Information Sciences Letters

Volume 12
Issue 3 *Mar. 2023*

Article 47

2023

An Efficient Firefly Algorithm for Optimizing Task Scheduling in Cloud Computing Systems

Ahmed Y. Hamed
Department of Computer Science, Faculty of Computers and Artificial Intelligence, Sohag University, Sohag, 82524, Egypt, L.Maghrabi@ubt.edu.sa

M. Kh. Elnahary
Department of Computer Science, Faculty of Computers and Artificial Intelligence, Sohag University, Sohag, 82524, Egypt, L.Maghrabi@ubt.edu.sa

Hamdy H. El-Sayed
Department of Computer Science, Faculty of Computers and Artificial Intelligence, Sohag University, Sohag, 82524, Egypt, L.Maghrabi@ubt.edu.sa

Louai Maghrabi
Department of Software Engineering, College of Engineering, University of Business and Technology, Jeddah, Saudi Arabia, L.Maghrabi@ubt.edu.sa

Information Sciences Letters
An International Journal

http://dx.doi.org/10.18576/isl/120348

An Efficient Firefly Algorithm for Optimizing Task Scheduling in Cloud Computing Systems

Ahmed Y. Hamed[1], *M. Kh. Elnahary*[1], *Hamdy H. El-Sayed*[1] *and Louai Maghrabi*[2,*]

[1]Department of Computer Science, Faculty of Computers and Artificial Intelligence, Sohag University, Sohag, 82524, Egypt
[2]Department of Software Engineering, College of Engineering, University of Business and Technology, Jeddah, Saudi Arabia

Received: 27 Jul. 2022, Revised: 2 Nov. 2022, Accepted: 5 Nov. 2022.
Published online: 1 Mar. 2023.

Abstract: As user service demands change constantly, task scheduling becomes an extremely significant study area within the cloud environment. The goal of scheduling is distributing the tasks on available processors in order to achieve the shortest possible makespan while adhering to priority constraints. In heterogeneous cloud computing resources, task scheduling has a large influence on system performances. The various processes in the heuristic-based algorithm of scheduling will result in varied makespans when heterogeneous resources are utilized. As a result, a smart method of scheduling must be capable of establishing precedence efficacy for each task to decrease makespan time. In our study, we develop a novel efficient method of scheduling tasks according to the firefly algorithm to tackle an essential task and schedule a heterogeneous cloud computing problem. We evaluate the performance of our algorithm by putting it through three situations with changing amounts of processors and numbers of tasks. The findings of the experiment reveal that our suggested technique found optimal solutions substantially more frequently in terms of makespan time when compared with other methods.

Keywords: heterogeneous resources, firefly algorithm, task scheduling, cloud computing.

1 Introduction

The distributed computing concept underpins the model of the cloud and consists of a collection of virtual machines that may be dynamically linked to build computing resources. Unutilized computing resources should be employed internationally to raise the rate of utilization and gain resources by improving their economic efficiency. The primary objective of the model of the cloud is to share users' data and resources. There are three forms of cloud computing: the first is IaaS, which refers to Infrastructure as a Service; the second is SaaS, which refers to Software as a Service; and the third is PaaS, which refers to Platform as a Service [1]. All of the services are available to users on a pay-as-you-go basis, and computing resources, applications, servers, networks, and data storage are shared. In a service SaaS, the licensed program is made available to the user on a subscription basis. The services can be accessed via a web browser from any device. In a service PaaS, the user can construct customized applications utilizing the services of the cloud and then publish them on their device. In a service IaaS, the client's organizational architecture is accessible online. The consumer is not required to comprehend the internal structure of the required infrastructure in order to use this service [1]. Rather than purchasing the complete corporate infrastructure, the consumer accepts only the necessary services as they are required. Because the number of cloud users has expanded in recent years, by default, the number of tasks that must be managed have increased for scheduling activities [1].

To address the issue of scheduling tasks more efficiently, we suggest a new approach based on the firefly algorithm to minimize the makespans of user requests on resources. Our Proposed Firefly Algorithm (PFA) minimizes makespans. In the firefly algorithm, the representation of a vector is a continuous value, so we use five manners that convert the continuous value to a discrete value. The priority is generated randomly to preserve the precedence constraints. These modifications improve the global optimization of the firefly algorithm's performance. We assess the performance of the PFA by putting it through three situations with changing amounts of processors and numbers of tasks. In terms of makespan, the results confirm that the PFA reaches the best solutions faster and more efficiently than other algorithms.

*Corresponding author e-mail: L.Maghrabi@ubt.edu.sa

The structure of this paper follows. The notations are provided in Section 2. Section 3 presents some related work on the task scheduling problem for various system architectures. The description of the issue of scheduling is presented in Section 4. The firefly algorithm is stated in Section 5. Our PFA is shown in Section 6. The results were obtained by applying the PFA and the comparisons with other results, as described in Section 7. The discussion is offered in Section 8, and Section 9 includes the conclusion of the paper and future work.

2 Notations

Tak_i	The Task i
Pro_i	The Processor i
MPR	The Processor count
NTK	The task count
$CCO(Tak_i, Tak_j)$	The cost between Tak_i and Tak_j
$STA(Tak_i, Pro_j)$	Tak_i start time on a Pro_j
$FNT(Tak_i, Pro_j)$	Tak_i finish time on a Pro_j
$RYT(Pro_i)$	Ready time of the Pro_i
LST	A list of jobs arranged in DAG topological order
$DATVE(Tak_i, Pro_j)$	Tak_i time of data arrival at Pro_j

3 Related Works

The cloud is a relatively new platform for managing and delivering internet-based services. This novel topic has recently received a great deal of interest from researchers. High efficiency in a cloud is the result of optimized task scheduling. Meta-inference methods of scheduling are often used where the scheduling of tasks are non-deterministic polynomial (NP) problems. A robust and optimized algorithm of genetics was presented by to improve the solutions [2]. The program combined the benefits of evolving genetic algorithms with experimental techniques.

Task scheduling is critical for scaling up dispersed electronic business systems and electronic science systems. This usage typically includes the communication of tasks that are performed on virtual resources or virtual machines. The basic goal of any scheduling approach is to decrease the breadth of the configuration, which represents the makespan of the exit job. A model called H3CSA was proposed in this regard [3].

Cloud service providers provide diverse virtual computers to conduct complicated tasks designated by consumers. A meta-hybrid method that used a genetic algorithm and thermodynamic simulated annealing algorithms (GATSA) was presented to solve the problem; the global and local search tendencies of the two algorithms compensated for each other's weaknesses. A novel theory was introduced and used to generate a semi-conducted starting population [4].

Proper task scheduling is necessary for optimal efficiency in cloud computing; however, traditional experimental methods in this context lack the necessary efficiency because task scheduling in a cloud is entirely NP. As a result, the bulk of recently suggested task scheduling methods have concentrated on a hybrid descriptive technique of job scheduling. A meta-heuristic technique algorithm known as HEFT was proposed [5].

Other researchers suggested inferential algorithms such as the ICA and FA to tackle the issue. Although ICA and FA can approach reasonable solutions, scalability, time of CPU, stability, and balancing of load are necessary to attain the most effective outcomes. A smart method that is based on the marriage of FA and ICA to obtain the desired result was proposed; FA's local search can enhance the ICA [6].

A new strategy was proposed to overcome the challenge of scheduling resources in cloud computing; it employed a model that improved the parallel scheduling of tasks while maintaining the relationships between serial tasks [7]. The primary characteristic of dynamic tasks was that users could assign varied and equal priorities to activities performed in sequential order and arrange them in a scheduling queue. FA was utilized for scheduling subtasks to show improvement in a short amount of time. The most critical factors to evaluate were fairness and efficiency; the strategy known as DSFFA was used to solve the task scheduling problems.

The scheduling of tasks is a primary concern within the cloud to prevent diminished system performance. A significant method that organizes the requirements of the user and achieves several aims, the technique aims to lessen the workload makespan and the cost of execution while increasing resource usage [8].

4 Problem Descriptions

During this work, the task scheduling model is delineated as distributed NTK's to be implemented on MPR processors that may be different. A task graph is mapped to describe the problem structure—a DAG composed of NTK's Tak_1, Tak_2, Tak_3, …Tak_n. Every node of the graph is identified as a task. We know that a task is a series of instructions that are implemented by the processor in successive order. A task may have pre-requested data; once all the inputs needed for the execution are received, the task can be put into the processor for implementation. These inputs are intended to be supplied after the completion of certain other activities, as these tasks assess them. We refer to this as relying on task dependency. If a task (t) is dependent on data from other tasks, then those tasks are recognized as the fathers of the task (t), and task (t) is their child. A task that has no father is called an "entry task," and a task that has no child is called an "exit task" [9]. The computing cost of a job is based on its execution time. Whenever the cost of computation of a Tak_i is indicated by weight (Tak_i, Pro_j), the DAG has additional edges that represent a partial order between the tasks. The partial order between the tasks introduces constrained precedence in a DAG such that if Tak_j is a child of Tak_i ($Tak_i \rightarrow Tak_j$), it cannot begin its execution until its father Tak_i finishes its execution. The value on the edge represents the cost of communication between the tasks and is indicated by CCO(Tak_i, Tak_j). The communication cost is considered only if Tak_i and Tak_j are assigned to different processors; otherwise, it is calculated to be zero. In that case, Tak_i and Tak_j are given to the processor itself. If a Tak_i is given to $Pros_j$, the task's start and finish times are denoted by STA(Tak_i, Pro_j) and FNT(Tak_i, Pro_j), respectively. After scheduling the tasks, the makespan is defined as max {FNT(Tak_i, Pro_j)} throughout all processors. The problem of scheduling tasks involves coordinating the schedules of the tasks in the different processors, decreasing the makespan for all of those schedules, and ensuring that the task dependency constraints are preserved. Task dependency constraints state that no task can begin before all fathers have terminated. Assuming that Pro_j is the processor and that the KP^{th} parent task Tak_{kp} of task Tak_i is scheduled, the DATVE of Tak_i at processor Pro_j is when the prerequisite data for the task execution becomes available. This is defined in [9] by the following:

$$DATVE(Tak_i, Pro_j) = \max\{FNT(Tak_{kp}, Pro_j) + CCO(Tak_i, Tak_{kp})\} \quad (1)$$

where kp = 1,2, 3, …. Parent Number

$$STA(Tak_i, Pro_j) = \max\{RYT(Pro_j), DATVE(Tak_i, Pro_j)\} \quad (2)$$

$$FNT(Tak_i, Pro_j) = STA(Tak_i, Pro_j) + weight(Tak_i, Pro_j) \quad (3)$$

$$RYT(Pro_j) = FNT(Tak_i, Pro_j) \quad (4)$$

$$Makespan = \max\{FNT(Tak_i, Pro_j)\} \quad (5)$$

5 Firefly Algorithms

Yang [10] created the firefly algorithm, which was modeled on the motion of flashing fireflies. For the sake of simplicity, we incorporate this concept into the rules listed below:

- Because all fireflies are unisex, all fireflies are attracted to all other fireflies.
- Their appeal is related to their luminosity. As a result, when two fireflies are flashing, the one with the lower brightness will move toward the one that is brighter. The attraction is based on the brightness, which decreases as the distance between the objects increases. If no firefly is brighter, the firefly moves randomly.
- The goal function's landscape to be optimized influences or determines the brightness of a firefly or the ferocity of a firefly.

There are two major concerns in the FA: the modulation of the intensity of light, and the creation of appeal. In the simplest example of the greatest improvement issue, the brightness IB of a firefly at a given position y will be selected as IB(y) α g(y). However, the attractiveness δ is subjective; it must be viewed according to the other firefly. As a result, it must change depending on the R_{ij}, which is the separation between fireflies i and j. The intensity of light diminishes in relation to the distance from the source, and light is also absorbed inside the medium. That is:

$$IB = IB_0 \exp(-\varphi R) \quad (6)$$

where IB_0 represents the initial brightness of the light and φ is the coefficient of light absorption. The attractiveness of a firefly is shown by:

$$\delta = \delta_0 \exp(-\varphi R^2) \quad (7)$$

A firefly i's attraction to a more attractive firefly j is determined by:

$$y_i = y_i + \delta_0 \exp(-\varphi R_{ij}^2)(y_j - y_i) + \alpha\varepsilon_i \quad (8)$$

Firefly Algorithm

g(y) is the objective function where $y = (y_1, ..., y_d)^T$

Initialize a firefly's y_i where i is equal from 1 to n

The IB_i is computed by using $g(y_i)$

While (iteration <Maximum of generation)

{

For i equal 1 to n

{

For j equal 1 to i

{

If ($IB_j > IB_i$)

{

The firefly(i) should be moved to j in all dimensions.

}

Through exp [$-\varphi R$]

Update the intensity of the light and evaluate newly discovered solutions

}

}

To find current best, rank fireflies

}

6 The Proposed Algorithm

PFA begins with the first set of initializations. Then, by applying some operators, the best solution is chosen according to the value of the goal function. In PFA, we state the following steps:

6.1 Initialization

The initialization is randomly generated according to this equation:

$$POP_{ij} = LB + rand*(UB - LB) \quad (9)$$

Where UB and LB refer to the upper and lower boundaries, we consider that the value of the UB is equal to the number of processors and the LB value is equal to one. Because the representation of a vector is a continuous value, we will use the LPV rule [11] and the SPV rule [12], as shown in Figure 1.

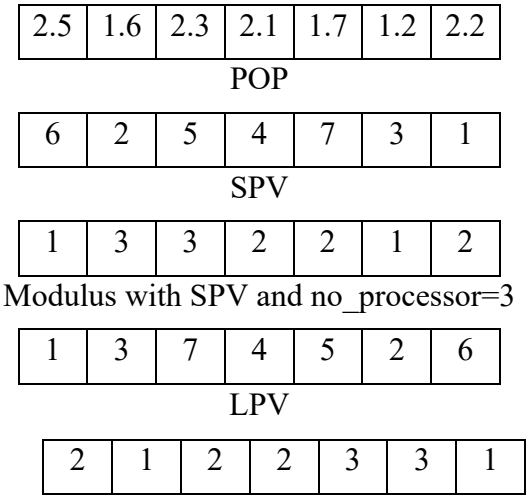

Fig.1: An example of the proposed schedule.

6.2 Priority Operations

Task prioritization plays a big role in task scheduling and calculating the makespan. The proposed priority is randomly generated in order to preserve the precedence constraints.

6.3 The Objective Function

The scheduling problem's primary goal is to lower the makespan, which is calculated according to equation 5. That is:

Objective Function = Makespan (10)

Algorithm 1: Calculate the makespan of the scheduled task using the Standard Genetic Algorithm (SGA) [9]

Input the schedule of tasks as shown in Figure 1

Output objective function

RYT[Pro_j] = 0 where j = 1, 2, ……MPR.

For i = 1 to NTK

{

// LST is generated randomly in an order that preserves precedence constraints

Take the first Tak_i from LST

For j equal 1 to MPR

{

If Tak_i scheduled to Pro_j

{

Calculate start time according to equation 2

Calculate finish time according to equation 3

Calculate ready time according to equation 4

}

```
        }
}
Calculate makespan according to equation 5
Calculate the objective function according to equation 10
```

6.4 The Proposed Operations

Algorithm 3: The whole PFA

```
Input the DAG with computation cost and communication cost
Output the best solution
Initialize the parameters α, δ0, φ, ∂, N, max_iteration
Generate the initial population according to equation 9
Convert a continuous value to a discrete value by using Algorithm 2
Calculate the objective function g(yi) by using Algorithm 1
For iteration=1 to max_iteration
{
  For i equal 1 to N
  {
    For j equal 1 to N
      {
       If ( g(yi) > g(yj))
        {
         Update the location using equation 8
         Convert a continuous value to a discrete value of the new solution by using Algorithm 2
         Compute the makespan of the newly obtained solution by using Algorithm 1 and update it
           if better
        }
       }

  }
}
```

Algorithm 2: The algorithm that transforms continuous values into discrete values.

```
Function Convert (s)
A = random number between  [1…5]
If  (A==1)
     Use the SPV rule
Else if (A==2)
     Use the LPV rule
Else if (A==3)
```

Use the round nearest function

Else if (A==4)

Use the floor nearest function

Else

Use the ceil nearest function

End if

End function

7 Evaluation of the PFA

We show the effectiveness of the PFA by applying it to three cases. The first case consists of ten tasks and three heterogeneous processors. The second case consists of ten tasks and three heterogeneous processors. The third case consists of eleven tasks and three heterogeneous processors. PFA was implemented as a system by MATLAB 2016. We set the values of the initial parameters $\alpha = 1.0$, $\delta_0 = 1.0$, $\varphi = 0.01$, $\partial = 0.97$, N = 30, max_iteration = 50.

7.1 Case 1

Suppose we have ten tasks to be executed on three heterogeneous processors {Pro_1, Pro_2, Pro_3}. The executing cost of each task is shown in [3]. The best solution obtained by PFA is shown in Figure 2, and the schedule obtained by PFA and other algorithms is shown in Table 1. The outcomes of the PFA are compared with those obtained by the New Genetic Algorithm (N-GA) [2], Proposed Particle Swarm Optimization (PPSO) [13], and H3CSA [3]. The obtained results are illustrated in Table 2 and Figure 3 with the proposed task priority of PFA {Tak_1, Tak_4, Tak_3, Tak_2, Tak_6, Tak_7, Tak_5, Tak_9, Tak_8, Tak_{10}}, task priority of N-GA {Tak_1, Tak_4, Tak_2, Tak_3, Tak_7, Tak_5, Tak_6, Tak_9, Tak_8, Tak_{10}}, task priority of PPSO {Tak_1, Tak_2, Tak_3, Tak_4, Tak_5, Tak_6, Tak_7, Tak_8, Tak_9, Tak_{10}}, and task priority of H3CSA {Tak_1, Tak_4, Tak_3, Tak_7, Tak_6, Tak_2, Tak_5, Tak_9, Tak_8, Tak_{10}}.

1	1	1	1	2	3	1	2	3	2

Fig. 2: The best solution for case 1.

Table 1: Schedule obtained by PFA and other algorithms for case 1.

	N-GA			PPSO			H3CSA			PFA		
	1	2	3	1	2	3	1	2	3	1	2	3
Tak_1			0–4	0–7			0–7			0–7		
Tak_2			32–50	7–14			36–43			30–37		
Tak_3	43–60			14–31			13–30			13–30		
Tak_4			4–32	31–37			7–13			7–13		
Tak_5		61–78			32–49			54–71			48–65	
Tak_6			62–89	37–72					33–60			33–60
Tak_7	60–66			72–78			30–36			37–43		
Tak_8	113–133			84–104				79–119			79–119	
Tak_9			97–113		109–142				75–91			74–90
Tak_{10}		163–175			142–154			124–136			123–135	

Table 2: Comparative results based on case 1.

Algorithm	N-GA	PPSO	H3CSA	PFA
Makespan	175	154	136	135

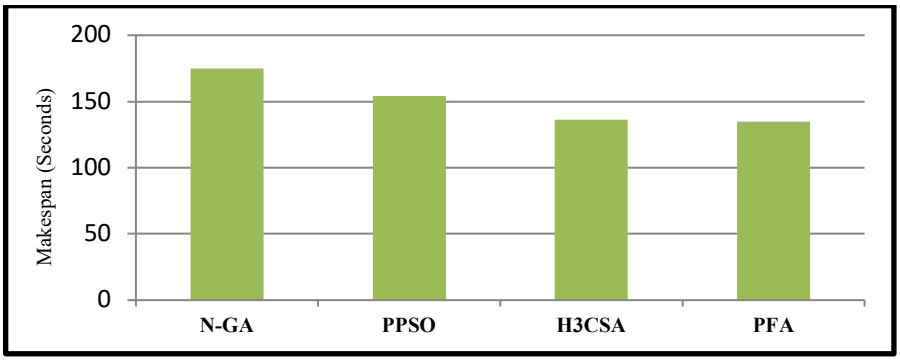

Fig. 3: Comparison of results for case 1.

7.2 Case 2

Suppose we have ten tasks to be executed on three heterogeneous processors {Pro$_1$, Pro$_2$, Pro$_3$}. The executing cost of each task is shown in [9]. The best solution obtained by PFA is shown in Figure 4, and the schedule obtained by PFA and other algorithms is shown in Table 3. The outcomes of the PFA are compared with those obtained by the WOA [14], GSA [15], Simulated Annealing (SA), GA, EGA-TS [16], Genetic Algorithm (GA) [8], and Hybrid Heuristic and Genetic (HHG) [17]. The obtained results are illustrated in Table 4 and Figure 5 with the proposed task priority of PFA {Tak$_1$, Tak$_6$, Tak$_4$, Tak$_5$, Tak$_2$, Tak$_3$, Tak$_8$, Tak$_9$, Tak$_7$, Tak$_{10}$}, task priority of WOA {Tak$_1$, Tak$_3$, Tak$_5$, Tak$_2$, Tak$_4$, Tak$_6$, Tak$_7$, Tak$_8$, Tak$_9$, Tak$_{10}$}, task priority of SA {Tak$_1$, Tak$_5$, Tak$_3$, Tak$_2$, Tak$_4$, Tak$_6$, Tak$_7$, Tak$_8$, Tak$_9$, Tak$_{10}$}, task priority of EGA-TS {Tak$_1$, Tak$_3$, Tak$_5$, Tak$_2$, Tak$_4$, Tak$_6$, Tak$_7$, Tak$_8$, Tak$_9$, Tak$_{10}$}, task priority of GSA {Tak$_1$, Tak$_3$, Tak$_2$, Tak$_6$, Tak$_4$, Tak$_5$, Tak$_7$, Tak$_8$, Tak$_9$, Tak$_{10}$}, task priority of GA {Tak$_1$, Tak$_3$, Tak$_2$, Tak$_6$, Tak$_4$, Tak$_5$, Tak$_7$, Tak$_8$, Tak$_9$, Tak$_{10}$}, task priority of GA {Tak$_1$, Tak$_2$, Tak$_4$, Tak$_5$, Tak$_9$, Tak$_3$, Tak$_7$, Tak$_6$, Tak$_8$, Tak$_{10}$}, and task priority of HHG {Tak$_1$, Tak$_2$, Tak$_6$, Tak$_3$, Tak$_4$, Tak$_5$, Tak$_8$, Tak$_7$, Tak$_9$, Tak$_{10}$}.

2	2	2	1	3	1	2	1	3	1

Fig. 4: The best solution for case 2.

Table 3: Schedule obtained by PFA and other algorithms for case 2.

	WOA			SA			EGA-TS			GSA			GA			GA			HHG			PFA		
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Tak$_1$		0 – 21			0 – 21			0 – 21			0 –21			0 –21		0 – 22				0 – 21			0 – 21	
Tak$_2$			38 – 56			38 – 56	38 – 60					38 – 58			38 – 58	22 – 44				21 – 39			21 – 39	
Tak$_3$		21 –48			21 –48			21 – 48			21 – 48			21 – 48		44 – 76				39 – 66			39 – 66	
Tak$_4$		48 –58			48 –58			48 – 58		50 – 57			50 – 57				51 –61				50 – 54	64 –71			
Tak$_5$	34 – 63			34 – 63					34 – 69			56 – 91			56 – 91			35 – 70			54 – 89			34 – 69
Tak$_6$		58			58			58			48			48			61		38			38		

		–75			–75			–75			–65			–65			–78		–64			–64		
Tak_7	64–78			64–78				75–100		64–78			64–78			76–90				66–91			66–91	
Tak_8		75–98			75–98		80–109				68–91			68–91			78–101		65–94			71–100		
Tak_9	86–101			86–101					90–98			91–99			91–99			74–82			89–97			78–86
Tak_{10}		108–124			108–124		109–122				106–122			106–122			101–117		104–117			100–113		

Table 4: Comparative results based on case 2.

Algorithm	WOA	SA	EGA-TS	GSA	GA[5]	GA[9]	HHG	PFA
Makespan	124	124	122	122	122	117	117	113

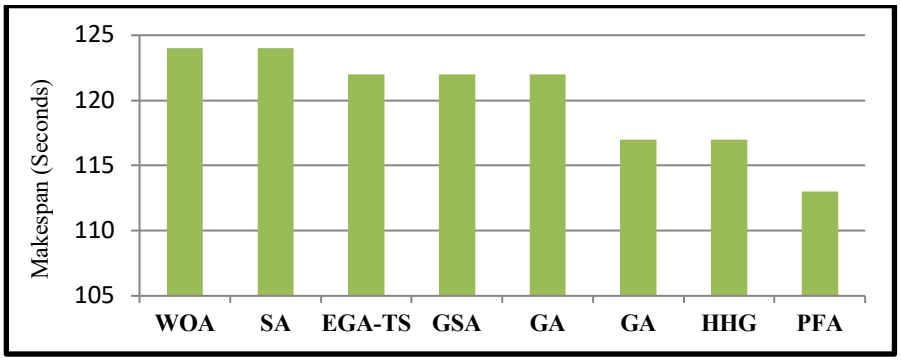

Fig. 5: Comparison of results for case 2.

7.3 Case 3

Suppose we have eleven tasks to be executed on three heterogeneous processors {Pro_1, Pro_2, Pro_3}. The executing cost of each task is shown in [18]. The best solution obtained by the PFA is shown in Figure 6, and the schedule obtained by the PFA and other algorithms is shown in Table 5. The outcomes of the PFA are compared with those obtained by three heuristic HEFT algorithms. MPQMA [18]. The obtained results are illustrated in Table 6 and Figure 7 with the proposed task priority of PFA {Tak_0, Tak_2, Tak_1, Tak_3, Tak_4, Tak_6, Tak_5, Tak_7, Tak_8, Tak_9, Tak_{10}}, task priority of HEFT-T {Tak_0, Tak_4, Tak_2, Tak_3, Tak_1, Tak_5, Tak_6, Tak_7, Tak_8, Tak_9, Tak_{10}}, task priority of HEFT-B {Tak_0, Tak_4, Tak_3, Tak_2, Tak_1, Tak_7, Tak_6, Tak_5, Tak_9, Tak_8, Tak_{10}}, task priority of HEFT-L {Tak_0, Tak_3, Tak_4, Tak_2, Tak_1, Tak_7, Tak_6, Tak_5, Tak_9, Tak_8, Tak_{10}}, task priority of MPQMA {Tak_0, Tak_1, Tak_2, Tak_5, Tak_3, Tak_4, Tak_7, Tak_9, Tak_6, Tak_8, Tak_{10}}.

2	3	1	2	2	1	1	2	1	2	1

Fig. 6: The best solution for case 3

Table 5: Schedule obtained by PFA and other algorithms for case 3.

	HEFT-T			HEFT-B			HEFT-L			MPQMA			PFA		
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Tak_0			0–5			0–5			0–5			0–5		0–7	

Tak_1			22–30		23–32				20–28			5–13			25–33
Tak_2			15–22	19–27			19–27					13–20	21–29		
Tak_3	20–31					15–30			5–20	20–31				7–20	
Tak_4			5–15			5–15		16–23			16–23			20–27	
Tak_5			30–38		37–47				37–45			20–28	47–53		
Tak_6		39–51		27–45			27–45					28–43	29–47		
Tak_7	40–57					30–42		38–48			49–59			27–37	
Tak_8		52–72				61–72			56–67			43–54	53–67		
Tak_9	57–69					42–52		48–56			59–67			37–45	
Tak_{10}	85–94					72–89	80–89				67–80		67–76		

Table 6: Comparative results based on case 3.

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

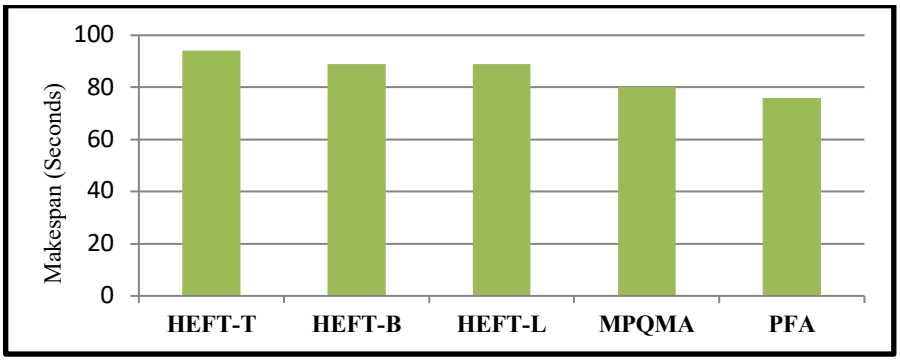

Fig.7: Comparison of results for case 3.

8 Discussions

The outcomes in Figure 3 and Table 2 show that the makespan of the PFA is reduced by (22.85%), (12.33%), and (0.73%) for N-GA, PPSO, and H3CSA, respectively. The outcomes in Figure 5 and Table 4 show that the makespan of the PFA is reduced by (8.87%), (8.87%), (7.37%), (7.37%), (3.41%), and (3.41%) for WOA, SA, EGA-TS, GSA, GA, and HHG, respectively. The outcomes in Figure 7 and Table 6 show that the makespan of the PFA is reduced by (19.14%), (14.60%), (14.60%), and (5%) for HEFT-T, HEFT-B, HEFT-L, and MPQMA, respectively.

9 Conclusions and Future Work

In order to achieve near-optimal results for the issue of scheduling tasks in a cloud, efficient strategies for the optimal mapping of the tasks are required. In this study, we suggest a novel efficient scheduling task method in the cloud that uses the firefly algorithm to address the issue of scheduling tasks. The system is made up of a small number of fully linked heterogeneous virtual machines. The method has been compared to other algorithms according to makespan. The comparative analysis illustrates that the PFA is significantly better in all cases. In our future work, we will develop an efficient algorithm to tackle scheduling tasks in a cloud using the algorithm of cuckoo search.

References

[1] Dubey, K., Kumar, M. & Sharma, S. C. (2018). Modified HEFT algorithm for task scheduling in cloud environment. *Procedia Computer Science*., **(125)**, 725–732 (2018).

[2] 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. *Journal of Systems and Software.*, **(124)**, 1-21 (2017).

[3] Mishra, A., Sahoo, M. N. & Satpathy, A. (2021). H3CSA: A makespan aware task scheduling technique for cloud environments. *Transactions on Emerging Telecommunications Technologies*., 1-20 (2021).

[4] Tanha, M., Shirvani, M. H. & Rahmani A. M. (2021). A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments. *Neural Computing and Applications*., 1-34 (2021).

[5] Kamalinia, A. & Ghaffari, A. (2017). Hybrid Task Scheduling Method for Cloud Computing by Genetic and DE Algorithms. *Wireless Personal Communications*., **(97)**, 6301–6323 (2017).

[6] Kashikolaei, S. M. G., Hosseinabadi, A. A. R., Saemi, B., Shareh, M. B., Sangaiah, A. K. & Bian, G. B. (2020). An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm. *Journal of Supercomputing*., **(76)**, 6302-6329 (2020).

[7] Saleh, I. A., Alsaif, O. I., Muhamed, S. A. & Essa, E. I. (2019). Task Scheduling for cloud computing Based on Firefly Algorithm. *Journal of Physics: Conference Series.*, **(1294)**, 042004 (2019).

[8] Hamed, A. Y. & Alkinani, M. H. (2021). Task Scheduling Optimization in Cloud Computing Based on Genetic Algorithms. *Computers, Materials & Continua*., **(69)**, 3289-3301 (2021).

[9] Younes, A., BenSalah, A., Farag, T., Alghamdi, F. A. & Badawi, U. A. (2019). Task Scheduling Algorithm for Heterogeneous Multi Processing Computing Systems. *Journal of Theoretical and Applied Information Technology*., **(97)**, 3477-3487 (2019).

[10] Yang, X.-S. (2010). Firefly algorithm, stochastic test functions and design optimization. International Journal of Bio-Inspired *Computation*., **(2)**, 78-84 (2010).

[11] Wang, L., Pan, Q. & Tasgetiren, F. M. (2011). A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Computers & Industrial Engineering.*, **(61)**, 76-83 (2011).

[12] Dubey, I. & Gupta, M. (2017). Uniform mutation and SPV rule based optimized PSO algorithm for TSP problem. 4th *International Conference on Electronics and Communication Systems (ICECS)*., 168–172 (2017).

[13] Biswas, T., Kuila, P. & Ray, A. K. (2020). A novel workflow scheduling with multi-criteria using particle swarm optimization for heterogeneous computing systems. *Cluster Computing*., **(23)**, 3255-3271 (2020).

[14] Thennarasu, S. R., Selvam, M. & Srihari, K. (2020). A new whale optimizer for workflow scheduling in cloud computing environment. *Journal of Ambient Intelligence Humanized Computing*., **(12)**, 3807-3814 (2020).

[15] Biswas, T., Kuila, P., Ray, A. K. & Sarkar, M. (2019). Gravitational search algorithm based novel workflow scheduling for heterogeneous computing systems. Simulation Modelling Practice and Theory., (96), 101932 (2019).

[16] Akbari, M., Rashidi, H. & Alizadeh, S. H. (2017). An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems. *Engineering Applications of Artificial Intelligence*., **(61)**, 35-46 (2017).

[17] Sulaiman, M., Halim, Z., Lebbah, M., Waqas, M. & Tu, S. (2021). An Evolutionary Computing-Based Efficient Hybrid Task Scheduling Approach for Heterogeneous Computing Environment. *Journal of Grid Computing*., **(19)**, 1-31 (2021).

[18] 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)**, 1650119 (2016).