The Journal of Supercomputing (2022) 78:740–765
https://doi.org/10.1007/s11227-021-03915-0

Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing

Laith Abualigah[1,2] · **Muhammad Alkhrabsheh**[1]

Accepted: 24 May 2021 / Published online: 31 May 2021

Abstract

The central cloud facilities based on virtual machines offer many benefits to reduce the scheduling costs and improve service availability and accessibility. The approach of cloud computing is practical due to the combination of security features and online services. In the tasks transfer, the source and target domains have differing feature spaces. This challenge becomes more complicated in network traffic, which leads to data transfer delay, and some critical tasks could not deliver at the right time. This paper proposes an efficient optimization method for task scheduling based on a hybrid multi-verse optimizer with a genetic algorithm called MVO-GA. The proposed MVO-GA is proposed to enhance the performance of tasks transfer via the cloud network based on cloud resources' workload. It is necessary to provide adequate transfer decisions to reschedule the transfer tasks based on the gathered tasks' efficiency weight in the cloud. The proposed method (MVO-GA) works on multiple properties of cloud resources: speed, capacity, task size, number of tasks, number of virtual machines, and throughput. The proposed method successfully optimizes the task scheduling of a large number of tasks (i.e., 1000–2000). The proposed MVO-GA got promising results in optimizing the large cloud tasks' transfer time, which reflects its effectiveness. The proposed method is evaluated based on using the simulation environment of the cloud using MATLAB distrusted system.

Keywords Cloud computing · Task scheduling · Multi-verse optimizer · Genetic algorithm · Hybrid method

✉ Laith Abualigah
Aligah.2020@gmail.com

Muhammad Alkhrabsheh
muhammad@gmail.com

[1] Faculty of Computer Sciences and Informatics, Amman Arab University, Amman 11953, Jordan

[2] School of Computer Sciences, Universiti Sains Malaysia, 11800 Gelugor, Pulau Pinang, Malaysia

1 Introduction

Cloud computing is defined as managing information and services of users and organizations using central technology such as storage, CPU, and network [1, 2]. Cloud computing aims to centralize the technology facilities to provide many benefits such as avoiding data redundancy, reducing physical costs, speeding up the services, and information flow. Technically, the cloud architecture is divided into three levels of services [3]; (1) the software as a service (SaaS), which represents the user services through system interfaces, (2) platform as a service (PaaS), which represents the operating system of the cloud, and (3) infrastructure as a service (IaaS), which contains the hardware facilities such as storages and network. The PaaS operates the data gathering between SaaS and IaaS through the network facilities [4].

The cloud takes one of three main deployment approaches; the first approach is the private cloud applied in the local environment like organizations or cities [5, 6]. The second approach is the public cloud that allows cloud deployment as a global network (i.e., cross the countries). The hybrid deployment approach mixes between the private and public approaches. The public cloud connected with the private cloud before delivering the service to the users through the private cloud's SaaS. Despite the benefits of the hybrid cloud, one of the main challenges of this approach is scheduling tasks [7]. Task transfer scheduling is difficult due to the differences in clouds' facilities specifications, such as storage capacity, transfer rate, throughput, and processors speed [8]. These differences may cause transfer workload, delivery time, and efficiency of resource usability [9, 10].

The cloud environment offers many benefits for businesses by centralizing technology facilities' operations (i.e., software and hardware). The facility's centralization could reduce the physical costs, speed up the services, avoid data redundancy, and improve the data's relationship schemes [11, 12]. The cloud deploys in wide applications to structure the data gathering between clouds. Organizations such as universities, banks, and hospitals may have their cloud that operated locally. However, these organizations book facilities of the public cloud to cut the operational costs of the cloud. It is hard to schedule the tasks to address the minimum makespan. A useful scheduler must apply beneficial strategies to handle the changing environment and the types of tasks. One of the primary essential issues of the tasks' transfer is task scheduling efficiency [13]. The scheduling algorithm should distribute the tasks based on the available cloud resources to minimize makespan without violating precedence constraints. In the cloud, efficient task scheduling would utilize all available resources to improve the transfer system performance. This increases the scheduling decisions' complexity to assure the efficiency of task transfer [12, 13]. Thus, the scheduling decision of task transfer is complicated due to the difficulty of addressing transfer efficiency. Hence, this paper proposes an optimization algorithm to improve task transfer performance based on the cloud environment. The task transfers' performance needs to cover the efficiency properties to provide effective scheduler decisions in the tasks' queue under the dynamic changing of the active tasks in the cloud.

Several kinds of research have been conducted to solve the transfer workload of heterogonous cloud tasks. The transfer efficiency is the main issue of scheduling the tasks. The efficiency focuses on transfer the tasks in response time based on the available cloud resources. Thus, faster tasks would transfer in first under the condition of full or effective cloud deployment. There are many properties included in the computations of transfer efficiency to schedule the transfer priority based on the delivery time. The nature-inspired algorithms, such as Dragonfly Algorithm (DA) [14], Gray Wolf Optimizer (GWO) [15], Marine Predators Optimizer (MPO) [16, 17], Salp Swarm Algorithm (SSA) [18], Grasshopper Optimization Algorithm (GOA) [19], Harmony Search (HS) [20], Sine Cosine Optimizer (SCO) [21], Particle Swarm Optimization (PSO), Firefly Algorithm (FA) [22], Krill herd Algorithm (KHA) [23, 24], Moth–flame Optimization (MOA) [25], Gradient-based Optimizer (GBO) [26], Group Search Optimizer (GSO) [27], Bat Algorithm (BA) [28], Aquila Optimizer [29], and Arithmetic Optimization Algorithm (AOA) [30], can be applied to solve the task schedule of heterogonous tasks transfer.

This paper aims to improve the nature-inspired algorithm (multi-verse optimizer) by multi-objective properties to improve the effectiveness of the transfer scheduler of heterogonous cloud tasks. The proposed method hybridized the multi-verse optimizer with genetic algorithm (GA) to enhance its searchability and avoid the trap in the optimum local problem. The main motivations are in twofold: firstly, the heterogeneous tasks' transfer cost could be reduced through assign tasks for all available cloud resources. The efficiency of the task transfer would be computed based on the transfer time of the tasks and depend on many properties such as throughput, the capacity of cloud machines, transfer rate, and processor speed. Secondly, the integration between the MVO and GA would provide effective processes to schedule cloud tasks transfer by optimizing the transfer time. The MVO-GA could schedule the transfer tasks based on the workload of the available cloud resources. On the other hand, the GA could improve the conventional MVO by applying the crossover and mutation processes to enhance the initiated tasks schedule by MVO. Experiments are conducted using two task scheduling scenarios to validate the effectiveness of the proposed method. The results show that the proposed method got better results compared to other well-known methods.

The contributions of this paper are summarized as follows.

- A new adaptive task scheduling method in the cloud-computing environment to reach the minimum time of transfer the tasks on available resources.
- Hybrid the multi-verse Optimizer with the genetic algorithm to improve its searchability.
- Apply various evaluation criteria and several scenarios to evaluate the proposed scheduling method.

The remaining section of this paper is organized as follows. Section 2 presents the previous work on the task scheduling area. Section 3 presents the proposed method hybrid multi-verse optimizer and genetic algorithm for solving the task scheduling problem. Section 4 discusses the experiments and results. Section 5 shows the

conclusion and future works. Finally, Sect. 6 shows the Research Implications and Limitations.

2 Related work

This section overviews the cloud computing definition, services, approaches, importance, and limitations to give a good understanding of cloud computing technology [31]. The definition of cloud computing is simplified as the execution of working tasks all or many departments of an organization using a central virtual machine that contains all information, services, and hardware requirements rather than using a single IT system for each group of employees or department inside an organization [32]. There are three main services provided by cloud computing which are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [33, 34]. Figure 1 presents the architecture of cloud computing services.

The descriptions of these services are given as follows. SaaS: this service represents users' and employees' interfaces that involve organizational activities of various domains like education and health [35]. These interfaces connected with working the application such as documentation of application, information-retrieving application, and reports application [36]. The users can insert, update, remove, and update their records to benefit from the business' services through user interfaces. Contrariwise, the employees and administrators can use SaaS to complete and manage their daily working tasks.

Moreover, the managers and decision-makers can use SaaS to generate statistics, reports, and analytical data to support their decisions and control their business. SaaS can be deployed using only monitors without storage and CPU. PaaS: This service represents a cloud computing operating system which processes and manages the gathered information between SaaS (system users) and IaaS (systems storage and infrastructure). IaaS: this service represents the hardware components of cloud computing, such as CPUs, storage, and network facilities. Thus, IaaS processes send and store applications data supporting PaaS management using network facilities.

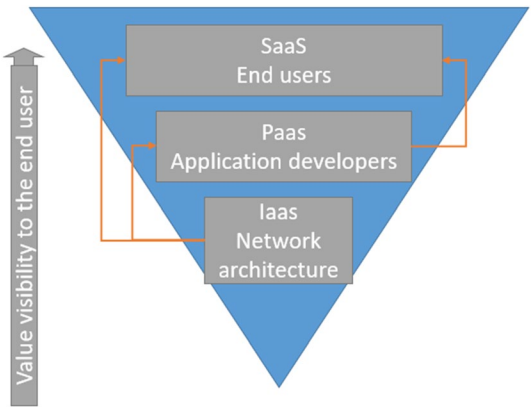

Fig. 1 Services of Cloud computing

Additionally, IaaS stores the applications deployed by PaaS and used by users' at the SaaS level.

As mentioned in the previous section, the job scheduler of heterogeneous cloud tasks should be multi-objective. The transfer efficiency and fairness need to be addressed. The job scheduler's multi-objective purpose increases the difficulty of (1) map the transfer tasks based on the importance weights and (2) reduces the transfer costs. This section presents the related algorithms that could be applied to schedule the transfer of heterogeneous cloud tasks.

Zeng in [37] mentioned that the MapReduce method is not investigated effectively in the applications of transferring the heterogeneous cloud tasks. The MapReduce works to reduce the tasks' transfer time though segment the large tasks as small blocks. The blocks can be delivered via different cloud paths. The MapReduce conducts based on two main processes: (1) map the transfer criteria through estimate the transfer time (input data of all tasks/blocks sizes), and (2) reduce the transfer time though allocate all available blocks in the cloud to transfer the data. The tasks that require fewer data will be transferred first. They found that the MapReduce can save 25–50% of the transfer time comparing with other transfer methods such as SLA agnostic. [38] proposed the multi-objective optimization to schedule the cloud tasks transfer based on many task properties. Although multi-objective optimization is adopting the genetic algorithm processes, the GA processes are managed in many stages based on nature's properties. Traditionally, the multi-objective optimization process starts with chromosome encoding for mapping and assignment of the task to the virtual machine. The next step is fitness calculation as per the objective of the functions. Genetic operators such as selection, crossover, and mutation are used during evolution. The evolution continues until the termination criteria are met. At this point, the Pareto front solutions from the evolution are retrieved and optimized to further process tasks. At each stage of multi-objective, a set of tasks, objectives, or properties will be evaluated. The multi-objective optimization is simulated based on two stages depending on two objectives which are the response time and resource consumption. The simulation results confirm that multi-objective optimization is useful for scheduling the task's transfer based on complex objectives or properties.

The Directed Acyclic Graph (DAG) is suggested to schedule the transfer of efficient tasks based on the transfer time costs [39]. DAG works like a decision tree to compute the tasks' transfer time based on many priorities, such as available cloud resources, transfer paths, and transfer throughput. DAG is effective in calculating the transfer time of complex tasks (i.e., a large set of tasks of heterogeneous resources). The time costs of all tasks are calculated to estimate the average transfer cost. The minimum and maximum costs can be determined based on specific formulas. Thus, the task scheduler will arrange the transfer priority based on the transfer time (the minimum costs are transferred firstly). The simulation result shows that DAG is useful to schedule the large set of tasks of heterogeneous resources based on the transfer cost (i.e., time).

[40] proposed a genetic algorithm (GA) to schedule the transfer of heterogeneous tasks based on the priority vectors of the tasks (transfer fairness). The users compute the transfer tasks' fitness function based on many properties, such as the trust of tasks and transfer repetition. The candidate tasks are considered as the initial

chromosomes. The higher priority task is selected based on the fitness of the task's properties, which is analyzed through crossover and mutation processes. This means the task priority is computed dynamically based on the property's appearance of current active tasks. The simulation results show the effectiveness of GA in optimizing the task priority based on the fitness function of the task properties detected in the population. They argued that task transfer based on the priority is the best compared with other scenarios, such as transfer the shortest tasks first.

[41] presented a cost-efficient task scheduling algorithm using two heuristic strategies. The first strategy dynamically maps tasks to the most cost-efficient VMs based on the concept of Pareto dominance. The second strategy, a complement to the first strategy, reduces the monetary costs of non-critical tasks. The simulation results showed that the algorithm could substantially reduce monetary costs while producing the makespan as good as the best known task scheduling algorithm can provide. However, this algorithm is inapplicable to multiple types of VMs with different pricing models. [42] proposed a Cloud scheduler based on Ant Colony Optimization (ACO). The scheduler's goal is to minimize the weighted flow time of a set of PSE jobs while minimizing makespan. Simulated experiments performed with real PSE job data and other Cloud scheduling policies indicate that this proposal allows for a more agile job handling while reducing PSE completion time. Besides, the evaluation results showed that ACO performs better than random and best effort algorithms. [43] proposed a trust dynamic level-scheduling algorithm named Cloud-DLS by integrating the existing DLS algorithm. This study's main contribution is extending the traditional formulation of the scheduling problem so that both execution time and reliability of applications are simultaneously accounted for. Theoretical analysis and simulations proved that the Cloud-DLS algorithm could efficiently meet the requirement of Cloud computing workloads in trust, sacrificing fewer time costs and assuring the execution of tasks securely.

[44] proposed a new priority-based job-scheduling algorithm in cloud computing based on multiple criteria decision-making model. This scheduling algorithm consists of three levels of priorities, including scheduling level (objective level), resource level (attribute level), and job level (alternative level). The algorithm calculates the priority vector of scheduling jobs (PVS), then chooses a job with a maximum priority value based on PVS and allocates appropriate resources. The result of this paper indicated that the proposed algorithm has reasonable complexity. All of the above-related works focused on the efficiency of the tasks scheduler by transferring the heterogeneous tasks in response time based on the available cloud resources.

Therefore, this paper's main challenge is to optimize the transfer of the heterogeneous task depending on the efficiency factors. The proposed optimization algorithm should work on multi properties of cloud resources such as speed, capacity, task size, number of tasks, number of virtual machines, and throughput. One of the most suitable reviewed algorithms for this paper is the multi-verse optimizer (MVO). MVO is a recently proposed evolutionary algorithm that works to optimize the solutions based on universe theory [45]. All universe objects (i.e., cloud tasks) belong to the initial hole called a white hole. Based on specific conditions, many objects could move to an inverse hole called a black hole. The black hole may receive other objects from the universe called a wormhole. The white hole's object characteristics are differing from the

characteristics of the objects in the worm universe. The black hole responsible for the schedule all received objects based on the various characteristics. Keep in mind that not all universes objects are allowed to move to a black hole (i.e., the movement is under control conditions).

Based on the reviewed works of heterogeneous task scheduling, the transfer scheduler should address the transfer efficiency through transfer the queued tasks as fast as possible. The conducted works have lacked an effective optimization algorithm to schedule the transfer of the task based on multi-objective problems (efficiency properties). This paper tries to fill the research gap by applying the MVO algorithm to schedule heterogeneous tasks transfer via the hybrid cloud. MVO algorithm expected to handle the efficiency properties to schedule the transfer tasks regarding its size based on the available cloud resources. On the other hand, the GA will be applied to optimize the scheduled tasks' transfer time. The integration between MVO and GA could help schedule the transfer of cloud tasks based on efficient transfer time.

3 The proposed hybrid multi-verse optimizer with genetic algorithm (MVO-GA)

This section presents the research methodology including the research design, proposed properties and algorithm, simulation environment, and evaluation processes. The research methodology is constructed based on the related works in the domain of transfer heterogeneous cloud tasks to assure useful achievement of the research objectives. The research methodology is the set of related processes and settings that should be accomplished to address the research objectives. Based on the research objective, questions, and objectives, the methodology was planned straightforwardly to develop the research.

3.1 Methodology design

This paper focuses on scheduling heterogeneous based on the cloud environment. The main research challenge is identified as to schedule the transfer of the task-based efficiency properties. Several previous works and studies have been reviewed to determine competency characteristics that can be applied to scheduling the priority of transferring heterogeneous tasks. The most important properties are the network transfer rate, transfer throughput, path distance, storage capacity, and processor speed. Based on the efficiency properties, the tasks should be scheduled based on the available cloud resources to produce an efficient workload of heterogeneous tasks transfer. For this purpose, this paper proposes a multi-verse optimization algorithm with a genetic algorithm, called (MVO-GA).

3.2 Task scheduling problem

The cloud task scheduling problem is defined as scheduling and allocating various tasks to numerous virtual machines (VMs) practically and making all the tasks

accomplished in a short execution period. Consider the cloud system (CS) consists of *Npm* physical machines (PM), and each physical machine consists of *Nvm* virtual machines (VMs) [7].

$$\mathrm{CS} = \left[\mathrm{PM1}, \mathrm{PM2}, ..., \mathrm{PM}_i, ..., \mathrm{PM}_{\mathrm{Npm}}\right] \tag{1}$$

where *PMi* (i = 1,…., *Npm*) denotes the *PMs* presented in the cloud, and it can be represented as in Eq. (2):

$$\mathrm{PM}i = \left[\mathrm{VM1}, \mathrm{VM2}, ..., \mathrm{VM}k, ..., \mathrm{VM}_{\mathrm{Nvm}}\right] \tag{2}$$

where *VMk*, $k = 1,2, \ldots,$ *Nvm* represents the k_{th} virtual machine. *Nvm* is the number of virtual machines and VM_k denotes the k_{th} virtual machine resource in the cloud environment. The feature of VM_k is defined as:

$$\mathrm{VM}k = \left[\mathrm{SIDV}_k, \mathrm{MIPS}_k\right] \tag{3}$$

where $SIDV_k$ is the serial number of virtual machines and $MIPS_k$ is the information processing speed of virtual machines (unit: *millions-of-instructions-per-second*, *mips*).

$$\mathrm{T} = \left[\mathrm{Task1}, \mathrm{Task2}, ..., \mathrm{Task}l, ..., \mathrm{Task}_{\mathrm{Ntsk}}\right] \tag{4}$$

where *Ntsk* is the number of tasks submitted by the users. *Taskl* represents the *lth* task in the task sequence. The feature of *Taskl* is defined as:

$$\mathrm{Task1} = \left[\mathrm{SIDl}, \mathrm{tasklengthl}, \mathrm{ECT}l, \mathrm{PI}l\right] \tag{5}$$

where *SIDT* l is the serial number of tasks and task length l is the instruction length of the task (unit: million instructions). Time *ECT* l refers to the expected completion time for the Task l; *PI* l refers to the task priority the number of tasks for *Ntsk*, the number of virtual machines for *Nvm*. The Expect Complete Time (ECT) matrix of size *Ntsk* × *Nvm* denotes the execution time required to run the task on each computing resource (virtual machine) that can be calculated by the following matrix:

$$ECT = \begin{bmatrix} ECT1,1 & \mathrm{ECT1,2} & \mathrm{ETC1,3} & \mathrm{ETC1, Nvm} \\ ECT2,1 & \mathrm{ECT2,2} & \mathrm{ETC\ 2,3} & \mathrm{ETC2, Nvm} \\ ECT3,1 & \mathrm{ECT3,2} & \mathrm{ETC\ 3,3} & \mathrm{ETC3, Nvm} \\ \mathrm{ECTNtsk,1} & \mathrm{ECTNtsk,2} & \mathrm{ECTNtsk,3} & \mathrm{ECTNtsk,Nvm} \end{bmatrix} \quad \ldots\ldots\ldots(6)$$

The main objective function is to reduce the makespan by locating the best set of tasks to be executed on VMs.

$$\mathrm{ETC} = \frac{\mathrm{TL}i}{\mathrm{MIPS}k}, k = 1, 2, 3, \ldots., \mathrm{Nvm}, i = 1, 2, 3, \ldots .., \mathrm{Ntsk} \tag{7}$$

where *ECTlk* refers to the required execution time of lth task on *kth* VM where *Nvm* is the number of VMs and *Ntsk* is the number of tasks. *TLi* is the task–length *i*. The fitness value is defined as:

$$\text{fit} = \max\left\{\text{ECT}_{lk}\right\}, \forall l \in \left[1, N_{\text{tsk}}\right] \text{ mapped } to\, kth\ VM, k = 1, 2, 3, ..., \text{Nvm} \quad (8)$$

3.3 Multi-verse optimizer

MVO applied for a structured section of the population that will involve the computation of the fitness solution of efficient task scheduling. For example, suppose {V1, V2, V3, V4} is the cloud virtual machines, and suppose {T1, T2, T3, and T4} are the active transfer tasks (as shown in Table 1). The population of MVO (White hole) is all active tasks in array [i][j], where i the number of tasks and j the number of virtual machines. Then, the fitness solution of each column in the array is computed. Hence, the optimal virtual machine for each task based on the transfer speed will be estimated. For example, T1/V2, T2/V3, T3/V4, and T4/V2 are the optimal solutions for each task in the array (i.e., black hole). Based on the black hole population, the GA will be applied to optimize the task scheduling based on the available cloud resources according to the intersection between the transfer tasks. The solutions of the MVO are updated using Eq. (9):

$$\begin{cases} x_k^j r1 < Ni(Ui) \\ X_i^j r1 \geq Ni(Ui) \end{cases}$$

$$x_i^j = \begin{cases} \text{if}(r_2 < \text{WEP}) & \begin{cases} if\ (r3 < 0.5)X_j + TDR \times ((ub_j - Ib_j) \times r4 + Ib_j) \\ \text{if } (r3 < 0.5)X_{\ j} + TDR \times ((ub_j - Ib_j) \times r4 + Ib_j \end{cases} \\ \text{if}(r_2 \geq \text{WEP}) & x_i^j \end{cases} \quad (10)$$

$$TDR = 1 - \frac{l1/p}{L1/p} \quad (11)$$

$$\text{WEP} = \min + l\left(\frac{\max - \min}{L}\right) \quad (12)$$

where x_{ji} indicates the j_{th} parameter of i_{th} universe, Ui shows the i_{th} universe, NI(Ui) is normalized inflation rate of the ith universe (the fitness values), r1 is a random number in [0, 1], and x_{jk} indicates the j_{th} parameter of k_{th} universe selected by a

Table 1 Example of VMs and tasks population

	T1	T2	T3	T4
V1	T1/V1	T2/V1	T3/V1	T4/V1
V2	T1/V2	T2/V2	T3/V2	T4/V2
V3	T1/V3	T2/V3	T3/V3	T4/V3
V4	T1/V4	T2/V4	T3/V4	T4/V4

roulette wheel selection mechanism. The above equation is for exchange the objects of universes i.

3.4 Genetic algorithm

The genetic algorithm is used to achieve an accurate schedule to process tasks in the Cloud. GA would be applied to the black hole prepared by MVO (MVO works to reduce the population boundaries). Hence, GA could find optimal solutions due to the reduced number of tasks population. The main operators of the genetic algorithm are the following:

- Selection method: this method is used to select pair random solutions to apply the crossover and mutation operator based on its probability [46].
- Crossover: this operator collects a pair of chromosomes to generate the chromosomes of the next generation using two-point method. Some characteristics of the first parent are transmitted to the new chromosome, but the rest of the characteristics come from the rest of the parents [47]. The crossover probability used in this paper is 0.8.
- Mutation: mutation operator in a genetic algorithm is used to maintain the diversity of the population by changing the chromosome with a small probability from the interval [0, 1], which is known as the probability of mutation (Pm), fixed is 0.2. Besides, two mutation points Mp1 and Mp2 should be defined to perform mutation operation. The mutation points are a random number between 0–1.
- Termination condition: this condition will apply if there are no further improvements to the Fitness value and the termination criteria best chromosome in the population is reached. More details for the used GA are given in [48, 49].

3.5 Hybrid multi-verse optimizer with genetic algorithm (MVO-GA)

On the other hand, the genetic algorithm (GA) is a search heuristic inspired by Charles Darwin's natural evolution theory. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction to produce offspring of the next generation. The GA includes many processes and computations to find the fitness solution based on the study environment. Figure 2 illustrates the methodology design of this paper. MVO and GA could be collaborative to support this paper through two main stages. The MVO would reduce the task's boundaries, TB = (V, E), where V is the set of tasks to be executed. E is the transfer paths of the tasks via the cloud virtual machines (VM). On the other hand, the GA will be applied to each task set in black holes to optimize the task scheduling based on efficiency properties as given in Fig. 3. The next section explains the details of the research methods.

Multi-verse optimizer (MVO) is a proposed evolutionary algorithm that works to optimize the solutions based on universe theory [45]. All universe objects (i.e., cloud tasks) belong to the initial hole called a white hole. Based on specific conditions, many objects could move to a smaller hole called a black hole. The black

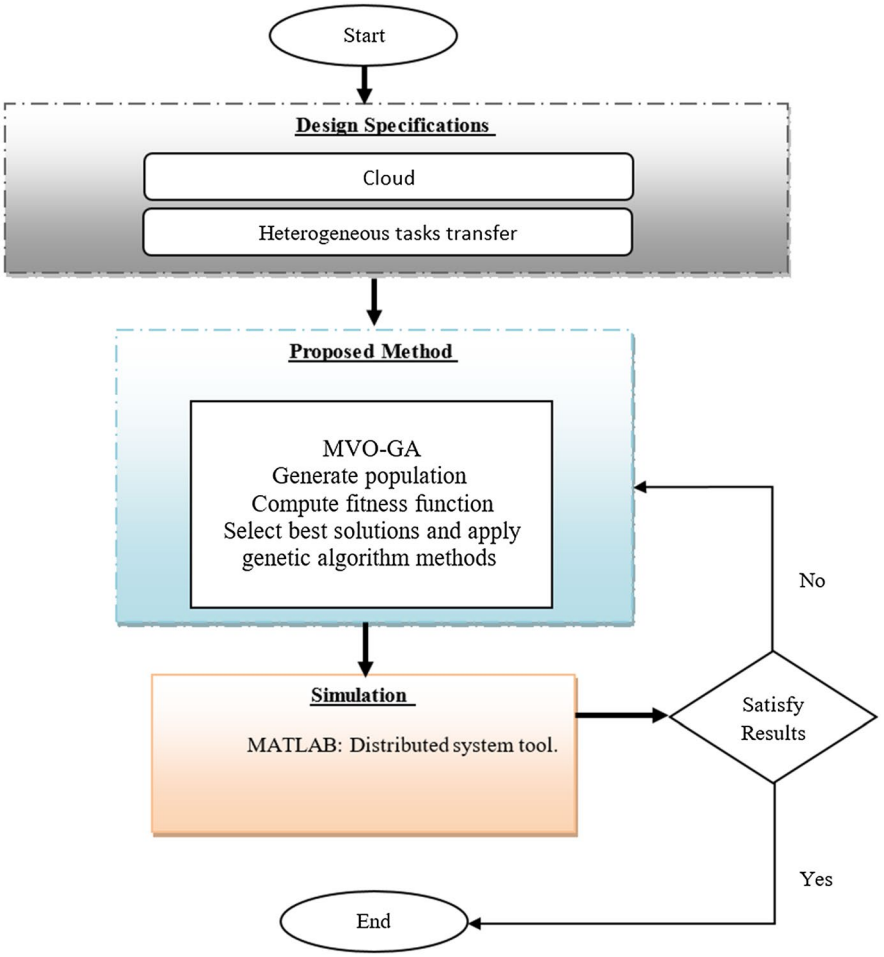

Fig. 2 Research design

hole may receive other objects from another universe called a wormhole. The white hole's object characteristics are differing from the characteristics of the objects in the worm universe. The black hole responsible for scheduling all received objects based on the various characteristics. Keep in mind that not all universes objects are allowed to move to a black hole (i.e., the movement is under control conditions).

4 Experimental results

This section provides the experimental settings of the proposed algorithms for scheduling the cloud task transfer. Based on the experiments, the system results are discussed using various datasets of transfer tasks. Besides, this section presents

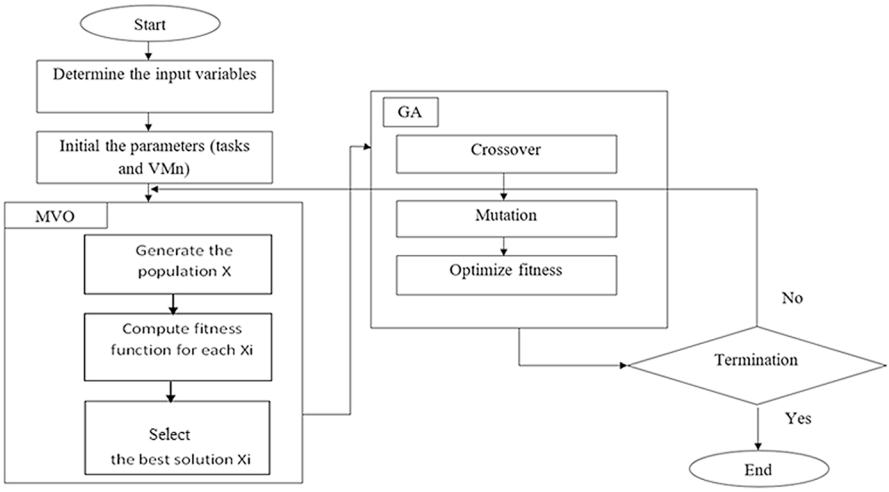

Fig. 3 Flowchart of the proposed hybrid MVO and GA method

the products' conclusion compared with other studies related to cloud task transfer scheduling.

As explained in Sect. 3, this paper applies the multi-verse objective (MVO) and genetic algorithm (GA), called MVO-GA, to optimize the cloud tasks. The proposed MVO-GA algorithm's main aim is to schedule the cloud transfer tasks in terms of transfer time. Thus, the cloud tasks need to be designed on the available virtual machines to reduce the currently active tasks' transfer time. The MVO-GA is proposed to distribute the transfer tasks based on the workload of the available virtual machines. Thus, the transfer tasks dataset is allocated based on the transfer properties of virtual machines such as machine storage, throughput, CPU, and transfer speed. After rescheduling the transfer tasks according to machine workload, the GA is applied to enhance task scheduling based on crossover and mutation processes. The output of the MVO-GA is considered as the initial population of the GA. Hence, the crossover and mutation processes work to optimize the task scheduling by looking for better possibilities to transfer some tasks through other virtual machines. In summary, MVO works to assure the virtual machines' workload, and the GA works to enhance the workload of these machines.

4.1 Simulation environment

The experiments are conducted through a simulation of a hybrid cloud environment and heterogeneous task transfer. The proposed method is evaluated using two experiments attributes to understand the accuracy of the proposed methods in this paper. Table 2 summarizes the simulation attributes in this paper. For effective simulation, a MATLAB toolbox called a distributed system would be utilized. This toolbox is useful to simulate a distributed network such as a hybrid cloud environment.

All experiments will be executed for ten different runs, and each run includes 500 iterations.

4.2 Experimental settings

This paper uses three modes of cloud task transfer scheduling; schedule 600 tasks, 1000 tasks, and 2000 tasks. The variety of dataset sizes could support the complex testing of the task transfer scheduling. The scheduling challenge is increased as the dataset size is increased. The datasets are imported for the simulation environment using MATLAB 2017a software.

Each task in the dataset is represented by four main parts: the task ID, the task size, the expected time to transfer the task (ECT), and the priority of the task transfer. This paper uses the task ID as a unique transfer identifier. On the other hand, the task size is used for the computational transfer time. Furthermore, the expected ECT is used to compare the real and expected transfer time in the context of MVO and GA processes. The transfer priority is not used in the scheduling process due to focusing on transfer time rather than transfer priority.

In a cloud transfer environment, the dataset placement would be managed using two main strategies: (1) transmit the dataset directly using the virtual machines, (2) transmit the dataset across data centers. The transfer across the data center is essential in a large number of datasets due to its effectiveness in managing the scheduling queue in a short time. Thus, this paper uses two data centers to schedule the dataset queue for MVO processes.

The experimental settings are adopted from related works to the scheduling of cloud transfer tasks. The experiments are conducted based on two virtual machines under the specifications mentioned in Table 4. The CloudSim version 3.0.3 is utilized in the experiments to conduct task transfer scheduling on MATLAB 2017a

Table 2 Simulation attributes

Entity	Parameter	Values of settings
Cloudlets	Number of cloudlets	100–2000
	Length	1000–2000
Virtual machine	RAM	512 MB
	MIPS	100–1000
	Size	10,000
	Bandwidth	1 Gb/s
	Policy type	Time shared
	Operating system	Windows
	No of CPU	1
Hosts	No of hosts	2
	RAM	2048 MB
	Storage	100 GB
	Bandwidth	1 Gb/s
	Policy Type	Time shared
Data center	No of data center	2

software in a distributed environment. The CloudSim is implemented to handle the prototyping of MVO and GA. The settings of CloudSim apply to two virtual machines of 512 RAM and 100–1000 MIPS.

On the other hand, the number of cloudlets is 100–2000 of length 1000–2000. The settings include two data centers, and each data center holding two hosts. The hosting capacity is there 1 TB, and the RAM capacity is 2048 MB. The simulation is conducted on Windows 2010 platform using a time-shared policy to simulate the reality of transfer time. The used machine has one CPU of dual-core properties for the hosted virtual machines.

4.3 Experimental results

The experiments were applied based on three modes of the dataset: 600 tasks, 1000 tasks, and 2000 tasks. The datasets are managed using two data centers in order to be processed by the MVO algorithm. MVO works to schedule the transfer tasks depending on the expected ECT to assure the virtual machines' balanced workload. The GA works on optimizing the transfer time of the dataset through the crossover and mutation processes.

Figure 4 shows the ECT of an optimized data transfer schedule for 600 tasks based on two basic algorithms (i.e., MVO and GA). Firstly, the MVO is applied without GA to understand the effectiveness of the MVO-GA, which will be explained later in this section. Based on MVO processes, the average ECT was computed for the two virtual machines' total executions. The results indicated that the optimized ECT based on MVO is about 5.6 s in the best case and about 654 s in the worst case. In other words, the MVO reduces the ECT time of the transfer tasks by about 649 s. Secondly, the optimized ECT for the two virtual machines based

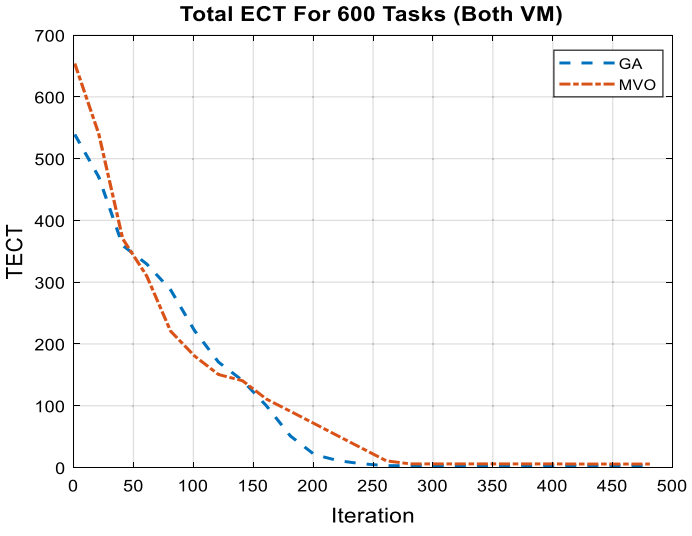

Fig. 4 MVO and GA in separate using 600 tasks

on GA is about 2.96 s in the best case and 593 s in the worst case. Both algorithms are practical to optimize the ECT of the transfer tasks (i.e., minimum solution). In other meaning, the GA reduces the ECT time by more than 590 s. In summary, of schedule 600 transfer tasks, both MVO and GA are useful in optimizing the ECT of the dataset. However, the GA is more effective than the MVO in the worst and best cases of the optimized ECT.

Figure 5 shows the ECT of the optimized task schedule for 1000 tasks based on two basic algorithms (i.e., MVO and GA). The average ECT was computed for both VMs. The MVO optimization results show that the total ECT of 1000 transfer tasks is about 5.3 s compared with 205 as initial ECT. Hence, the MVO reduces the transfer ECT by about 200 s. On the other hand, the optimized ECT based on GA for 1000 tasks records about 4.9 s comparing with 191 s as initial ECT. Thus, the GA reduces the ECT by about 146 s for the dataset of 100 tasks.

In conclusion, the GA is more effective than the MVO in optimizing the ECT of the transfer 1000 tasks using the two virtual machines. Although the optimized ECT is useful for MVO and GA, it can be noticed that the optimization of each algorithm is less effective when handling a larger dataset. The optimization of MVO and GA is better for the 600 tasks than the optimization for 1000 tasks.

To understand the effect of dataset size on the optimization effectiveness of MVO and GA, both algorithms are applied to a dataset of 2000 transfer tasks. Figure 6 shows the ECT of the optimized data transfer schedule for 2000 tasks based on two different algorithms (MVO and GA) using both VMs. The MVO optimization results based on 2000 tasks indicated that the optimized ECT is about 29.8 s in the best case comparing with about 610 s in the worst case. On the other hand, the optimized ECT based on GA is about 25 s in the best case comparing with 585 s in the initial case. The results show that the MVO and GA

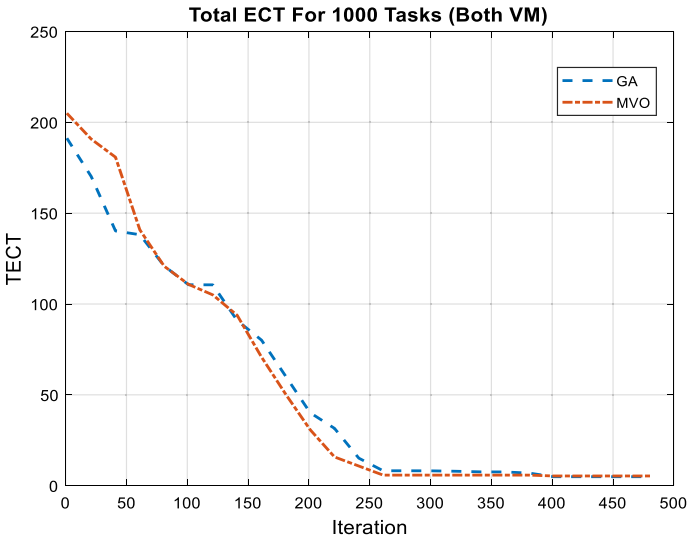

Fig. 5 MVO and GA in separate using 1000 tasks

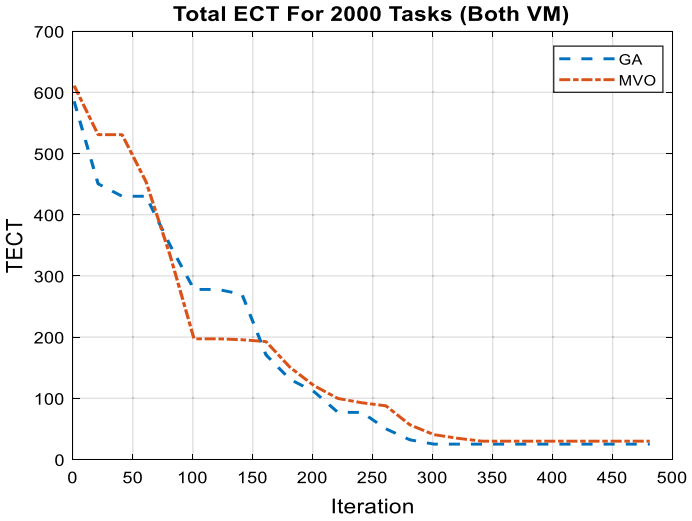

Fig. 6 MVO and GA in separate for 2000 tasks

effectively optimize the ECT for a dataset of 2000 tasks. However, the ECT is increased as the dataset size is increased. The optimized ECT using MVO and GA is better for 1000 tasks than a dataset of 2000 tasks.

From the above three figures, it can be concluded that the optimization is more difficult when the dataset size is more extensive. Hence, further analyses were performed on the dataset of 2000 assignments to understand scheduling behavior in the large data set. Figure 7 shows the ECT of each virtual machine using only the MVO algorithm based on a dataset of 2000 tasks. The results indicate that the VM1 would transfer the 2000 tasks in about 86 s in the best case. On the other hand, VM2 would transfer the same tasks in about 90 s in the best case. These results show the effectiveness of MVO to address the workload balancing of the available VMs. MVO approximately balances the distribution of transfer tasks on the two VMs based on the ECT of these tasks (i.e., 86 s for VM1 and 90 s for VM2).

Figure 8 shows the ECT of each virtual machine using only the GA algorithm for the dataset of 2000 tasks. The results indicate that the total ECT of VM1is about 27 s in the best case. On the other hand, the total ECT of VM2 is about 27 s. Hence, the GA is more useful to optimize the ECT than the MVO. However, the MVO virtually ensures the balanced workload of the transfer of the task using the available resources of VMs. Based on the above results, the integration between MVO and GA could effectively enhance the ECT optimization of the large dataset, i.e., 2000 tasks. The MVO would address the balance transfer scheduling by utilizing the available resources effectively. Consequently, the GA would improve ECT optimization based on the performed transfer schedule produced by MVO.

Figure 9 clarifies the comparisons between the GA and MVO in optimizing the ECT of the transfer tasks using a single VM (i.e., VM1 or VM2). The results indicate that the GA can optimize the solutions better than MVO when applying a large

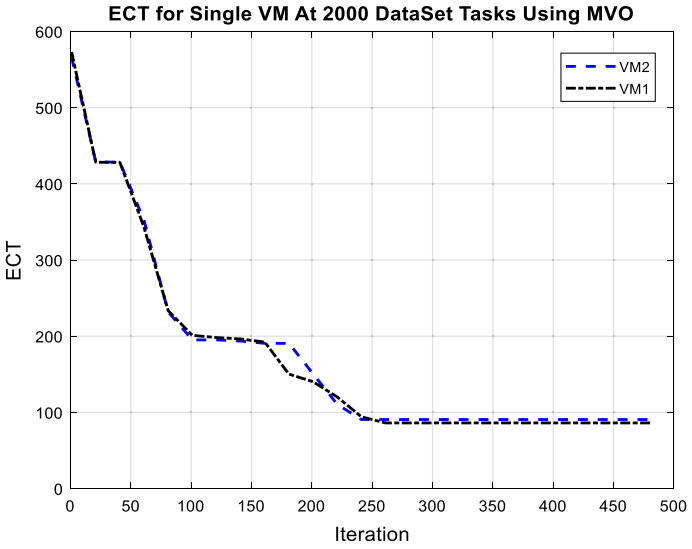

Fig. 7 Single-VM optimization based on MVO for 2000 tasks

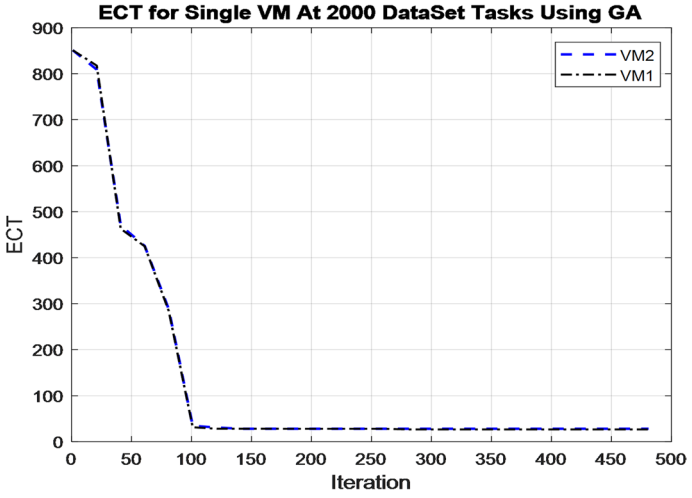

Fig. 8 Single VM optimization based on GA for 2000 tasks

dataset of 2000 tasks. However, the MVO processes would improve the GA optimization by scheduling the transfer tasks effectively.

Table 3 summarizes the optimized ECT based on the various experiment scenarios explained by now in this section. The experiment scenarios applied the MVO and GA in the separated mode for 600, 1000, and 2000 transfer tasks (Fig. 10).

In summary, the MVO-GA provides better ECT for both VM to transfer 2000 tasks. The MVO-GA records about 4 s as ECT for 2000 tasks comparing with 29 s

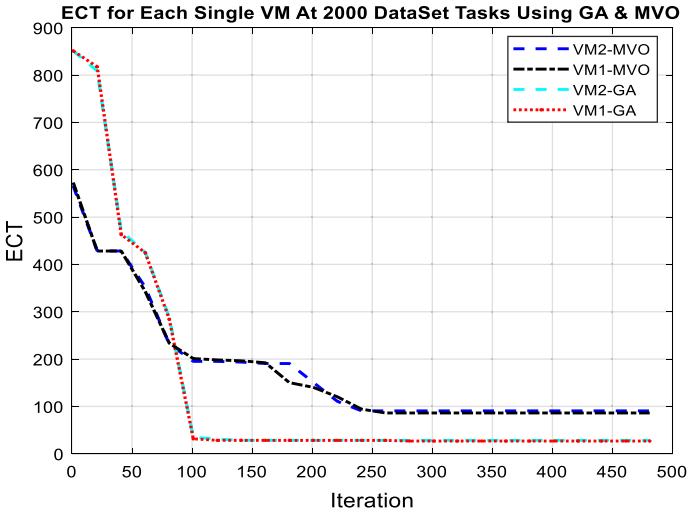

Fig. 9 Comparisons between GA and MVO using single VM on 2000 tasks

based on only MVO and 25 s for the only GA. To confirm the analyzed data, Fig. 11 shows the MVO-GA to optimize the ECT for 1000 transfer tasks using two virtual machines. The result indicates that the MVO-GA record about 1.5 s to transfer the 1000 scheduled tasks using two virtual machines. Thus, the MVO-GA would better optimize the ECT than working on MVO or GA in a separate mode. The simulation using only MVO records about 5.3 s as optimized ECT and the GA records about 4.9 s as optimized ECT.

In Table 4, the MVO-GA is applied using a dataset of 2000 tasks to study the features' effects of MVO-GA on large datasets' ECT optimization. Figure 10 shows the optimization results using two VMs of the 2000 transfer tasks depend on the MVO-GA. The result indicates that the MVO with partial GA (i.e., only crossover process) records 5 s ECT as an optimized solution. The MVO is supporting full GA records about 4 s ECT as an optimized solution. The 2000 tasks take a longer transfer time to eliminate the crossover process (about 10 s when keeping mutation processes and

Table 3 Summary of ECT experimental results of conventional methods

Algorithm Task		GA			MVO		
Both VMs		Best	Worst	Avg	Best	Worst	Avg
With Crossover	600	**2.98628**	539.170	109.7498	**5.6347**	653.965	119.8299
	1000	**4.9259**	191.2557	55.1608	**5.3305**	205.003	55.7378
	2000	**25.050**	585.328	158.8518	**29.869**	610.94	166.5817
Single VMs		Best	Worst	Avg	Best	Worst	Avg
2000 Tasks	VM1	26.86071	817.520	136.8136	86.1670	572.990	173.3205
	VM2	28.3611	808.936	135.7598	90.64236	565.9419	176.4578

The bold font refers to the obtained best results

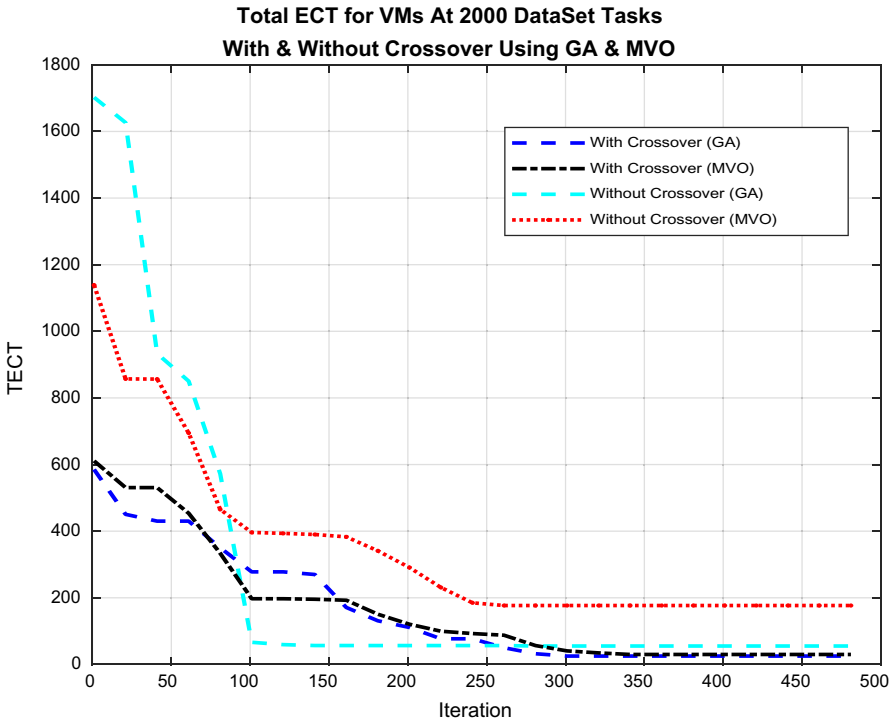

Fig. 10 MVO-GA for 2000 task

198 s without mutation processes). These results show the importance of crossover processes in enhancing ECT optimization. On the other hand, the integration between MVO provides effective ECT for the large dataset rather than applying each algorithm separately. In comparison with other methods (i.e., Gray Wolf Optimizer and Arithmetic Optimization Algorithm), the proposed MVO-GA got better results. Figure 12 shows clearly that the proposed method outperformed all other comparative methods when used 1000 tasks as well as when used 2000 as shown in Fig. 13.

This section explains the experimental results based on various scenarios using the MVO and GA to optimize cloud transfer tasks' ECT. The next section discusses the recorded results in the conducted simulation. The experimental results show that the dataset of small size, i.e., 600 transfer tasks, can be addressed using a single optimization algorithm such as MVO or GA. The scheduling challenge of task transfer appears clearly in a large dataset, i.e., more than 1000 tasks. Using a single algorithm (MVO or GA) could improve the ECT of the transfer tasks. However, further enhancement in optimizing the ECT is required.

Hence, the MVO-GA is proposed to improve the ECT optimization for large datasets. The proposed integration method performs the scheduling of cloud task transfer faster than using a single algorithm. For example, the ECT of 2000 transfer tasks is about 4 s using the MVO-GA. For the same dataset, the best ECT record of MVO is 29 s, and the best record of GA is 25 s. This indicates the effectiveness in optimize

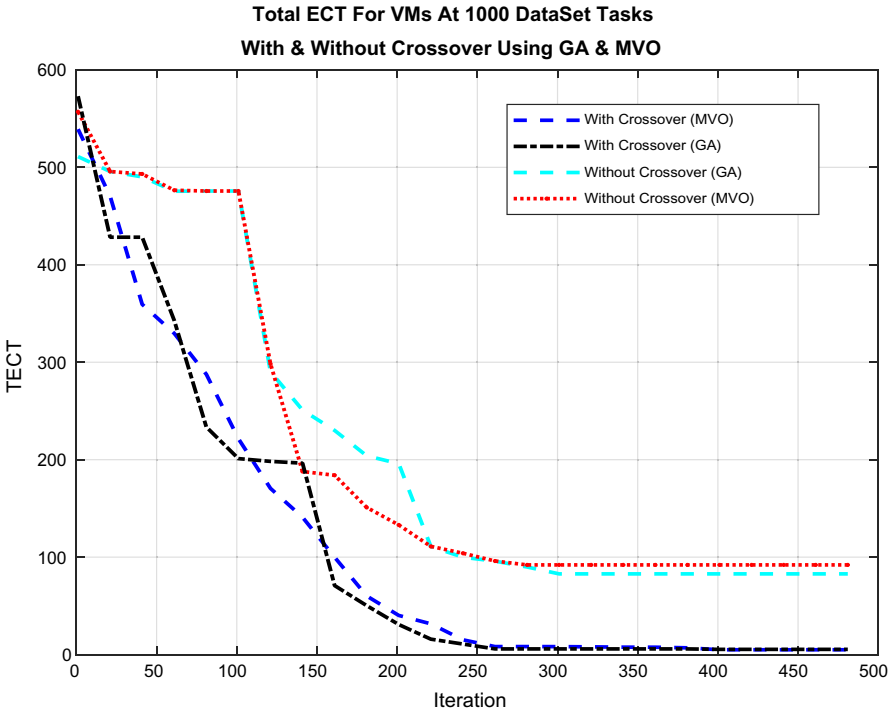

Fig. 11 MVO-GA for 1000 tasks

Table 4 Summary of ECT experimental results of the comparative methods

Algorithm	Results of ECT			
	Task size	Best	Worst	Average
MVO-GA	1000	1.4892	189.41	46.846
	2000	4.0313	559.67	120.92
MVO With Only Crossover	1000	4.9140	195.61	158.62
	2000	4.9259	189.01	153.752
MVO With Only Mutation	1000	5.3051	206.53	60.8325
	2000	9.8914	506.93	213.44
MVO Without GA	1000	25.371	576.94	142.95
	2000	198.36	810.71	169.53
Grey Wolf Optimizer	1000	1.6952	199.441	80.154
	2000	9.0548	325.67	112.692
Arithmetic Algorithm Optimization	1000	1.59582	199.952	79.2547
	2000	8.658	320.654	111.59

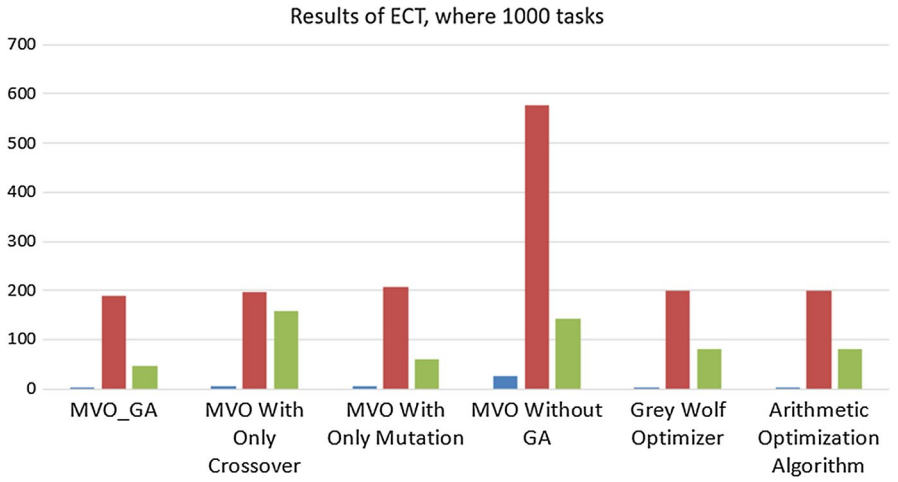

Fig. 12 the results of comparative methods in terms of ECT using 1000 tasks

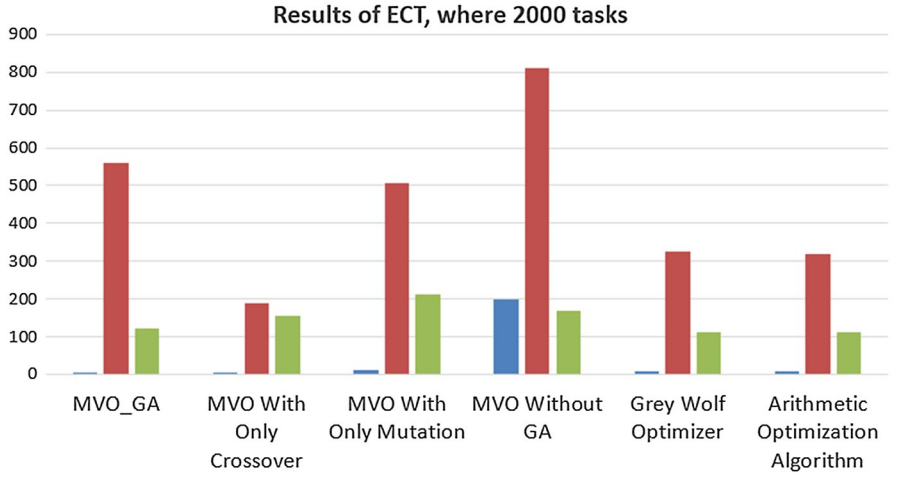

Fig. 13 the results of comparative methods in terms of ECT using 2000 tasks

the ECT of the transfer task using two optimization stages. Firstly, the MVO stage works on schedule the tasks dataset in terms of balance workload depend on the available machine's resources. Secondly, the GA processes (crossover and mutation) work on finding better solutions to optimize the transfer tasks' ECT by replacing the task allocations on the available resources to improve the total ECT of the virtual machines.

The recorded results in this paper for 2000 tasks optimized the ECT by 13.79% compared to using only MVO, i.e., 29 s, and the MVO-GA, i.e., 4 s. Also, the recorded results for 2000 tasks optimized the ECT by about 16% compared to only GA, i.e., 25 s, and the integration between MVO and GA, i.e., 4 s. In other meaning,

Elaziz et al. (2019) and Abualigah and Diabat (2020) record about 1–2 s as ECT to transfer 2000 cloud tasks using methods such as MALO, ALO, PSO, and MSA comparing with 4 s as ECT to transfer the same dataset using MVO-GA. Although the compared studies' results are more effective, the use of MVO-GA is also effective due to a good percentage in optimizing the ECT of large datasets.

This section clarifies the experimental results of the proposed MVO-GA to optimize the cloud tasks. This section also explains the implemented datasets and the experimental settings to perform the results based on various scenarios. In summary, the optimization of ECT for cloud transfer tasks faced a challenge in transferring many tasks, i.e., 2000 tasks. The utilizing of MVO-GA provides adequate optimization time comparing with utilizing only MVO or GA. The discussion of various testing scenarios shows that the integration between MVO-GA is applicable in scheduling the cloud transfer tasks based on efficient ECT. MVO is applicable to schedule the transfer tasks to depend on the available machine resources and balance workload. On the other hand, GA applies to improve the ECT optimization by finding better resources to transfer the scheduled tasks using crossover and mutation processes.

In general, this paper is essential for efficient scheduling of cloud transfer tasks to reduce the extensive data transfer time using different specifications of cloud resources. The main implications of this paper are the following:

1. Theoretically, this paper explains the properties that could affect the efficiency of cloud tasks transfer time. The cloud transfer time is controlled by cloud resources' efficiency, such as capacity, speed, bandwidth, and length of tasks. The efficiency of the cloud characteristics must be carefully analyzed to address the cloud transport schedule's optimization time.
2. Theoretically, this paper explains the MVO-GA algorithms' effectiveness to schedule the cloud transfer tasks in terms of transfer time. This is the main contribution of this paper. There is a research gap in implementing integration between various optimization algorithms to address cloud transfer tasks.
3. Practically, the MVO is an effective algorithm to assure the workload of the available cloud resources. This algorithm can handle the distribution of the cloud tasks through estimate the various specifications of the cloud resources, such as the speed and capacity of the virtual machines.
4. Practically, the GA is sufficient to enhance the transfer tasks' schedule rather than schedule the tasks themselves. According to available cloud resources, the crossover and mutation processes in GA are designed to find better possibilities to allocate the transfer tasks. Thus, the GA would play an essential role in supporting missions rather than handling the complete mission of scheduling the task's transfer.
5. Practically, the MVO-GA promises a solution to schedule a large set of cloud transfer tasks in terms of transfer time efficiency. The features of these two algorithms can be designed based on two stages: (1) prepare the initial tasks transfer schedule using MVO to address the workload of the cloud resources, and (2) enhance the scheduling using GA through finding better possibilities to optimize

the transfer time of the large number of transfer tasks using different specifications of cloud resources.

5 Conclusion and future works

Cloud technology is used widely to equip virtual machines as central storage and processors to avoid IT equipment costs in organizations or homes. Thus, it is necessary to transfer the large volume of cloud tasks in response time. The weakness in the cloud tasks schedule could lead to delay in gathering the tasks, which cause problems in delivering the services at the right time. Hence, it is essential to optimize the cloud tasks by scheduling the tasks queue using the cloud resources. This paper applies MVO-GA algorithms to schedule cloud tasks. The proposed algorithms work to optimize the heterogonous cloud tasks. This paper presents the main challenge of schedule the transfer cloud tasks, which is the heterogonous transfer of the cloud environment. The difference between cloud resources specifications (i.e., storage capacity and speed) could lead to slow data transfer. The optimization algorithms would resolve this issue by scheduling the tasks based on the available cloud resources and transfer time efficiency.

In terms of the second research objective, the MVO-GA algorithms are proposed due to these algorithms' optimization features. The MVO is proposed to schedule the transfer tasks queue according to available cloud machines. The MVO could work effectively to balance the distribution of a large set of transfer tasks depends on the efficiency properties of cloud machines such as capacity, speed, and bandwidth. On the other hand, the GA was proposed to improve the workload of the cloud resources to reduce the required transfer time of the scheduled tasks. The transfer tasks could transfer more efficiently using other machines or gates through the crossover and mutation processes. Therefore, the output of the MVO is considered as the initial population of the GA. The crossover and mutation processes work on optimizing the transfer time by looking for better possibilities to transfer some tasks through more free virtual machines. According to the third research objective, the proposed algorithm is simulated based on cloud task transfer efficiency prorates. The simulation was conducted using MATLAB software to judge the optimization efficiency of transfer time using the MVO-GA algorithm. The simulation results show that the integration between the proposed algorithms performs the scheduling of cloud task transfer faster than using MVO or GA separate. The proposed algorithm's success in optimizing the transfer time of many tasks (1000–2000) is about 15%. Compared with other studies, the proposed MVO-GA records excellent results in optimizing the large cloud tasks, reflecting the effectiveness of this paper. The next section presents the most important implications of this paper. Based on the outcomes of this paper, the following are suggested future works:

1. Apply the MVO-GA on a larger dataset of cloud data transfer (i.e., 10,000 million tasks) to study the proposed algorithm's effectiveness based on the real environment of cloud transfer tasks to optimize the transfer time.

2. Test the integration between other optimization algorithms such as greedy algorithm and GA, MVO and PSO, greedy algorithm, and PSO to schedule the cloud transfer tasks in terms of transfer time. The standpoint of the integration between the features of the algorithm must structure based on two stages that are: (1) assure the balance load of transfer scheduling and the available cloud resources; (2) enhance the workload of the cloud resources through optimizing the transfer time of the scheduled tasks in the first stage.

Funding No funding was received for conducting this study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

Human and animal rights No animal studies were carried out by the authors for this article.

References

1. Kumar M, Sharma S (2018) Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. Comput Electr Eng 69:395–411
2. Wickremasinghe B, Calheiros RN, Buyya R (2010) Cloudanalyst: a cloudsim-based visual modeller for analysing cloud computing environments and applications. in 2010 24th IEEE international conference on advanced information networking and applications. 2010. IEEE
3. Bokhari MU, Makki Q, Tamandani YK (2018) A survey on cloud computing. Big Data Analytics. Springer, pp 149–164
4. Li J et al (2020) OKCM: improving parallel task scheduling in high-performance computing systems using online learning. J. Supercomput 1–24
5. Linthicum DS (2016) Emerging hybrid cloud patterns. IEEE Cloud Computing 3(1):88–91
6. Manickam M, Rajagopalan S (2019) A hybrid multi-layer intrusion detection system in cloud. Clust Comput 22(2):3961–3969
7. Abualigah, L, A Diabat (2020) A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Cluster Comput 1–19
8. Yuan H, J Bi, M Zhou (2019) Profit-sensitive spatial scheduling of multi-application tasks in distributed green clouds. IEEE Transac Automation Sci Eng
9. Abualigah L et al (2020) TS-GWO: IoT Tasks Scheduling in Cloud Computing Using Grey Wolf Optimizer, in Swarm Intelligence for Cloud Computing. Chapman and Hall/CRC. p. 127–152.
10. Mansouri N, Javidi M, Zade BMH (2020) A CSO-based approach for secure data replication in cloud computing environment. J Supercomput, 1–52
11. Alguliyev RM, Imamverdiyev Y, Abdullayeva FJ (2019) PSO-based load balancing method in cloud computing. Autom Control Comput Sci 53(1):45–55
12. K Sreenu, M Sreelatha (2019) W-Scheduler: whale optimization for task scheduling in cloud computing. Cluster Comput, 1–12
13. Toosi AN, Sinnott RO, Buyya R (2018) Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka. Futur Gener Comput Syst 79:765–775
14. Alshinwan M et al (2021) Dragonfly algorithm: a comprehensive survey of its results, variants, and applications. Multimedia Tools and Applications, 1–38

15. Safaldin M, Otair M, Abualigah L (2021) Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks. J Ambient Intell Humaniz Comput 12(2):1559–1576
16. Eid A, Kamel S, Abualigah L (2021) Marine predators algorithm for optimal allocation of active and reactive power resources in distribution networks. Neural Comput Appl, 1–29
17. Al-Qaness MA et al (2020) Marine predators algorithm for forecasting confirmed cases of COVID-19 in Italy, USA, Iran and Korea. Int J Environ Res Public Health 17(10):3520
18. Abualigah L et al *Selection scheme sensitivity for a hybrid Salp Swarm Algorithm: analysis and applications.* Engineering with Computers, 2020: p. 1–27
19. L Abualigah, A Diabat (2020) A comprehensive survey of the Grasshopper optimization algorithm: results, variants, and applications. Neural Comput Appl. 1–24
20. Abualigah L, Diabat A, Geem ZW (2020) A comprehensive survey of the harmony search algorithm in clustering applications. Appl Sci 10(11):3827
21. Abualigah L, Diabat A(2021) Advances in sine cosine algorithm: a comprehensive survey. Artificial Intell Rev, 1–42
22. Altabeeb AM et al (2021) Solving capacitated vehicle routing problem using cooperative firefly algorithm. Applied Soft Computing, 107403
23. Abualigah L et al (2021) A parallel hybrid krill herd algorithm for feature selection. Int J Mach Learn Cybern 12(3):783–806
24. Abualigah LMQ (2019) Feature selection and enhanced krill herd algorithm for text document clustering. Springer.
25. Shehab M et al (2020) Moth–flame optimization algorithm: variants and applications. Neural Comput Appl 32(14):9859–9884
26. Jiang Y et al (2021) An efficient binary Gradient-based optimizer for feature selection. Math Biosci Eng 18(4):3813–3854
27. Abualigah L (2020) Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. Neural Comput Appl, 1–24
28. Alsalibi B, Abualigah L, Khader AT (2021) A novel bat algorithm with dynamic membrane structure for optimization problems. Appl Intell 51(4):1992–2017
29. Abualigah L et al (2021) Aquila Optimizer: A novel meta-heuristic optimization Algorithm. Comput Indus Eng, **107250**.
30. Abualigah L et al (2020) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609
31. Mapetu JPB, Kong L, Chen Z (2020) A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing. J Supercomput, 1–42
32. Jovevski D (2011) Impact of cloud computing on the business worldwide, the level of use in Macedonian companies. Methodius University, Skopje, Faculty of Economics
33. Hayes B (2008) Cloud computing. ACM New York, NY, USA
34. Pallis G (2010) Cloud computing: the new frontier of internet computing. IEEE Internet Comput 14(5):70–73
35. Khurana S, Verma AG (2013) Comparison of cloud computing service models: SaaS, PaaS, IaaS. Int J Elect Commun Technol IJECT, **4**.
36. Boksebeld R (2010) The impact of cloud computing on enterprise architecture and project success. Apeldoorn: Hogeschool Utrecht Faculty Science and Engineering
37. Zeng X et al (2018) Cost efficient scheduling of MapReduce applications on public clouds. J comput Sci 26:375–388
38. Naik K, Gandhi GM, Patil S (2019) Multiobjective virtual machine selection for task scheduling in cloud computing. Computational Intelligence: Theories, Applications and Future Directions-Volume I. Springer, pp 319–331
39. Chen W et al (2017) Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems. Futur Gener Comput Syst 74:1–11
40. M Ashouraie, NJ Navimipour (2015) Priority-based task scheduling on heterogeneous resources in the Expert Cloud. Kybernetes, 2015
41. Su S et al (2013) Cost-efficient task scheduling for executing large programs in the cloud. Parallel Comput 39(4–5):177–188
42. Mateos C, Pacini E, Garino CG (2013) An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments. Adv Eng Softw 56:38–50

43. Wang W et al (2012) Cloud-DLS: dynamic trusted scheduling for cloud computing. Expert Syst Appl 39(3):2321–2329
44. Ghanbari S, Othman M (2012) A priority based job scheduling algorithm in cloud computing. Procedia Eng 50:778–785
45. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513
46. Abualigah LMQ, Hanandeh ES (2015) Applying genetic algorithms to information retrieval using vector space model. Int J Comput Sci Eng Appl 5(1):19
47. L Abualigah, AJ Dulaimi (2021) A novel feature selection method for data mining tasks using hybrid sine cosine algorithm and genetic algorithm. Cluster Comput, 1–16.
48. Abualigah LM, Khader AT (2017) Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. J Supercomput 73(11):4773–4795
49. CB Şahin, Ö Dinler, L Abualigah (2021) Prediction of software vulnerability based deep symbiotic genetic algorithms: Phenotyping of dominant-features. Appl Intell, 1–17.