Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/160606

Optimization Task Scheduling Bee Colony Algorithm for Heterogeneous Cloud Computing Systems

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

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

Received: 1 Aug. 2022, Revised: 22 Sep. 2022, Accepted: 24 Sep. 2022.
Published online: 1 Nov. 2022.

Abstract: The primary purpose of the task scheduler is to assign tasks to available processors to produce a minimum Makespan without violating precedence constraints. In heterogeneous cloud computing resources, task assignments and schedules significantly impact system operation. In the experimental task scheduling algorithm, a different mapping of the process will result in a different maximum completion time of a batch of tasks (Makespan) on heterogeneous cloud computing resources. Thus, a scheduling algorithm has to define a schedule considering the precedence of child tasks depending on the resources required to reduce makespan. In this paper, we propose an Efficient Artificial Bee Colony Optimization Algorithm (EABCOA) to solve heterogeneous cloud computing resources' task assignment and scheduling problems. The basic idea of this process is to exploit the advantages of meta-heuristic algorithms to get the optimal solution for makespan. We evaluate our algorithm's performance by applying it to three cases with a different number of tasks and processors. The results show that the proposed approach significantly outperforms other methods in finding the optimal solution for makespan.

Keywords: Heterogeneous processors, Artificial bee colony optimization algorithm, Task scheduling, multiprocessing, cloud computing.

1 Introduction

Execution time is one of the key pre-performance measurements of any computing system. in order to reduce execution time, processors were developed faster. Still, they had physical limitations, so a multiprocessing system was used. In a multiprocessing system, the program is divided into tasks to perform each task on one of the processors. Assigning tasks and processors is called scheduling tasks in a multiprocessing system. in order to Achieve optimal task scheduling and processor utilization in a heterogeneous multiprocessing system is a computationally challenging goal. The term improvement may refer to several goals combined. Usually, the main goal is to reduce the length of the schedule (Makespan). Finding the optimal task schedule is an NP [1,2] hard problem. Accordingly, meta-heuristic algorithms are a good candidate to tackle this problem.

Many articles have applied the artificial bee colony optimization algorithm to solve task scheduling. This paper presents the proposed Efficient Artificial Bee Colony Optimization Algorithm (EABCOA) for heterogeneous cloud computing resources. Heterogeneous cloud computing resources have different processing capabilities. The task processing time can only be specified when assigned to a specific processor; that is, the task processing time depends on the processor. The proposed Efficient Artificial Bee Colony Optimization Algorithm (EABCOA) is presented to find the optimal task scheduler assigned to heterogeneous cloud computing resources. The basic idea of this technique is to exploit the advantages of meta-heuristic algorithms to get the optimal solution for makespan. The proposed algorithm assigns tasks to processors randomly, and the task priority is generated randomly in order to preserve the precedence constraints. We evaluate our algorithm's performance by applying it to three cases with a different number of tasks and processors. The results show that the proposed approach succeeded in finding the optimal solution for makespan.

The convergence speed of the Artificial Bee Colony Optimization Algorithm (ABCOA) is slow. The algorithm has strong exploration performance, but poor exploitation performance since its solution update formula only alters one component of the solution vector each time. In order to solve the above drawbacks and accelerate the rate of convergence,

[*]Corresponding author e-mail: hamdy2006x@gmail.com

a new strategy is proposed to find novel candidate solutions in the employed bee search step of the proposed EABCOA. This strategy accelerates the convergence rate of the algorithm and improves the algorithm's exploitation ability. In this strategy, we use a method that randomly selects one solution from an existing solution and generates random numbers according to those generated numbers. It takes the values from the randomly chosen solution and puts them into the current solution without changing the randomly selected solution. We present the same strategy in the onlooker bees phase. This strategy improves the algorithm's exploitation ability and enhances the solution's precision.
The rest of this paper is presented as follows. The notations are given in Section 2. Section 3 presents some work related to the problem of task scheduling for various systems architectures. A description of the problem is given in section 4. In section 5, Artificial Bee Colony Optimization Algorithm is described. Our EABCOA approach to finding optimal task scheduling for heterogeneous cloud computing resources is described in Section 6. The results were obtained by applying EABCOA, and their comparison with other results is presented in Section 7. Section 8 concludes the paper and future work.

2 Notations

G	A task graph
DAG	A Directed Acyclic Graph
Ti	Task i
Pi	Processor i
M	Number of processors
N	Number of tasks
Ni	Node i
C(Ti, Tj)	Communication Cost between task i and task j
ST(Ti, Pj)	Start time of task i on a processor Pj
FT(Ti, Pj)	Finish time of task i on a processor Pj
RT(Pi)	Ready time of the processor i
LT	A list of tasks according to the topological order of DAG.
DAT(Ti, Pj)	The Data Arrival Time of task i at processor j
Pop_size	Number of the initial population
Max_iter	The maximum number of iterations

3 Related Work

Recently, cloud computing has emerged as a widely used platform to supply compute, storage, and analytics services to end-users and organizations on a pay-as-you-use basis, with high skill, accessibility, scalability, and resiliency. It allows people and organizations to access a large pool of high-processing resources without establishing a high-performance computing (HPC) platform. For the last few years, task scheduling in cloud computing has been an outstanding resource for researchers. The Task Aware Scheduling Algorithm (TASA) and Proactive Simulation-based Scheduling and Load Balancing (PSSLB) are presented during this research work. The author investigated and empirically compared some of the most prominent state-of-the-art scheduling heuristics regarding Makespan, Average Resource Utilization Ratio (ARUR), Throughput, and energy consumption [3].
Cloud computing is an efficient technology to serve the needs of big data applications. It minimizes the makespan of the cloud system, whereas increasing resource utilization is essential to reduce costs. In this case, task scheduling is challenging to satisfy the requirement because it needs effectiveness and efficiency. The particle swarm optimization (PSO) algorithm with many discrete variants was presented for task scheduling in cloud computing [4].
Cloud computing is an emerging distributed, low-cost computing paradigm with a large collection of heterogeneous autonomous systems. It provides on-demand, flexible, and scalable services to customers on a pay-per-use basis. The general performance of cloud infrastructure depends on task assignment and scheduling. Efficient task scheduling decreases the power consumption of the cloud infrastructure and increases service providers' profit by reducing the processing time of the user's job. An efficient task scheduling algorithm using a multi-objective Artificial Bee Colony Algorithm (TA-ABC) is presented. The algorithm optimizes the cloud computing environment's energy, cost, resource utilization, and processing time [5].
Task scheduling is one of the significant problems in a cloud computing system. Efficient task scheduling is substantial for achieving cost-efficient execution and improving resource utilization. A particle swarm optimization (PSO) using heuristic algorithms has been proposed [6]. In order to initialize the PSO, an improved initialization of the longest job to the fastest processor (LJFP) and minimum completion time (MCT) algorithms are used. The performance of the LJFP-PSO and MCT-PSO algorithms is evaluated by minimizing the makespan, total execution time, degree of imbalance, and total energy consumption metrics.
For the past few years, cloud computing has been considered an attractive high-performance computing platform for individuals and organizations. To accommodate the requirements of cloud users, cloud service providers (CSPs) are setting up their data centers with high-performance computing resources. Users are mainly interested in response time, whereas cloud service providers are more concerned about revenue generation. Concerning these needs, the task scheduling for the users' applications in cloud computing has attained focus from the research community. A resource-aware dynamic task scheduling approach is proposed and implemented [7]. The DRALBA algorithm has revealed significant improvements in attained ARUR, Throughput, and Makespan.

With the rapid development of cloud computing and the internet, load balancing techniques are becoming more and more critical than ever. A good scheduling algorithm is a significant way to resolve load balance problems. A new load balance algorithm is proposed based on the ABC algorithm [8], which can be seen as a new scheduling method based on a swarm intelligence algorithm.

The combination of the Swarm Intelligence algorithm of an artificial bee colony with a heuristic scheduling algorithm, called Heuristic Task Scheduling with Artificial Bee Colony (HABC), was proposed [9]. This algorithm enhances cloud computing's virtual machines (VMS) scheduling solutions in homogeneous and heterogeneous environments. It has been introduced to minimize the makespan and balance the loads.

The general problem of multiprocessor scheduling can be defined as scheduling a task graph on a multiprocessor system so that the length of the schedule can be optimized. Several exploratory approaches have been developed in the literature that obtains suboptimal solutions in less polynomial time. Recently, genetic algorithms have gained a lot of awareness because they are powerful and guarantee a good solution. In this paper, the author has developed a genetic algorithm based on the principles of evolution found in nature to find the optimal solution [10].

Constrained application scheduling takes precedence over a distributed heterogeneous computing system to reduce response or total execution time. The author designed and studied the effectiveness of a micro Genetic Algorithm (microGA) based scheduling algorithm [11].

Effective scheduling of tasks in heterogeneous computing systems is paramount for implementing high-performance programs. Programs are considered multiple sequences of tasks presented as directed acyclic graphs (DAGs). Each task has its execution timeline that incorporates various processors. Furthermore, each edge on the graph represents constraints between sequential tasks. The author proposes a new list of scheduling algorithms that schedule tasks represented in the processor DAG and that better reduce the overall execution time by taking into account the limitations of cross-processing. This goal will be achieved in two main phases: (a) computing priorities for each task performed and (b) selection of the processor who will take over each task [12].

A new static scheduling algorithm, Communication Leveled DAG with Duplication (CLDD), is proposed to schedule tasks on heterogeneous distributed computing systems efficiently. It solves most of the limitations of existing algorithms. The algorithm focuses on reducing the range and provides better performance than other algorithms in acceleration, efficiency, and time complexity [13].

A new task scheduling algorithm for heterogeneous computing, called HSIP (Heterogeneous Scheduling Algorithm with Optimized Task Priority), whose function is based on three pillars: (1) an optimized strategy for priority tasks based on standard deviation with optimizing size as a weight to calculate and weigh the delivery cost to make scheduling priority more reasonable; (2) the policy of choosing the duplication of the entry task to make the period shorter; and (3) an improved optimization policy based on the introduction of idle periods (ITS) to make task scheduling more efficient [14].

Hybrid list-based task scheduling using the Duplication (HLTSD) algorithm for heterogeneous processors. The HLTSD algorithm has the same time complexity as modern algorithms; however, it produces a lower cost schedule than other related methods. This work also presents a mathematical formulation for finding priority tasks. The processor selection phase is improved through input task repetition, insertion-based policy, primary task repetition at other levels, and load balancing on each processor [15].

A proposed genetic algorithm (PGA) was presented to solve homogeneous and heterogeneous multiprocessing task attribution and scheduling problems. The basic idea of this process is to exploit the advantages of heuristic algorithms to reduce the search for space and time to get the best solution [1].

A task scheduling algorithm based on a Genetic Algorithm (GA) is presented [16] to assign and execute different tasks. The algorithm aims to decrease both (Makespan) and execution costs of tasks and increase resource utilization, Speedup, and Efficiency.

Bee colony optimization (BCO) has been applied as a local search in the proposed memory algorithm [17].

A quantum genetic algorithm with spin angle optimization is presented in the literature for scheduling tasks based on distributed systems such as cloud data centers [18].

4 Problem Description

The task scheduling model numbers in this work, which can be described as N distributed tasks to be performed on M processors, can be general processors with different computing capabilities. A task graph can be set to describe the structure of the problem. Task graph G is a directed acyclic graph (DAG) consisting of tasks T1, T2, T3 ... Tn. Each node in the graph is described as a task. A task is supposed to be a set of instructions performed sequentially on a particular processor. The task (node) may have the necessary (input) data before it can be executed. When all entries are received, the node can be run for execution.

These entries are expected to be delivered after some other task has been completed, and these tasks are evaluated for them. We call it task dependency. If the task (Ti) is dependent on other essential data, we consider the tasks to be tasks that act as significant parents (Tk) → task (Ti) as their child. A node that does not contain an entry node is called an entry node, and a node that does not have a child is called an end node [1]. The time required to perform a task is what we call the computational cost. At any point, the cost of calculating Ti is indicated by weight (Ti, Pj). The graph also has straight E edges representing a partial order between the tasks. The partial system introduces a DAG constrained by precedence and means that if (Ti → Tj), then Tj is a child and cannot start until the parent Ti ends. The weight on edge represents the communication cost between the tasks indicated by C(Ti, Tj), and the communication cost is only considered if Ti and Tj are set on different processors; otherwise, it is counted as zero, in which case Ti and Tj are set on the processor itself. If node Ti is set on processor Pj, the node's start and finish times are indicated by ST(Ti, Pj) and FT(Ti, Pj), respectively. After scheduling the tasks into the processors, the makespan is defined as the maximum {FT(Ti, Pj)} across all processors.

The problem with task scheduling is finding the task schedule in the processors so that the makespan is reduced across the possible schedules, as task dependency constraints are kept. Task dependency restrictions state that no task can be started until all parents have completed it. Let the Pj be the wizard in which the original Tk task was a cog from the scheduled task Ti. The data arrival time (DAT) in the Pj processor is the time that data is available for each required to perform the task, as defined in [1] as follows:

$$DAT(Ti, Pj) = \max\{FT(Tk , Pj) + C(Ti, Tk)\} \qquad (1)$$

where k=1, 2, … Number_Parent

$$ST(Ti, Pj) = \max\{RT(Pj), DAT(Ti, Pj)\} \qquad (2)$$

$$FT(Ti, Pj)= ST(Ti, Pj)+weight(Ti, Pj) \qquad (3)$$

$$Makespan = \max\{FT(Ti, Pj)\} \qquad (4)$$

where i=1, 2,…, N and j=1,2,…., M

5 Artificial Bee Colony Optimization Algorithm

It is a definition guide for solving integrative optimization problems. The behavior of bees in nature inspires the Artificial Bee Colony Optimization Algorithm (ABCOA). In a honeybee colony, bees search the environment for flower paths and, if they find a good food source, share it with other bees. When foraging bees return to the hive, they share the information they have discovered about food sources through a particular movement called an oscillation dance. Studies of this type of bee dance show that in the center of this dance, certain information, such as direction, distance, quantity, and quality of the food source, is shared with other bees [17,19,20].

ABCOA is a population-based algorithm. The artificial bee community is looking for the perfect solution. Each artificial bee generates one solution to the problem. The algorithm consists of two alternating stages: forward pass and backward pass. During each forward swipe, each bee explores the search space. It applies a predetermined number of movements, which build and improve the solution, resulting in a new solution. After obtaining new partial solutions, the bees return to the nest and begin the second stage, the so-called back passage. During the back pass, all the bees share information about their solutions.

In nature, bees would perform a dancing ritual, informing other bees about the amount of food they found and the patch's proximity to the nest. In the search algorithm, the bees announce the quality of the solution, i.e., the value of the objective function. During the backward pass, every bee decides with a certain probability whether it will advertise its solution. The bees with better solutions have more chances to promote their solutions. The remaining bees have to decide whether to continue exploring their solution in the next forward pass or to start exploring the neighborhood of one of the advertised solutions. Similarly, this decision is taken with probability, so better solutions are more likely to be explored. The two phases of the search algorithm, forward and backward pass, are performed iteratively until a stopping condition is met. To continue, this is the pseudo-code of the ABCOA algorithm:

```
1. Initialization: every bee is set to an empty solution
2. For every bee, do the forward pass:
Evaluate all possible constructive moves
According to the evaluation, choose one move using the roulette wheel
3. All bees are back in the hive
4. Sort the bees by their objective function value
5. Every bee decides randomly whether to continue its exploration and become a recruiter or to become a follower (bees with higher objective function values have a greater chance to continue their exploration)
6. For every follower, choose a new solution from recruiters by the roulette wheel
7. If the stopping condition is not met, Go To Step 2
8. Output the best result.
```

6 The Proposed Efficient Artificial Bee Colony Optimization Algorithm

The proposed Efficient Artificial Bee Colony Optimization Algorithm (EABCOA) starts with the first possible solutions. Then, by applying some factors. The best solutions are determined according to the value of the objective function. In the proposed algorithm (EABCOA), we note that the six components are: (1) an initialization method. (2) Priority operation. (3) The objective function. (4) Employed phase. (5) Onlooker phase. (6) The scout stage.

6.1 Initialization

The initialization is randomly generated according to the number of processors. Suppose we have 3 processors, so the generated schedule is between 1 and 3, as shown in Fig. 1.

3	2	3	1	2	1	3	1	2

Fig. 1: The proposed schedule

T4, T6, and T8 are scheduled into processor 1, T2, T5, and T9 into processor 2, and T1, T3, and T7 are scheduled into processor 3.

6.2 Priority Operation

Task priority plays a significant role in task scheduling and calculating makespan. The proposed priority is randomly generated in order to preserve the precedence constraints $\{T_{Entry}, \ldots\ldots\ldots\ldots, T_{Exit}\}$.

6.3 The Objective Function

The main objective of the scheduling problem is to reduce makespan. That is:

Objective_Function = Makespan (5)

Where makespan is calculated by Eq. 4.

Algorithm1: The pseudo-code for the task schedule in the Standard Genetic Algorithm (SGA) [1] is as follows:

```
Input the schedule as shown in Fig. 1
Output Objective_function = Makespan
// ready time initialization for all processors
For all processors, Pj RT(Pj)=0
    For i = 1 to N
     {
       // LT is generated randomly in order that preserves
           precedence constraints
        Remove the first task Ti form list LT
       For j = 1 to M
        {
           If Ti is scheduled to processor Pj
              // start time for the task
              ST(Ti, Pj) = max{RT(Pj), DAT(Ti, Pj)}
              // finish time for the task
              FT(Ti, Pj) = ST(Ti, Pj) + weight(Ti, Pj)
              // ready time for the processor
              RT(Pj) = FT(Ti, Pj)
          End If
        }
    }
Makespan=max{FT(Ti, Pj)}
```

6.4 The Proposed Operations

6.4.1 Employed Bee Phase

In this phase, the number of employed bees equals the food sources and is applied to each solution. In addition, an employed bee generates a food source at the current source by the following steps:

Step 1: Select the current food source

Step 2: Select a food source randomly but not be the same as the current food source

Step 3: Generate random numbers between 1 to the number of tasks

Step 4: Copy from the randomly chosen food source without changing the randomly chosen source and replace the values of the current food source when it is equal to the generated random numbers as shown in Fig. 2. Consider the generated numbers to be {2, 6, 8, 9}

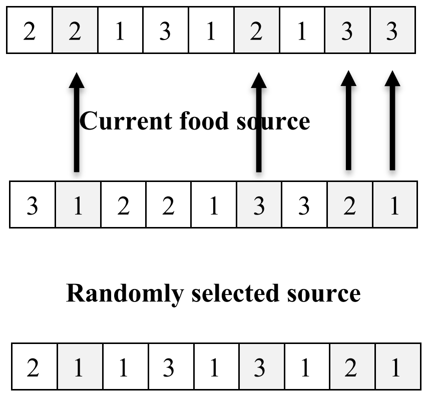


Fig. 2: New obtained source

Step 5: Calculate the objective function of the new obtained source

Step 6: If the new source is better than the current source, replace the current one. Or else increase the trial by one

Step 7: Apply the previous steps 1- 6 for each food source

6.4.2 Onlooker Bee Phase

In this phase, each onlooker bee selects an employed bee to improve its solution, and the selection is made according to fitness value.

$$\text{Pro}_i = \frac{fit(i)}{\sum_{j=1}^{n} fit(j)} \quad (6)$$

Where fit is a fitness function, and it is calculated as (7)

fit = 1/objective function (7)

Generate a random number if the value of that number is less than the fitness value, then improve the solution as follows:

Step 8: Select the current food source

Step 9: Select a food source randomly but not be the same as the current food source

Step 10: Generate random numbers between 1 to the number of tasks

Step 11: Copy from the randomly chosen food source without changing the randomly chosen source and replace the values of the current food source when it is equal to the generated random numbers as shown in Fig. 3. Consider the generated numbers to be {1, 5, 7, 9}

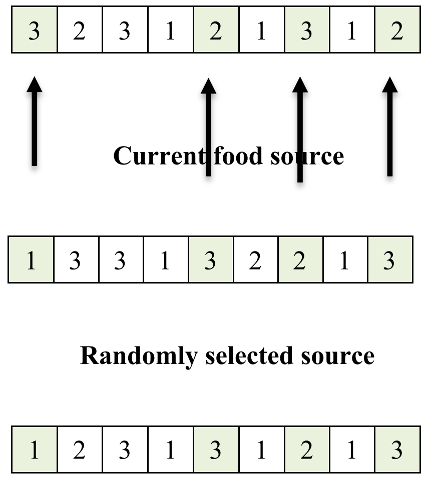


Fig. 3: New obtained source.

6.4.3 Scout Bee Phase

In this phase, the employed bee becomes a scout bee, which cannot improve its solution until the trial is equal to the limit. If the trial is greater than or equal to the limit, it will generate a new solution, as shown in Fig. 1.

Algorithm2: The whole EABCOA

```
Set the parameters pop_size, max_iter, trial, and limit
Input the DAG with computation cost, communication cost, and number of processors
Generate the initial population (schedule) as in the initialization section
Calculate the objective function for the initial population (schedule) Algorithm 1
For iter=1 to max_iter
// Employed Bee Phase
    For i=1 to pop_size
        Select the current food source
          Select food source randomly but not be the same as a current food source
          Generate random numbers between 1 and the number of tasks //between{1, ….., (number of tasks/2)}
          To obtain the new source copy from the randomly chosen food source without changing the randomly chosen source, replace the values of the current food source when it is equal to the generated numbers, as shown in Fig. 2
        Calculate the objective function for the new obtained source by using Algorithm 1
        If the objective function of the new source < objective function of the current source
            Update the current source with the new obtained source
            Update the objective function of the current source with the new objective function
            trial = 0
        Else
            trial = trial + 1
        End if
    End For
// Onlooker Bee Phase
    Calculate Pro_i according to Eq. 6
    For i=1 to pop_size
        If rand < Pro(i)
          Select the current food source
          Select food source randomly but not be the same as a current food source
          Generate random numbers between 1 to the number of tasks //between{1, ….., (number of tasks/2)}
          To obtain the new source copy from the randomly chosen food source without changing the randomly chosen source and replace the values of the current food source when it is equal to the generated number, as shown in Fig. 3
          Calculate the objective function of the new obtained source by using Algorithm 1
          If the objective function of the new source < objective function of the current source
            Update the current source with the new obtained source
            Update the objective function of the current source with the new objective function
            trial = 0
          Else
            trial = trial + 1
          End if
        End if
    End for
// Scout Bee Phase
    If trial >= limit
        Generate a new population and replace it with the old
        Calculate the objective function for the newly generated population and replace it with the old
    End if
    Keep the best solution
End for
```

7 Evaluation of the Proposed Artificial Bee Colony Optimization Algorithm

In this section, we show the effectiveness of the EABCOA by applying it to three cases. The first case consists of 8 tasks and three heterogeneous processors. The second case is 11 tasks and three heterogeneous processors. The third one consists of 11 tasks and three heterogeneous processors.

EABCOA was implemented as a system by MATLAB 2016. We run our system one more time with the initial values of the parameters pop_size = 100, max_iter = 100, trial = 0, limit = 10 for all cases.

7.1 Case 1

We consider a case of 8 tasks {T0, T1, T2, T3, T4, T5, T6, T7} to be executed on three heterogeneous processors {P1, P2, P3}. The execution cost of each task on different processors is shown in [21]. The results obtained by the proposed EABCOA are compared with those obtained by Heterogeneous Earliest Finish Time (HEFT). It is a list-based scheduling heuristic in which a task priority list is first built so that optimal allocation decisions are made locally for every task based on the task's estimated completion time [22]. Critical Path on Processor (CPOP) is a listed-based scheduling heuristic. It uses a different attribute for setting the task priorities and different strategies for determining the most effective processor for every selected task [22]. A Basic Genetic Algorithm (BGA) [23] is a genetic algorithm based on the principles of evolution found in nature to obtain an optimal solution for task graph scheduling problems. MicroGAS [11] is designed to investigate the effectiveness of a micro genetic-based scheduling algorithm to resolve the precedence task graph scheduling problem in distributed computing systems. A Genetic Algorithm (GA) based on Task scheduling to assign and execute different tasks. It aims to decrease the makespan and execution cost of tasks and increase resource utilization, Speedup, and Efficiency [16]. The results are shown in Table 1 and Fig. 4, with the proposed task priority of EABCOA {T0, T4, T1, T2, T3, T6, T5, T7}, task priority of HEFT {T0, T4, T1, T3, T2, T5, T6, T7}, task priority of CPOP {T0, T4, T3, T2, T1, T6, T5, T7}, task priority of BGA {T0, T3, T2, T1, T4, T6, T5, T7}, task priority of MicroGAS {T0, T3, T4, T1, T2, T6, T5, T7}, and task priority of GA {T0, T1, T4, T3, T2, T6, T5, T7}.

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

	HEFT			**CPOP**			**BGA**			**microGAS**			**GA**			**EABCOA**		
	P1	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT
T0			0-9	0-11			0-11					0-9			0-9			0-9
T1		20-35				35-46		22-37				19-30		20-35		20-30		
T2			28-42		28-40		22-31				26-38				19-33		26-38	
T3	23-34					25-35	11-22					9-19			9-19			28-38
T4			9-28	11-26					22-41	20-35			20-35					9-28
T5			53-58			50-55			54-59			48-53			48-53			51-56
T6	55-65			48-58					41-54	35-45			35-45					38-51
T7			78-88	76-87					59-69			58-68			58-68			56-66

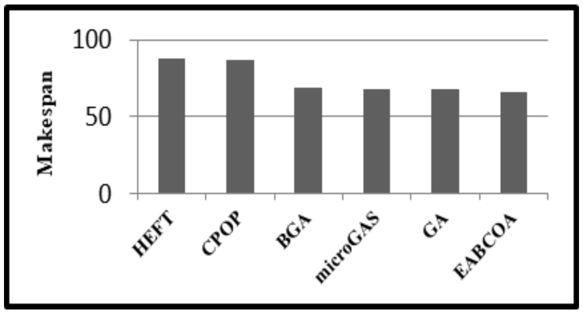


Fig. 4: Comparison of results for case 1.

7.2 Case 2

The results obtained by the proposed EABCOA are compared with those obtained by GA [16] and Enhanced Genetic Algorithm for Task Scheduling (EGA-TS) [25]. It is a genetic-based algorithm and meta-heuristic technique to handle static task scheduling for processors in heterogeneous computing systems. Multiple Priority Queues and a Memetic Algorithm (MPQMA) uses a genetic algorithm (GA) along with hill climbing to assign a priority to every subtask and to search for a solution for the task to processor mapping, using a heuristic-based Earliest Finish Time (EFT) approach. New Genetic Algorithm (NGA) [24] to improve the task scheduling solutions in a cloud computing environment, a powerful and improved genetic algorithm is proposed. The algorithm uses the evolutionary genetic algorithm's advantages and heuristic approaches. The

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

	EGA-TS			MPQMA			NGA			GA			EABCOA		
	P1	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT
T0	0-10			0-10			0-10				0-11			0-11	
T1		35-47				38-51			38-51	38-49			36-47		
T2			38-51		29-37			29-37			11-19			11-19	
T3			20-38			20-38			20-38		39-49			19-29	
T4	10-37			10-37			10-37				19-39			29-49	
T5		47-59				51-69			51-69	49-64			47-62		
T6	61-70			47-56			47-56			29-38					29-48
T7		67-79			67-79			67-79			49-61			59-71	
T8			51-66		37-47			37-47				28-43		49-59	
T9		82-94			79-91			79-91			73-85			71-83	
T10		94-104			91-101			91-101			85-95			83-93	

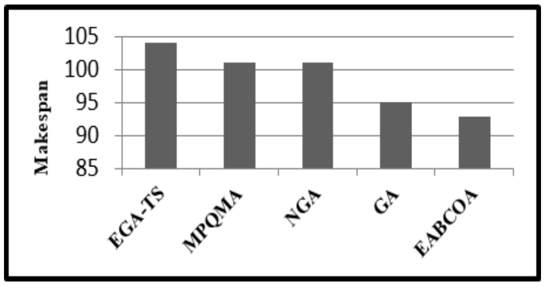


Fig. 5: Comparison of results for case 2.

7.3 Case 3

In this case, the number of tasks {T0, T1, T2, T3, T4, T5, T6, T7, T8, T9, T10} to be executed on three heterogeneous processors {P1, P2, P3}. The cost of executing each task on different processors is shown in [26] results obtained by the proposed EABCOA results are shown in Table 2 and Fig. 5, with the proposed task priority of EABCOA {T0, T2, T3, T4, T1, T6, T8, T7, T5, T9, T10}, task priority of EGA-TS {T0, T4, T3, T1, T2, T5, T8, T6, T7, T9, T10}, task priority of MPQMA {T0, T4, T3, T2, T8, T1, T6, T5, T7, T9, T10}, task priority of NGA {T0, T4, T3, T2, T8, T1, T6, T5, T7, T9, T10}, GA {T0, T2, T6, T4, T1, T3, T8, T7, T5, T9, T10}.

The results obtained by the proposed EABCOA are compared with those obtained by GA [16] and three heuristic HEFT algorithms. Upward Ranking Priority approach: Every subtask is assigned a weight from exit to entry. In the Downward Ranking Priority approach, every subtask is assigned a weight from the entry node to the exit node. Level ranking Priority approach, every subtask is placed in a sorting list based on its level value in increasing order [27]. Quantum Genetic Algorithm with Rotation Angle Refinement (QGARAR) for scheduling tasks based on distributed systems such as cloud data centers [18]. The results are shown in Table 3 and Fig. 6, with the proposed task priority of EABCOA {T0, T1, T2, T5, T3, T4, T6, T9, T8, T7, T10}, task priority of Level Priority {T0, T2, T1, T6, T3, T5, T4, T9, T7, T8, T10}, task priority of Upward Priority {T0, T2, T1, T6, T3, T5, T4, T7, T9, T8, T10}, task priority of Downward Priority {T0, T1, T2, T3, T5, T6, T4, T7, T8, T9, T10}, task priority of QGARAR {T0, T1, T4, T2, T3, T5, T6, T7, T8, T9, T10}, and task priority of GA {T0, T1, T3, T4, T2, T5, T7, T6, T9, T8, T10}

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

	Level			Upward			Downward			GA			QGARAR			EABCOA		
	P1	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**	**P1**	**P2**	**P3**
	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT	ST-FT
T0	0-7			0-7			0-7			0-7			0-7			0-7		
T1	12-22			12-22			7-17			7-17			7-17			7-17		
T2	7-12			7-12			17-22				21-28			21-28		17-22		
T3		30-38				30-37	22-28			17-23					25-32			25-32
T4			37-43			37-43			32-38			32-38	17-27					32-38
T5	34-45			34-45				28-41		23-34			27-38			22-33		
T6	22-34			22-34			28-40				28-43			28-43		33-45		
T7			49-56			43-50			39-46			38-45			35-42			38-45
T8		52-61			52-61			41-50				45-55	38-46				40-49	
T9	45-60			45-60				54-65			46-57			50-61		45-60		
T10	71-79			68-76				65-74			62-71			61-70		60-68		

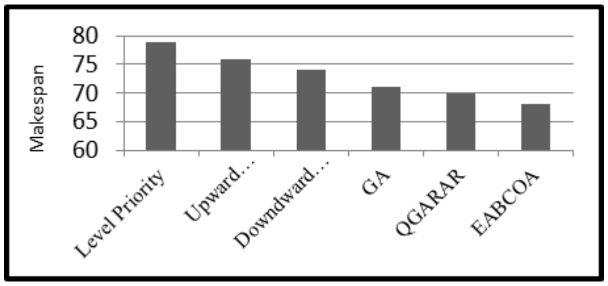


Fig. 6: Comparison of results for case 3.

7.4 Discussion

According to the results in Fig. 4 and Table 1, it is found that the Makespan of the proposed EABCOA is reduced by (25%), (24.13%), (4.34%), (2.94%), (2.94%) about HEFT, CPOP, BGA, microGAS, and GA, respectively. According to the results in Fig. 5 and Table 2, it is found that the Makespan of the proposed EABCOA is reduced by (10.57%), (7.9%), (7.9%), (2.10%) about EGA-TS, MPQMA, NGA, and GA, respectively. According to the results in Fig. 6 and Table 3, it is found that the Makespan of the proposed EABCOA is reduced by (13.92%), (10.52%), (8.10%), (4.22%), (2.85%) about Level Ranking Priority, Upward Ranking Priority, Downward Ranking Priority, GA, and QGARAR, respectively.

8 Conclusion and Future Work

In order to get near-optimal results for the problem of task scheduling in a cloud computing environment, efficient strategies for the optimal mapping of the tasks are required. This paper proposed an efficient artificial bee colony optimization algorithm (EABCOA) to solve the task scheduling problem in a cloud computing environment. The system consisted of a limited number of fully connected heterogeneous processors. The comparison of the algorithm has been made against the algorithms in terms of makespan. The comparative analysis explained that the performance of the proposed algorithm is significantly better in all cases. In our future work, we will develop an efficient task scheduling algorithm using the Cuckoo search algorithm in a cloud computing environment.

Conflicts of Interest
The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] A. Younes, A. BenSalah, T. Farag, F. A.Alghamdi and U. A. Badawi, Task Scheduling Algorithm for Heterogeneous Multi Processing Computing Systems, *Journal of Theoretical and Applied Information Technology.*, **97(12)**, 3477-3487 (2019).

[2] Q. M. Hussein and A. N. Hasoon, *Dynamic process scheduling using genetic algorithm*, in Proc. NTICT, Baghdad, Iraq., 111-115 (2017).

[3] M. Ibrahim, S. Nabi, A. Baz, H. Alhakami, M. S. Raza and et. al, An in-depth Empirical Investigation of state-of-the-art Scheduling Approaches for Cloud Computing, *IEEE Access*., **8**, 128282-128294 (2020).

[4] X. Huang, C. Li, H. Chen and D. An, Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies, *Cluster Computing*., **23(2)**, 1137-1147 (2020).

[5] R. K. Jena, Task scheduling in cloud environment: A multi-objective ABC framework, *Journal of Information & Optimization Sciences*., **38(1)**, 1-19 (2017).

[6] S. A. Alsaidy, A. D. Abbood and M. A. Sahib, Heuristic initialization of PSO task scheduling algorithm in cloud computing, *Journal of King Saud University-Computer and Information Sciences*., (2020).

[7] S. Nabi, M. Ibrahim and J. M. Jimenez, DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing, *IEEE Access*., **9**, 61283-61297 (2020).

[8] L. Tang, J-S. Pan, Z. Wang, K. Ma and H. Zhao, Novel Artificial Bee Colony Algorithm Based Load Balance Method In Cloud Computing, *Journal of Information Hiding and Multimedia Signal Processing*., **8(2)**, 460-467 (2017).

[9] B. Kruekaew and W. Kimpan, Enhancing of Artificial Bee Colony Algorithm for Virtual Machine Scheduling and Load Balancing Problem in Cloud Computing, *International Journal of Computational Intelligence Systems*., **13**, 496-510 (2020).

[10] R. Hwang, M. Gen, and H. Katayamaa, A Performance Evaluation of Multiprocessor Scheduling with Genetic Algorithm, *Asia Pacific Management Review*., **11(2)**, 67-72, (2006).

[11] J. E. Pecero, P. Bouvry, H. J. F. Huacuja, J. D. T. Villanueva, M. A. R. Zuñiga and C. G. G. Santillán, *Task Scheduling in Heterogeneous Computing Systems Using a MicroGA*, 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing., 618-623 (2013).

[12] S. AlEbrahim and I. Ahmad, Task scheduling for heterogeneous computing systems, *Journal of Supercomputing*., **73**, 2313–2338 (2017).

[13] A. A. Nasr, N. A. El-Bahnasawy and A. El-Sayed, Task Scheduling Algorithm for High Performance Heterogeneous Distributed Computing Systems, *International Journal of Computer Applications*., **110(16)**, 0975 – 8887 (2015).

[14] G. Wang, Y. Wang, H. Liu, and H. Guo, HSIP: a novel task scheduling algorithm for heterogeneous computing, *Scientific Programming*., 1–11 (2016).

[15] M. Sulaiman, Z. Halim, M. Waqas and D. Aydin, A hybrid list-based task scheduling scheme for heterogeneous computing, *Journal of Supercomputing*., **4**, 1-37 (2021).

[16] A. Y. Hamed and M. H. Alkinani, Task Scheduling Optimization in Cloud Computing Based on Genetic Algorithms, *Computers, Materials & Continua*., **69(3)**, 3289-3301 (2021).

[17] M. H. Kashani, M. Jamei, M. Akbari and R. M. Tayebi, *Utilizing Bee Colony to Solve Task Scheduling Problem in Distributed Systems*, 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks., 298-303 (2011).

[18] T. Gandhi, Nitin and T. Alam, *Quantum genetic algorithm with rotation angle refinement for dependent task scheduling on distributed systems*, 2017 Tenth International Conference on Contemporary Computing (IC3), 1-5 (2017).

[19] D. Teodorovic, T. Davidovic and M. Selmic, Bee Colony Optimization: The Applications Survey, *ACM Transactions on Computational Logic*., 1-20 (2011).

[20] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, *Applied Soft Computing*., **8**, 687–697 (2008).

[21] Y. Lee and A. Zomaya, A novel state transition method for metaheuristic-based scheduling in heterogeneous computing systems, *IEEE Transactions Parallel Distributed Systems*., **19(9)**, 1215–1223 (2008).

[22] E. H. Houssein, A. G. Gad, Y. M. Wazery and P. N. Suganthan, Task Scheduling in Cloud Computing based on Meta-heuristic: Review, Taxonomy, Open Challenges, and Future Trends, *Swarm and Evolutionary Computation*., **62** (2021).

[23] S. Gupta, G. Agarwal, and V. Kumar, *Task scheduling in multiprocessor system using genetic algorithm*, 2010 Second International Conference on Machine Learning and Computing (ICMLC)., 267–271 (2010).

[24] B. Keshanchi, A. Souri, and N. Navimipour, 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).

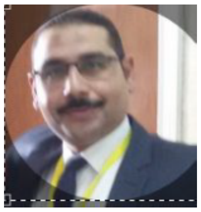
[25] M. Akbari, H. Rashidi, and S. Alizadeh, An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems, *Engineering Applications of Artificial Intelligence*., **61**, 35-46 (2017).

[26] M. Shirvani, A new shuffled genetic-based task scheduling algorithm in heterogeneous distributed systems, *Journal of Advances in Computer Research*., **9(4)**, 19–36 (2018),

[27] Y. Xu, K. Li, J. Hu, and K. Li, A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, *Information Sciences*., **270**, 255-287 (2014).

A. Younes received his Ph.D. degree in Sept. 1996 from South Valley University, Egypt. His research interests include Artificial Intelligence and genetic algorithms, specifically in computer networks. Recently, he has started conducting research in the area of Image Processing. Currently, he works as a Professor at Sohag University, Egypt. Younes always publishes the outcome of his research in international journals and conferences.

M. Kh. Elnahary Received a B.S degree from the computer science department, Sohag University, Egypt. His interests are in task scheduling and computer networks.



Hamdy H. El-Sayed Received a Ph.D. in wireless ad hoc network routing protocols from the computer science department Sohag University, Egypt, in march 2015. His research interests are ad hoc routing protocols and sensor networks, the Internet of Things, cloud computing, and security.