A New Task Scheduling Algorithm based on Water Wave Optimization for Cloud Computing

Dina A. Amer
Computer and System Engineering Dept,
Faculty of Engineering, Zagazig University, Egypt

Gamal Attiya
Computer Science and Engineering Dept,
Faculty of Electronic Engineering, Menoufia University, Egypt

Ibrahim Ziedan
Computer and System Engineering Dept,
Faculty of Engineering, Zagazig University, Egypt

Aida A. Nasr
Robotics and Intelligent Machines Dept,
Faculty of Artificial Intelligence,
Kafrelsheikh University, Egypt

ABSTRACT

Nowadays cloud computing provides many benefits for organizations. Businesses can ensure reliable calamity recovery and backup solutions without the spat of tuning them up on a physical machine. For many companies, exploiting complex calamity recovery plans can be an expensive guarantee, and backing up data is time exhaustion. The cloud itself is built in such a way that the data stored more than one time in servers, so that if any server fails, the data is backed up immediately. The capability of accessing data readily is available after handling the failure. However, still, cloud computing resources face many problems such as scheduling problems. This paper tackles the resource scheduling problem and presents a new efficient algorithm, called Improved Water Wave Optimization (IWWO), to address such a problem. The main idea is the enhancement/improvement of the Water Wave Optimization (WWO) algorithm by using reinforcement learning to overcome the local optimality of the conventional WWO during the searching process. The proposed IWWO is implemented in the CloudSim toolkit and evaluated by considering a real data set and a randomly generated data set. The results are compared with the results of the Genetic Algorithm (GA) and Ant Colony Optimization (ACO) algorithm. The obtained results show that the IWWO can solve the resource scheduling with minimum schedule length and a high balance degree.

Keywords

Cloud computing, task scheduling, optimization, and water wave optimization

1. INTRODUCTION

Cloud computing gives companies a high level of flexibility over the presented services. There are unlimited virtual resources with various capacities to provide users with different functions. Moreover, several new services, like Big Data as a Service (BDaaS), are now available to users [**1**]. Many users, companies, and governments are toward transferring their works and data to the cloud aiming to save cost and time. Nevertheless, the widespread use of cloud computing in different fields causes many challenges as load balancing, power consumption, security, and resource scheduling. An important factor affecting cloud performance is the task scheduling technique. Weak algorithms waste the computing power of the cloud resource. On the contrary, developing a smart algorithm can increase the performance of cloud computing and save time and money [**2**] because it has an important role in optimizing the utilization of the available resources. It refers to the process of distributing tasks of a given application onto available resources (virtual machines /VMs) in the cloud. Since the number of VMs is limited and has different capabilities, there is a need for an efficient scheduling method for carefully assigning tasks to virtual machines [**3**]. In a cloud environment, the number of user tasks and the number of available resources can grow rapidly. This requires task scheduling to play a major role in enhancing the stability and reliability of the cloud system. The primary objective of cloud task scheduling is to schedule user tasks at the same time and provide the job with efficient resources to meet QoS parameters such as reducing execution time for all submitted tasks [**4,5**]. Recently, several methods are proposed to solve the scheduling problem [**6,7**]. Nevertheless, most of the existing methods tackle one or more performance parameters without considering the limitations of the available resources. Further, the scheduling algorithms that achieved a minimum schedule length provide a large computational time, and vice versa. Briefly, the algorithms concerned with reaching the optimal schedule length, take more time to schedule the cloudlets, while the algorithms concerned with reducing computation time, fall into local optimality of the solution. The research paper's key motivation is to closely address the problem of multi-objective scheduling that selects the optimal resource for user's cloudlets to enhance required parameters of quality of services (QoS) such as processing time, makespan, throughput, and a high degree of balance through distributes the user's tasks onto the available virtual machines in an efficient way that in turn enhances cloud computing resources utilization. This paper introduces a new meta-heuristic optimization approach, named Improved Water Wave Optimization (IWWO) algorithm, to tackle the scheduling problem. The introduced algorithm is derived from a new nature-inspired optimization algorithm called Water Wave Optimization (WWO). The WWO is based on shallow water wave theory and imitates wave motion to solve optimization problems. The main objective of the proposed IWWO algorithm is to improve the WWO algorithm by using reinforcement learning to overcome the local optimality problem of the conventional WWO [**8**]. Furthermore, another

enhancement to the proposed IWWO is done by applying the Max-Min algorithm as the breaking stage of the IWWO algorithm. The proposed IWWO is implemented in the CloudSim toolkit and evaluated by considering a real data set and a randomly generated data set. The results are compared with those obtained with the GA and ACO algorithms. The obtained results prove that the IWWO can solve the resource scheduling with minimum schedule length, high throughput, and high balance degree.
The main contributions of this paper are as follows:

- Formulating WWO and IWWO algorithms for the scheduling problem taking into account the availability of cloud resources.
- Developing an Improved Water Wave Optimization (IWWO) algorithm to tackle the scheduling problem and solving the drawbacks of the conventional WWO.
- IWWO scheduling algorithm has been implemented and tested at Cloudsim simulator by submitting random independent tasks and real NASA-iPSC data set, and results demonstrate that IWWO improved various QOS parameters.

The rest of this paper is organized as the following. Sect. **2** presents the cloud computing model and scheduling process. Section **3** formulates the scheduling problem as an optimization problem. Section **4** presents a literature survey of related work. Section **5** describes the WWO algorithm in some detail. The proposed IWWO is presented in Section **6**. Section **7** introduces the experimental results and discussion. The last section introduces the concluding remarks and future work of this research.

2. CLOUD COMPUTIONG AND SCHEDULING PROBLEM

Within cloud data centers, cloud infrastructure consists of a limited number of heterogeneous physical servers. Each server can host one or more VMs that are run in a parallel way using one of the main sharing (time-shared or space-shared) policies [**9**]. Further, each data center has a data center broker that has a scheduler, the backbone of the scheduling process, which is responsible for assigning arrived jobs/cloudlets onto the available VMs. When a cloud user submits a cloudlet to the cloud, it firstly enters the task management component that organizes incoming requests. The task manager then transmits the sent tasks to the task scheduler, which distributes the incoming tasks to the convenient and relevant VMs by applying the scheduling algorithm. The scheduler takes its scheduling decision depending on the current state of the VMs provided by the Cloud Information System (CIS). That is, if a VM is not available, the tasks will wait in the task queue. When the virtual machine finishes processing, the existing tasks, it can be used for other tasks, and so [**10**]. Figure 1 shows a sample of cloud environments and cloudlets scheduling.

3. SCHEDULING PROBLEM FORMULATION

As mentioned earlier, the scheduling problem refers to the process of assigning user tasks/cloudlets onto the available VMs in cloud.

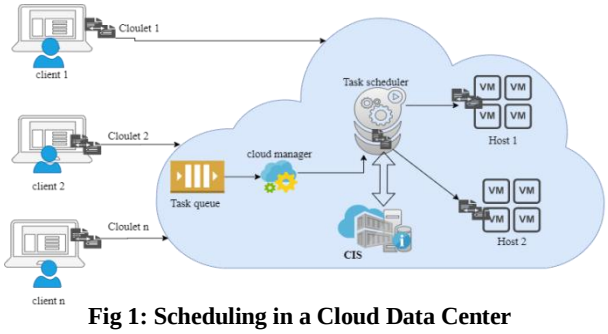

Fig 1: Scheduling in a Cloud Data Center

Since the number of submitted tasks (n) is greater than the number of available VMs (m) and different tasks have different requirements and the resources availability in cloud have dynamic nature, the scheduling problem is defined as NP-complete. This section presents the formulation of the scheduling problem as an optimization problem to be solved by an optimization algorithm.
The cloud user submits n cloudlets $\{Cl_1, Cl_2, \ldots\ldots Cl_n\}$, each cloudlet has a specific length $\ell(Cl_i)$ in Million Instruction (MI) for processing in the physical host Ph= $\{Ph_1, Ph_2, \ldots\ldots, Ph_M\}$ in the cloud data center. Each physical host has m virtual machines $VM = \{v_1, v_2, \ldots\ldots v_m\}$ each v_i has specific configurations such as main memory (v_{mem}), storage v_{st}, processing power (v_{mips}) in MIPS or Million Instruction Per Second, and number of cores (v_{cpus}). Our objective is to find the convenient mapping of each task in cloud resources so that it is possible to enhance QoS parameters.
Let x_{ij} be a binary decision variable as:

$$x_{ij} = \begin{cases} 1 & if\ Cl_i\ allocated\ to\ v_j \\ 0 & otherwise \end{cases}$$

Then, the scheduling problem may be formulated minimizing the objective function that is the task processing time (CPT) as:

$$min\ CPT = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathrm{ET}(Cl_i, v_j)\, x_{ij} \qquad (1)$$

Subject to

$$\sum_{i=1}^{n} \ell(Cl_i)\ x_{ij} \leq Total_{MIPS}(v_j) \qquad (2)$$

$$\sum_{i=1}^{n} mem(Cl_i)\ x_{ij} \leq (v_{j_{mem}}) \qquad (3)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad for\ task\ t_i \qquad (4)$$

The objective function (CPT) in this model is to minimize the total execution time of all submitted tasks. Where, Task execution time $\mathrm{ET}(Cl_i, v_j) = \frac{\ell(Cl_i)}{Total_{MIPS}(v_j)}$. The constraints are to satisfy task requirements without wasting cloud resources. Equation (**2**) represents the first constraint that is to prevent overloading at any virtual machine (v_j) and maintaining system balance. It ensures that the required load for all tasks assigned to a virtual machine(v_j) doesn't exceed the processing power of that virtual machine. It also ensures that the total number of tasks assigned to a virtual machine at a time must be less than or equal to $Total_{MIPS}(v_j)$. Where, $Total_{MIPS}(v_j) = v_{mips} * v_{cpus}$. The second constraint, Eq. (**3**), guarantees that the required memory for processing all tasks assigned to a virtual machine doesn't exceed its available memory. Finally, the third constraint, Eq. (**4**), assures that each task is allocated to only one VM while more than one task may be assigned to the same VM at a time. The tasks are non-preemptive so that each task must be executed

without any interruption [**11**].

4. RELATED WORK

The scheduling problem is solved by implementing mathematical techniques like exhaustive search algorithm [**12**] and branch-and-bound [**13**] to achieve the optimal solution. However, the time complexity exponentially increases as the number of tasks and/or VMs increases. In [**14**], a heuristics method, called FCFS, is presented to solve the problem. It first collects all users' tasks in a queue and then the scheduler determines which task will be mapped on VM depending on the task arrival time. The FCFS technique is the default method in the cloud computing system, but it leads to a very high schedule length as it does not consider any criteria for allocating tasks into VMs. Many heuristic algorithms have been designed and implemented to resolve the scheduling problem, but it is difficult to choose the best algorithm to solve problems of task assignment because the techniques are developed under different assumptions. In [**15**], The author discusses heuristic approaches for task scheduling and provides a distinction between them. The proposed results prove that Min-Min approach is the best suited for improving cost, makespan and throughput. The Max-Min algorithm exhibits a good performance for achieving the optimal task scheduling in the IaaS cloud model. The MET algorithm achieves better results for the degree of imbalance of the optimal task scheduling. For obtaining the optimal results with multi-objective scheduling problems, the author recommended the hybridization of heuristic and meta-heuristic algorithms. Therefore, in this paper Max-Min algorithm is implemented as a breaking operator for enhancing the performance of the WWO. For obtaining the optimal results with multi-objective scheduling problem, the author recommended combining heuristic and meta-heuristic algorithms. Therefore, in this paper Max-Min algorithm is implemented as a breaking operator for enhancing the performance of the WWO. The Max-Min algorithm's concept is to begin scheduling tasks with the longest completion time to the available resource. This algorithm consists of two phases. It starts with estimating the completion time for each task in the task list with a different VM. In the second phase, the task with the longest expected completion time is identified and allocated to the resource that gives the shortest completion time and then deleting the chosen task from the task list. This process is repeated until all tasks are scheduled [**16**,**17**]. The meta-heuristic techniques are based on simulating the behavior of natural phenomena. These algorithms are used for solving a problem with low time complexity [**18-22**]. Genetic Algorithm (GA) [**23**] is one of the most common optimization techniques. It is imitating on evolutionary and chromosomal formation concepts. The standard GA works through several operations: the forming of an initial population, the assessment of the initial population's solutions, determination of the best one, and procreation to generate a new population. Crossover and mutation are two operations that are applied to create alternative solutions. For each operation, the algorithm formulates a new solution (child) and adds this solution to a new population according to a specific fitness function. In [**24**], a modified GA is proposed to improve the execution time of all tasks, minimize the total execution cost, and maximize the resource utilization.

In [**25**], the authors proposed a new parallel bi-objective hybrid GA that minimizing the makespan and energy consumption. GA-based task scheduling for dynamic resource provisioning has been presented in [**26**]. Based on the results of workload forecasting, the author presented a cost-optimized resource scheduling strategy in a cloud computing environment to minimize the total cost of renting virtual machines. The proposed scheduling module was tested using NASA Ames iPSC / 860 data set and Google data tracking and the obtained results were compared with the FCFS, Max-Min, and Min-Min scheduling algorithms. The ACO algorithm simulates the cooperative behavior of real ants. The ACO algorithm was applied to solve combinatorial optimization problems, and it is very successful in solving various problems [**27**]. The main concern for ant behavior is the collective behavior among ants to perform complex tasks such as transporting food and finding the shortest path to food sources. The ant colony reaches the food source by tracing some of them from the colony's nest to the discovered food source. The ants follow others during their trips by leaving a chemical trace (pheromone) on the ground to determine the shortest path to food. Pheromone is an olfactory and volatile substance that loses its concentration over time. The role of this trace is to direct the other ants to the target point. The greater the value of pheromone on a given path, the greater the probability (p) that ants will choose the same path. For an ant, its path is determined according to the value of pheromone on it.

In [**28**], ACO used for task scheduling in cloud computing, the authors introduced a new improvement in the ACO algorithm for achieving the load balancing through task scheduling depending on the past result in task scheduling. The new meta-heuristics algorithms are also proposed for solving the scheduling problem, like the FGMTS algorithm presented in [**29**]. Where, Gray Wolf Optimizer algorithm is combined with existing fractional theory. As well as the formulation of the multi-objective function to solve the scheduling problem. The objective of the proposed improvement considers parameters, such as execution time, communication time and cost, implementation cost, energy consumption, and resource utilization. While in [**30**], the SA is combined with the Harris Hawk optimizer (HHO) for improving the local search process in the exploration phase and improving the convergence rate and the solution quality of the conventional HHO algorithm. The authors state that the HHOSA outperforms other algorithms, namely, PSO, SSA, MFO, FA, and HHO. Also, in [**31**], the whale optimization algorithm is presented. The author introduces an optimized version of the Whale optimization algorithm to solve the early convergence problem in the conventional whale optimization algorithm.

5. WATER WAVE OPTIMIZATION (WWO) ALGORITHM

The traditional water wave optimization algorithm simulates a shallow water wave model. This is a population based algorithm, where each solution in the population is similar to a wave, and the search area is similar to the seabed area. The fitness value of the wave is influenced by the depth of the sea floor, the short distance from the stable water level and the high water level [**32**]. Each solution is equivalent to a unique wave position in the seabed area. Each wave characterizes by specific wave height and wavelength. The wave height is an integer number that represents the wave energy while wavelength belongs to real numbers and corresponding to the searching area [**33**]. The most fitness wave has a small wavelength and large wave height (high energy) as shown in Figure **2**. The searching process of the WWO depends on three operators: propagation, breaking, and refraction.

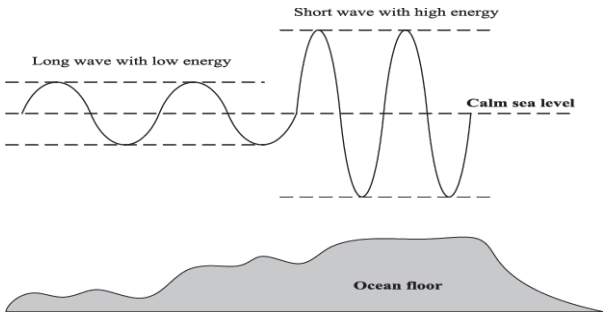

Fig 2: Different shapes for the wave in deep and shallow water [32]

i) Propagation Operator

The propagation operator represents the global search process, where each wave is propagated only once in each generation. A new dimension of the new propagated wave w' expressed by shifting a dimension d in the original according to Eq. (**5**):

$$w'(d) = w(d) + \lambda_w . r. L(d) \quad (5)$$

Where, r is a uniform distribution (random values [-1,1]), $L(d)$ is the d_{th} dimension length of the search space ($1 \leq d \leq n$) and λ_w is the wavelength of the propagated wave w. If the new position is outside the valid range, it is repositioned in a new random position in the range. Then the wave length is initialized with 0.5 for all waves and then updated after each generation based on Eq. (**6**):

$$\lambda = \lambda . \alpha^{-(f(w)-f_{min}+\varepsilon)/(f_{max}-f_{min}+\varepsilon)} \quad (6)$$

Where, $f(w)$ is the fitness value of the wave w, f_{max} and f_{min} are the maximum fitness and minimum fitness values in the current population, ε is a small positive number to avoid division by zero and α is the wavelength reduction coefficient.

ii) Refraction Operator

After applying the propagation factor, the wave height may be increased or decreased according to the fitness of the generated wave. The wave height will be increased if the generated wave fitness is better, which refers to the wave propagates from deep water to shallow water. Otherwise, the wave height will be reduced. When the wave height reaches zero, the wave has not improved for generations. It loses its momentum and discards. Then by applying the refractive operator according to the Eq. (**7**), a new wave can be generated at a random location between the old wave and the best-known solution w^*.

$$w'(d) = N\left(\frac{w^*(d)+w(d)}{2}, \frac{|w^*(d)-w(d)|}{2}\right) \quad (7)$$

Where, w^* is the best solution, and $N(\mu, \sigma)$ is a Gaussian random number with standard deviation σ and mean value μ. The height of the new wave is reset to maximum height h_{max} and its wavelength is updated by Eq. (**8**).

$$\lambda' = \lambda \frac{f(w)}{f(w')} \quad (8)$$

iii) Breaking Operator

When the propagation operator generates a new wave (w') better than the best-known wave (w^*), the WWO implements the breaking operator to conduct a local search around the wave multiple times. The first step is to randomly define the k dimension (where k is a random number between 1 and the predetermined number k_{max}), and in each d dimension it will generate a solitary wave w' according to Eq. (**9**):

$$w'(d) = w(d) + N(0,1).\beta\, L(d) \quad (9)$$

Where, β is the breaking coefficient. If none of the solitary waves is better than w^*, then it will remains, otherwise, w^* is replaced by the fittest wave between the solitary waves. The overall steps of the conventional WWO algorithm are given in table 1. In [**34**], an adaption for the WWO is proposed to address the TSP. As the propagation is carried out by subsequence reversal, breaking is performed by local search on a swap-based neighborhood structure. Results indicate that the modified WWO has better performance than GA and BBO. In [**35**], a similar approach is used to adapt the WWO to solve the permutation Flow-shop Scheduling Problem (FSP) except that local search is conducted by the NEH reinsertion method. In this paper, the conventional WWO procedure is the same as those adapted for TSP. The propagation operator is achieved by generating a random real number r (between 0 and 1) for each dimension d and comparing it to the wavelength λ to determine if the wave propagated or not. In refraction, it makes waves absorb some of the best-known wave features. This is achieved by transferring the randomly selected subsequence from the best known solution w* to the corresponding portion of the refracted solution w. While the breaking operation is applied to each newly found best solution w*, then directly generate k_{max} solitary waves, each of which is obtained by alternating two randomly selected components.

Table 1: The Conventional WWO algorithm framework

Algorithm 1: Conventional WWO Algorithm	
***Input**: tasks list, VMs list **Output**: scheduling solution*	
1.	Initialize a population of n random waves
2.	WHILE (stop criterion is not met) DO
3.	FOR each wave in population DO
4.	Propagate w to w' based on Eq. 5
5.	if (f(w') > f(w)) then
6.	Replace w with w'
7.	if (f(w') > f(w*)) then
8.	Break w' based on Eq. 9
9.	Update w* with w'
10.	end if
11.	else
12.	Decrease wave height by one
13.	end if
14.	if (wave height ==0) then
15.	refract w to w' based on Eq. 7 and Eq. 8
16.	Update the wavelength based on Eq. 6
17.	ENDFOR
18.	ENDWHILE
19.	return w*

6. PROPOSED TECHNIQUE

6.1 Improved Water Wave Optimization

The main idea of the Improved Water Wave Optimization (IWWO) algorithm is to improve the traditional WWO algorithm in the process of movement from a solution to the neighbor solution by using a scientific intelligent method called *Reinforcement Learning (RL).* In the proposed improvement, the properties of the solution (wavelength and wave height) are guided by the RL to make the WWO method more intelligent in dealing with the problem. Where the RL makes these properties to learn from its old movement to move towards the best solution and avoid bad solutions and a local minimum in the search space. *Reinforcement Learning (RL)* is a method of machine learning. In the RL, the agent moves in all possible directions to reach a solution, and then an evaluation is performed to determine the quality of that solution as well as determine the reward or punishment. The agent should maximize the expected reward by improving the obtained solution quality [**36**]. In the proposed IWWO, the wave is rewarded by improving its quality through increasing

wave height and adding it to the population and punished by decreasing its wave height. Table 2 shows the proposed IWWO algorithm. The searching process of the proposed IWWO depends on four operators: propagation, breaking, refraction, and learning. These operators are clearly discussed in the following sections through the rewarded function steps.

Table 2: The proposed IWWO algorithm

Algorithm 2: IWWO algorithm
***Input**: tasks list, VMs list **Output**: scheduling solution*
1. Initialize a population W of n random waves
2. WHILE (stop criterion is not met)
3. FOR each wave in population
4. Propagate w to w'
5. if (w' rewarded) then
6. Break w'
7. Else
8. if (wave height ==0) then
9. Refract w and reset its wave height to h_{max}
10. Update wavelength
11. End if
12. if (f(w') < f(w*)) then
13. Update w* with w'
14. Break w to w'
15. If (w' rewarded)
16. Repeat steps from step 8 to step 12
17. ENDFOR
18. ENDWHILE
19. return w*

i) Propagation Operator

As mentioned previously, at each generation, the wave propagated once according to its wavelength value. In the proposed approach, the propagation operator is implemented as in the TSP solving. First, it generates a random number r (r is uniform distributed in the range between 0 and 1). If $r < \lambda$, the subsequence of the wave w [d, d + L] is reversed, and L is a random integer in the range [1, n-d]. The comparison of wavelength satisfies that the low fitness wave, with a large wavelength, has a large probability for propagating, while the high fitness wave, with a small wavelength, has a small probability for propagating and vice versa. The pseudo-code of the propagation operator is given in Table 3.

Table 3: Propagating Algorithm

Algorithm 3: propagation
Input: wave and wavelength **Output**: new wave
1. Given a wave $w \epsilon W$ and its wavelength λ
2. for each dimension d of w do
3. Generate r uniformly distributed in [0, 1]
4. if r < λ do
5. Generate an integer L in [0, n - d]
6. Reverse w [d, d + L], obtain w'
7. return w'

ii) Refraction Operator

As mentioned previously, a wave is refracted if its height approaches zero. In the conventional WWO, the wave is reset into a new random position between old and best positions. In the proposed algorithm, we want the wave to learn from the best obtained wave by determining the similar dimensions with the optimal wave to follow them and avoiding its old bad characteristics that causing a bad generation. New wave height is set to h_{max} and the wavelength is updated based on Eq. (8). After implementing refraction, the wave w is replaced with w' in the population W. The refraction algorithm is given in Table 4.

Table 4: Refraction Algorithm

Algorithm 4: refraction
Input: wave and wavelength **Output:** new wave with maximum height
1. Given a wave $w \epsilon W$ and its wavelength λ
2. Count(s_n) the similar dimension between w* and w'
3. Generate a uniformly distributed in [0, n]
4. Generate an integer b in [0, s_n*(n – a)]
5. Replace w [a, a+b], obtain w'
6. return w'

iii) Breaking Operator

The breaking operator represents the exploiting stage of the WWO as it has the role of local search for the current optimal solution to generate a better solution than last obtained. So, if we discover that the newly generated solution is better the breaking will be activated. Here, we have implemented the steps of the Max-Min heuristic algorithm as a local search. Where the new propagated wave is rewarded and then is broken to enhance its quality based on the Max-Min steps. The main concept of the Max-Min algorithm is declared in section 4 and its implementation steps are given in table 5.

Table 5: Max_Min Algorithm

Algorithm 5: Max-Min
Input: task list **Output**: scheduled list
1. Given a task list T
2. For each submitted task (t_i) in task list (T) // start phase 1
3. For each resource (R_j in available resource list
4. Compute completion time $CTij = ET_{ij} + r_j$
5. End for
6. End for
7. While task list isn't empty // start phase 2
8. Find task with maximum CT and assign it to resource that gives minimum ET
9. Remove this task from list
10. Update ready time for selected resource
11. Update CTij for unselected tasks
12. End while
13. End

iv) Learning Process (Rewarded decision)

In the proposed algorithm, the learning process is implemented by applying the principle of *reinforcement learning*. In other words, determining the reward or punishment decision for the newly generated wave is decided by reinforcement learning. As mentioned previously, the higher the wave height, the higher the fitness wave, the better solution, and vice versa. Therefore, when a new generated wave improves solution, it must be rewarded by increasing its height; otherwise, it must be punished by reducing its height. There are possible four cases for the new wave position and the reward function:

Case 1: The fitness value of the new wave is better than the fitness of the current best wave. Here, the new wave will be rewarded with the maximum possible reward by replacing the current best wave with the new wave and increasing its height to the maximum available height (h_{max}) and including the new wave into the population.

Case 2: the new wave is not better than the current best wave but it better than the old wave. In this case, the new wave

must be rewarded by adding it to the population because it surpassed the old wave. Further, punishing it is done by reducing its height by one because it does not learn from the current best wave to outperform it.

Case 3: the new wave is not better than the old wave but it does not exceed the threshold value. In this case, the new wave will be punished by reducing its length by two and adding it to the population.

Case 4: the new wave exceeds the threshold limit. This means that this wave is very bad and will not be improved and does not reach a better position. Therefore, the new wave will be punished by neglecting and removing it from the population.

6.2 Performance Evaluation

This section introduces the experimental results of scheduling different number of user tasks into a different number of VMs. The results of the proposed IWWO are compared with those obtained by the conventional WWO, ACO, and GA in terms of schedule length, computation time, memory usage, balancing degree, and throughput.

6.3 Experimental Environment

A Core i5 laptop with 8GB RAM and a 64-bit Windows 7 operating system was used for the simulation. In this assessment, the well-known CloudSim tool kit is used to simulate a cloud computing environment [**37**]. The CloudSim provides the main classes used for building the cloud, such as data center class, host class, cloudlet class, etc. Parameters WWO and IWWO; Population size, number of iterations, maximum wave height, and initial wavelength were set as 100, 50, 6, and 0.5, respectively. The ACO parameters; the ants' number, iterations number, Q, ρ, and initial pheromone are set to 10, 70, 100, 0.7, and 0.3 respectively. The data center and host configurations are given in table 6 while the characteristics of VMs are given in table 7.

Table 6: Data center and host configuration

Cloud entity	Characteristic	value
Data Center	No. of Data Centers	1
	No. of Hosts	3
	No. of users	1
Host	Storage	1 TB
	RAM	2 GB
	BW	10 GB
	Shared policy	Space shared policy

Table 7: VM characteristics

Characteristic	Value
No. of VMs	10,25, 50 and 75
v_{mips}	500 to 2000
v_{mem}	500 MB
BW	0.5Gb/S
VMM	Xen
Size	100 MB
v_{cpus}	2,8,4,16,32

6.4 Data Sets

In this paper, two cases are considered in the experimental test:

Case 1: Application tasks are randomly generated with different lengths in range [1000, 10000] *million instructions* while the virtual machines are generated with a number of cores between 1 and 4, and MIPS from 400 to 1500.

Case 2: Several sets of real applications are used from the harmonized standard workload [**38**]. These data sets were taken from records in the NAS Division of Numerical Aerodynamic Simulation Systems (NAS) at the NASA Ames Research Center. NASA-iPSC-1993-1.1-cln.swf uses the replaced cleaning log on 1 Aug. 2006 for trial. The records contain 3-month accounting records for the 128 iPSC / 860 process item. After cleaning up a total of 43,910 records, this log file used for research work contains only 42,264 records [**39**]. The record workload from NASA Ames iPSC / 860 in SWF format is shown in Fig 3. The workload on iPSC / 860 is a mixture of interactive and batch functionality (development and production) that mainly consists of computational aviation science applications.

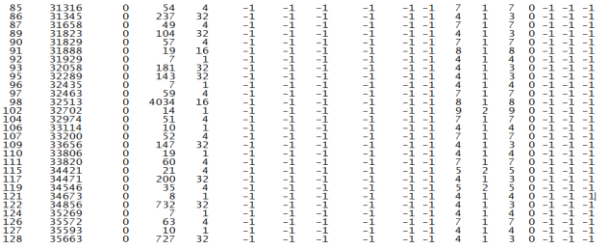

Fig 3: Sample of log file "NASA-iPSC-1993-1.1-cln-6.swf" of the application model.

6.5 Performance Metrics

The metrics applied to measure the performance of the introduced algorithm are schedule length, computation time, used memory, balance degree, and throughput.

Schedule Length (SL): SL is defined as the amount of time, from start to finish execute a set of submitted tasks on the most loaded VM, i.e. the maximum completion time of all submitted tasks. It is considered an important measure of the quality of results obtained with any scheduling algorithm [**40**]. Since the SL including the waiting time and processing time, all techniques used in the scheduling field aim to minimize SL values to reduce the waiting time of the user task.

Computation time: The computation time of the scheduling algorithm is defined as the amount of time it takes to resolve the scheduling problem and obtain the scheduling decision for each task. Or it is defined as the runtime of the scheduler to reach the solution. A high computation time refers to the high complexity of the used technique.

Used memory: The used memory is the amount of memory used by the scheduling algorithm during the searching process for completing the task scheduling. Heavily used memory indicates a waste of system memory and is a weakness in the algorithm.

Throughput: Throughput is the maximum number of completed tasks at one time [**41**]. It measures the scheduling technique efficiency, as high throughput value yields low response time and high execution rate.

(BD) Balance Degree: BD is the degree of balancing the workload on all VMs after applying the scheduling process. The higher BD refers to a more efficient scheduling algorithm and a higher load balancing system. The BD is calculated by Eq. (**10**):

$$BD = SL_{opt}/SL_{fin} \quad (10)$$

Where, SL_{fin} is the final SL after applying the scheduling decision [**42**], SL_{opt} is the optimal schedule length. $SL_{opt} = MI_t/MIPS_t$, where, MI_t is the sum of MI for all submitted tasks and $MIPS_t$ is the sum of all available MIPS.

6.6 Experimental Results

6.6.1 Schedule Length (SL)

Figures 4, 5, 6, and 7 show the SL of scheduling different tasks (100, 250, 500, 1000, 2000) onto different VMs (10, 25, 50, and 75) by using the ACO, GA, WWO, and IWWO. Figures 4.a, 5.a, 6.a, and 7.a show the results of the real data set while the Figures 4.b, 5.b, 6.b, and 7.b show the results of the random data set.

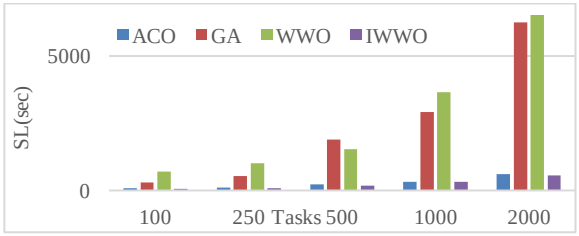

Fig 4.a: SL of scheduling real tasks on 10 VMs

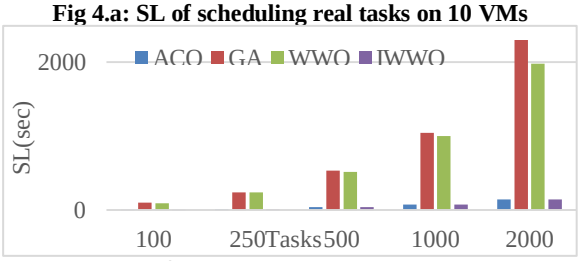

Fig 4.b: SL of scheduling random tasks on 10 VMs

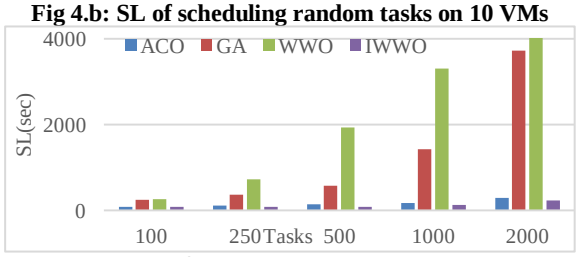

Fig 5.a: SL of scheduling real tasks on 25 VMs

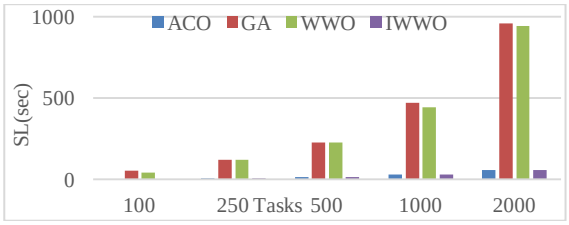

Fig 5.b: SL of scheduling random tasks on 25 VMs

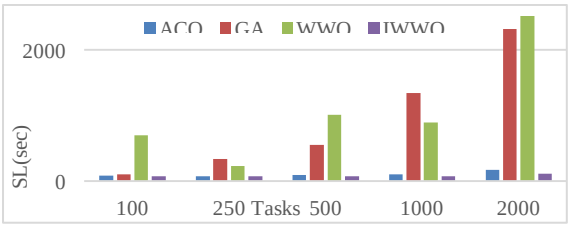

Fig 6.a: SL of scheduling real tasks on 50 VMs

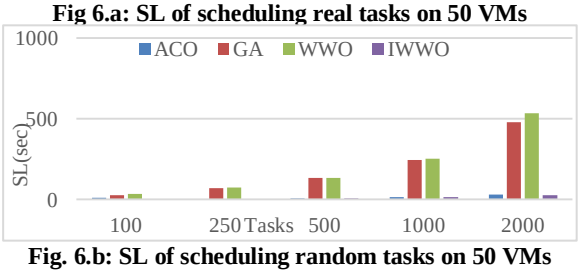

Fig. 6.b: SL of scheduling random tasks on 50 VMs

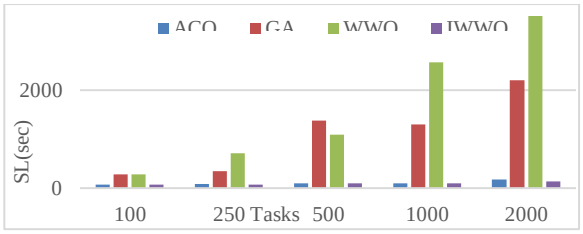

Fig. 7.a: SL of scheduling real tasks on 75 VMs

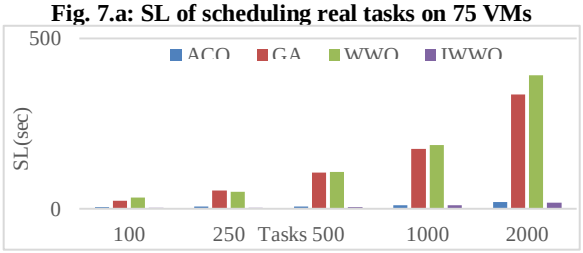

Fig. 7.b: SL of scheduling random tasks on 75 VMs

Form the figures; the proposed IWWO algorithm achieves SL less than that of ACO, GA, and WWO in all cases. The results satisfy the goal of the improvement in the WWO algorithm that leads to overcoming the local optimum point problem of the SL value as declared in figures 4, 5, 6, and 7. Where the SL of WWO is the largest one in most cases and IWWO gives the best SL value but this improvement in SL leads to a problem with computation time as will be discussed in the following section.

6.6.2 Computation time

Table 8 presents the computation time (in seconds) of the WWO, IWWO, ACO, and GA for the real data set. From the table, the results indicate that the WWO requires less time since most of its computational steps depend on the reverse operation of a part of the wave which leads to falling into the local maximum point. While the IWWO algorithm takes more time in the rewarding process which leads to an improvement in the result. As well, the time of running Max-Min algorithm steps as the breaking operation. However, the obtained results achieve a great improvement in the SL value as declared in the previous section and also the throughput and balance degree. In table 9 the computation time of scheduling 100, 250, 500, 1000, and 2000 random tasks onto 10, 25, and 50 VMs.

Table 8: Computation time (sec) of scheduling real tasks

No. of VMs	Algorithm	Number of Tasks				
		100	250	500	1000	2000
10	**IWWO**	**0.031**	**0.063**	**0.172**	**0.577**	**2.199**
	WWO	**0.01**	**0.015**	**0.031**	**0.062**	**0.141**
	GA	0.032	0.063	0.094	0.203	0.353
	ACO	0.405	2.355	9.219	35.740	144.566
25	**IWWO**	**0.016**	**0.109**	**0.203**	**0.655**	**2.324**
	WWO	**0.015**	**0.047**	**0.078**	**0.141**	**0.281**
	GA	0.047	0.109	0.156	0.312	0.624
	ACO	0.206	12.729	50.653	200.055	791.873
50	**IWWO**	**0.047**	**0.124**	**0.296**	**0.827**	**2.574**
	WWO	**0.031**	**0.078**	**0.125**	**0.250**	**0.515**
	GA	0.078	0.156	0.327	0.530	1.139
	ACO	7.941	49.796	191.858	768.410	3048.011
75	**IWWO**	**0.095**	**0.233**	**0.403**	**0.485**	**1.892**
	WWO	**0.030**	**0.078**	**0.188**	**0.359**	**0.743**
	GA	0.093	0.203	0.390	0.780	1.581
	ACO	17.534	108.031	427.161	1722.418	8002.913

Table 9: Computation time (sec) of scheduling random

tasks

No. of VMs	Algorithm	Number of Tasks 100	250	500	1000	2000
10	**IWWO**	**0.025**	**0.063**	**0.16**	**0.69**	**2.29**
	WWO	**0.016**	**0.031**	**0.47**	**0.78**	**0.125**
	GA	**0.016**	**0.062**	**0.078**	**0.125**	**0.218**
	ACO	**0.41**	**2.324**	**9.03**	**37.09**	**142.74**
25	**IWWO**	**0.031**	**0.078**	**0.23**	**0.64**	**2.37**
	WWO	**0.016**	**0.031**	**0.08**	**0.14**	**0.266**
	GA	**0.031**	**0.063**	**0.09**	**0.19**	**0.42**
	ACO	2.08	12.5	49.56	195.765	792.97
50	**IWWO**	**0.047**	**0.125**	**0.28**	**0.81**	**2.79**
	WWO	**0.031**	**0.078**	**0.14**	**0.29**	**0.52**
	GA	0.047	0.078	0.19	0.25	0.66
	ACO	7.97	48.14	191.63	769.57	3049.7

6.6.3 *Balance Degree*

Figures 8, 9, and 10 show the BD of scheduling 100, 250, 500, 1000, and 2000 tasks respectively into different numbers of *VMs*. From the Figs., the presented IWWO scores the highest *BD* ratio for all the examination cases because it gives the minimum makespan in all experiment test (real and random) cases. Results prove that the proposed approach can distribute the different number of user tasks onto the available *VMs* with a higher *BD* ratio that leads to improve resources utilization.

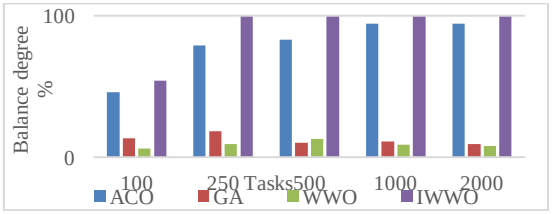

Fig 8.a: BD of different real tasks on 10 VMs

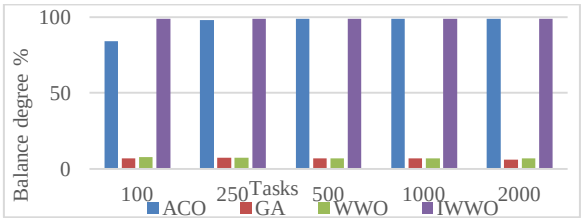

Fig 8.b: BD of different random tasks on 10 VMs

6.6.4 *Used Memory*

Figures 11, 12, 13, and 14 shows the used memory by the WWO, IWWO, ACO, and GA algorithms for scheduling different real tasks (100, 250, 500, 1000, and 2000) onto different VMs. The memory usage metric is measured in CloudSim toolkit where used memory is the subtraction result of total memory before running the algorithm from the total memory after running the algorithm. From figures, the proposed IWWO used more memory than the traditional WWO as a result of applying the learning features that lead to increase the used memory as well the used memory by Max-Min algorithm steps as the breaking operation. This point considers as a weak point in the proposed IWWO but the improvement in other performance metrics is very high compared with the metrics of WWO especially in the SL value and throughput that leads to high improvement in overall system performance.

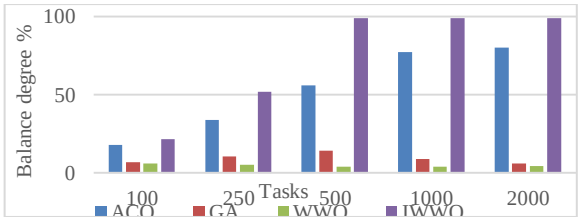

Fig 9.a: BD of different real tasks on 25 VMs

Fig 9.b: BD of different random tasks on 25 VMs

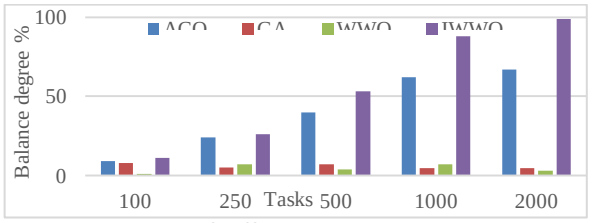

Fig 10.a: BD of different real tasks on 50 VMs

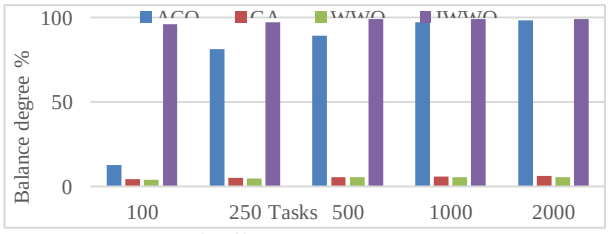

Fig 10.b: BD of different random tasks on 50 VMs

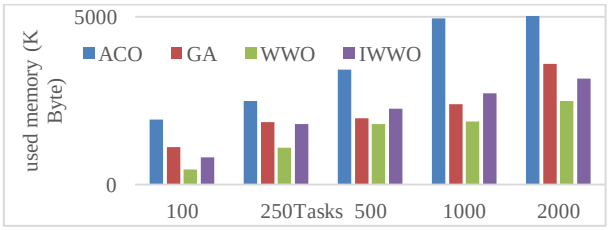

Fig 11: Used memory in the case of using 10 VMs

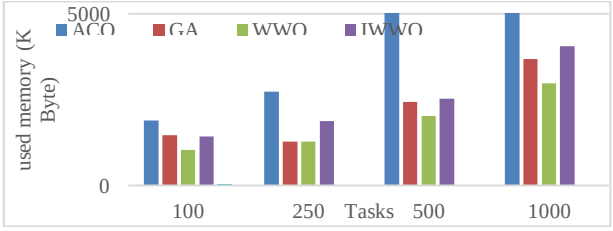

Fig 12: Used memory in the case of using 25 VMs

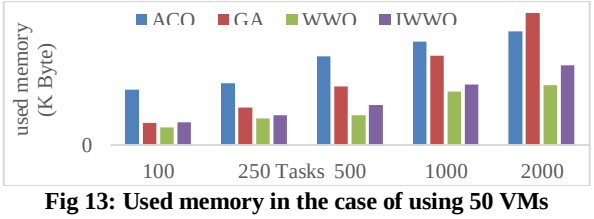

Fig 13: Used memory in the case of using 50 VMs

6.6.5 Throughput

Figures 15, 16, 17, and 18 show the throughput obtained when applying the IWWO, WWO, GA, and ACO to schedule different tasks (100, 250, 500, 1000, and 2000) onto different VMs (10, 25, 50, and 75) considering real time and randomly generated tasks. From the Figs. 15,16,17, and 18, the IWWO technique achieves the higher throughput. The proposed IWWO technique can efficiently explore the available number of VMs and improve the WWO performance. From the figures, the IWWO achieves the maximum throughput and this value increases as the number of tasks and number of VMs increase.

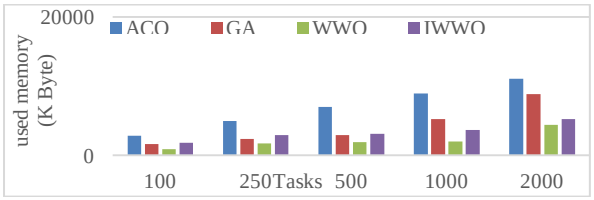

Fig 14: Used memory in the case of using 75 VMs

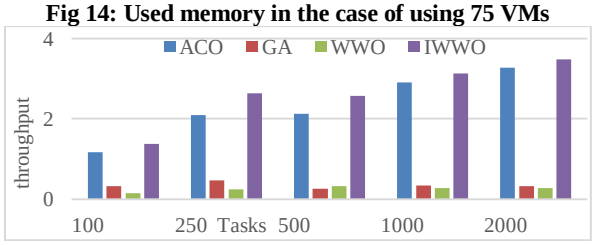

Fig 15.a: Throughput of t real tasks on 10 VMs

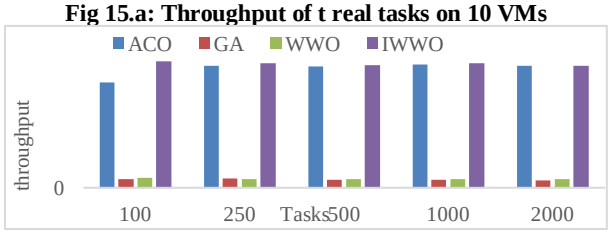

Fig 15.b: Throughput of random tasks on 10 VMs

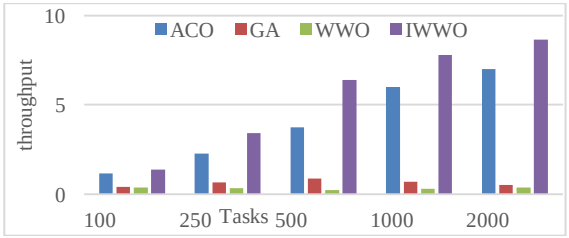

Fig 16.a: Throughput of t real tasks on 25 VMs

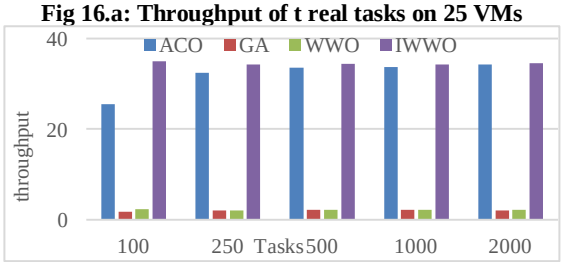

Fig 16.b: Throughput of random tasks on 25 VMs

7. EXPERIMENTAL RESULTS SUMMARY

From the previous results, the IWWO algorithm enhances conventional WWO performance. It achieves better solutions in all cases with a different number of tasks and different number of VMs. On the other hand, the IWWO achieves near-optimal SL, higher balance degree, and high throughput compared with the WWO, GA, and ACO this achieves the main objectives from our research in improving the QoS parameters. However, it has a large memory and high computation time as a result of running the Max-Min algorithm as the breaking operator.

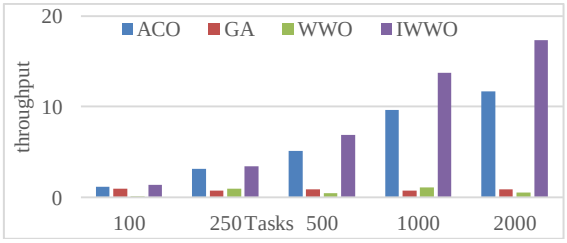

Fig 17.a: Throughput of t real tasks on 50 VMs

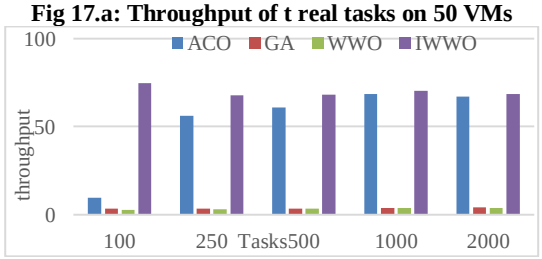

Fig. 17.b: throughput of random tasks on 50 VMs

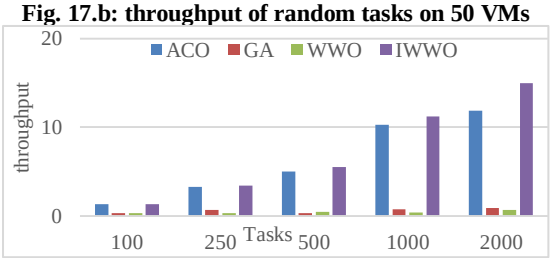

Fig 18.a: Throughput of t real tasks on 75 VMs

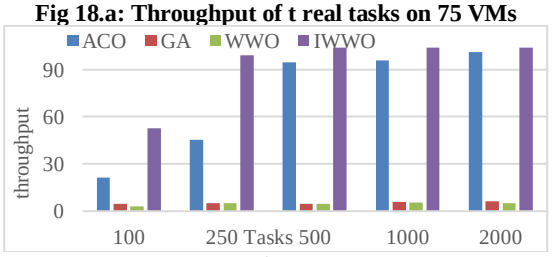

Fig 18.b: Throughput of random tasks on 75 VMs

Moreover, the improvement in the value of SL, BD, and throughput is great with respect to WWO and IWWO. This improvement reaches 90% in SL results for most experimental cases and up to 95% in throughput. While the increment in the computation time does not reach 15% compared with 50% with WWO.

8. CONCLUSION

The scheduling problem has a strong impact on the performance of cloud computing. Therefore, there is a need to apply an efficient scheduling strategy to improve cloud performance. This article developed a new scheduling algorithm, called IWWO for scheduling problem in cloud computing. It is intended to efficiently schedule tasks onto VMs based on applying some features of reinforcement learning to improve the performance of the WWO. The obtained results prove that the presented IWWO can assign many tasks with SL less than that of the WWO as well it gives better results than the WWO algorithm. This because the proposed IWWO first applies some features of RL and then applies a heuristic algorithm Max-Min in the breaking operation to enhance the algorithm local search phase. The proposed IWWO can be improved by dispensing the refraction operator or by applying multithreading programming to minimize the computation time and the used memory.

9. REFERENCES

[1] Zheng, Zibin, Jieming Zhu, and Michael R. Lyu. "Service-generated big data and big data-as-a-service: an overview", the IEEE BigData Congress, 2013.

[2] Alkhanak, E. Nabiel, S. P. Lee, R. Rezaei, and R. M. Parizi, "Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues", Journal of Systems and Software, Vol.113, pp. 1-26, 2016.

[3] L. Mei, W.K. Chan, and T. H. Tse, "A tale of clouds: paradigm comparisons and some thoughts on research issues," Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC'08), pp. 464-469, 2008.

[4] B. Keshanchi and N.J. Navimipour, "Priority-based task scheduling in the cloud systems using a memetic algorithm", Journal of Circuits, Systems and Computers 25(10) (2016), 1–33.

[5] F. Ramezani, J. Lu and F. Hussain, "Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization", Proceeding of the International Conference on Service-Oriented Computing, Springer, 2013, pp. 237–251.T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment." Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 658–664, 2014.

[6] S. K. Panda, I. Gupta, and P. K. Jana, "Allocation-aware task scheduling for heterogeneous multi-cloud systems," Procedia Comput. Sci., vol. 50, pp.176–184, 2015.

[7] Zhang, J., Zhou, Y. and Luo, Q, "Nature-inspired approach: a wind-driven water wave optimization algorithm", Appl. Intell., Vol. 49, pp. 233-252, 2019.

[8] S.K. Mishra, B. Sahoo and P. P. Parida, "Load balancing in cloud computing: A big picture", Journal of King Saud University – Computer and Information Sciences, Vol. 32, pp.149–158, 2018.

[9] I. Strumberger , M. Tuba, N. Bacanin and E. Tuba, "Cloudlet Scheduling by Hybridized Monarch Butterfly Optimization Algorithm", J. Sensor and Actuator Network, Vol. 8: doi:10.3390/jsan8030044 , 2019.

[10] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, and A. El-Sayed, "Using the TSP Solution Strategy for Cloudlet Scheduling in Cloud Computing", Journal of Network and Systems Management, Vol. 27, Issue 2, pp. 366-387, 2019.

[11] F. Mahmoodi and K.Dooley "A comparison of exhaustive and non-exhaustive group scheduling heuristics in a manufacturing cell," Int. J. Prod. Res, Vol. 29, pp. 1923–1939, 1991.

[12] S. Toumi, B. Jarboui, M. Eddaly, and A. Rebaı, "Branch-and-bound algorithm for solving blocking flowshop scheduling problems with makespan criterion," Int. J. Math. Oper. Res, Vol. 10, pp. 34–48, 2017.

[13] B. Pavithra, and R. Ranjana, "A comparative study on performance of energy efficient load balancing techniques in cloud," International Conference in Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 1192–1196, 2016.

[14] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," PLoS One, vol. 12, no. 5, pp. 1–26, 2017, doi: 10.1371/journal.pone.0176321.

[15] G. Ming and H. Li, "An Improved Algorithm Based on Max-Min for Cloud Task Scheduling," Recent Advances in Computer Science and Information Engineering in Springer Berlin Heidelberg, vol. Volume 125 of the series Lecture Notes in Electrical Engineering, pp. 217-223, January 2012.

[16] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications Vol. 50, pp.22-27, July 2012.

[17] H. Gamal El Din Hassan Ali, I. A. Saroit, and A. M. Kotb, "Grouped tasks scheduling algorithm based on QoS in cloud computing network," Egypt. Informatics J., vol. 18, no. 1, pp. 11–19, 2017, doi: 10.1016/j.eij.2016.07.002.

[18] R. Zhang, F. Tian, X. Ren, Y. Chen, K. Chao, R. Zhao, B. Dong, W. Wang, "Associate multi-task scheduling algorithm based on self-adaptive inertia weight particle swarm optimization with disruption operator and chaos operator in cloud environment," Serv. Oriented Comput. Appl. 2018. https://doi.org/10.1007/s11761-018-0231-72018.

[19] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm", Journal of Global Optimization, Vol. 39, pp.459–471, 2007.

[20] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," Egypt. Informatics J., vol. 16, no. 3, pp. 275–295, 2015, doi: 10.1016/j.eij.2015.07.001.

[21] A. F.S. Devaraj, M. Elhoseny, S. Dhanasekaran, E. LaxmiLydia and K.Shankar," Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization algorithm for energy efficient load balancing in Cloud Computing environments," Journal of Parallel and Distributed Computing, Vol. 142, pp. 36-45, 2020.

[22] Keshanchi, Bahman, A. Souri, and N. J. 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, Vol. 124, pp. 1-21, 2017.

[23] S. A. Hamad and F.A. Omara, "Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment," International Journal of Advanced Computer Science and Applications, Vol. 7, pp. 550-556, 2016.

[24] K. L. D. S. Valli, "Multi - objective heuristics algorithm for dynamic resource scheduling in the cloud computing environment," J. Supercomput., no. 0123456789, 2021, doi: 10.1007/s11227-020-03606-2

[25] M.Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E.-G. Talbi, A.Y. Zomaya and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," Journal of

Parallel and Distributed Computing, Vol. 71,pp. 1497-1508, 2011.

[26] M. Dorigo, and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., Vol. 1, pp. 53–66, 1997.

[27] K. Li, G. Xu, G. Zhao, Y. Dong and D. Wang, "Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization," 2011 Sixth Annual Chinagrid Conference, Liaoning, 2011, pp. 3-9, doi: 10.1109/ChinaGrid.2011.17.

[28] K. Sreenu and S. Malempati, "MFGMTS: Epsilon Constraint-Based Modified Fractional Grey Wolf Optimizer for Multi-Objective Task Scheduling in Cloud Computing," IETE J. Res., vol. 65, no. 2, pp. 201–215, 2019, doi: 10.1080/03772063.2017.1409087.

[29] I. Attiya, M. Abd Elaziz, and S. Xiong, "Job Scheduling in Cloud Computing Using a Modified Harris Hawks Optimization and Simulated Annealing Algorithm," Comput. Intell. Neurosci., vol. 2020, 2020, doi: 10.1155/2020/3504642.

[30] F. Hemasian-Etefagh and F. Safi-Esfahani, "Dynamic scheduling applying new population grouping of whales meta-heuristic in cloud computing," J. Supercomput., vol. 75, no. 10, pp. 6386–6450, 2019, doi: 10.1007/s11227-019-02832-7.

[31] Yu-Jun Zheng, "Water wave optimization: A new nature-inspired metaheuristic", Computers &Operations Research, Vol. 55, pp.1–11, 2015.

[32] Jinzhong Zhang, Yongquan Zhou and Qifang Luo, "An improved sine cosine water wave optimization algorithm for global optimization," Journal of Intelligent & Fuzzy Systems, Vol.34, pp. 2129–2141, 2018.

[33] Xiao-Bei Wu, Jie Liao and Zhi-Cheng Wang, "Water wave optimization for the traveling salesman problem" in: D.-S. Huang, V. Bevilacqua, P. Premaratne (Eds.), Intelligent Computing Theories and Methodologies, Springer, Cham, pp. 137–146.

[34] X. Yun, X. Feng, X. Lyu, S. Wang, and B. Liu, "A novel water wave optimization based memetic algorithm for flow-shop scheduling" IEEE Congress on Evolutionary Computation, pp. 1971–1976, 2016.

[35] A. Gosavi, "A tutorial for reinforcement learning," Missouri University of Science and Technology, Tech. Rep., September 2019.

[36] T. Goyal, A. Singh and A. Agrawal, "Cloudsim: simulator for cloud computing infrastructure and modeling," Procedia Engineering, Vol. 38, pp. 3566-3572, 2012.

[37] D. G. Feitelson, D. Tsafrir, D. Krakov, "Experience with using the parallel workloads archive," J. Parallel Distrib. Comput. 74 (10), 2967–2982 (2014).

[38] http://www.cs.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/index.html#usage.

[39] K. Jansen, K.-M. Klein, and J. Verschae, "Closing the gap for makespan scheduling via sparsification techniques," arXiv preprint arXiv:1604.07153 (2016).

[40] A. V. Lakra and D. K. Yadav, "Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization," Procedia Computer Science, vol. 48, pp. 107 – 113, 2015.

[41] A.A. Nasr, A. T. Chronopoulos, N.A. El-Bahnasawy, G. Attiya, and A. El-Sayed, "A Novel Water Pressure Change Optimization Technique for Solving Scheduling Problem in Cloud Computing" Journal of Cluster Computing, Vol. 22, Issue 2, pp. 601-617, 15 June 2019.