionary methods have been developed to address it. Because population-based algorithms have a slow convergence rate, they are combined with local search algorithms. Thus, in this study (Dordaie & Navimipour, 2018), a hybrid particle swarm optimization and hill-climbing method are suggested to improve the task scheduling makespan.

4. Problem Description

In cloud computing, task scheduling is represented as a graph with NTAS tasks (TAS_1, TAS_2, TAS_3, ..., TAS_{NTAS}). Each node (task) with GRT and E-directed edges represents a subset of the tasks' requests (Hamed & Alkinani, 2021). Each node (task) represents an instruction that may be executed sequentially on the same virtual machine as other instructions; it may have one or more inputs. The availability of the inputs triggers the execution of an exit or entry task. A partial request with a precedence constraint ($TAS_i \rightarrow TAS_j$), i.e., TAS_i precedes TAS_j in the implementation process. The execution time of a task TAS_i is denoted by (TAS_i) weight. Let $COM\_CO(TAS_i, TAS_j)$ be the cost of communication of an edge, and it will be equal to zero if TAS_i and TAS_j are scheduled on the same virtual machine. Start and finish times are denoted by $Sta\_Time(TAS_i, VTM_j)$ and $Fts\_Time(TAS_i, VTM_j)$, respectively [6]. The Data_Arr of TAS_i at virtual machine VTM_j is given by:

$$Data\_Arr(TAS_i. VTM_j) = \max\{Fts\_Time(TAS_k, VTM_j) + COM\_CO(TAS_i, TAS_k)\} \quad (1)$$

Where k = 1.2, ..., number of Parents

The task scheduling problem in cloud computing may be defined as determining the best time to allocate or schedule the start times of the specified tasks on virtual machines. While maintaining precedence is restricted, the completion time (schedule length) and execution cost decrease. The completion time is defined as the schedule length or finishes time calculated as follows:

$$\text{Scheduled Length} = \max(Fts\_Time(TAS_i, VTM_j)) \quad (2)$$

$$Fts\_Time(TAS_i, VTM_j) = Sta\_Time(TAS_i, VTM_j) + WET_{ij} \quad (3)$$

Where i = 1.2. ...., NTAS, and j = 1,2, …NVTM

Algorithm 1: To find the schedule length (Hamed & Alkinani, 2021)

Input the schedule of tasks as shown in Table 1

$Rey\_Time[VTM_j] = 0$ where $j = 1, 2, \ldots NVTM$.

For i = 1 : NTAS

{

From DALT take the first task TAS_i to be executed and remove it from DALT.

For j = 1 : NVTM

{

If TAS_i is scheduled to virtual machine VTM_j

$$Sta\_Time(TAS_i, VTM_j) = \max\{Ret\_Time(VTM_j), Data\_Arr(TAS_i, VTM_j)\}$$

$Fts\_Time(TAS_i, VTM_j) = Sta\_Time(TAS_i, VTM_j) + WET(TAS_i, VTM_j)$

$Rey\_Time(VTM_j) = Fts\_Time(TAS_i, VTM_j)$

End If

}

}

Schedule length = max(Fts_Time)

5. Jaya Algorithm

Let G(y) be the objective function that has to be minimized (or maximized). Assume that there are 'm' number of design variables (i.e. j=1,2,...,m) and 'n' number of possible solutions (i.e. population size, k=1,2,...,n) at any iteration i. Let the best candidate acquire the best value of G(y) (i.e., G(y)best) in all candidate solutions, and the worst candidate obtain the worst value of G(y) (i.e., G(y)worst) in all candidate solutions. If the value of the jth variable for the kth candidate during the ith iteration is Yj,k,i, then this value is updated according to the following Eq (4) (Venkata Rao, 2016).

$$Y'_{j,k,i} = Y_{j,k,i} + ran_{1,j,i}\,(Y_{j,best,i} - |Y_{j,k,i}|) - ran_{2,j,i}\,(Y_{j,worst,i} - |Y_{j,k,i}|) \quad (4)$$

where $Y_{j,best,i}$ is the variable j value for the best candidate and $Y_{j,worst,i}$ is the variable j value for the worst candidate $Y'_{j,k,i}$ is the updated value of $Y_{j,k,i}$, and $ran_{1,j,i}$ and $ran_{2,j,i}$ are the two random values for the j^{th} variable in the range [0, 1] during the i^{th} iteration. The word "$ran_{1,j,i}$ $((Y_{j,best,i} - Y_{j,k,i})$" denotes the solution's propensity to get closer to the best solution, whereas the term "- $ran_{2,j,i}$ $(Y_{j,worst,i} - Y_{j,k,i})$" denotes the solution's inclination to avoid the worst solution. If $Y'_{j,k,i}$ yields a superior function value, it is accepted. At the end of the iteration, all of the acceptable function values are kept, and these values form the input to the following iteration (Venkata Rao, 2016).

Jaya algorithm

```
Initialize population size, number of design variables, and termination criterion
Iteration=1
While (iteration <= termination criterion)
        Identify the best    and the worst solutions in the population
        Modify the solutions based on the best and the worst solutions by using Eq.(4)
        If the solution of Y'j,k,i is better than Yj,k,i
            Update the old solution with the new obtained solution
        End if
        Iteration =iteration +1
End while
```

6. The Proposed Algorithm

Because it is evident that the vector representation in the Jaya algorithm is in continuous value form, we will utilize the five ways to transform these continuous values into discrete values. The first rule is the Smallest Position Value (SPV) (Dubey & Gupta, 2017), the second is the Largest Position Value (LPV) (Wang, Pan, & Tasgetiren, 2011), the third is the round nearest function, the fourth is the floor nearest function, and the fifth is the Ciel nearest function. Table 1 shows how we will utilize the modulus function with the number of virtual machines in the SPV and LPV to raise the result by one.

Table 1. Convert Continuous Values to Discrete Values

Population	1.0	1.6	1.4	3.0	2.3	1.9	2.0
SPV rule	1	3	2	6	7	5	4
modulus with SPV and NVRM=3	2	1	3	1	2	3	2
LPV rule	4	5	7	6	2	3	1
modulus with LPV and NVRM=3	2	3	2	1	3	1	2
round nearest function	1	2	1	3	2	2	2
floor nearest function	1	1	1	3	2	1	2
ceil nearest function	1	2	2	3	3	2	2

Algorithm 2: The function that converts a continuous value to a discrete value

```
Function convert_to_discrete (u)
Rand=random number between [1…5]
If (Rand == 1)
        Transform the continuous values by the SPV rule
Else if (Rand == 2)
        Transform the continuous values by the LPV rule
Else if (Rand == 3)
        Transform the continuous values by round the nearest function
Else if (Rand == 4)
        Transform the continuous values by the nearest function
Else
        Transform the continuous values by ceil nearest function
End if
End function
```

Algorithm 3: EJA

```
Input DAG with communication and computation cost
Initialize population size, number of design variables, lower bound, upper bound, and termination criterion
Initialize population by using population = lower bound + rand * (upper bound – lower bound)
Convert the Initialize population by using Algorithm 2
Calculate the schedule length by using Algorithm 1
Iteration=1
While (iteration <= termination criterion)
Identify the best and the worst solutions in the population
```

Modify the solutions based on the best and the worst solutions by using Eq.(4)
Convert the obtained solution by using **Algorithm 2**
Calculate the schedule length by using **Algorithm 1**
If the solution of Y'j,k,i is better than Yj,k,i
Update the old solution with the new obtained solution
End if
Iteration =iteration +1
End while

7. Evaluation of the EJA

We demonstrate the EJA's performance by applying it to three instances. The first scenario has ten tasks and three heterogeneous virtual machines, and the second scenario has ten tasks and three heterogeneous virtual machines. The third is made up of three heterogeneous virtual machines and eleven tasks.

$$\text{Speedup} = \min_{VTM_j} \left(\sum_{TAS_i} \frac{WET_{i,j}}{\text{schedule length}} \right) \quad (5)$$

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{NVTM}} \quad (6)$$

$$\text{Throughput} = \frac{\text{NTAS}}{\text{Schedule Length}} \quad (7)$$

7.1 Case 1

In this case, the tasks {TAS_1, TAS_2, TAS_3, TAS_4, TAS_5, TAS_6, TAS_7, TAS_8, TAS_9, TAS_{10}} are executed on three heterogeneous virtual machines {VTM_1, VTM_2, VTM_3}. The cost of executing each task on different virtual machines is shown in Table 2 (Younes, Ben Salah, Farag, Alghamdi, & Badawi, 2019). The schedule obtained by EJA is shown in Table 3. The results obtained by the EJA are compared with those obtained by the Whale Optimization Algorithm (WOA) (Thennarasu, Selvam, & Srihari, 2021), Gravitational Search Algorithm (GSA) (Biswas, Kuila, Ray, & Sarkar, 2019), and Hybrid Heuristic and Genetic-based scheduling task algorithm for heterogeneous computing (HHG) (Sulaiman, Halim, Lebbah, Waqas, & Tu, 2021). The results obtained by the EJA and WOA, GSA, and HHG are illustrated in Table 4. The task priority of EJA {TAS_1, TAS_6, TAS_4, TAS_5, TAS_2, TAS_3, TAS_8, TAS_9, TAS_7, TAS_{10}}. Figure 1, Figure 2, Figure 3, and Figure 4 represent the results obtained by the EJA, WOA, GSA, and HHG in terms of makespan, speedup, efficiency, and throughput.

Table 2. Computation Cost for Case 1

TAS/ VTM	VTM_1	VTM_2	VTM_3
TAS_1	22	21	36
TAS_2	22	18	18
TAS_3	32	27	43
TAS_4	7	10	4
TAS_5	29	27	35
TAS_6	26	17	24
TAS_7	14	25	30
TAS_8	29	23	36
TAS_9	15	21	8
TAS_{10}	13	16	33

Table 3. Schedule Obtained by EJA for Case 1

	VTM_1		VTM_2		VTM_3	
	Sta_Time	Fts_Time	Sta_Time	Fts_Time	Sta_Time	Fts_Time
TAS_1	-	-	0	21	-	-
TAS_2	-	-	21	39	-	-
TAS_3	-	-	39	66	-	-
TAS_4	64	71	-	-	-	-
TAS_5	-	-	-	-	34	69
TAS_6	38	64	-	-	-	-
TAS_7	-	-	66	91	-	-
TAS_8	71	100	-	-	-	-
TAS_9	-	-	-	-	78	86
TAS_{10}	100	113	-	-	-	-

Table 4. The Comparative Results for Case 1

Algorithm	Makespan
WOA	122
GSA	122
HHG	117
EJA	113

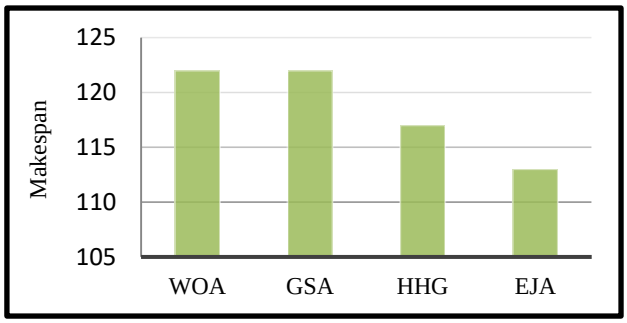

Figure 1. Comparison of Makespan for Case 1

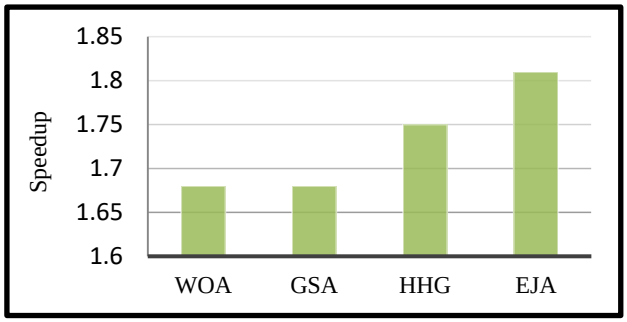

Figure 2. Comparison of Speedup for Case 1

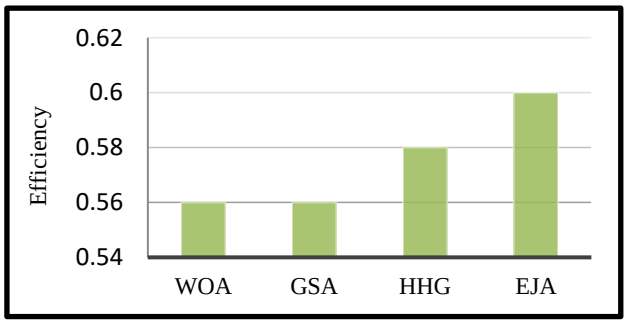

Figure 3. Comparison of Efficiency for Case 1

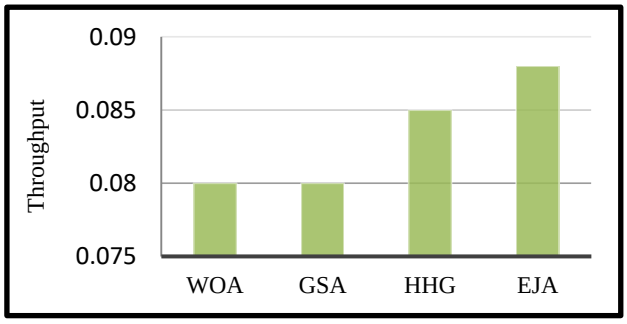

Figure 4. Comparison of Throughput for Case 1

7.2 Case 2

In this case, the tasks {TAS_1, TAS_2, TAS_3, TAS_4, TAS_5, TAS_6, TAS_7, TAS_8, TAS_9, TAS_{10}} are executed on three heterogeneous virtual machines {VTM_1, VTM_2, VTM_3}. The cost of executing each task on different virtual machines is shown in Table 5 (Younes, Ben Salah, Farag, Alghamdi, & Badawi, 2019). The schedule obtained by EJA is shown in Table 6. The results obtained by the EJA are compared with those obtained by the Ant Colony Optimization (ACO) (Tawfeek, El-Sisi, Keshk, & Torkey, 2015), Heterogeneous Earliest Finish Time (HEFT) (Topcuoglu, Hariri, & Wu, 2002), and Critical Path on Processor (CPOP) (Topcuoglu, Hariri, & Wu, 2002). The results obtained by the EJA and ACO, HEFT, and CPOP are illustrated in Table 7. The task priority of EJA { TAS1, TAS4, TAS3, TAS2, TAS5, TAS6, TAS9, TAS8, TAS7, TAS10}. Figure 5, Figure 6, Figure 7, and Figure 8 represent the results obtained by the EJA, ACO, HEFT, and CPOP in terms of makespan, speedup, efficiency, and throughput.

Table 5. Computation Cost for Case 1

TAS/ VTM	VTM_1	VTM_2	VTM_3
TAS_1	14	16	9
TAS_2	13	19	18
TAS_3	11	13	19
TAS_4	13	8	17
TAS_5	12	13	10
TAS_6	13	16	9
TAS_7	7	15	11
TAS_8	5	11	14
TAS_9	18	12	20
TAS_{10}	21	7	16

Table 6. Schedule Obtained by EJA for Case 2

	VTM_1		VTM_2		VTM_3	
	Sta_Time	Fts_Time	Sta_Time	Fts_Time	Sta_Time	Fts_Time
TAS_1	-	-	-	-	0	9
TAS_2	-	-	-	-	9	27
TAS_3	21	32	-	-	-	-
TAS_4	-	-	18	26	-	-
TAS_5	-	-	26	39	-	-
TAS_6	-	-	-	-	27	36
TAS_7	32	39	-	-	-	-
TAS_8	-	-	55	66	-	-

TAS_9	-	-	43	55	-	-
TAS_{10}	-	-	66	73	-	-

Table 7. The Comparative Results for Case 2

Algorithm	Makespan
CPOP	86
HEFT	80
ACO	78
EJA	73

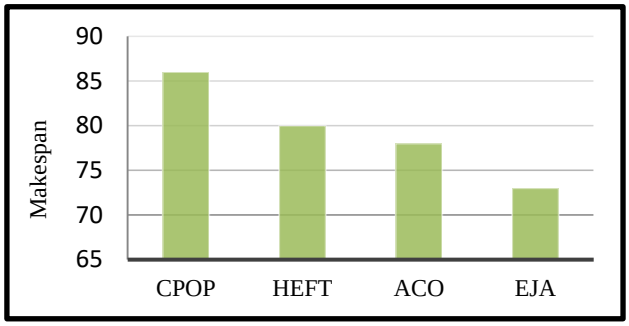

Figure 5. Comparison of Makespan for Case 2

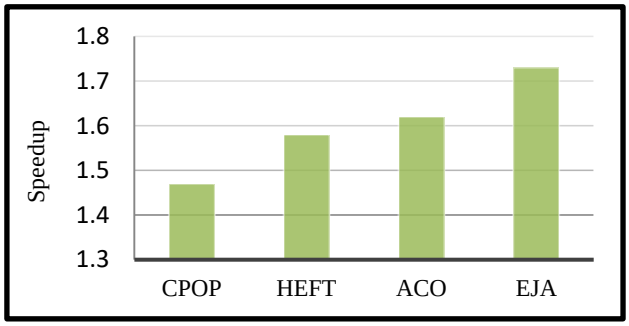

Figure 6. Comparison of Speedup for Case 2

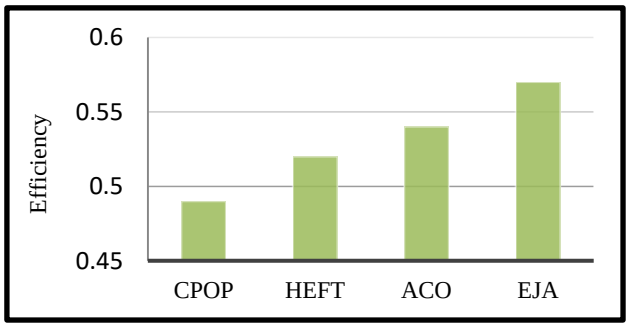

Figure 7. Comparison of Efficiency for Case 2

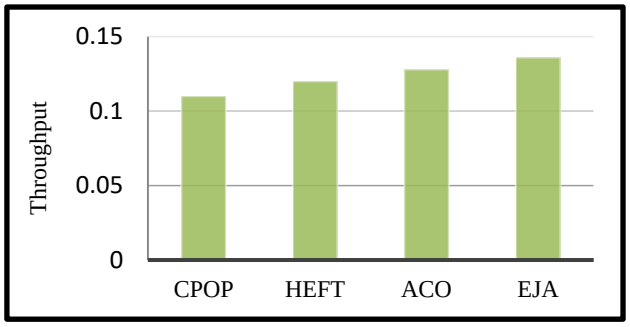

Figure 8. Comparison of Throughput for Case 2

7.3 Case 3

In this case, the tasks {TAS_1, TAS_2, TAS_3, TAS_4, TAS_5, TAS_6, TAS_7, TAS_8, TAS_9, TAS_{10}} are executed on three heterogeneous virtual machines {VTM_1, VTM_2, VTM_3}. The cost of executing each task on different virtual machines is shown in Table 8 (Keshanchi, Souri, & Navimipour, 2017). The schedule obtained by EJA is shown in Table 9. The results obtained by the EJA are compared with those obtained by the Multiple Priority Queues and a Memetic Algorithm (MPQMA) (Keshanchi, Souri, & Navimipour, 2017), a New Genetic Algorithm (NGA) (Keshanchi, Souri, & Navimipour, 2017). The results obtained by the EJA, MPQMA, and NGA are illustrated in Table 10. The task priority of EJA {TAS_0, TAS_2, TAS_3, TAS_4, TAS_1, TAS_6, TAS_8, TAS_7, TAS_5, TAS_9, TAS_{10}}. Figure 9, Figure 10, Figure 11, and Figure 12 represent the results obtained by the EJA, MPQMA, and NGA in terms of makespan, speedup, efficiency, and throughput.

Table 8. Computation Cost for Case 3

TAS / VTM	VTM_1	VTM_2	VTM_3
TAS_0	10	11	12
TAS_1	11	12	13
TAS_2	12	8	13
TAS_3	14	10	18
TAS_4	27	20	19
TAS_5	15	12	18
TAS_6	9	14	19
TAS_7	19	12	14
TAS_8	14	10	15
TAS_9	15	12	15
TAS_{10}	18	10	17

Table 9. Schedule OBTained by EJA for CAse 3

	VTM_1		VTM_2		VTM_3	
	Sta_Time	Fts_Time	Sta_Time	Fts_Time	Sta_Time	Fts_Time
TAS_0	-	-	0	11	-	-
TAS_1	36	47	-	-	-	-
TAS_2	-	-	11	19	-	-
TAS_3	-	-	19	29	-	-
TAS_4	-	-	29	49	-	-
TAS_5	47	62	-	-	-	-
TAS_6	-	-	-	-	29	48
TAS_7	-	-	59	71	-	-
TAS_8	-	-	49	59	-	-
TAS_9	-	-	71	83	-	-
TAS_{10}	-	-	83	93	-	-

Table 10. The Comparative Results for Case 3

Algorithm	Makespan
MPQMA	101
NGA	101
EJA	93

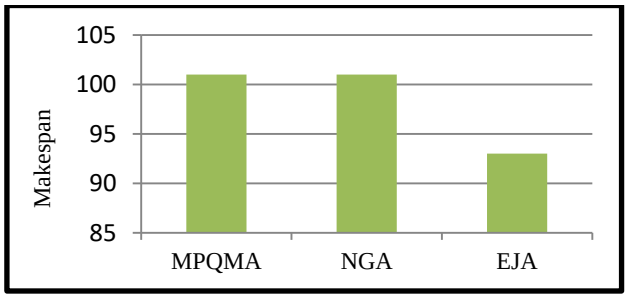

Figure 9. Comparison of Makespan for Case 3

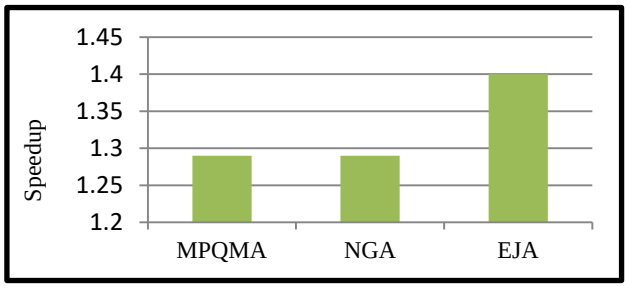

Figure 10. Comparison of Speedup for Case 3

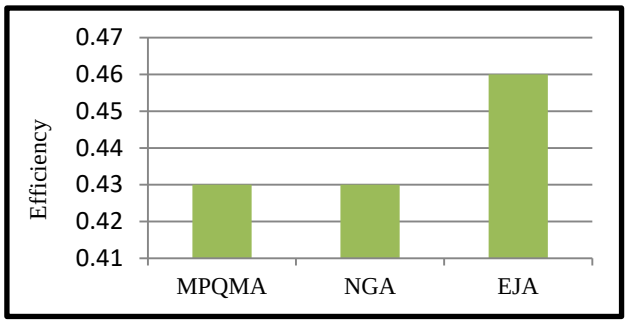

Figure 11. Comparison of Efficiency for Case 3

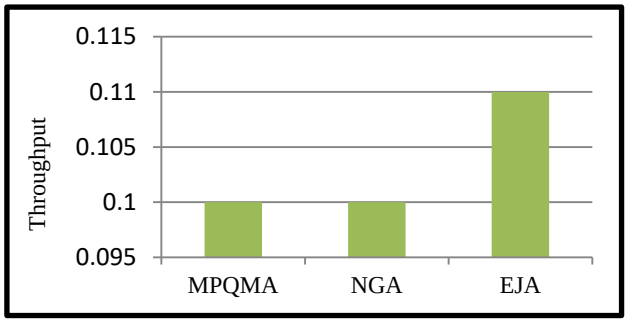

Figure 12. Comparison of Throughput for Case 3

8. Conclusion and Future Work

The suggested efficient Jaya algorithm allocates or schedules subtasks to available virtual machines in a cloud computing context. According to the findings obtained on DAGs of various situations, the efficient Jaya algorithm outperforms other algorithms in terms of makespan, speedup, efficiency, and throughput. In the future, we will create an algorithm based on DAGs that will consider resource load balancing.

References

Alkayal, E. S., Jennings, N. R., & Abulkhair, M. F. (2016). Efficient Task Scheduling Multi-Objective Particle Swarm Optimization in Cloud Computing. *Proc. - Conf. Local Comput. Networks, LCN.* (2016), 17-24. https://doi.org/10.1109/LCN.2016.024

Biswas, T., Kuila, P., Ray, A. K., & Sarkar, M. (2019). Gravitational search algorithm based novel workflow scheduling for heterogeneous computing systems. *Simul. Model. Pract. Theory. 96*(2019), 101932. https://doi.org/10.1016/j.simpat.2019.101932

Dordaie, N., & Navimipour, N. J. (2018). A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments. *ICT Express*, *4*(2018), 199-202. https://doi.org/10.1016/j.icte.2017.08.001

Dubey, I., & Gupta, M. (2017). Uniform mutation and SPV rule based optimized PSO algorithm for TSP problem. *Proc. 2017 4th Int. Conf. Electron. Commun. Syst. ICECS 2017*, *17*(2017), 168-172. https://doi.org/10.1109/ECS.2017.8067862

Hamed, A. Y., & Alkinani, M. H. (2021). Task scheduling optimization in cloud computing based on genetic algorithms, Comput. Mater. *Contin*, *69*(2021), 3289-3301. https://doi.org/10.32604/cmc.2021.018658

June, A. (2014). Research Paper on Optimized Utilization of Resources Using PSO and Improved Particle Swarm Optimization (IPSO). *Algorithms in Cloud Computing*, *2*(2014), 499-505.

Kaur, S., & Verma, A. (2012). An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment. *Int. J. Inf. Technol. Comput. Sci. 4*(2012), 74-79. https://doi.org/10.5815/ijitcs.2012.10.09

Keshanchi, B., Souri, A., & Navimipour, N. J. (2017). An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. *J. Syst. Softw. 124*(2017), 1-21. https://doi.org/10.1016/j.jss.2016.07.006

Sulaiman, M., Halim, Z., Lebbah, M., Waqas, M., & Tu, S. (2021). An Evolutionary Computing-Based Efficient Hybrid Task Scheduling Approach for Heterogeneous Computing Environment. *J. Grid Comput. 19*(2021). https://doi.org/10.1007/s10723-021-09552-4

Tawfeek, M., El-Sisi, A., Keshk, A., & Torkey, F. (2015). Cloud task scheduling based on ant colony optimization. *Int. Arab J. Inf. Technol. 12*(2015), 129-137.

Thennarasu, S. R., Selvam, M., & Srihari, K. (2021). A new whale optimizer for workflow scheduling in cloud computing environment. *J. Ambient Intell. Humaniz. Comput. 12*(2021), 3807-3814. https://doi.org/10.1007/s12652-020-01678-9

Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst. 13*(2002), 260-274. https://doi.org/10.1109/71.993206

Venkata Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput. 7*(2016), 19-34. https://doi.org/10.5267/j.ijiec.2015.8.004

Wang, L., Pan, Q. K., & Tasgetiren, M. F. (2011). A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Comput. Ind. Eng. 61*(2011), 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. *Inf. Sci. (Ny). 270*(2014), 255-287. https://doi.org/10.1016/j.ins.2014.02.122

Younes, A., Ben Salah, A., Farag, T., Alghamdi, F. A., & Badawi, U. A. (2019). Task scheduling algorithm for heterogeneous multi processing computing systems. *J. Theor. Appl. Inf. Technol. 97*(2019), 3477-3487.