An Evolutionary Algorithm for Task scheduling Problem in the Cloud-Fog environment

An Evolutionary Algorithm for Task scheduling Problem in the Cloud-Fog environment

Mohammed Najm Abdulredha[*1], Bara'a A. Attea[1], Adnan Jumaa Jabir[1]

[1] Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq.
Mohammed.najm.422@gmail.com

Abstract

The rapid and enormous growth of the Internet of Things, as well as its widespread adoption, has resulted in the production of massive quantities of data that must be processed and sent to the cloud, but the delay in processing the data and the time it takes to send it to the cloud has resulted in the emergence of fog, a new generation of cloud in which the fog serves as an extension of cloud services at the edge of the network, reducing latency and traffic. The distribution of computational resources to minimize makespan and running costs is one of the disadvantages of fog computing. This paper provides a new approach for improving the task scheduling problem in a Cloud-Fog environment in terms of execution time(makespan) and operating costs for Bag-of-Tasks applications. A task scheduling evolutionary algorithm has been proposed. A single custom representation of the problem and a uniform intersection are built for the proposed algorithm. Furthermore, the individual initialization and perturbation operators (crossover and mutation) were created to resolve the inapplicability of any solution found or reached by the proposed evolutionary algorithm. The proposed ETS (Evolutionary Task Scheduling algorithm) algorithm was evaluated on 11 datasets of varying size in a number of tasks. The ETS outperformed the Bee Life (BLA), Modified Particle Swarm (MPSO), and RR algorithms in terms of Makespan and operating costs, according to the results of the experiments.

Keywords: task scheduling, cloud computing, fog computing, Evolutionary algorithm, BOT.

1 Introduction

Cloud computing refers to the computer resources and systems available on-demand over the network to provide several integrated computing services[1]. Cloud computing offers three basic types of service models, namely Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS)[2]. Cloud computing can be implemented as a layered architecture and comes in four main development models: Private, Public, Hybrid, and Community clouds[3][4].

Cloud computing provides high flexibility and speed in expanding or reducing the infrastructure, anytime, anywhere computing, efficient allocation and distribution of resources to keep capital spending from wasting unused resources, and pay as use charging strategy. However, the salient downside of the cloud system is the high transmission delay as the data destined to cloud servers should traverse a long-distance, unstable, and usually congested internet connection[5]. As a result, CISCO announced Fog computing

which brings the computation capabilities to the user vicinity. The fog nodes, which are characterized by low computing and storing capabilities in comparison with the cloud high-performance server, are placed at the edge network, such that the lightweight tasks can be processed locally without the burden of communicating with the far cloud nodes[5]. The use of fog nodes reduces both the communication latency and processing cost which in turn satisfied the user demands by reducing both the processing cost and transmission latency. However, the combination between cloud and fogs encounters various issues to meet the user demands as they deliver dissimilar services, e.g., expensive and low latency cloud services versus cheap and high latency fog services.

In cloud computing, various cloud users require different services depending on their changing needs, thus the most important challenge is how the user tasks are scheduled efficiently to satisfy different criteria for both parties, named cloud provider and cloud users. The cloud providers aim to increase their revenue without affecting the cloud system performance; this can be achieved by resource utilization, load balancing, reducing the energy consumption, and increasing the system throughput[6]. Contrarily, the cloud users look for a low cost, low latency services which are not exceeding their tasks budget and deadline limits. In general, the key issues in cloud environments have been categorized into seven major categories: resource management, load balancing, privacy and security, transition to the cloud, availability and scalability, energy efficiency, interoperability, and compatibility[6]. Scheduling in a cloud computing environment means that many tasks can be performed on the pool of available computing resources in the most optimal way. This process relies on several improvement criteria such as reliability, outreach, load balancing, implementation cost, budget, and resource utilization[7]. In the task scheduling process, the tasks are delivered from the users to the cloud scheduling software, and then the cloud scheduler explores the state of the resources from the cloud information service, as follows: the Datacenter Broker (DB) monitors all resources presented in the cloud system and collects the current status of all available resources and all remaining resources that might be available in the cloud system. The cloud scheduler then determines the target resource that should be chosen and allocated to the task based on the exploration stage information received[8]. The task scheduling can be mainly categorized into static and dynamic fashions. In the former, all the task specifications are known in advance which allows scheduling all tasks before executing any task, while in dynamic scheduling, such tasks information is not available which makes a cloud provider can't make an efficient plan before execution.

On the other hand, the tasks may appear in different models depending on the task interrelationship. One kind of task is the bag of tasks (BoT), where the tasks can be processed in parallel in a single processing level with zero dependencies among each other.

Task scheduling has been proven to be a typical NP-hard optimization problem[9]that cannot be effectively addressed by conventional methods. Recent studies have therefore stepped up research on the use of heuristic and metaheuristic algorithms for task scheduling optimization due to their effectiveness in solving problems of high complexity and large size[10]. For example, several researchers have formulated the BoT scheduling as a single objective problem to address the cost problem[11]and make the optimization process[12] with some limitations, such as the implementation deadline and the task budget.

In this paper, we focus on the task scheduling problem in a Cloud-Fog environment, a highly distributed computing platform, to address large scale BoT applications. An Evolutionary Task Scheduling algorithm (ETS) algorithm has been proposed to solve this problem. The main goal of the ETS algorithm is to achieve a good trade-off between execution time and the financial cost of completing a set of tasks in the Cloud-Fog system. Also, this algorithm can flexibly adapt to the requirements of different users as one person wants to prioritize execution time at present, while others want to complete their tasks with a limited budget.

In what follows, we summarize the main aim of this paper:

- The problem of task scheduling in Cloud-Fog environment is addressed in this paper as an optimization problem.
- A single objective evolutionary algorithm (EA) is developed to tackle the formulated problem.

Our approach was evaluated experimentally and compared with the Bee Life algorithm (BLA), the modified Particle Swarm algorithm (MPSO), and the optimized Round Robin (RR) algorithm on several datasets of different sizes. The results showed that the proposed algorithm achieved better QoS and was better at balance between time and cost efficiency verse all algorithms compared to it.

The rest of the paper is organized as follows. Section 2 describes the related scenarios proposed in the literature for solving the task scheduling problem in the Cloud-Fog. Section 3 defines the proposed algorithm. In section 4, the proposed evolutionary algorithm is evaluated. Finally, concluding remarks are summarized and some future works are given in Section 5.

2 Literature Review

The BoT applications are typical embarrassingly parallel types of applications consisting of many independent tasks, which can be processed in parallel without synchronization or communication. The BoT applications are widely spread in computer imaging, computational biology, parameter sweeps, fractal calculations, data mining, and Monte Carlo simulations[13]. Several researches have tackled the problem of finding the best scheduling solution for BoT tasks according to various optimization criteria and constraints. In literature, various works have been conducted to optimize the total scheduling cost for the BoT tasks concerning different constraints.

Zhang et al [14] proposed a greedy heuristic (GH) approach to minimize the total cost for the due-date-constrained Bag-of-Tasks Scheduling Problem (BTSP) in hybrid clouds. It encompasses two phases, task ordering and task scheduling. The task ordering is accomplished by an Earlier Latest Start Time First method (ELSTF), while the task scheduling is achieved by a Task Dispatching method (TD). The GH method outperformed the RoundRobin method in terms of total cost minimization.

A bio-inspired modified particle swarm optimization (MPSO) algorithm was proposed by Domanal et al[15] to find the minimum execution cost in a heterogeneous infrastructure of the cloud, where the backpropagation artificial neural network (ANN) was used to forecast the spot instances' future values using the past data of a particular region. Their proposed scheduling algorithm outperformed various state-of-the-art algorithms like Round Robin, First Come First Serve, Ant Colony Optimization, and Genetic Algorithm.

Zhang et al [16] considered the due date constraints while reducing the total cost for BoT the hybrid clouds. They proposed three task rescheduling heuristics by considering the data transmission cost and developed two effective acceleration methods to further enhance the efficiency.

Abdi et al [12] proposed a modified particle swarm (MPSO) scheme to optimize the task scheduling in terms of the completion time. MPSO has shown better performance in comparison with PSO and GA algorithms in the cloud environment.

Zhang et al[17] proposed an effective heuristic (EH) to find the minimum Makespan according to the budget in hybrid clouds. To accomplish the required task execution cost minimization, EH divided the optimization problem into task sequencing and scheduling phases. The longest Task First method was applied to generate a task sequence followed by the task assignment method to schedule all tasks in the obtained sequence. EH scheme outperformed the round-robin in terms of finding the minimum Makespan within the task budget.

A bee's life algorithm (BLA) was proposed by Bitam et al [18] for task scheduling problems to efficiently utilize resources in the fog network by optimizing both the execution time and memory usage

An online algorithm based on the Lyapunov optimization technique was proposed by Nan et al[19] to online adjust the tradeoff between average response time and average cost. Their proposed method outperformed both Fog-First and Cloud-First algorithms which were implemented as the performance benchmarks in their experiments.

Sindhu et al[20] proposed a novel deadline-constrained bi-objective genetic algorithm-based scheduler (DBOGA) to schedule a BoT application in a cloud environment to find an optimal trade-off between execution time and execution cost. DBOGA has defined a new fitness based on the deflection of the individual fitness from the mean of lower and upper bounds, which contributed to finding the optimal solution quickly. DBOGA outperformed both the standard genetic algorithm (SGA) and BaTS in terms of execution cost and time.

Recently, Nguyen et al [21] presented a genetic algorithm for optimization of job scheduling in a Cloud-Fog environment to reduce the makespan and operating cost. Their proposed method was compared with the Bee Life Algorithm (BLA), Modified Particle Swarm Optimization (MPSO) and Round Robin (RR) approaches in terms of makespan and execution cost. However, the main downsides of their scheme are the high complexity of chromosome representation, and both the conflicting objectives (cost and makespan) are modeled as a single objective function.

The above-mentioned scheduling schemes (BoT) are either intended for use in a single domain, i.e. cloud or fog domain, assume monolithic nodes with similar properties and unlimited capabilities, or study different optimization objectives separately. Current research work has also focused on addressing BoT scheduling.

3 The Proposed Task Scheduling algorithm

Task scheduling means the best use of the available resources. The high performance of the cloud computing environment depends greatly on the efficiency of resource scheduling. Most current scheduling algorithms take into account various parameters such as time (i.e. makespan) and cost[22]. However, there are other essential parameters, such as reliability and availability[23] to be considered.

The task scheduling problem can be expressed as a three tuple system of (*T, P, M*), where T represents the set of tasks, P is the set of processing nodes, and M is the optimal mapping between tasks and processing units. The mapping should satisfy the best execution time and cost.

The Cloud–Fog computing infrastructure consists of Cloud nodes and Fog nodes, which have the same attributes such as CPU rate, CPU usage fee, bandwidth usage fee, and memory usage fee. However, Cloud nodes are typically more powerful than the Fog nodes, but the cost of using them is greater. The set of P processors including nC Cloud nodes and nF Fog nodes in the system ($\aleph = nC \cup nF$) is expressed as:

$$\mathrm{P} = \{P_1, \mathrm{P}_2, \mathrm{P}_3, \dots, \mathrm{P}_{\aleph}\} \quad (1)$$

Makespan and monetary cost metrics are the main optimization metrics used for scheduling both BoT and workflow. In the BoT scheduling problem, the goal is to find an optimal mapping between independent tasks and corresponding computing cloud-fog nodes. Let $T = \{T_1, T_2, T_3, \dots, T_m\}$ is a set of m independent tasks which have various characteristics. Each task T_k ($T_k \in T$) is assigned to the processor P_i ($P_i \in P$), which is represented as T_k^i, where each processing node can be utilized to process a set of one or more tasks:

$$P_iT = \{T_x^i, T_y^i, \dots, T_z^i\} \quad (2)$$

For a set of P_iT tasks, the execution time (ET) that node P_i needs to complete all assigned tasks is:

$$ET(P_i) = \sum_{T_k^i \in P_iT} ET(T_k^i) \quad (3)$$

Where $ET(T_k^i)$ is the execution time of the task T_k on the processing node P_i.

$Makespan\ (M)$ is the total time for the system to complete all tasks, defined from the time when the request is received to the time that the last task is completed, or the time when the last machine finishes. It can be expressed as:

$$\mathrm{M} = \max_{1 \le i \le m} [ET(P_i)] \quad (4)$$

When one task is executed in the Cloud–Fog system, a monetary amount must be paid for processing cost, memory usage cost, and bandwidth usage cost. The cost estimated when node P_i processes the task T_k is expressed as:

$$Cost(T_k^i) = C_p(T_k^i) + C_m(T_k^i) + C_b(T_k^i) \quad (5)$$

In Equation (11), each cost is calculated as follows. Processing cost is defined as:

$$C_p(T_k^i) = c_1 * ET(T_k^i) \quad (6)$$

where c_1 is the CPU usage cost per time unit in node P_i, and $ET(T_k^i)$ is defined as Equation (9). Letting c_2 be the memory usage cost per data unit in node P_i and $Mem(T_k)$ the memory required by the task T_k, the memory usage cost is:

$$C_m(T_k^i) = c_2 * Mem(T_k^i) \quad (7)$$

Task T_k processed in node P_i needs an amount of bandwidth $B_w(T_k)$, which is the sum of input and output file size. Let c_3 be the bandwidth usage cost per data unit, the bandwidth usage cost is defined as follows:

$$C_b(T_k^i) = c_3 * B_w(T_k^i) \quad (8)$$

Total cost for all tasks to be executed in Cloud–Fog system is calculated as follows:

$$TotalCost = \sum_{P_i \in P} \sum_{T_k^i \in P_i\mathrm{T}} Cost(T_k^i) \quad (9)$$

We can define a utility function that computes the trade-off between the Makespan and total cost as follows:

$$F = \alpha * \mathrm{M} + \alpha * TotalCost \quad (10)$$

3.1. Algorithmic framework for the proposed algorithm

In this section, we present the task scheduling problem as a new approach to improving the task scheduling problem for Bag-of-Tasks applications in a Cloud-Fog environment in terms of execution time and running costs. The distinct components of the proposed GA, specifically formulas of the single initialization mechanism, recombination, and mutation operators, were designed to properly fit the problem-solving. As populations are prepared and assessed, GA then operates in generational cycles, each with solution selection, recombination, mutation, new population evaluation, and termination testing.

3.1.1 Encoding of Chromosomes

In this approach, each individual *I* is represented by a chromosome in the genetic algorithm and thus represents a solution to the task. In chromosome coding, we use the dimensional **m**, the number of genes (tasks), and the other dimension we use the א represents the number of nodes available to process the existing tasks. Each gene has a value between [1 - א]. For example Figure 1 shown in a chromosome consisting of 10 tasks and the values contained within each gene represent the nodes assigned to this task. Node 1 executes the set of tasks {1, 4, 8}, the set of tasks {5, 7, 9} is assigned to be processed in node 2, while node 3 is responsible for the set of tasks {2, 3, 6, 10}. This representation is in BoT and Workflow.

1	2	3	4	5	6	7	8	9	10
1	3	3	1	2	3	2	1	2	3

Figure 1 Encoding of Chromosomes.

3.1.2 Fitness functions

The fitness function is used to assess the degree of bad or good chromosomes in a population. It plays an important role in EA because it not only affects the convergence speed of the algorithm but also decides whether the optimal solution can be obtained. Usually, the fitness function is designed based on the objective function of the problem to be addressed. In this study, the Fitness 1 (makespan) for BoT calculated by the utility function shown in Equation (4) in section 3 and the Fitness 2 (Cost) calculated by the utility function shown in Equation (9) in section 3.

3.1.3 Selection

The selection operator is used to select solutions from the current population that will be used to form the next population of solutions (this being the basis for the next iteration of the algorithm). Parent Selection is the process of selecting parents who mate and recombine to create offsprings for the next generation. Parent selection is very crucial to the convergence rate of the EA as good parents drive individuals to a better and fitter solutions. However, care should be taken to prevent one extremely fit solution from taking over the entire population in a few generations, as this leads to the solutions being close to one another in the solution space thereby leading to a loss of diversity. Maintaining good diversity in the population is extremely crucial for the success of an EA.

3.1.4 Crossover

The crossover operator is analogous to reproduction and biological crossover. In this, more than one parent is selected and one or more off-springs are produced using the genetic material of the parents. Crossover is usually applied in an EA with a high probability (pc). using Uniform Crossover In this type, we don't divide the chromosome into segments, rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring. We can also bias the coin to one parent, to have more genetic material in the child from that parent. For example Figure 2 shown Uniform Crossover.

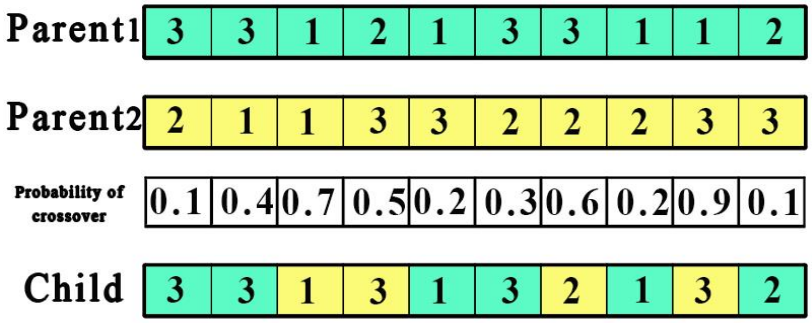

Figure 2 Uniform Crossover

3.1.5 Mutation

The mutation is another important genetic factor. Few changes are made to one or some chromosome genes and in this way the diversity in the population is preserved. We have developed two types of mutation, the first a uniform mutation, which is a factor that replaces the value of the selected gene (the node that processes the task to be mutated) with a uniform random value that is determined between the upper and lower boundaries (nodes available from the cloud and fog) specified for that gene, The Procedure of uniform mutation is described in algorithm1.

Procedure Uniform Mutation (chromosome)

```
1 :  Set  M   ← number of tasks
2 :  Set nC ← number of Cloud nodes
3 :  Set  nF  ← number of Fog nodes
4 :  Set Pm = 0.1 ← Probability of Mutation
5 :  For gene = 1 to M do
6 :      Rand ← RandInt(0,  1)
7 :      IF rand <= Pm then
8 :          NewGene = Rondom(1, nC + nF)
9 :          chromosome [gene] = NewGene
10 :     End IF
11 :  End For
```

End procedure

Algorithm 1 pseudocode Uniform Mutation

4 Experimental Results and Discussion

In this section, we present the simulation setup and evaluation of the proposed algorithm based on the results of conducted tests.

4.1 Modified Particle Swarm Optimization

is one of the heuristic algorithms (particle swarm optimization). This evolutionary algorithm, developed by Kennedy and Eberhart [24], simulates the social actions of flocks of birds or groups of fish on their way to their desired destination. Modified Particle Swarm Optimization (MPSO)[12], which can adapt to proposed model, can be described by adding or removing layers in the PSO algorithm while maintaining the general idea of PSO. Particle velocities are updated on a daily basis based on pBest and gBest, the best experience and leader. The kth particle's velocity is modified using the equation below.

$$V_k^{(t+1)}(i,j) = w \cdot V_k^t(i,j) + c_1 r_1 (p\text{Best}_k^t\ (i,j) - X_k^t(i,j)) + c_2 r_2 (g\text{Best}_k^t\ (i,j) - X_k^t(i,j)) \quad (11)$$

The new position of kth particle is defined as follows.

$$X_k^{(t+1)}(i,j) = \begin{cases} 1, & \text{if } \left(V_k^{(t+1)}(i,j) = \max_i \left\{V_k^{(t+1)}(i,j)\right\}\right), i \in \{1,2,\cdots,m\} \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

The utility function F shown in Equation(13) is used to measure the fitness value in this problem.

$$F = \alpha * \frac{\text{MinMakespan}}{\text{Makespan}} + (1-\alpha) * \frac{\text{Min Total Cost}}{\text{Total Cost}} \rightarrow \text{Max} \quad (13)$$

4.2 Bee Life Algorithm (BLA)

In 2005, the artificial bee life algorithm [25] was created as a novel meta-heuristic approach. The simulation of the minimalistic foraging model of honey bees in the search process for solving real-parameter, non-convex, and non-smooth optimization problems [26] is inspired by honey bees' intelligent foraging behavior.

In [18] authors work on the fog computing paradigm, the objective is to optimized task scheduling using the bee swarm algorithm. The aim is to find the best balance between CPU execution time and allocated memory for fog computing, The execution time (makespan) is defined in the following equation:

$$\text{CPU Execution Time}\left(FN_j\text{Tasks}\right) = \sum_{\substack{1<k<r \\ i \in \text{jobs of selected tasks}}} JTask_{ik}^j.\text{StartTime} + JTask_{ik}^j.\text{ExeTime} \quad (14)$$

The cost is defined in the following equation:

$$\text{Cost function(FNTasks)} = \min\left[\sum_m^{j=1}\left(\text{Cost function}\left(\text{JTask}_{ik}^j, \text{FN}_j\right)\right)\right] \quad (15)$$

Where

$$\begin{aligned} &\text{Cost\_function (JTask}\ {}_{ik}^{j}, FN_j) = \\ &\text{w1.CPU\_Execution\_Time}\left(FN_j\ \text{Tasks}\right) + w2.\ \text{Memory}\left(FN_j\ \text{Tasks}\right) \end{aligned} \quad (16)$$

4.3 Round Robin Algorithm

Regardless of the load on the VM, the round robin algorithm assigns tasks to the next VM in the queue. The Round Robin strategy does not take into account resource capabilities, priority, or task duration. As a result, higher priority and longer tasks result in longer response times.[27][28]. The execution time (makespan) is defined in the following equation:[27]

$$\text{Makespan}\ = max\left\{\frac{CT_j}{j \in \text{VMs}}\right\} \quad (17)$$

The cost is defined in the following equation:

$$\text{TotalCost}\ = [\sum_{x=1}^{n}\ \text{Cycles of Tasks}\ (x) *\ \text{cost of}\ VM\ \text{Instance}\ ] \quad (18)$$

4.4. Experimental Settings

Cloud and Fog nodes have different processing power as well as resource usage cost. We assumed that each node has its processing capacity represented by processing rate (measured by MIPS (million instructions per second)), along with CPU, memory, and bandwidth usage cost. Thirteen processing nodes constructed the Cloud–Fog system, with the characteristics shown in Table 1. In the Fog layer, Fog nodes such as routers, gateways, workstations, or personal computers have limited processing capacity. In the Cloud layer, servers or virtual machines in high-performance data centers are responsible for handling tasks. Therefore, the processing speed of Cloud nodes is much faster than Fog nodes. In contrast, the cost of using resources in the Cloud is more expensive than in the Fog. These costs are calculated according to Grid Dollars (G$)—a currency unit used in the simulation to substitute for real money[21].

Table 1- Characteristics of the Cloud–Fog infrastructure[21].

Parameter	Fog Environment	Cloud Environment	Unit
Number of nodes	10	3	Node
CPU rate	[500, 1500]	[3000, 5000]	MIPS
CPU usage cost	[0.1, 0.4]	[0.7, 1.0]	G$/s
Memory usage cost	[0.01, 0.03]	[0.02, 0.05]	G$/MB
Bandwidth usage cost	[0.01, 0.02]	[0.05, 0.1]	G$/MB

All user requests are tasks executed in a cloud-fog where each request analyzes or estimates what resources it needs to complete a currency, and from this we conclude that each request has attributes such as the number of instructions, memory required, the size of the input file and the size of the output file. In BoT, the tasks are randomly generated and use the attributes shown in the table2.

Table 2- Attributes of tasks[21].

Property	Value	Unit
Number of instructions	[1, 100]	109 instructions
Memory required	[50, 200]	MB
Input file size	[10, 100]	MB
Output file size	[10, 100]	MB

In an experiment, we test our proposed scheduling method using Matlab 2013a software based on the running environment of Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz and 8 GB RAM.

4.5. Experimental Results

In this section, we will examine the performance of the proposed single evolutionary algorithm for solving the task scheduling problem in a cloud–fog computing environment. Parameter settings that affect the performance of the algorithm are summarized in Table 3.

Table 3- Settings used for testing the efficiency of the proposed algorithms.

Parameter name	**Acronym**	**Possible settings**
Number of systems	nSystem	3
Number of runs	nRun	10
Number of tasks	m	{40, 80, 120, 160, 200, 250, 300, 350, 400, 450, 500}
Number of nodes	nC, nF	{3, 10 }
Population size	I	100
Number of generations	max_g	500
Probability of crossover	pc	0.5
Probability of mutation	pm	0.1, 0.3

Moreover, for each system model, we define 3 different cloud-fog systems with m tasks and nodes (nC + nF = 13). Moreover, for each system, the algorithms implement 10 different operations. We compared the performance of both algorithms, i.e., the evolutionary task scheduling algorithm (ETS), Bee Life (BLA), Modified Particle Swarm (MPSO), and RR algorithm.

Table 4 and Table 5 show the results of ETS, BLA, MPSO, and RR executions of 3 systems each consisting of 10 runs. For each run consisting of tasks m={40, 80, 120, 160, 200, 250, 300, 350, 400, 450, 500} and the cloud-fog system model with nC=3, nF=10. Competitive results are given in bold. Table 4 shows the makespan function and Table 5 shows the cost function.

Table 4- Performance of ETS algorithms in terms of system's **makespan** of 10 different runs for each system, where m is set to various number of tasks. Successful results were marked with bold against their counterparts.

Test#	m	ETS	BLA*	MPSO*	RR*
1	40	**178.921**	207.57	215.27	456.11
2	80	**383.651**	416.09	445.35	963.08
3	120	**566.593**	654.92	686.2	1495.66
4	160	**780.537**	917.67	932.33	1912.08
5	200	**974.886**	1067.49	1065.94	1905.7
6	250	**1166.4**	1490.65	1479.84	3054.8
7	300	**1460.072**	1765.49	1712.76	3309.14
8	350	**1712.174**	2010.74	1911.27	3638.19
9	400	**2014.118**	2421.98	2300.51	4347.64
10	450	**2247.181**	2840.79	2751.29	5350.35
11	500	**2435.121**	3174.39	3067.26	6023.74

* Results of these algorithms from the research paper for Nguyen et al [21].

Table 5- Performance of ETS algorithms in terms of system's **cost** of 10 different runs for each system, where m is set to various number of tasks. Successful results were marked with bold against their counterparts.

Test#	m	ETS	BLA*	MPSO*	RR*
1	40	**213.767**	730.88	740.38	755.98
2	80	**443.587**	1540.85	1553.43	1581.82
3	120	**645.495**	2367.59	2381.16	2418.9
4	160	**893.9865**	3183.8	3197.01	3246.69
5	200	**1135.027**	3767.37	3778.27	3835.4
6	250	**1413.187**	5017.17	5007.59	5079.06
7	300	**1686.916**	5877.41	5862.52	5935.94
8	350	**2007.761**	6653.37	6632.38	6738.19
9	400	**2265.633**	7816.04	7759.95	7875.39
10	450	**2548.507**	8926.57	8876.3	9016.59
11	500	**2872.858**	9995.97	9921.76	10,097.75

*Results of these algorithms from the research paper for Nguyen et al [21].

From the results reported in Tables 3 and Tables 4, we can easily see that ETS outperforms BLA, MPSO, and RR in makespan and cost optimization criteria.

Figure 3 and Figure 4 show the comparison of makespan and cost trade-off of four algorithms. It was obvious that our proposed algorithm, ETS, dominated the first position in every dataset with a high average makespan and cost, which was better than MPSO, BLA, and RR, respectively.

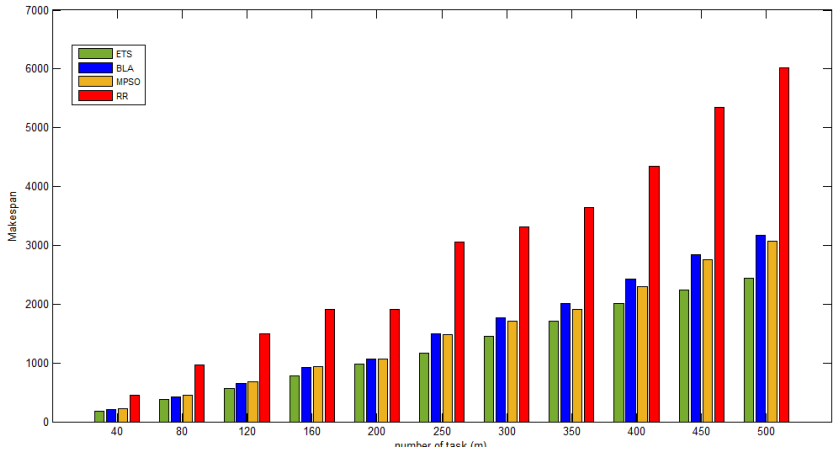

Figure 3. makespan comparison of the four algorithms.

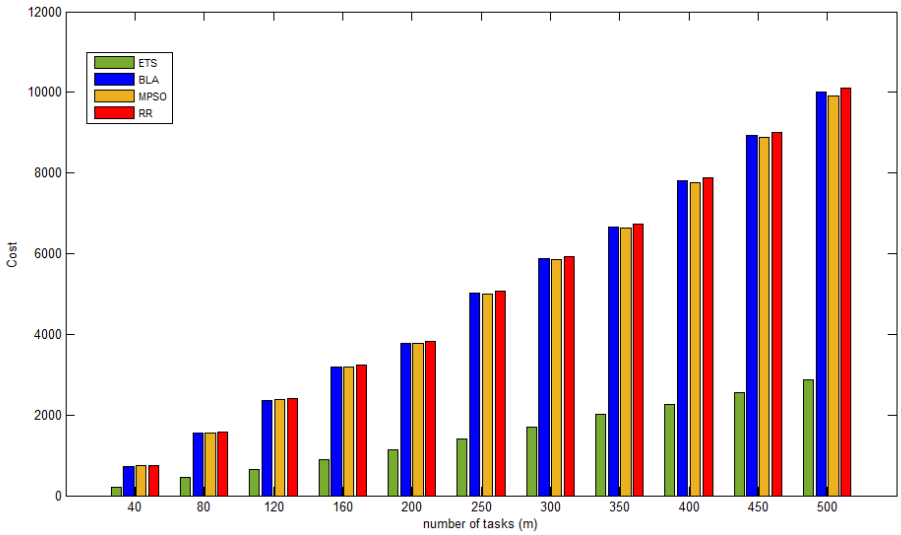

Figure 4. cost comparison of the four algorithms.

5 Conclusions

In this work, we focused on task scheduling problems for BoT applications in the cloud–fog computing environment. The ETS algorithm is designed to solve the problem as a single objective optimization problem to achieve a good trade-off between execution time and the financial cost of completing a set of tasks in the Cloud-Fog system. The problem is solved by adopting the mechanism of an evolutionary algorithm. The ETS outperformed three other methods, namely MPSO, BLA, RR, in Cloud–Fog system over 11 sets of tasks in terms of trade-off between makespan and cost execution.

In the future, we will research, improve and apply more algorithms to solve scheduling problems, especially evolutionary algorithms. Besides, we plan to expand the scheduling problem by focusing on optimizing many other objectives, such as Resource utilization, Load balancing, Budget, Deadline, and Energy efficiency to satisfy users.

References

[1] F. F. Moghaddam, M. Ahmadi, S. Sarvari, M. Eslami, and A. Golkar, "Cloud computing challenges and opportunities: A survey," in *2015 1st International Conference on Telematics and Future Generation Networks (TAFGEN)*, 2015, pp. 34–38.
[2] A. A. Laghari, H. He, I. A. Halepoto, M. S. Memon, and S. Parveen, "Analysis of quality of experience frameworks for cloud computing," *IJCSNS*, vol. 17, no. 12, p. 228, 2017.
[3] V. Kumar, A. A. Laghari, S. Karim, M. Shakir, and A. A. Brohi, "Comparison of fog computing & cloud computing," *Int. J. Math. Sci. Comput.*, vol. 5, no. 1, pp. 31–41, 2019.
[4] A. R. DAR, D. Ravindran, and S. Islam, "Fog-based Spider Web Algorithm to Overcome Latency in Cloud Computing," *Iraqi J. Sci.*, pp. 1781–1790, 2020.
[5] X. Tan and B. Ai, "The issues of cloud computing security in high-speed railway," in *Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology*, 2011, vol. 8, pp. 4358–4363.
[6] R. A. Al-Arasi and A. Saif, "Task scheduling in cloud computing based on metaheuristic techniques: A review paper.," *EAI Endorsed Trans. Cloud Syst.*, vol. 6, no. 17, p. e4, 2020.
[7] N. Ranaldo and E. Zimeo, "Time and cost-driven scheduling of data parallel tasks in grid workflows," *IEEE Syst. J.*, vol. 3, no. 1, pp. 104–120, 2009.
[8] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *2016 18th Asia-Pacific network operations and management symposium (APNOMS)*, 2016, pp. 1–4.
[9] D. Rahbari and M. Nickray, "Scheduling of fog networks with optimized knapsack by symbiotic organisms search," in *2017 21st Conference of Open Innovations Association (FRUCT)*, 2017, pp. 278–283.
[10] N. Soltani, B. Soleimani, and B. Barekatain, "Heuristic algorithms for task scheduling in cloud computing: a survey," *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 8, p. 16, 2017.
[11] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds," in *2011 IEEE third international conference on cloud computing technology and science*, 2011, pp. 320–327.
[12] S. Abdi, S. A. Motamedi, and S. Sharifian, "Task scheduling using modified PSO algorithm in cloud computing environment," in *International conference on machine learning, electrical and mechanical engineering*, 2014, pp. 8–9.
[13] Y. Zhang and J. Sun, "Novel efficient particle swarm optimization algorithms for solving QoS-demanded bag-of-tasks scheduling problems with profit maximization on hybrid clouds," *Concurr. Comput. Pract. Exp.*, vol. 29, no. 21, p. e4249, 2017.
[14] Y. Zhang, J. Sun, and J. Zhu, "An effective heuristic for due-date-constrained bag-of-tasks scheduling problem for total cost minimization on hybrid clouds," in *2016 International Conference on Progress in Informatics and Computing (PIC)*, 2016, pp. 479–486.
[15] S. G. Domanal and G. R. M. Reddy, "An efficient cost optimized scheduling for spot instances in heterogeneous cloud environment," *Futur. Gener. Comput. Syst.*, vol. 84, pp. 11–21, 2018.
[16] Y. Zhang, J. Zhou, and J. Sun, "Scheduling bag-of-tasks applications on hybrid clouds under due date constraints," *J. Syst. Archit.*, vol. 101, p. 101654, 2019.
[17] Y. Zhang, J. Sun, and Z. Wu, "An heuristic for bag-of-tasks scheduling problems with resource demands and budget constraints to minimize makespan on hybrid clouds," in *2017 fifth international conference on advanced cloud and big data (CBD)*, 2017, pp. 39–44.
[18] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterp. Inf. Syst.*, vol. 12, no. 4, pp. 373–397, 2018.
[19] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "Cost-effective processing for delay-sensitive applications in cloud of things systems," in *2016 IEEE 15th international symposium on network computing and applications (NCA)*, 2016, pp. 162–169.
[20] S. Sindhu and S. Mukherjee, "An evolutionary approach to schedule deadline constrained bag of tasks in a cloud," *Int. J. Bio-Inspired Comput.*, vol. 11, no. 4, pp. 229–238, 2018.
[21] B. M. Nguyen, H. Thi Thanh Binh, and B. Do Son, "Evolutionary algorithms to optimize task

scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Appl. Sci.*, vol. 9, no. 9, p. 1730, 2019.
[22] S. Kaur, P. Bagga, R. Hans, and H. Kaur, "Quality of Service (QoS) aware workflow scheduling (WFS) in cloud computing: A systematic review," *Arab. J. Sci. Eng.*, vol. 44, no. 4, pp. 2867–2897, 2019.
[23] S. S. K. Kumar and P. Balasubramanie, "Dynamic scheduling for cloud reliability using transportation problem," *J. Comput. Sci.*, vol. 8, no. 10, p. 1615, 2012.
[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, 1995, vol. 4, pp. 1942–1948.
[25] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Citeseer, 2005.
[26] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, vol. 39, no. 3, pp. 459–471, 2007.
[27] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks," *Sci. world J.*, vol. 2016, 2016.
[28] P. Pradhan, P. K. Behera, and B. N. B. Ray, "Modified round robin algorithm for resource allocation in cloud computing," *Procedia Comput. Sci.*, vol. 85, pp. 878–890, 2016.