Enhanced multi-verse optimizer for task scheduling in cloud computing environments

Sarah E. Shukri, Rizik Al-Sayyed, Amjad Hudaib, Seyedali Mirjalili

PII: S0957-4174(20)30950-7
DOI: https://doi.org/10.1016/j.eswa.2020.114230
Reference: ESWA 114230

To appear in: *Expert Systems With Applications*

Received date : 13 February 2020
Revised date : 2 November 2020
Accepted date : 2 November 2020

Enhanced multi-verse optimizer for task scheduling in cloud computing environments

Sarah E. Shukri[a], Rizik Al-Sayyed[a], Amjad Hudaib[a], Seyedali Mirjalili[b]

[a]*King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan*
sarah.shukri92@gmail.com ,{r.alsayyed,ahudaib}@ju.edu.jo
[b]*Centre for Artificial Intelligence Research and Optimisation, Torrens University Australia, Fortitude Valley, Brisbane, 4006 QLD, Australia*
ali.mirjalili@gmail.com

Abstract

Cloud computing is a trending technology that allows users to use computing resources remotely in a pay-per-use model. One of the main challenges in cloud computing environments is task scheduling, in which tasks should be scheduled efficiently to minimize execution time and cost while maximizing resources' utilization. Many meta-heuristic algorithms are used for task scheduling in cloud environments in the literature such as Multi-Verse Optimizer (MVO) and Particle Swarm Optimization (PSO). In this paper, an Enhanced version of the Multi-Verse Optimizer (EMVO) is proposed as a superior task scheduler in this area. The proposed EMVO is compared with both original MVO and the PSO algorithms in cloud environments. The results show that EMVO substantially outperforms both MVO and PSO algorithms in terms of achieving minimized makespan time and increasing resources' utilization.

Keywords: Cloud computing, task scheduling, Multi-Verse Optimizer, makespan, virtual machines

1. Introduction

A cloud can be viewed as a huge pool of accessible and virtualized resources. These resources can be dynamically assigned to different tasks or jobs, in an aim to achieve optimal resource utilization (Mell et al., 2011). These resources are typically offered for users by a pay per-use model, in which all agreements between end user and the infrastructure provider are managed by customized Service Level Agreements "SLAs" (Vaquero et al., 2008).

Enhanced multi-verse optimizer for task scheduling in cloud computing environments

**[a], **[a], **[a], **[b]

[a]**
,{}@xx.xxx.xx
[b]**
**

Abstract

Cloud computing is a trending technology that allows users to use computing resources remotely in a pay-per-use model. One of the main challenges in cloud computing environments is task scheduling, in which tasks should be scheduled efficiently to minimize execution time and cost while maximizing resources' utilization. Many meta-heuristic algorithms are used for task scheduling in cloud environments in the literature such as Multi-Verse Optimizer (MVO) and Particle Swarm Optimization (PSO). In this paper, an Enhanced version of the Multi-Verse Optimizer (EMVO) is proposed as a superior task scheduler in this area. The proposed EMVO is compared with both original MVO and the PSO algorithms in cloud environments. The results show that EMVO substantially outperforms both MVO and PSO algorithms in terms of achieving minimized makespan time and increasing resources' utilization.

Keywords: Cloud computing, task scheduling, Multi-Verse Optimizer, makespan, virtual machines

1. Introduction

A cloud can be viewed as a huge pool of accessible and virtualized resources. These resources can be dynamically assigned to different tasks or jobs, in an aim to achieve optimal resource utilization (Mell et al., 2011). These resources are typically offered for users by a pay per-use model, in which all agreements between end user and the infrastructure provider are managed by customized Service Level Agreements "SLAs" (Vaquero et al., 2008).

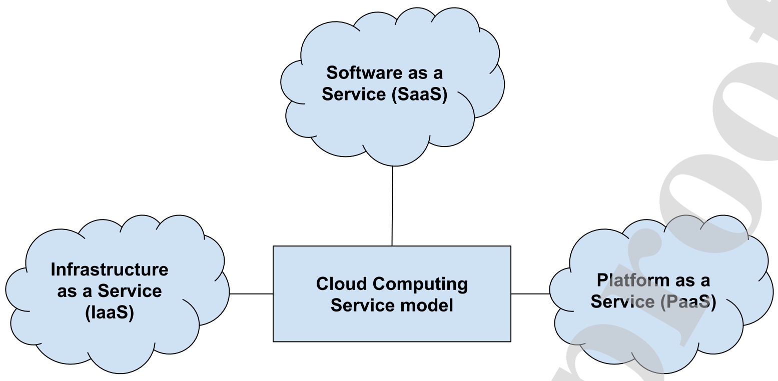

Software as a Service (SaaS)
Infrastructure as a Service (IaaS)
Cloud Computing Service model
Platform as a Service (PaaS)

Figure 1: Service models of cloud computing

In this era of rapid growing technology, new opportunities are open for businesses, where new technologies are replacing old ones. Cloud Computing allows the users to scale their resources quickly (in minutes or even seconds), compared to days or even weeks using traditional systems. This process avoids underutilization of resources when the servers are idle, also avoids over-utilization when all the servers are active and busy, and there are no idle servers (Klems et al., 2008).

Typically, a cloud ecosystem consists of three principal components: Cloud Consumer, Cloud Provider and Cloud Services. Cloud Consumer consumes the cloud services offered by the cloud provider (Mei et al., 2008). Depending on the type of the provided services, the type and capability of the service provider, the cloud services can be provided using different service models: Infrastructure as a Service “IaaS”, Platform as a Service “PaaS” and Software as a Service “SaaS” (See Figure 1) (Subashini and Kavitha, 2011).

Cloud computing environment still faces major challenges. One of these major challenges is the task (or job) scheduling. It refers to the process of assigning tasks to potential resources in order to minimize execution time and cost as possible and to gain better performance (Singh et al., 2014). Resources in the cloud are processors, firewalls, applications or networks and they are dynamically assigned to tasks. The assignment process is dynamic due to several reasons such as: the flow, sequence and nature of tasks are uncertain and the number of available resources is also uncertain.

Scheduling of tasks in the cloud environment is a critical aspect; since inefficient scheduling of tasks might lead to an overall performance degradation of the cloud. Thus; scheduling of tasks in the cloud environment is different than that in a traditional environment. This is because it depends on different parameters, such as: task length, processing power, and computing cost (Kaur and Verma, 2012). Scheduling of tasks is mainly performed by assigning priorities to the available tasks and then assigning these tasks with respect to their priority to the available resources.

In literature, different scheduling algorithms have been proposed. Some of which were built based on traditional task scheduling algorithms such as: Round Robin (RR), First Come First Served (FCFS) or the Shortest Job First (SJF). Other algorithms used the meta-heuristic algorithms to schedule tasks such as: Particles Swarm Optimization (PSO), Genetic Algorithm (GA), and Ant Colony Optimization (ACO).

The importance of adapting new solutions for cloud computing is due to the fact that many enterprises are moving their business and infrastructure to be cloud based, rather than having them on-premises. The importance of cloud-based services and applications is rapidly increasing. Thus, the quality of the provided service needs to be addressed carefully. One of the major aspects of the quality is the response time of requested services. Cloud users expect services to be done as like the service executed locally (Garrison et al., 2012). Thus, implementing an efficient approach for scheduling the tasks would be useful for both cloud user and cloud service provider.

Also, cloud computing services can be deployed in small to large organizations. These organizations exhibit highly competitive advantage and restrict governance regulations. Under such circumstances, cloud computing helps in reducing the operational cost and increases the agility and scalability of the business. Hence, providing them with computing, networking and data storage solutions (Tripathi and Mishra, 2011).

In addition to that, the cloud environment enables the user to utilize a large number of virtual resources for every requested task, which makes the manual and traditional scheduling techniques not an efficient solution that introduce the need to have new efficient scheduling solutions (Arunarani et al., 2019).

In this paper, an enhanced version of the well-known Multi-Verse Optimizer (MVO) (Mirjalili et al., 2016) is proposed. The proposed algorithm will be adapted to solve the task scheduling algorithm for cloud computing environment. The enhancement of the algorithm will be by adding new operation that saves some of the best solutions at each iteration, and after a certain number of iterations, these solutions are fed back to the algorithm as a new solution. The main factors to be assessed of the proposed approach is the length of the task and its required cost and power. And the main objective is to minimize the execution time. In order to enhance the performance of task scheduling problem in the cloud computing. The proposed approach will exploit the local and global search capabilities along with the main operators performed by the MVO to avoid the challenges and drawbacks of traditional task scheduling algorithms.

The rest of the paper is organized as follows: Section 2 lists some of the related work in task scheduling for cloud computing, section 3 describes the preliminaries used in meta-heuristic based task scheduling, section 4 describes the proposed approach, section 5 presents and analyzes the experimental environment and results, and finally the paper is concluded.

2. Related works

Cloud computing is a trending technology that is replacing existing systems nowadays. Many challenges face it, however. One of the main challenges is the scheduling of the submitted tasks into the cloud. Many solutions were proposed in the literature using different on-line mode heuristic scheduling algorithms such as max-min (Bhoi et al., 2013), first come first served (Agarwal et al., 2014); another research was proposed by Verma and Kaushal (2014) that is called "Deadline and Budget Distribution based Cost-Time Optimization Algorithm", the algorithm focused on two main factors: the deadline for executing the task and the budget. The algorithm met the objective by executing the tasks before the deadline ends and this execution has minimized the cost.

Some set of researches applied task scheduling to environments similar to the cloud environment, such as in Dai et al. (2019); where researchers proposed scheduling approach based on improved earliest deadline first algorithm for autonomous driving in Mobile Edge Computing (MEC). The main purpose is to schedule the incoming tasks submitted by autonomous vehicles to the appropriate MEC server. In terms of minimized response time. Finally, the results show that the algorithm can schedule tasks effectively and efficiently. Another research by Lin et al. (2019) applied in the manufacturing sector to support production decisions made in smart factories. The main purpose is to schedule coming tasks with considering the edge computing. They adapted the deep Q network (DQN) to solve the scheduling for complex job shop scheduling problems (JSP) in edge computing, and the results show that the approach can perform better than other algorithms.

Another set of researches utilized the usage of meta-heuristic algorithms for the task scheduling problem; one of the well-regarded algorithms is the Genetic Algorithm (GA) that was introduced by Holland in 1975 (Holland, 1975), GA is population based algorithm and it stimulates the evolution process of nature. A recent application of GA in task scheduling was proposed by Jena and Mohanty (2018), the research proposed a GA-based approach to allocating and scheduling incoming tasks into proper resources, while taking in consideration the requirements of customers and the limitations of the resources. The approach was divided into two main stages. In the first stage, GA-based resource allocation was used. In the second stage, the authors proposed a shortest task first scheduling. The objectives considered were minimizing makespan time and maximizing customer satisfaction. And the results showed that the proposed approach exhibited better than existing scheduling algorithms. Another researchers adapting GA for solving task scheduling problem are proposed in Joseph et al. (2015), Shojafar et al. (2015), Wu et al. (2012), and Wang et al. (2012).

A recent research for automating big data task scheduling in cloud environment was proposed by Rjoub et al. (2019), the main purpose is to help the cloud users deal with their tasks with good performance. They proposed new technique called Deep learning Smart Scheduling (DSS), it combines Deep Reinforcement Learning (DRL) and Long Short-Term Memory (LSTM). The proposed approach aims at scheduling incoming tasks to the suitable virtual resource. In order to improve the overall performance and reduce the resource's cost. Finally, experiments were conducted using real-world datasets from the Google Cloud Platform, and the results showed that the proposed solution minimizes the CPU usage cost and enhanced the RAM memory usage cost when compared to the Shortest Job First (SJF), Round Robin (RR) and improved PSO algorithms. Another similar work was done by Rjoub et al. (2020a), in which a trust-aware scheduling was proposed called BigTrustScheduling. The authors demonstrated that the proposed approach reduced the makespan time and decreased the monetary cost compared to other scheduling algorithms.

Since the task scheduling problem can be considered as an NP-hard problem, and it is a challenging task. Madni et al. (2019) proposed a new hybrid gradient descent cuckoo search (HGDCS) algorithm to optimizer task scheduling problems in Infrastructure as a Service cloud computing service model. The proposed solution was evaluated using several metrics, such as: makespan time, throughput, load balancing and performance improvement rate. And for proofing the efficiency of their proposed solution, HGDCS algorithm was compared to existing meta-heuristic algorithms. Finally, the HGDCS performed well in different cases and on all the datasets, and had better simulation and statistical results than a wide range of optimization algorithms.

As the importance of cloud computing becoming more critical as it is considered as a main computing and storage platform nowadays, task scheduling into appropriate resources is a critical task. To solve such a problem, Rjoub et al. (2020b) proposed a new solution based on using four deep and reinforcement learning-based scheduling approaches in an attempt to optimize the process of task scheduling when dealing with large-scale workloads onto cloud computing resources. The execution time and maximize resource's utilization was consider as the objective in the optimization process. The proposed reinforcement approaches utilised by them are Deep Q networks, reinforcement learning, Recurrent Neural Network Long Short-Term Memory, and Deep Reinforcement learning combined with LSTM.

Another similar work was done by Jacob (2014), in which Bat Algorithm (BA) was used in the task scheduling environment considering the objective of minimizing makespan time. The algorithm was compared to GA, which demonstrated its higher accuracy and better efficiency in terms of minimal makespan time. Researchers in (Raghavan et al., 2015) proposed n new approach based on the Bat algorithm in order to adapt the scheduling problem in cloud computing, aiming to reduce the whole cost. The algorithm was compared to Best Resource selection (BRS) algorithm and shown better results.

3. Preliminaries and essential definitions

3.1. Multi-Verse Optimizer Algorithm

Multi-Verse optimizer algorithm (MVO) is a nature-inspired algorithm proposed by Mirjalili et al. (2016). MVO algorithm is inspired by three concepts of cosmology: White holes, black holes and worm holes, MVO algorithm mathematically modelled on these concepts to find the optimal solution in the huge space of candidate solutions using exploration, exploitation and local search.

MVO is inspired by a recent theory called the multiverse theory (Davies et al., 2003) that emerged after the Big Bang theory. The Big Bang theory states that there was a massive explosion that led to the existence of the universe we are living in. While on the other hand, the Multiverse theory implies that there were several big bangs, and each one led to the existence of a different universe. Thus, Multiverse believes that there are several and other universes than the one we are living in. Moreover, the Multiverse theory believes that these universes can interact and collide with each other and each universe has different properties.

MVO was mainly inspired by three concepts derived from the Multiverse theory: white holes, black holes and wormholes. White holes have never been seen,

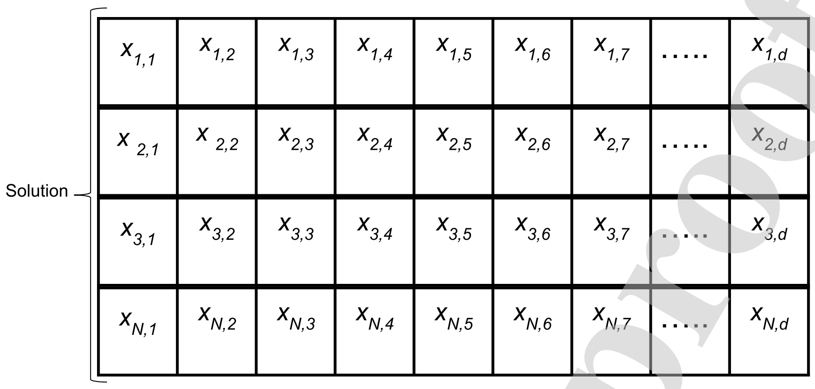


Figure 2: Encoding of solutions - MVO algorithm

but it is argued by physicists that they can be viewed as a big bang or collisions between different parallel universes. Black holes is observed and attract everything toward them due to their high gravitational force. Finally, wormholes connect all the parts of the universe to each other and act as a time/space tunnel to enable objects travel among them. In the original MVO paper, it was mentioned that every universe has an inflation rate that causes the expansion through space.

MVO is a population based algorithm, and it goes through two phases during the search process: exploration and exploitation. In MVO algorithm; solutions are assumed to act as universes and variables in a solution assumed to be like variables in a universe. Figure 2 illustrates an example of the encoding of a universe. Where x represents an object of the i^{th} universe, d represents the number of objects and n represents the number of universes. In addition, each universe is assigned an inflation rate, which refers to the value of the fitness function for that universe.

White/black tunnels are established in MVO algorithm; in order to enable universes to exchange objects between them. Moreover, universes with higher inflation rate have white holes and universes with less inflation rate have black holes. Thus, objects travel through white/black tunnel from the white hole of the source universe to the black hole of the destination universe. This mechanism increases the overall inflation rate average of all universes by moving objects from higher inflation rate universes that acquire more white holes into universes with lower inflation rate.

In order to mathematically model the exchanging mechanism of objects, first of all; the universes are sorted by their inflation rate (fitness value), where the universes with higher inflation rate are sorted on the front. On the other hand, universes with lower inflation rate are sorted at the rear. Then, one of them is

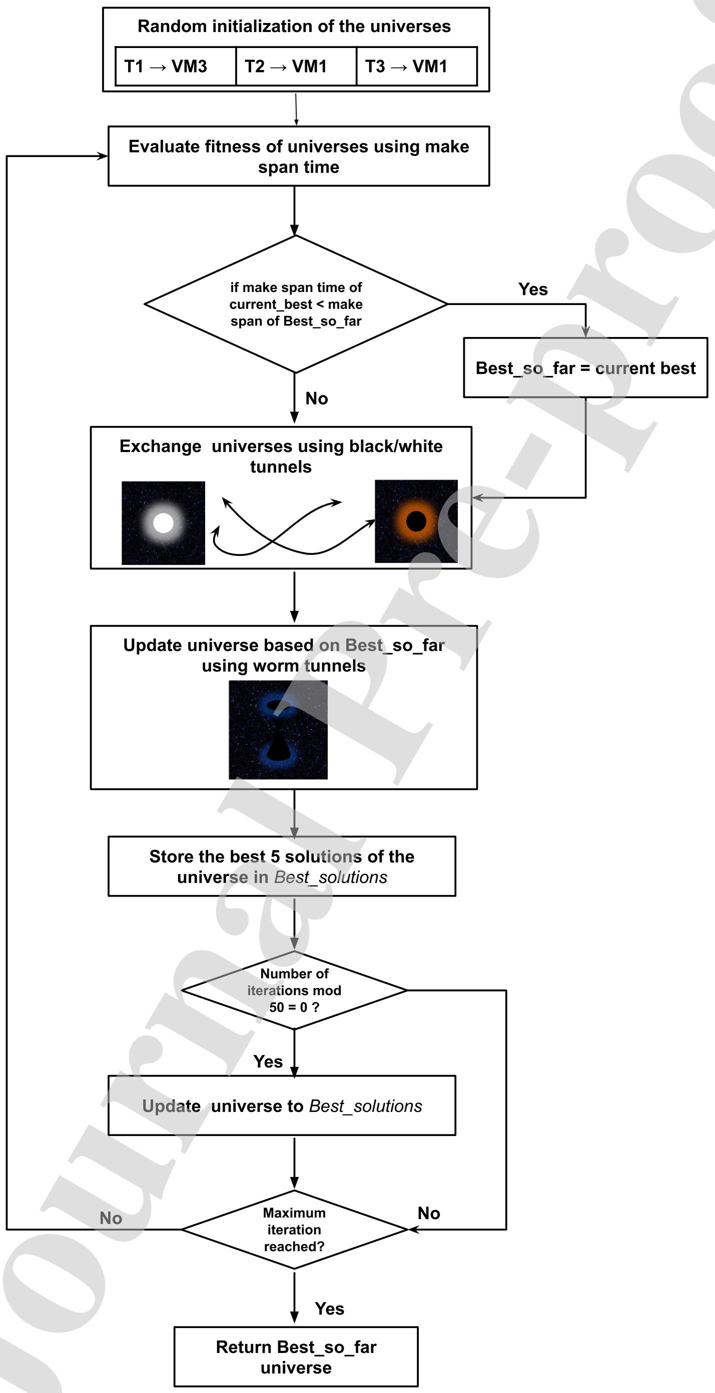

Random initialization of the universes
T1 → VM3
T2 → VM1
T3 → VM1
Evaluate fitness of universes using make span time
if make span time of current_best < make span of Best_so_far
Yes
Best_so_far = current best
No
Exchange universes using black/white tunnels
Update universe based on Best_so_far using worm tunnels
Store the best 5 solutions of the universe in Best_solutions
Number of iterations mod 50 = 0 ?
Yes
Update universe to Best_solutions
No
Maximum iteration reached?
No
Yes
Return Best_so_far universe

Figure 3: Main steps of the Enhanced MVO algorithm

picked to have a white hole using roulette wheel mechanism. By implementing this exchanging mechanism, exploration is held; since universes are required to exchange objects to explore the search space.

On the other hand, exploitation may be performed using the wormholes; each universe is required to transform objects through the search space randomly without relying on the inflation rate. Worm tunnels are established between a universe and the best universe obtained so far. Thus, using this mechanism the diversity of the solution can be guaranteed. And may be able to expand the local search and improve the overall inflation rate of the universe.

4. Task scheduling based on MVO

The main purpose of our proposed algorithm is to map a number of tasks into the available resources in accordance with adaptable time. For time minimization, the MVO algorithm is a suitable approach to the problem. MVO algorithm is a search heuristic algorithm that is inspired by the nature and can be used efficiently for optimization and searching problems. It was applied in different areas such as: static and dynamic clustering (Shukri et al., 2018), feature selection (Faris et al., 2018), estimation of energy consumption (Wang et al., 2018), identifying optimal parameters (Fathy and Rezk, 2018) and other applications. MVO algorithm generates a set of solution agents called universes, these universes are subjected to different operators until the best solution is reached. The output of the task scheduling problem is an array of universes with assignment of tasks into proper VMs. The proposed algorithm will go through the following main steps:

1. Randomly initialize first population by assigning tasks to different VMs. As shown in Figure 3 for example, task 1 is assigned to VM 3, while tasks 2 and 3 are assigned to VM1.
2. Evaluate the fitness of universes by calculating the makespan time.
3. Perform re-production operations using white, black, and warm holes.
4. Repeat process 2 and 3 until end criteria happens.

In order to adapt the MVO algorithm to solve the task scheduling problem in the cloud computing environment, two main criteria must be addressed: encoding of the universes, and the objective or fitness function.

- Encoding of Universes:

  The main objective of the enhanced MVO algorithm is to find the optimal mapping between a set of tasks into a set of resources, the resources are

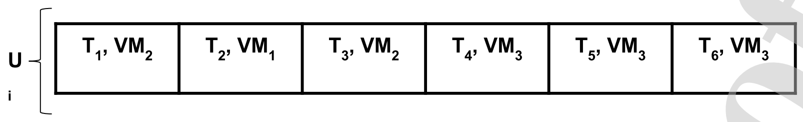


Figure 4: Representation of Universe (Enhanced MVO)

mainly Virtual Machines (VMs) and will be measured by their Millions Instructions Per Second (MIPS), while the tasks will be measured by their length (Number of instructions). The universes will be representing the assignment of tasks to potential VMs, the length of the universe is equal to the number of tasks, and the values will be representing which VM that has been assigned to this task. For example, if we have a scheduling problem where there are 3 VMs and 6 tasks, then the length of the universe will be equal to the number of tasks which is 6, and the values will be either VM1, VM2 or VM3. Figure 4 illustrates an example of the universe encoding for such a problem.

Where tasks 1 and 3 are assigned to VM2, task 2 is assigned to VM1, and tasks3, 4 and 5 are assigned to VM3.

- Objective Function:

  The main objective of the enhanced MVO algorithm is to assign a number of tasks to the potential resources in a way to achieve minimized execution time and increased resource utilization. This process involves finding a suitable sequence in which all tasks are executed in a minimized period of time. In this paper, the nature of the tasks is assumed to be independent, thus there is no communication time between tasks. The only time that will be assessed is the makespan time, which represents the completion and waiting time of all tasks. The formulation of the objective function is described in equation 1.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} E_{ij}X_{ij} \tag{1}$$

  Where E_{ij} represents the execution time of task i on VM_j, and X_{ij} is a Boolean representing if a task i is assigned to VM_j or not.

  The constraints of the objective function are described below:

  - Constraint 1: No two tasks can be assigned to the same VM at the same time

- Constraint 2: Each task can be assigned to only one VM at a time

The computational complexity of the proposed approach depends on different criteria such as: the number of universes, iterations, tasks, universes sorting mechanism and the roulette wheel selection mechanism, as used by the original paper by Sayyedali Mirjalili. The sorting mechanism will be done using the Quick Sort algorithm which has a complexity of $O(nlog(n))$ in the best case and $O(n^2)$ on average and worst case, the sorting of universes is done in every iteration. Roulette wheel selection is done for every variable contained in the universe,and based on our implementation of the roulette wheel selection mechanism, its complexity is $O(log(n))$, thus, the total complexity time of the proposed MVO based task scheduling algorithms is as follows:

$$O(i * n^2 + n * t * log(n)) \tag{2}$$

Where i is the total number of iterations, n is the number of universes, t is the number of tasks to be scheduled

4.1. Enhanced Multi-verse Algorithm

This section presents and discusses the enhanced version of MVO for the task scheduling problem. The aim of the enhanced version is to enhance the results of the original one in general, and for task scheduling in specific.

The enhanced MVO starts by generating a set of random solutions such as the original one. And at each iteration, the same operations performed by original MVO are performed here. In addition to that, at each iteration the best 5 solutions of the universe are selected and stored in a new variable called $Best\_solutions$, and every 10 iterations, the $Best\_solutions$ will have 50 new solutions stored in it, which will be sent to the next iteration as the input universe. The process will continue until the maximum number of iterations are reached. The enhanced MVO algorithm will go through the steps: 1, 2 and 3; as in the original MVO, and steps 4 and 5 are added to the enhanced version as in the following steps:

1. Randomly initialize the first population by assigning tasks to different VMs
2. Evaluate the fitness of the universes by calculating the makespan time
3. Perform re-production operations using white, black, and warm holes
4. Store the best 5 solutions of the universe in $Best\_solutions$
5. When the iteration number is a duplicate of 50 (i.e. its mod 50 = 0), send the $Best\_solutions$ as the input universe at the next iteration
6. Repeat process 2, 3, 4, and 5 until end criteria happens

Table 1: Parameters settings

Algorithm	Parameter	Value
MVO & EMVO	Universe Size	50
	WEP	0.2
	TDR	1
	Number of jobs	100-900
	Number of VMs	50
GA	Population Size	50
	Selection mechanism	Roulette wheel
	Cross over	0.9
	Mutation	0.2
PSO	Swarm size	50
	Acceleration constants	[2.1,2.1]
	Inertia weights	[0.9,0.6]

This modification is expected to enhance the performance of the original MVO algorithm, since it exploits the benefits of the original MVO by using its original operations at each iteration, such as exchanging elements using black, white, and worm holes. On the other hand, it explores a whole new set of solutions each 10 iterations where these solutions are the collection of the best solutions found so far.

5. Implementation and Experiments Results

In this section, the experimental environment and parameters are listed, the dataset description is provided and the implementation results are provided and discussed.

5.1. Experiment Setup

The algorithms were implemented using Matlab version 8.5.0, and all the experiments were run on a PC with the following specifications:

- Windows 10 Professional 64 bit operating system
- Intel(R) Core(TM) i7-8550U CPU
- 8 GB RAM

The parameters of the algorithms are listed in table 1.

The main objective of the proposed algorithm is to find the best sequence of tasks, where all tasks are assigned to the VMs and with minimized completion

Table 2: Number of jobs per datasets

Group	Number of tasks
Regular	100
	200
	300
	400
	500
	600
Big	700
	800
	900
	1000
	1100

time that is referred to as the makespan time. To formulate the problem, we used a number of tasks $(T1, T2, T3, T4, T5.\dots.T_n)$ that are running on a number of VMs $(VM1, VM2, VM3.\dots.VM_m)$, the speed power of VMs are expressed in Million Instructions Per Second (MIPS), and the tasks' lengths are expressed by their number of instructions.

For defining the tasks; the Google Cloud Jobs dataset (GoCJ) was used (Hussain and Aleem, 2018), the main benefit of using this dataset over artificial datasets is that it reflects real workload behavior as perceived in Google cluster traces and MapReduce logs from the M45 super-computing cluster.

5.2. Dataset Description

The GoCJ repository contains a set of datasets[1], each dataset represented as a text files, each of which contains a specific number of tasks, for the purpose of this research, the files are divided into two groups based on the number of tasks in each file, the groups are described in table 2. Each text file consists of a set of rows, where each row has a numeric value representing the size of a job in terms of MI. Each dataset consists of jobs with varying sizes, Figure 5 describes the distribution of the jobs' size in all the datasets.

5.3. Experiments Results and Analysis

This section illustrates and analyzes the experimental results for the enhanced MVO algorithms and compares it with the original MVO, in addition to the PSO algorithm for the task scheduling problem. In the following subsection, the experiment is divided into two main groups: the first one for running the algorithms on

[1]https://data.mendeley.com/datasets/b7bp6xhrcd/1

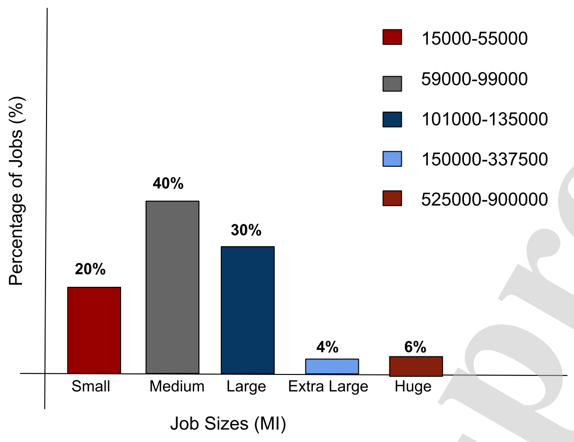


Figure 5: Jobs Sizes

regular-size datasets and the second one for big-size datasets. Each group has two sub-groups, the first one ran using a fixed number of processors for all the datasets and the second one used a variable number of processors for each dataset. In the following subsection, the results of regular-size and big-size datasets are compared and analyzed.

5.3.1. Experiments Results

1. Regular-size datasets: In this group, the results of all the datasets that are grouped as regular-size datasets are listed, the first experiment is run using the same number of VMs for all the datasets. Then the experiment is repeated using variable number of VMs for each dataset, depending on the number of tasks in each dataset. Each group is analyzed using the makespan time, throughput, and resource utilization metrics.

   - Fixed number of processors: This experiment was structured around a different number of tasks distributed among six datasets, each with a different number of tasks ranging from 100 to 600. The objective was to map these jobs to a fixed number of 50 VMs with MIPS ranging from 100 to 4000.

     (a) Makespan time results: The minimal makespan time results indicates better results in terms of better assigning the tasks to VMs with minimal execution and waiting time. Table **??** highlights the average makespan time results for choosing 100 to 600 tasks. It can be observed that for all the datasets, the EMVO outper-

Table 3: Makespan time results for regular-size datasets using fixed number of VMs

Number of Jobs	EMVO	MVO	PSO
100	187.23	187.75	224.10
200	387.64	453.52	439.42
300	542.92	661.05	672.11
400	768.51	782.12	817.67
500	875.41	1085.16	1124.70
600	1099.06	1286.73	1304.06

formed the original MVO and PSO, with an average improvement of EMVO over the original MVO and PSO by 11.39% and 15.23%, respectively. Thus, it can be significantly reported that the EMVO has performed better in scheduling tasks than the original one and PSO algorithm. It also has good capabilities in performing local and global search for finding the best solution. On the other hand, the original MVO and PSO had competitive results where MVO outperformed the PSO in 5 datasets, but with a very small improvement limited to 5.41% only. While the PSO had better results in only one dataset. These results still can't be a satisfying evidence that the MVO performed better task scheduling than PSO in terms of makespan time.

(b) Throughput results: Higher throughput values indicate more efficient scheduling algorithm. In which more tasks are being run per time unit. Table 4 highlights the throughput results for all the datasets over the same number of VMs. It is expected that every algorithm has converged results, since it assigns the tasks for the same number of VMs. For the EMVO, the average throughput value was 54.12% against 46.66% and 45.90% for the original MVO, and PSO, respectively. This indicates that the EMVO had also increased the efficiency of scheduling in terms of number of tasks processed by the VMs. It can also be noticed that the EMVO had the best results of 57% throughput for 500 tasks compared to only 51% and 48% for original MVO, and PSO, respectively. The results can indicate that the EMVO can be efficiently used for task

Table 4: Throughput results for regular-size datasets using fixed number of VMs

Number of Jobs	EMVO	MVO	PSO
100	53.41%	53.26%	44.62%
200	51.59%	44.10%	45.51%
300	55.26%	45.38%	44.64%
400	52.05%	51.14%	48.92%
500	57.12%	46.08%	44.46%
600	54.59%	46.63%	46.01%

scheduling for the regular size of tasks and for a fixed number of VMs.

(c) Resources utilization results: The last factor to be assessed in this experiment is the amount of utilization of used resources. Resources are represented by the used VMs, which are 50 VM for all the datasets. The used VMs have MIPS ranges from 100 to 4000. Table 5 shows the average resources' utilization for all the datasets. EMVO and MVO had competitive results among all the datasets with an average of 97.44% and 97.92%, respectively. The PSO also had good results with an average of 96.16%.
It can be concluded that for this experiment using datasets with regular numbers of tasks and fixed number of VMs, the EMVO had minimal makespan time and increased throughput and resource utilization when compared to both MVO and PSO algorithms. This means that the EMVO had an excellent behavior in task scheduling using its improved local and global search operations.

- Variable number of processors: The second experiment was structured around using the same datasets used in the first experiment, but with different number of VMs for each dataset. Where each dataset had different number of tasks ranging from 100 to 600, the number of VMs for each dataset was 10% of its number of tasks. Table 6 lists the number of VMs for each dataset. The aim of this experiment was to test the efficiency of changing the number of VMs and its effect on the assessed factors. (i.e. makespan, throughput and resource utilization)

Table 5: Resources Utilization results for regular-size datasets using fixed number of VMs

Number of Jobs	EMVO	MVO	PSO
100	80.80%	81.60%	78.00%
200	95.60%	97.60%	92.80%
300	98.00%	98.00%	96.40%
400	97.60%	98.00%	96.80%
500	98.00%	98.00%	96.80%
600	98.00%	98.00%	98.00%

Table 6: Number of VMs per regular-size datasets

Number of Tasks	Number of VMs
100	10
200	20
300	30
400	40
500	50
600	60

Table 7: Makespan time results for regular-size datasets using variable number of VMs

Number of Jobs	EMVO	MVO	PSO
100	551.05	608.52	641.68
200	653.90	789.24	791.47
300	845.94	853.48	923.54
400	828.51	908.50	978.65
500	826.17	1098.35	1086.10
600	882.40	1169.14	1250.88

(a) Makespan time results: As listed in table 7, the makespan time results for the datasets are shown for EMVO, MVO and PSO algorithms. First, it can be viewed that the makespan time results are higher than the results of the first experiment for the first four datasets. Which can be justified since the number of VMs had been reduced in this experiment for these datasets. While the last two datasets had equal and higher number of VMs and performed better in terms of minimized makespan time. Thus, the first conclusion is that the increased number of VMs can reduce the makespan time. For conducting the efficiency of the proposed algorithm, it can be seen that it has outperformed both algorithms for all the datasets. The improvement rates were 14.26% and 18.11% over MVO and PSO, respectively. These results indicate that the EMVO enhanced the performance and wasn't affected negatively by using variable number of VMs. While the MVO outperformed the PSO in five datasets with an enhanced rate of only 5.35%.

(b) Throughput results: The second factor of assessment in this experiment is the amount of tasks being processed by VMs at time unit. This is referred to as throughput, where higher throughput values indicate better results. The results are shown in table 8. Compared to the first experiment where a fixed number of VMs is used (table 4). This experiment shows worse throughput values for the first four datasets, which is justified since the less number of VMs has been used. But for the last two datasets,

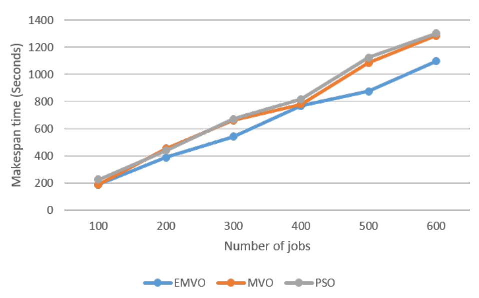
Table 8: Throughput results for regular-size datasets using variable number of VMs

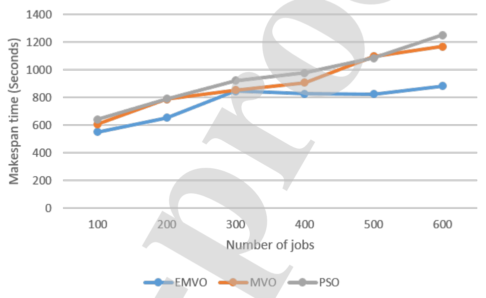
Number of Jobs	EMVO	MVO	PSO
100	18.15%	16.43%	15.58%
200	30.59%	25.34%	25.27%
300	35.46%	35.15%	32.48%
400	48.28%	44.03%	40.87%
500	60.52%	45.52%	46.04%
600	68.00%	51.32%	47.97%

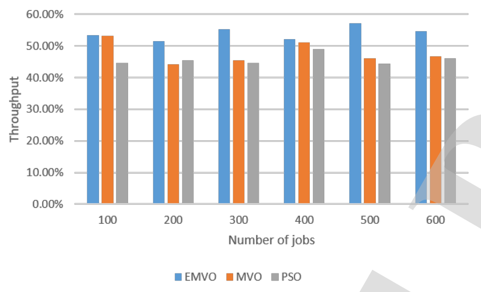
higher number of VMs produced higher throughput values, also the throughput values of the proposed EMVO were higher than both MVO and PSO algorithms. Thus, EMVO also performed better for variable number of VMs than both MVO and PSO, with decreased makespan time and increased throughput values.

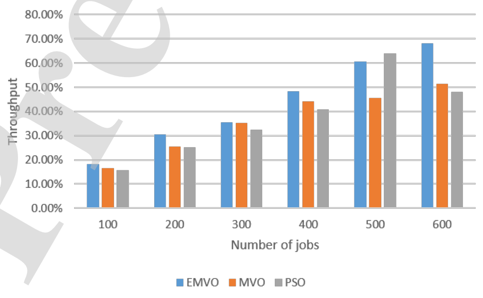
(c) Resource utilization results: For this experiment, the amount of Resources utilization is expected to be increased since the number of used resources is decreased. Thus, the exploited resources are expected to be higher; but it is not necessarily that this must be the case. Table 9 lists the resources utilization rates for the three algorithms, higher number of resources implied more utilization, also the results are very competitive for all the algorithms. It can be concluded that in addition to efficient makespan and throughput values; EMVO had efficiently exploited available resource with an average of 97.10% utilization. As a conclusion of this experiment, we might conclude that EMVO can be efficiently used for task scheduling on regular size datasets with a fixed or variable number of VMs. Figure 6 displays the overall results for all the evaluation criteria of the regular size datasets.

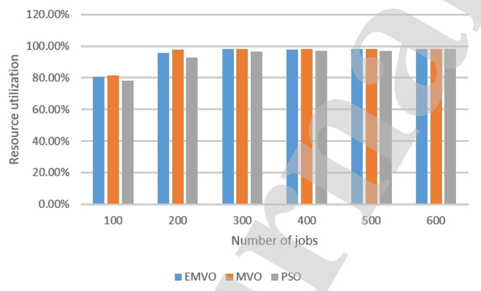
2. Big-size datasets: This group focused on datasets with a relatively big number of tasks, compared to the first group. The number of tasks in this group ranged from 700 to 1000 tasks. The experiment is split into two parts, the first part will be by assigning these tasks to a fixed number of 100 VMs in order to test the efficiency of the proposed algorithm's overall. While the


(a) Maekspan time: Fixed VMs


(b) Maekspan time: Variable VMs


(c) Throughput: Fixed VMs


(d) Throughput: Variable VMs


(e) Resources utilization: Fixed VMs

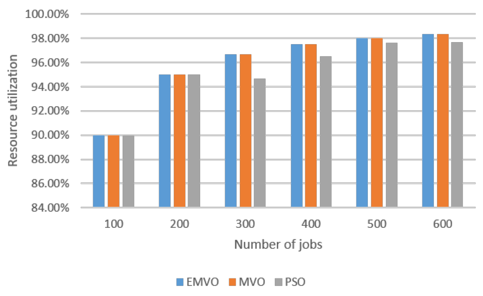

(f) Resources utilization: Variable VMs

Figure 6: Makespan time, throughput and resources utilization for fixed and variable number of VMs of regular size datasets

Table 9: Resource utilization results for regular-size datasets using variable number of VMs

Number of Jobs	EMVO	MVO	PSO
100	90.00%	90.00%	90.00%
200	95.00%	95.00%	95.00%
300	96.66%	96.66%	94.67%
400	97.50%	97.50%	96.50%
500	98.00%	98.00%	97.60%
600	98.33%	98.33%	97.67%

second part will assign the tasks to a variable number of VMs to test the effect of using different number of VMs for each dataset.

- Fixed Number of processors: In this experiment all the datasets are assigned to a fixed number of 100 VMs, with MIPS ranging from 100 to 4000, the makespan time, throughput and resource utilization are analyzed in order to test the efficiency of the proposed EMVO. Also, compared to the original MVO and PSO algorithms.

  1. Makespan time results: This experiment aimed at minimizing the makespan time results, since minimized execution and waiting time imply better scheduling algorithm. Table 11 lists the makespan time results for the three algorithms on the big-size datasets. Clearly (as shown in the table), the makespan time was increased by the increment of the number of tasks. It can also be seen that the EMVO outperformed the MVO and PSO algorithms in all the datasets for a fixed number of VMs. With an average enhancement of 11.23% and 20.64% over MVO and PSO, respectively. Thus, the proposed algorithm performed better than both MVO and PSO.
  2. Throughput results: For this experiment, throughput results indicate the efficiency of the algorithm in executing the tasks proportional to the number of used VMs. Since the number of VMs is fixed for all the datasets, the throughput values are expected not to vary a lot, EMVO exhibited excellent performance with throughout, ranging from 85% to 93%, event though both MVO and PSO had good results. These results indicate that the EMVO performed the best in local and global

Table 10: Makespan time results for big-size datasets using fixed number of VMs

Number of Jobs	EMVO	MVO	PSO
700	823.73	908.26	975.17
800	933.17	1038.64	1153.63
900	969.77	1158.42	1369.82
1000	1158.77	1275.73	1425.85

Table 11: Throughput results for big-size datasets using fixed number of VMs

Number of Jobs	EMVO	MVO	PSO
700	0.85	0.77	0.72
800	0.86	0.77	0.69
900	0.93	0.78	0.66
1000	0.86	0.78	0.70

search for finding the best solution with regard to minimized makespan time and maximized throughput results.

3. Resources utilization: The last factor to be assessed in this experiment is resources' utilization. It refers to the percentage of resources that have been used through the tasks' scheduling process. Higher resources utilization indicates better and more efficient scheduling, since the aim of the scheduling process is to use the resources in an efficient way. Table 12 lists the results. As can be seen from the table, EMVO outperformed both MVO and PSO with a limited enhancement rate, where all algorithms had competitive results. It can all also be concluded that these results are significantly higher than the results of regular-size datasets, where a maximum of 60 VMs were used in these experiments.

- Variable number of processors: The forth experiment was structured around using datasets with relatively big datasets. The aim was to map them for different number of VMs for each dataset. The datasets had different number of tasks ranging from 700 to 1000, the amount of VMs for each dataset was

Table 12: Resources utilization results for big-size datasets using fixed number of VMs

Number of Jobs	EMVO	MVO	PSO
700	99.00%	98.60%	98.40%
800	99.00%	99.00%	98.60%
900	98.80%	99.00%	98.00%
1000	99.00%	99.00%	98.60%

Table 13: Number of VMs per big-size datasets

Number of Tasks	Number of VMs
700	70
800	80
900	90
1000	100

10% of its number of tasks. Table 13 lists the number of VMs for each dataset. The aim of this experiment was to test the efficiency of changing the number of VMs and its effect on the assessed factors.

1. Makespan time results: The aim of this experiment was to map the tasks to variable VMs with minimal makespan time. Table 13 shows that the proposed algorithms had outperformed both algorithms. For the first three datasets with a high improvements rate that reached 13.23% and 19.63% over MVO and PSO, respectively. For the last dataset, both EMVO and MVO had competitive results while MVO outperformed with a limited negligible value. Thus, it can be concluded that EMVO wasn't affected when variable number of VMs was used, and it had significantly outperformed both MVO and PSO algorithms for the majority of datasets.
2. Throughput results: The second factor of assessment is the throughput value; higher throughput indicates better efficiency, the goal was to map the tasks into VMs with higher throughput values. The EMVO exhibited great performance with high throughput values with an average of 87.45%, while the MVO and the PSO had 77.54% and 69.24%,

Table 14: Makespan time results for big-size datasets using variable number of VMs

Number of Jobs	EMVO	MVO	PSO
700	1003.89	1279.65	1360.21
800	1042.09	1182.01	1420.56
900	1221.39	1303.51	1464.19
1000	1258.43	1242.11	1384.15

Table 15: Throughput results for big-size datasets using variable number of VMs

Number of Jobs	EMVO	MVO	PSO
700	84.98%	77.07%	71.78%
800	85.73%	77.02%	69.35%
900	92.81%	77.69%	65.70%
1000	86.30%	78.39%	70.13%

respectively.

3. Resources utilization results: As can be seen from table 16, all the algorithms produced almost similar results. This implies that when the number of jobs is large, the three algorithms behave almost the same with slight negligible differences. It can be also noticed that the results are very high and indicates an excellent utilization of the available results.

As a summary, the results show that for different sizes of datasets, we can see that the EMVO outperformed both the original MVO and the PSO (i.e. for big and regular dataset sizes). The EMVO out-performance was in most cases in terms of makespan time and throughout. In addition, the EMVO exhibited better performance in terms of higher resource utilization. Figure 7 displays the overall results of the big size datasets.

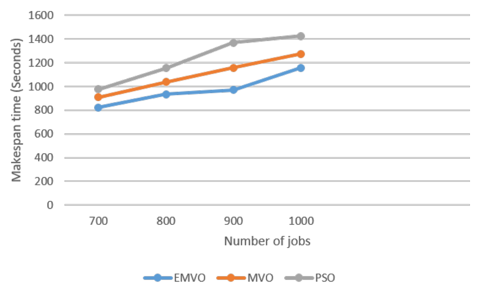
6. Conclusion

This paper proposed an Enhanced Multi-Verse Optimizer (EMVO) algorithm. The EMVO was applied to enhance task scheduling in a cloud environment. To

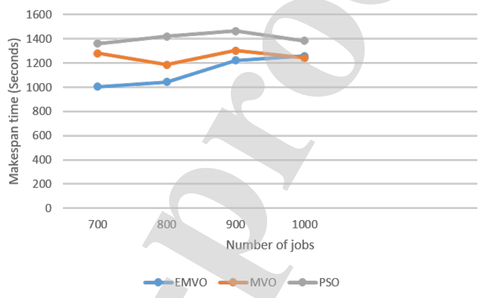
Table 16: Resources utilization results for big-size datasets using variable number of VMs

Number of Jobs	EMVO	MVO	PSO
700	98.57%	98.57%	97.43%
800	98.75%	98.75%	98.75%
900	98.89%	98.89%	98.44%
1000	99.00%	99.00%	99.00%

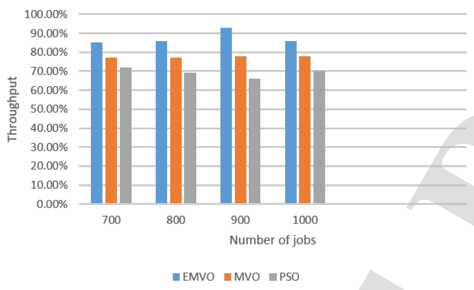
minimize makespan time, maximize throughput, and increase resource utilization, the EMVO was then used to map a number of tasks into a desired number of VMs. The EMVO was used to schedule tasks and address throughout and resource utilization. The EMVO performance was compared with both the original MVO and the PSO algorithms. Results show that the EMVO mapped tasks well with no extra overheads on throughput. It also outperformed the MVO and the PSO by achieving higher resource utilization.



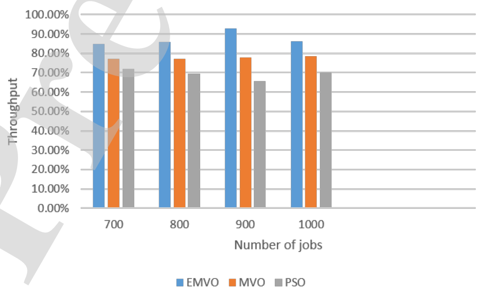
(a) Maekspan time: Fixed VMs



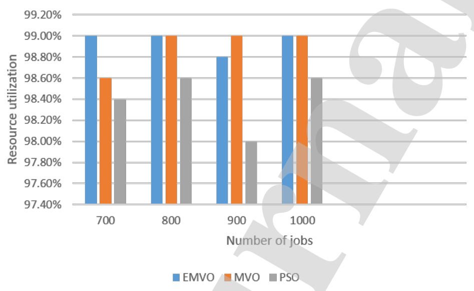
(b) Maekspan time: Variable VMs



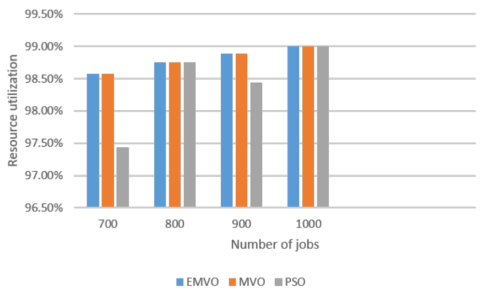
(c) Throughput: Fixed VMs



(d) Throughput: Variable VMs



(e) Resources utilization: Fixed VMs



(f) Resources utilization: Variable VMs

Figure 7: Makespan time, throughput and resources utilization for fixed and variable number of VMs of big size datasets

References

Agarwal, D., Jain, S., et al., 2014. Efficient optimal algorithm of task scheduling in cloud computing environment. arXiv preprint arXiv:1404.2076 .

Arunarani, A., Manjula, D., Sugumaran, V., 2019. Task scheduling techniques in cloud computing: A literature survey. Future Generation Computer Systems 91, 407–415.

Bhoi, U., Ramanuj, P.N., et al., 2013. Enhanced max-min task scheduling algorithm in cloud computing. International Journal of Application or Innovation in Engineering and Management (IJAIEM) 2, 259–264.

Dai, H., Zeng, X., Yu, Z., Wang, T., 2019. A scheduling algorithm for autonomous driving tasks on mobile edge computing servers. Journal of Systems Architecture 94, 14–23.

Davies, P., et al., 2003. A brief history of the multiverse .

Faris, H., Hassonah, M.A., Ala'M, A.Z., Mirjalili, S., Aljarah, I., 2018. A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture. Neural Computing and Applications 30, 2355–2369.

Fathy, A., Rezk, H., 2018. Multi-verse optimizer for identifying the optimal parameters of pemfc model. Energy 143, 634–644.

Garrison, G., Kim, S., Wakefield, R.L., 2012. Success factors for deploying cloud computing. Communications of the ACM 55, 62–68.

Holland, J.H., 1975. Adaptation in natural and artificial systems ann arbor. The University of Michigan Press 1, 975.

Hussain, A., Aleem, M., 2018. Gocj: Google cloud jobs dataset for distributed and cloud computing infrastructures. Data 3, 38.

Jacob, L., 2014. Bat algorithm for resource scheduling in cloud computing. Int J Res Appl Sci Eng Technol 2, 53–57.

Jena, T., Mohanty, J., 2018. Ga-based customer-conscious resource allocation and task scheduling in multi-cloud computing. Arabian Journal for Science and Engineering 43, 4115–4130.

Joseph, C.T., Chandrasekaran, K., Cyriac, R., 2015. A novel family genetic approach for virtual machine allocation. Procedia Computer Science 46, 558–565.

Kaur, S., Verma, A., 2012. An efficient approach to genetic algorithm for task scheduling in cloud computing environment. International Journal of Information Technology and Computer Science (IJITCS) 4, 74.

Klems, M., Nimis, J., Tai, S., 2008. Do clouds compute? a framework for estimating the value of cloud computing, in: Workshop on E-Business, Springer. pp. 110–123.

Lin, C.C., Deng, D.J., Chih, Y.L., Chiu, H.T., 2019. Smart manufacturing scheduling with edge computing using multiclass deep q network. IEEE Transactions on Industrial Informatics 15, 4276–4284.

Madni, S.H.H., Abd Latiff, M.S., Ali, J., et al., 2019. Hybrid gradient descent cuckoo search (hgdcs) algorithm for resource scheduling in iaas cloud computing environment. Cluster Computing 22, 301–334.

Mei, L., Chan, W.K., Tse, T., 2008. A tale of clouds: Paradigm comparisons and some thoughts on research issues, in: 2008 IEEE Asia-Pacific Services Computing Conference, Ieee. pp. 464–469.

Mell, P., Grance, T., et al., 2011. The nist definition of cloud computing .

Mirjalili, S., Mirjalili, S.M., Hatamlou, A., 2016. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Computing and Applications 27, 495–513.

Raghavan, S., Sarwesh, P., Marimuthu, C., Chandrasekaran, K., 2015. Bat algorithm for scheduling workflow applications in cloud, in: 2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV), IEEE. pp. 139–144.

Rjoub, G., Bentahar, J., Wahab, O.A., 2020a. Bigtrustscheduling: Trust-aware big data task scheduling approach in cloud computing environments. Future Generation Computer Systems 110, 1079–1097.

Rjoub, G., Bentahar, J., Wahab, O.A., Bataineh, A., 2019. Deep smart scheduling: A deep learning approach for automated big data scheduling over the cloud, in: 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE. pp. 189–196.

Rjoub, G., Bentahar, J., Wahab, O.A., Bataineh, A.S., 2020b. Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems .

Shojafar, M., Javanmardi, S., Abolfazli, S., Cordeschi, N., 2015. Fuge: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Computing 18, 829–844.

Shukri, S., Faris, H., Aljarah, I., Mirjalili, S., Abraham, A., 2018. Evolutionary static and dynamic clustering algorithms based on multi-verse optimizer. Engineering Applications of Artificial Intelligence 72, 54–66.

Singh, R.M., Paul, S., Kumar, A., 2014. Task scheduling in cloud computing. International Journal of Computer Scienceand Information Technologies (IJCSIT) 5, 7940–7944.

Subashini, S., Kavitha, V., 2011. A survey on security issues in service delivery models of cloud computing. Journal of network and computer applications 34, 1–11.

Tripathi, A., Mishra, A., 2011. Cloud computing security considerations, in: 2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), IEEE. pp. 1–5.

Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M., 2008. A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review 39, 50–55.

Verma, A., Kaushal, S., 2014. Deadline constraint heuristic-based genetic algorithm for workflow scheduling in cloud. International Journal of Grid and Utility Computing 5, 96–106.

Wang, X., Luo, D., Zhao, X., Sun, Z., 2018. Estimates of energy consumption in china using a self-adaptive multi-verse optimizer-based support vector machine with rolling cross-validation. Energy 152, 539–548.

Wang, X., Wang, Y., Zhu, H., 2012. Energy-efficient multi-job scheduling model for cloud computing and its genetic algorithm. Mathematical Problems in Engineering 2012.

Wu, G., Tang, M., Tian, Y.C., Li, W., 2012. Energy-efficient virtual machine placement in data centers by genetic algorithm, in: International conference on neural information processing, Springer. pp. 315–323.

Enhanced multi-verse optimizer for task scheduling in cloud computing environment

Highlights

- An improved algorithm for task scheduling in cloud computing environment.
- An Enhanced version of the Multi-Verse Optimizer (EMVO) algorithm is proposed.
- The enhanced algorithm produced good results when compared with other algorithms.
- The enhanced algorithm reduced makespan time and increased resources' utilization.

CRediT author statement

Sarah E. Shukri:, Methodology, Software implementation, Writing- Original draft preparation, validation.

Rizik Al-Sayyed: Conceptualization, Supervision, Reviewing and Editing, Visualization, Validation.

Amjad Hudaib: Conceptualization, Supervision, Reviewing writing, Visualization, Validation.

Seyedali Mirjalili: Conceptualization, Original idea validation, Investigation.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: