

Hybrid Chaotic Grey Wolf Optimizer for Task Scheduling in Cloud Computing

Erfan Fouladfar^{1,2*}, Mohammad-Reza Khayyambashi², Josep Solé-Pareta¹

¹Department of Computer Architecture, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

²Department of Computer Architecture, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

Email: M.R.Khayyambashi@eng.ui.ac.ir

Email: pareta@ac.upc.edu

*Corresponding Author: efouladf@ac.upc.edu

Abstract:

Cloud computing is a collection of distributed computers that allow users to access resources and services over the Internet. The optimal use of resources for various applications depends on how tasks are scheduled in the cloud environment, which has a significant impact on the efficiency of cloud service providers. Task scheduling is a key process in the cloud computing environment, which aims to execute user requests on resources in an efficient manner, taking into account other features of the cloud environment. The main goal of task scheduling is minimizing the makespan time and maximizing throughput as well as minimizing the cost. Since task scheduling is considered as an NP-complete problem, the necessity of using nondeterministic and meta-heuristic algorithms to optimize task scheduling is evident at a logical time. The major part of this problem is to design an efficient intelligent searching pattern to schedule the tasks in available virtual machines. In this paper, we propose a meta-heuristic algorithm called hybrid chaotic grey wolf optimizer (HCGWO) to tackle the problem of task scheduling in various heterogeneous virtual machines. This paper focuses on minimizing overall makespan and cost by modeling the swarm intelligence. The proposed algorithm prevents the local convergence and explores the global intelligent searching in finding the best optimized virtual machine for the user task among the set of virtual machines. We have made the simulation and performance evaluation using Cloudsim toolkit and compared the results with other swarm intelligent based algorithms such as Genetic Algorithm, Particle Swarm Optimization, and Cuckoo Search. The evaluation results show that there is a major improvement in minimizing the makespan and cost.

Keywords Cloud computing, Task scheduling, Virtual machine, Multi-objective Optimization, Grey wolf optimizer

1. Introduction

Cloud computing is a reliable computing model that offers highly efficient and highly scalable web-based applications, computing resources and services. These services can be accessed over the Internet which has attracted many users due to the reduction of operating cost [1]. Through strong virtualization capabilities, users can submit their requests to the cloud without any knowledge of the underlying structure of the platform [2].

In cloud systems, resources are distributed across multiple domains which increase the security of the system [3]. The main objective of cloud computing is to provide services in a fast and energy efficient manner using available resources. Therefore, tasks submitted by end-users are required to be scheduled by a scheduling mechanism that balances the load on available resources while minimizing their execution time and cost [4]. Three types of categories called infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) are provided by the cloud service model. Infrastructure as a Service (IaaS) is a self-service model which offers clients to access, monitor, and manage remote datacenter infrastructures, such as virtualized components, external storages, high-end computing resources, and networking services (e.g. firewalls). Platform as a service (PaaS) is a third-party provider, which delivers hardware and software tools and provides a platform and environment to allow developers to build applications and services over the Internet. Software as a service (SaaS) is a model for the distribution and programming models to analyze and process the data in a cloud environment [5]. These three cloud service models are shown in Fig. 1.

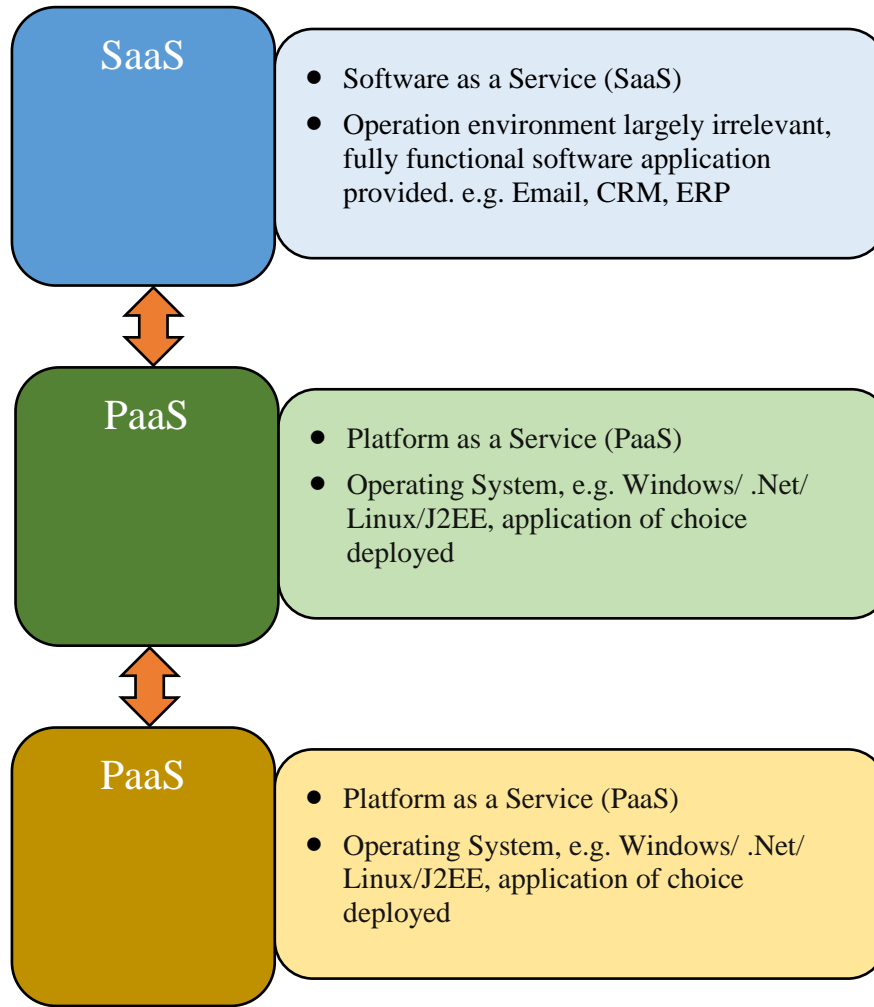


Fig. 1 Cloud service models.

Task scheduling problem is an NP-complete problem that takes into account different factors such as execution time, power consumption, and resource utilization [6]. The problem of obtaining a suitable balance between the duration and energy consumption is a multi-objective optimization problem. The process of task scheduling involves receiving the tasks from users and assigning them to available resources while considering the requirements and properties of each task [7]. Many studies solely try to minimize execution time as the single objective of the problem while ignoring the need for minimization of cost in cloud infrastructures.

For a reliable scheduler, it is advantageous to use meta-heuristic algorithms that find near-optimal solutions. Grey wolf optimizer (GWO) [8] is a recently introduced population-based optimizing algorithm that has shown to be a better performing algorithm than many other algorithms. The previous optimization techniques such as GA and PSO not only suffer from local minima but also have time complexity problem [9]. In order to improve previous studies and overcome the limitations in the existing research works like high makespan problem with GA and PSO, a multi-objective task scheduling approach called HCGWO for clouds using grey wolf optimizer is proposed in this study. The performance of the proposed HCGWO is evaluated through experimental studies by three sets of configurations. Finally, the results are compared with those provided by some meta-heuristic approaches like Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Cuckoo Search (CS).

The rest of the paper is organized as follows: a brief review of some of the literature works based on task scheduling in cloud computing is presented in Section 2. In Section 3 the background of the proposed method is explained in detail. Problem evaluation of our proposed method is given in Section 4. The experimental results and the performance evaluation discussion are provided in Section 5. Finally, the conclusions are summed up in Section 7.

2. Related Work

In recent years, a lot of research has been conducted in the area of task scheduling problem. Here, we review some of the relevant studies. Ramezani *et al.* [10] enhanced the quality of service in a cloud environment by decreasing the length of tasks queue in addition to reduction in task execution cost and power consumption. Traditionally, in many task scheduling algorithms using evolutionary algorithms, tasks get assigned to higher performance virtual machines while lower performance virtual machines get neglected which leads to unnecessary longer queues. Their proposed solution to this problem was developed based on multi-objective particle swarm optimization and multi-objective genetic algorithm.

To tackle energy consumption challenges in cloud, Zhao, *et al.* [11] introduced the Task Requirement Degree (TRD) to enhance resource utilization while reducing power consumption. Two main strategies to achieve this goal include avoiding frequent transmission of data which is fairly energy consuming and load balancing between machines to decrease inefficiency. Slave ant colony optimization (SACO) is a novel ant colony based algorithm proposed by Moon, *et al.* [12] that is tested on the task scheduling problem. The improvement made to the original optimizing algorithm is by avoiding long paths created by leading ants and reported results to show improvement in terms of preprocessing time and makespan over other ant colony based algorithms. Alla, *et al.* [13] proposed a novel technique by using both fuzzy logic and particle swarm optimization coupled with dynamic queues.

Keshanchi *et al.* [14] introduced a priority-based task scheduling approach in cloud environment which uses an improved variant of genetic algorithm called N-GA. An elitism schema is adopted to avoid premature convergence of solutions. In addition to that, the proposed model is verified using model checkers which showed model support for reliable and deadlock-free problems.

With emphasis on power efficiency and environmental concerns, Lin *et al.* [15] proposed a task scheduling algorithm that uses server power efficiency of tasks to guide the algorithm which can be described as the ratio of tasks executed to the amount of energy consumed in a specific amount of time. The main idea is to predict power efficiency of every task on every machine and using this parameter to guide task scheduling and mapping tasks to the machine with highest energy.

Since reliability is an important factor in quality of service in cloud computing, Yang, *et al.* [16] have selected reliability of nodes as the objective to be optimized. With the means of game theory, the problem of task scheduling is modeled as a sequential multi-stage game. The task scheduling algorithm based on sequential game mainly attempts to minimize the execution time of the task. Findings are promising for big data in cloud environments. An artificial bee colony algorithm that approaches the problem with multiple objectives of makespan, cost, energy, and load is proposed in [17].

A different set of objectives including energy, time, migration cost, and load utilization was the subject of the hybrid algorithm introduced by Gobalakrishan and Arun [18]. They proposed a hybrid algorithm by combining genetic algorithm and grey wolf optimizer. The genetic algorithm improves the efficiency of grey wolf optimizer while optimizing the execution of tasks in virtual machines which results in better performance than each algorithm individually.

Similarly, a genetic and bacteria foraging hybrid algorithm to tackle the balance between makespan and power consumption is modeled as a multi-objective optimization problem in [19]. Simulation results show that the scalability of the hybrid algorithm is promising to make it suitable for large-scale clouds.

Nasr *et al.* [9] have presented a new meta-heuristic method called water pressure change optimization (WPCO) to solve task scheduling problem. WPCO is inspired by the phenomenon of water density changing. WPCO can allocate tasks on available resources in low time complexity as well as improving memory usages and resource utilization.

A hybrid meta-heuristic algorithm based on fuzzy logic and particle swarm optimization along with dynamic dispatch queue is proposed in [20] to optimize cloud utilization and quality of service. The experimental results show an improvement in makespan, resource utilization, and load balancing.

3. Background

3.1. Grey Wolf Optimizer (GWO)

Grey Wolf Optimizer, proposed by Mirjalili *et al.* in 2014 [8], is a newly introduced swarm intelligence (SI) algorithm, inspired by the collective behavior of grey wolves (GWs) and based on social hierarchy and hunting procedure of GWs for prey in nature, like any other swarm intelligence algorithm. Grey wolves usually prefer to live in a pack with the size of 5 to 12 members on average. According to [8], all grey wolves in a pack have to follow very strict rules in a social dominant hierarchy. The hierarchy of grey wolves consists of four levels, gradually decreasing from α to ω according to fitness value [21]:

1. The alphas (α), a male and a female, are the most dominant wolves and in the highest level in the hierarchy. This means that they are leaders and responsible for making all decisions such as hunting. These decisions should be followed by the group. The alphas represent the best solution in GWO.
2. The betas making α of dominance in the group and help the second highest level are the (β) decisions or other activities. They are also advisors of α wolves and the best candidates to be the alpha.
3. The deltas (δ) are the third level in wolves group and submit information to alphas and betas, however, they dominate the omega. They usually responsible for watching the boundaries of territory, protecting and caring of the pack as well as hunting.
4. The omegas (ω) are the lowest level of dominance in the pack and the last wolves allowed to eat. They are considered as followers.

The principal steps executed in GWO are searching, encircling, hunting, and attacking the prey. These major steps in a mathematical formulation of GWO, presented in [8], are described as follows:

1. *Social hierarchy of GWO:*

According to the level of dominance in the pack, there are four kinds of solutions in the mathematical model for the GWO. The best and fittest solution is called the alpha (α). Beta (β) and delta (δ) represent the second and third best solutions respectively, and remaining solutions are known as ω .

2. *Encircling behavior of grey wolves:*

Initially, the GWs encircle the prey before hunting. The following equations represent the encircling behavior:

$$\vec{D} = \left| \vec{C} \cdot \vec{X}_p - \vec{X}(t) \right|, \quad (1)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}, \quad (2)$$

where \vec{D} represents the distance between grey wolf and prey, t indicates the current iteration number, \vec{A} and \vec{C} are coefficient vectors, \vec{X}_p represents the position vector of the prey, \vec{X} denotes the position vector of a grey wolf.

The vectors \vec{A} and \vec{C} are calculated as follows:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \quad (3)$$

$$\vec{C} = 2 \cdot \vec{r}_2 \quad (4)$$

where the value of \vec{a} is linearly reduced from 2 to 0 during the iterations, and \vec{r}_1, \vec{r}_2 are arbitrary vectors between $[0,1]$.

3. *Hunting prey:*

Usually, the alpha guides the hunt and the beta and delta occasionally join in hunting. To mathematically simulate the hunting behavior of grey wolves, the alpha (best solution) beta (the second best solution), and delta (the third best solution) are supposed to have better information regarding the potential position of the prey (optimum). Therefore, the first three best solutions determined so far are saved and oblige the other search

agents (including the omegas) to update their positions according to the position of the best search agents, α . The hunting behavior is mathematically formulated as follows [8]:

The updated position of α , β , and δ is as in Eq. (5):

$$\overline{X}(t+1) = \frac{\overline{X}_1(t) + \overline{X}_2(t) + \overline{X}_3(t)}{3} \quad (5)$$

where \overline{X}_1 , \overline{X}_2 , \overline{X}_3 are defined as in Eqs. (6), (7), and (8) respectively.

$$\overline{X}_1 = \overline{X}_\alpha - \overline{A}_1 \cdot (\overline{D}_\alpha), \quad (6)$$

$$\overline{X}_2 = \overline{X}_\beta - \overline{A}_2 \cdot (\overline{D}_\beta), \quad (7)$$

$$\overline{X}_3 = \overline{X}_\delta - \overline{A}_3 \cdot (\overline{D}_\delta), \quad (8)$$

where \overline{X}_α , \overline{X}_β , \overline{X}_δ are the first three best solutions in the population at iteration t , \overline{A}_1 , \overline{A}_2 , \overline{A}_3 are defined as in Eq. (3), and \overline{D}_α , \overline{D}_β , \overline{D}_δ are defined using Eqs. (9), (10), and (11) respectively.

$$\overline{D}_\alpha = \left| \overline{C}_1 \cdot \overline{X}_\alpha - \overline{X} \right|, \quad (9)$$

$$\overline{D}_\beta = \left| \overline{C}_2 \cdot \overline{X}_\beta - \overline{X} \right|, \quad (10)$$

$$\overline{D}_\delta = \left| \overline{C}_3 \cdot \overline{X}_\delta - \overline{X} \right|, \quad (11)$$

Where \overline{C}_1 , \overline{C}_2 , \overline{C}_3 are defined as in Eq. (4).

It is clear from above discussion that the search process initializes with generating a random population of GWs in the GWO mechanism. During the iterations, α , β , and δ wolves evaluate the probable position of the prey. Each candidate solution updates its distance from the prey.

4. Attacking the prey and search for prey (exploitation and Exploration):

Grey wolves attack the prey when it stops moving, then the hunting process is over. This process can result in the global optima; that is the exploitation. It can be understood from Eqs. (6) to (8) that vector \overline{A} forces GWO to exploit. This process can be modeled mathematically by decreasing the value of \overline{a} . Since the value of a is diminished from 2 to 0, \overline{A} is also diminished. In other words, \overline{A} is a random value in the range of $[-2a, 2a]$. When $|\overline{A}| < 1$, the grey wolves are compelled to attack the prey. On the other hand, the search for prey by the grey wolves is the ability of them in searching for various positions of the prey. That is, the search for prey is the exploration ability. The random values of \overline{A} are utilized to oblige the

search agent to diverge from the prey. For $|\vec{A}| < 1$, the grey wolves are compelled to diverge from the prey [22].

3.2. Proposed Hybrid Chaotic Grey Wolf Optimizer (HCGWO)

GWO algorithm is considered as the base algorithm in the proposed method due to its convergence power. There are two common aspects in most meta-heuristic algorithms: *exploration* and *exploitation*. The exploration is the ability of expanding search space, where the exploitation is the ability of finding the optimum around a good solution. In earlier iterations, a meta-heuristic algorithm explores the search space in order to find better solutions. To explore the search space thoroughly and to avoid getting trapped in a local optimum, the algorithm must use the exploration in the first few iterations. Hence, effective exploration is an important part in a meta-heuristic algorithm. Over time exploration fades out and exploitation fades in, so the algorithm can find an optimum around a good solution. Finding a suitable tradeoff between exploration and exploitation is critical. Even though almost all meta-heuristic algorithms employ the exploration and exploitation aspects but their approaches might differ. Hence proposing a new approach is desirable.

In GWO these two phases are controlled using the parameter \vec{a} . As mentioned in the previous section this parameter is decreased linearly. In earlier iterations of the algorithm, exploration is performed and in the later iterations of the algorithm, exploitation is carried out. By changing the linear behavior of \vec{a} , we can change the emphasis on exploration and exploitation and come up with a balanced approach between these two stages. Our new control parameter is as followed:

$$\vec{a}(t) = 2 - 2 \left(\frac{t}{t_{\max}} \right)^k \quad (12)$$

Where t indicated the current iteration, t_{\max} is the total number of iterations and k is a constant. Here the parameter \vec{a} is still decreased from 2 to 0 but in a non-linear manner. For the values of k between 0 and 1, the emphasis will be on exploitation, however adequate searching of the space might suffer. For values of more than 1, search space will be thoroughly explored and then the algorithm moves on to exploitation. A suitable value for k should be found by trial and error.

We expect to improve GWO's performance by the non-linear reduction of the control parameter, however, there is still room for further improvement. Another improvement that can be considered for GWO is hybridization. The main objective of the proposed method is to improve GWO by means of hybridization and adding more operators to enhance the exploratory capability [23]. For this reason, the proposed task scheduling method is composed of grey wolf optimizer and evolutionary algorithm (EA) [24]; as a result, crossover and mutation are added to GWO for better performance. For instance, the mutation and crossover operators from differential evolution (DE) are two effective operators to optimally explore and exploit a solution space. The DE-type mutation creates the self-adaptive characteristic of the solution population in DE known as contour matching [25] which refers to automatically explore the most promising area of the solution space [23].

The mutation is one of the most important operators in evolutionary algorithms such as DE. In this paper, the Michalewicz's nonuniform mutation [26] is used for GWO. This is obtained by Eq. (13):

$$X_i^{t+1} = \begin{cases} X_{gBest}^t + \Delta(t, X_{gBest}^t - X_i^t), & \text{if } rand_i \leq M_r \\ X_{gBest}^t - \Delta(t, X_i^t - X_{worst}^t), & \text{otherwise} \end{cases} \quad (13)$$

$$(14)$$

$$M_r = 0.05 \times \hat{F}_{i,best}$$

where t is current iteration and $rand_i$ is a random number in $[0,1]$ with uniform distribution. X_{gBest}^t and X_{worst}^t are the best and worst wolves ($i \neq gBest \neq worst$), respectively. The mutation probability M_r controls the

adaptive mutation process where $\hat{F}^{i,best} = F^i - F^{best}$, F^i is the fitness value of the i th wolf and F^{best} is the best wolf in the current iteration. The function $\Delta(t, X)$ takes a value in the interval $(0, w)$ and can be [27]calculated using Eq. 15.

$$\Delta(t, w) = w \left(1 - z^{\left(\frac{1-t}{T} \right)} \right) \quad (15)$$

where T is the maximum number of iterations, z is a random number in the $[0,1]$ with uniform distribution. In the initial generations with emphasize to exploration nonuniform mutation tends to search the space uniformly and for tuning the solution in the later generations it tends to search the space locally, that is, closer to its descendants.

The crossover mechanism is based on uniform or binomial crossover. A new crossover vector from the mutated vector $X_i^{t+1} = (X_{i1}^{t+1}, X_{i2}^{t+1}, \dots, X_{in}^{t+1})$ and the target vector $X_i^t = (X_{i1}^t, X_{i2}^t, \dots, X_{in}^t)$ is produced by Uniform or binomial crossover using Eq. (16).

$$U_{ij}^t = \begin{cases} X_{ij}^{t+1}, & \text{if } R_j \leq C_r \text{ or } j = \text{random}(i) \\ X_{ij}^t, & \text{if } R_j > C_r \text{ or } j \neq \text{random}(i) \end{cases} \quad (16)$$

where $j \in [0,1, \dots, D]$, $i \in [0,1, \dots, N_w]$, N_w is the number of wolves. $R_j \in [0,1)$ is the j th realization of a uniform random generator number. Unlike classical evolutionary algorithm (EA) and differential evolution (DE), the crossover probability C_r is not a constant and is defined using Eq. (17) [27],

$$C_r = 0.2 \times \hat{F}^{i,best} \quad (17)$$

$$\hat{F}^{i,best} = \frac{F^i - F^{best}}{F^{worst} - F^{best}}, \quad \forall i = 1, 2, \dots, N_w \quad (18)$$

where N_w is the number of grey wolves. F^{best} and F^{worst} are the best and the worst grey wolves in the current iteration, respectively, and F^i is the fitness or objective function value of the i th wolf.

At the end, the greedy strategy from differential evolution (DE) [28] is used for selection mechanism and is done using Eq. (19),

$$X_i^{t+1} = \begin{cases} U_{ij}^t, & \text{if } f(U_{ij}^t) \leq f(X_{ij}^t) \text{ and } \text{rand}_i < \rho \\ X_{ij}^t, & \text{otherwise} \end{cases} \quad (19)$$

where rand_i and ρ are random numbers in the interval $(0,1)$, $f(X_{ij}^t)$ represents the fitness of the last position, and $f(U_{ij}^t)$ is the new position obtained by Eq. (5). The value of ρ is randomly generated inside interval $(0,1]$ in each iteration. This greedy strategy ensures “survival of the fittest” which means the update is accepted only if a better objective is achieved.

4. Problem Definition with Solution Framework

This section provides the systems model of the proposed HCGWO algorithm for task scheduling in cloud computing.

4.1. The system model

Hybrid Chaotic Grey Wolf Optimizer for Task Scheduling in Cloud Computing

The cloud computing provides hardware infrastructure and software programs as on-demand high performance computing services as self-service access using virtualization techniques over the Internet or dedicated network, with delivery on demand, and payment based on usage. The low level cloud computing services range from full applications and development platforms to cloud servers, data storage, and virtual desktops [29]. The system model of task scheduling mechanism in the cloud computing is shown in Fig. 2. It includes cloud users who submit tasks, cloud interface called cloud broker, task scheduler and resource manager [30]. The resources contain one or more physical machines (PM) with one or more virtual machines (VMs), sharing memory, CPU, and storage. Each task is mapped to suitable VM according to the resource requirement such as CPU, memory, storage. The main steps of the working flow of this model are as follows:

- After interacting with the cloud infrastructure, users can access the resources through cloud broker which is responsible for authentication and uthorization. The users submit the task requests to the task manager. The task manager manages the database to store every user request.
- The Task Scheduler is major component, which act as a middleware between clients and resource manager to process the client request. It further takes all the information regarding the Task requirements posted by the clients and proceeds further for resource allocation using the inbuilt scheduling algorithm.
- Finally, the resource manager plays an important role which acts as a resource information system and provides services to resource scheduler in order to allocate the user tasks in best possible VM. In order to balance the load of each host machines, resource manager has an in built resource monitor which keeps tracks of the entire loads and tackles the unbalanced condition.

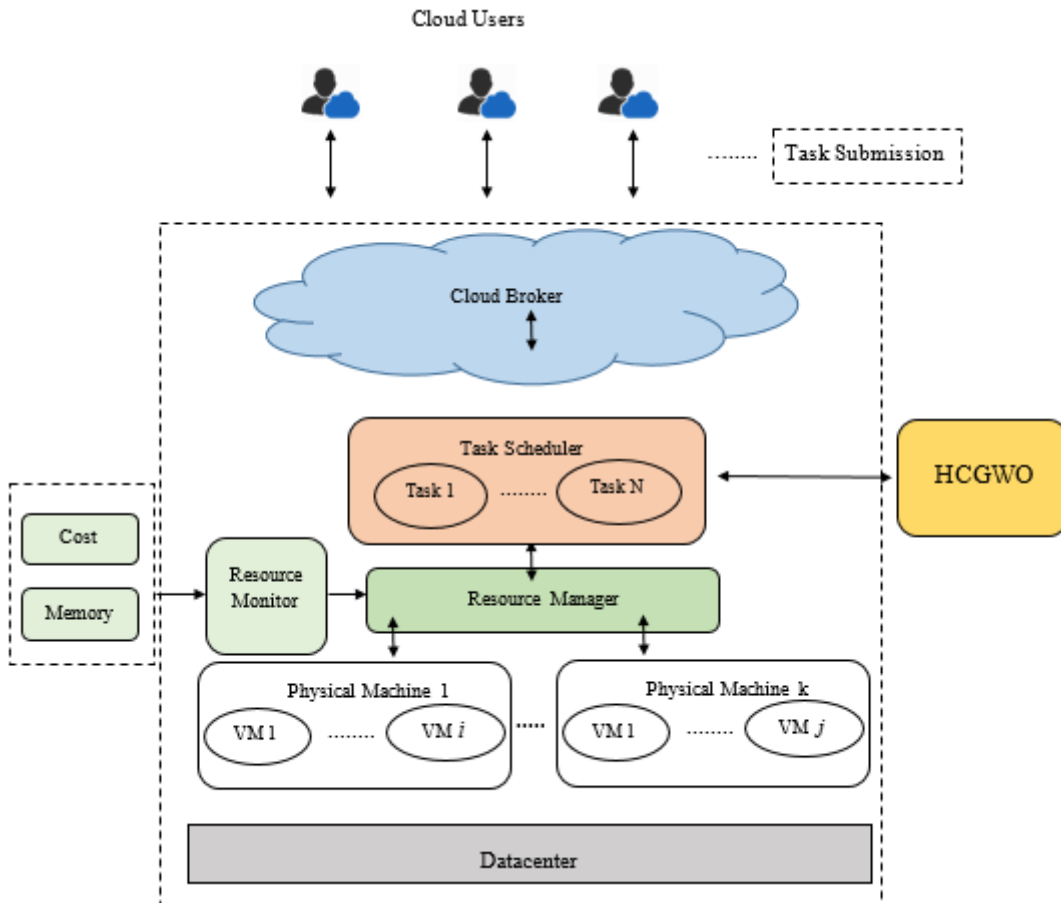


Fig. 2. System model of proposed task scheduling mechanism in the cloud computing.

4.2. The Task Scheduling Problem

The main purpose of task scheduling is to determine the allocation of tasks of a given request to a given virtual machine (VM) of the cloud service provider in order to minimize fitness function. The cloud scheduler is responsible to find the optimal resource i.e. cloud virtual machines (VMs) for the submitted tasks.

In this regard, the task scheduling problem in cloud computing is one of important aspects of design since the overall performance of the cloud computing provider's in terms of quality of service (QoS) depends on it. It is important to consider the part QoS plays in any task scheduler design. Most popular QoS factors of task scheduler design include execution time, transmission time, latency, resource utilization and makespan. Two QoS parameters including makespan and total cost are considered as two main objective of our task scheduling algorithm which have to optimized for better performance for the cloud providers and clients.

To model our problem mathematically, it should be noted that each given task contains various autonomous subtasks. Each task should be assigned to one virtual machine. Let us consider the cloud, C , consists of s physical machine, which can be represented using Eq. (20).

Cloud C : (20)

$$PM = \{PM_1, \dots, PM_s\}$$

where C represents the cloud and PM represents a set of physical machines presented in the cloud. For each physical machine, a set of virtual machines are available which can be represented by Eq. (21),

$$VM = \{VM_1, \dots, VM_j, \dots, VM_r\} \quad (21)$$

where VM represents a set of virtual machines presented in the physical machine PM_i . Each virtual machine has the central processing unit (CPU) and the memory. The user tasks can be represented by Eq. (22),

$$T = \{T_1, \dots, T_n\} \quad (22)$$

where T is the set of tasks, and each task contains a task set $T_i = \{t_1, \dots, t_k, \dots, t_m\}$. Each subtask t_k in T_i needs to be allocated to the corresponding virtual machine with minimum makespan and cost. The parameters used in the proposed method are given in Table 1. Each incoming task to the task scheduler contains different characteristics such as memory requirement, cost, and deadline.

4.3. Proposed task scheduling method based on HCGWO

The proposed task scheduling method for scheduling tasks to the virtual machines in the cloud computing is discussed in this section. The primary goal of the proposed method is to schedule user tasks efficiently using an improved version of grey wolf optimizer. Our task scheduling method is based on the multi-objective model [31] and grey wolf optimizer. The block diagram of the proposed task scheduling method is shown in Fig. 3.

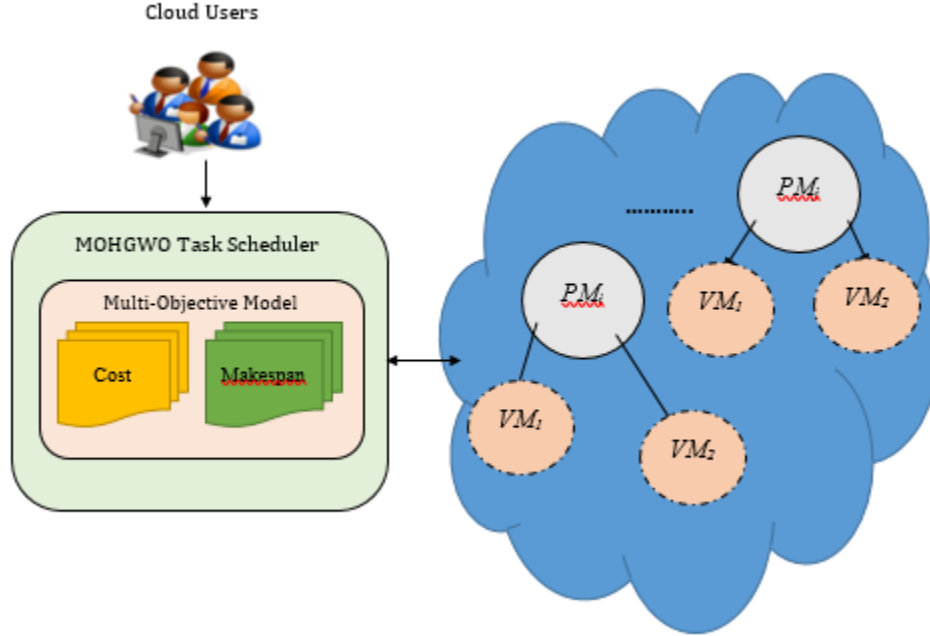


Fig. 3. Block diagram of the proposed HCGWO for task scheduling.

Step 1: Solution encoding

The first important process for task scheduling is solution encoding. Solution encoding represents the allocation of all tasks on virtual machines in task scheduling process. Each solution in the proposed method consists of a set of tasks and virtual machines. Depending on virtual machine's configuration such as capacity, tasks can be scheduled in order to minimize the cost and makespan. Assume that, there are four tasks (T_1, T_2, T_3, T_4) and each of them consists of two subtasks (t_1, t_2), then we have totally 8 subtasks. Also, we have two physical machines (PM_1, PM_2) that PM_1 has three virtual machines (VM_1, VM_2, VM_3) and PM_2 has two virtual machines (VM_4, VM_5). The initial solution, which is generated randomly, is given in Table 2. As it can be seen in Table 1, task t_1 and t_4 are allocated to VM_1 , task t_2 and t_7 are allocated to VM_2 , task t_5 is allocated to VM_3 , task t_3 and t_6 are allocated to VM_4 , and finally task t_8 is allocated to VM_5 .

Table 1: Initial solution encoding.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
VM_1	1	0	0	1	0	0	0	0
VM_2	0	1	0	0	0	0	1	0
VM_3	0	0	0	0	1	0	0	0
VM_4	0	0	1	0	0	1	0	0
VM_5	0	0	0	0	0	0	0	1

Step 2: Multi-objective model

To formulate the objective function of the proposed algorithm, control parameters are defined. The multi-objective model includes four control parameters. The problem can be formulated by using Eq. (23).

$$fitness = \min [c_1 \times C_{total} + c_2 \times M] \quad (23)$$

Where C_{total} is the total cost including the cost function of CPU, memory and migration, and M is the makespan. c_1 and c_2 are control parameters with the range of $[0,1]$. All these control parameters are assumed as 1 in this work for simplicity.

The total cost, C_{total} , is defined by using Eq. (24),

$$C_{total} = C_{CPU} + C_{Memory} + C_{Migration} \quad (24)$$

where C_{CPU} is the CPU cost of the virtual machine, C_{Memory} is the memory cost of the virtual machine, and $C_{Migration}$ is migration cost. C_{CPU} can be calculated using Eq. (25),

$$C_{CPU} = \sum_{j=1}^{|VM|} C_{CPU}(j) \quad (25)$$

where $C_{CPU}(j)$ represents the CPU cost of the virtual machine V_j , $|VM|$ is the total number of virtual machines. Then, $C_{CPU}(j)$ can be calculated using Eq. (26) by considering the expected time to complete (ETC_{ij}) of i th task (T_i) on j th virtual machine (VM_j) and the CPU cost of virtual machine ($VM_j^{CPU_Cost}$) for unit time,

$$C_{CPU}(i) = ECT_{ij} \times VM_j^{CPU_Cost} \quad (26)$$

The memory cost of the virtual machine V_j is calculated using Eq. (27) by considering the ETC_{ij} and the memory cost of virtual machine ($VM_j^{Memory_Cost}$) for unit time,

$$C_{CPU}(i) = ECT_{ij} \times VM_j^{Memory_Cost} \quad (27)$$

The migration cost $C_{Migration}$ consists of movement factor (M^F) and cost factor (C^F), and is calculated using Eq. (28) [32],

$$C_{migration} = \frac{M^F + C^F}{2} \quad (28)$$

$$M^F = \frac{1}{PM} \left[\sum_{j=1}^{|VM|} \left(\frac{\text{Number of movements}}{\text{Total VM}} \right) \right] \quad (29)$$

$$C^F = \sum_{j=1}^{|VM|} \left(\frac{\text{Cost to run} \times \text{Memory of task}}{VM \times PM} \right) \quad (30)$$

The second parameter of fitness function is the total makspan (M) which is the completion time of all task. Consider M_{ij} is the completion time of task T_i , $i = \{1, \dots, n\}$, on virtual machine VM_j , $j = \{1, \dots, r\}$, and initially can be calculated using Eq. (31),

$$M_{ij} = ECT_{ij} + W_i \quad (31)$$

where ETC_{ij} represents the expected time to complete of i th task (T_i) on j th virtual machine (VM_j) and W_i represents the waiting time of of i th task (T_i) on VM_j . Then, the total makspan (M) can be calculated by Eq. (32),

$$M = \sum_{i=1}^n M_{ij} \quad (32)$$

Algorithm 1: Pseudo-code of HCGWO algorithm for Task Scheduling

Input: User Tasks T , Virtual Machines V
 Parameters of HCGWO Algorithm
Output: The tasks are scheduled on the virtual machines

- 1: Begin
- 2: Randomly generate the initial population of wolves;
- 3: Evaluate the fitness of each wolf;
- 4: Set X_α to be the best wolf;
- 5: Set X_β to be the second best wolf;
- 6: Set X_δ to be the third best wolf;
- 7: **while** (Current_Iteration < Max_Iterations) **do**
- 8: **for** each wolf **do**
- 9: Update the position using Eq. 5;
- 10: Perform Greedy Search strategy using Eq. 19;
- 11: **End for**
- 12: Calculate the mutation and crossover probability of each individual;
- 13: **for** each wolf **do**
- 14: **if** rand(0,1) < C_r **then**
- 15: Perform crossover using Eq. 16;
- 16: **End if**
- 17: **if** rand(0,1) < M_r **then**
- 18: Perform mutation using Eq. 13;
- 19: **End if**
- 20: **End for**
- 21: Update a , A , and C using Eq. (12), (3), and (4) respectively;
- 22: Calculate the fitness of wolves;
- 23: Update X_α , X_β and X_δ
- 24: Current_Iteration = Current_Iteration + 1;
- 25: **End while**
- 26: Return X_α ;

5. Result and Discussion

This section analyzes the efficiency of presents the experimental results of Hybrid Chaotic grey wolf optimizer for task scheduling in cloud environment and presents the experimental results and the comparative discussion of the proposed method with other scheduling methods.

5.1. Experimental Setup

To illustrate the effectiveness of the proposed method, we have used a personal computer with Intel Core i7 processor and 6GB memory running a 64-bit version of Windows 10 operating system. The proposed method is implemented on the CloudSim Cloud Simulator [33]. Cloudsim is a highly extensible simulation framework that allows modeling and simulating of Cloud computing infrastructures.

5.2. Performance Metric

To evaluate the performance of the proposed method, makespan and cost are used as the evaluation metrics. As a recall, they are described as follows:

- *Makespan*: The makespan represents the total length of the schedule that is the total time required for executing all the tasks.
- *Cost*: The cost represents the total cost required for scheduling tasks to virtual machines.

5.3. Experimental Results

In this section, the proposed method is evaluated based on makespan and cost. Here, at first, we assign the N number of task and M number of resources to schedule the task based on the makespan, the cost and fitness function. In this work, we have taken two set of configurations. Then the performance analysis is taken based on two sub divisions such as (1) Performance analysis with the population size of 10 and 25 iterations, (2) Performance analysis with the population size of 20 and 50 iterations. Simulation parameters are presented in Table 2.

Table 2. Simulation Parameters

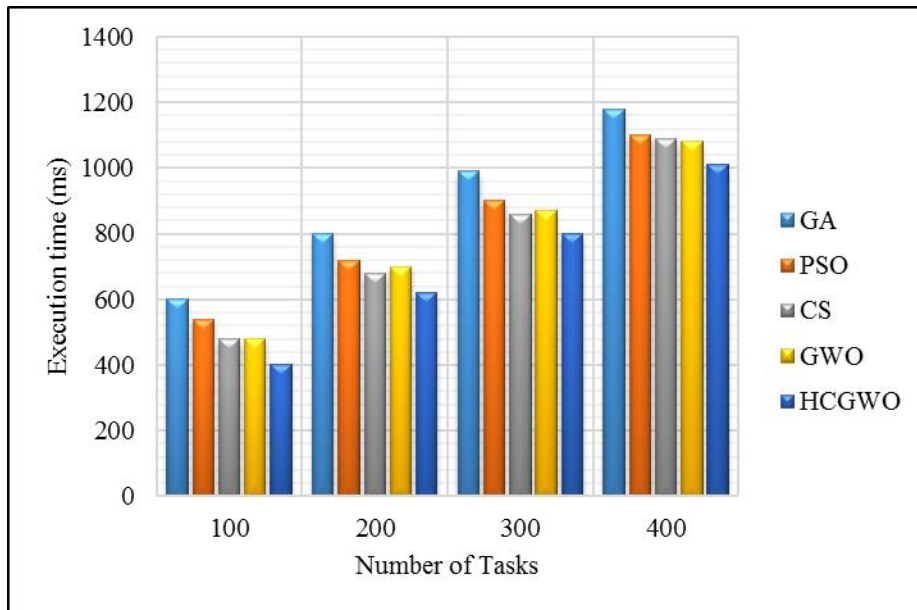
Number of datacenter	2
Number of host in datacenter	15
Number virtual machines	50
Number of tasks	100-400
Length of task	5000-10000 MI
MIPS of processing element	1000-2000 (MIPS)
Number of PE per VM	5-10
Number of Pes requirement	2-5
Bandwidth	500-1000 (bit)
RAM	512-2048 MB

5.3.1 Performance Analysis on makespan

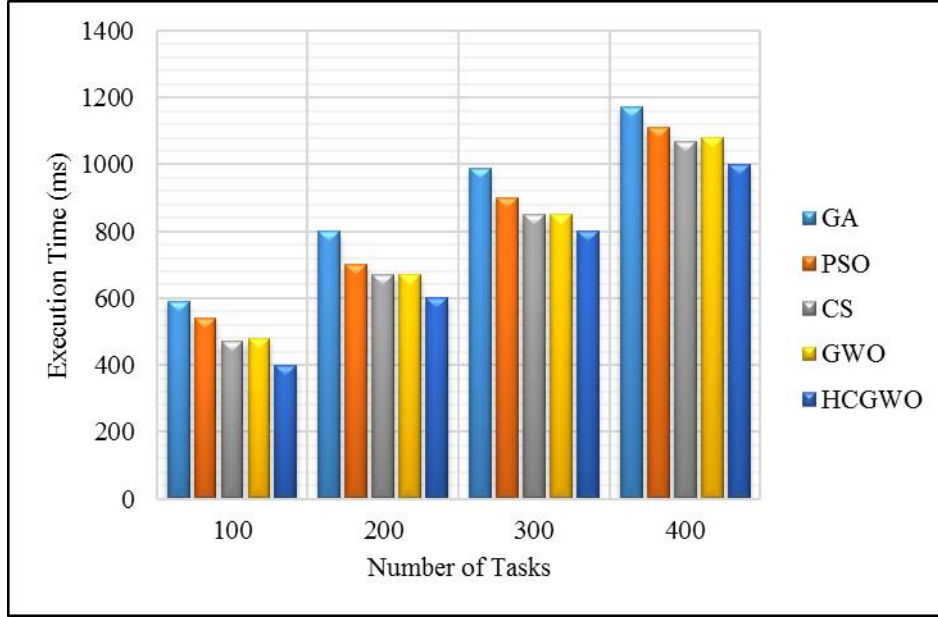
At first, the proposed method is evaluated using the population size of 10 and 25 iterations. Here, we compare our proposed HCGWO algorithm based multi-objective scheduling with Genetic Algorithm (GA), Particle Swarm Optimization, Cuckoo Search (CS), and Grey wolf Optimizer (GWO) based scheduling algorithm. Fig. 4 represents

Hybrid Chaotic Grey Wolf Optimizer for Task Scheduling in Cloud Computing

the makespan of the proposed method along with other methods for this experiment. The makespan of the proposed HCGWO and other algorithms on 25th iteration for population size of 10 is shown in Fig. 4a. As it can be seen, the performance of the proposed method based on makespan is the minimum in all cases. When the number of tasks is 100, the makespan of the proposed HCGWO is 400 (ms) while it is 600 for using GA, 540 for using PSO, 480 for using CS, and 480 for using GWO. The average makespan improvement of proposed HCGWO is 23.15% in comparison with GA, PSO, CS, and GWO when the number of tasks is 100. When the number of tasks is 200, the makespan of the proposed HCGWO is 620 (ms) while it is 720 for using GA, 680 for using PSO, 700 for using CS, and 620 for using GWO. The average makespan improvement of proposed HCGWO is 14.16% in comparison with GA, PSO, CS, and GWO when the number of tasks is 200. When the number of tasks is 300, the makespan of the proposed HCGWO is 800 (ms) while it is 990 for using GA, 900 for using PSO, 680 for using CS, and 700 for using GWO. The average makespan improvement of proposed HCGWO is 11.33% in comparison with GA, PSO, CS, and GWO when the number of tasks is 300. When the number of tasks is 400, the makespan of the proposed HCGWO is 1010 (ms) while it is 1180 for using GA, 1100 for using PSO, 1090 for using CS, and 1080 for using GWO. The average makespan improvement of proposed HCGWO is 9.10% in comparison with GA, PSO, CS, and GWO when the number of tasks is 400. Fig. 4b represents the makespan of the proposed method and other methods on 50th iteration with the population size of 20. In this experiment, the performance of the proposed method based on makespan is the minimum in all cases, although the average improvement is less than the last experiment. With the increase of population size and iteration, the performance of all methods improves. When the number of tasks is 100, the makespan of the proposed HCGWO is 400 (ms) while it is 590 for using GA, 540 for using PSO, 470 for using CS, and 480 for using GWO. The average makespan improvement of proposed HCGWO is 22.42% in comparison with GA, PSO, CS, and GWO when the number of tasks is 100. When the number of tasks is 200, the makespan of the proposed HCGWO is 600 (ms) while it is 800 for using GA, 700 for using PSO, 670 for using CS, and 670 for using GWO. The average makespan improvement of proposed HCGWO is 15.04% in comparison with GA, PSO, CS, and GWO when the number of tasks is 200. When the number of tasks is 300, the makespan of the proposed HCGWO is 800 (ms) while it is 990 for using GA, 900 for using PSO, 850 for using CS, and 850 for using GWO. The average makespan improvement of proposed HCGWO is 10.51% in comparison with GA, PSO, CS, and GWO when the number of tasks is 300. When the number of tasks is 400, the makespan of the proposed HCGWO is 1000 (ms) while it is 1170 for using GA, 1110 for using PSO, 1070 for using CS, and 1080 for using GWO. The average makespan improvement of proposed HCGWO is 9.10% in comparison with GA, PSO, CS, and GWO when the number of tasks is 400. In overall, the proposed HCGWO can achieve 14.41% improvement for minimizing the makespan, compared to GA, PSO, CS, and GWO.



(a)



(B)

Fig. 4. Illustration of the makespan of the proposed HCGWO and other algorithms. (a) Makespan of the proposed HCGWO and other algorithms for the population size of 10 on iteration 25. (b) Makespan of the proposed HCGWO and other algorithms for the population size of 20 on iteration 50.

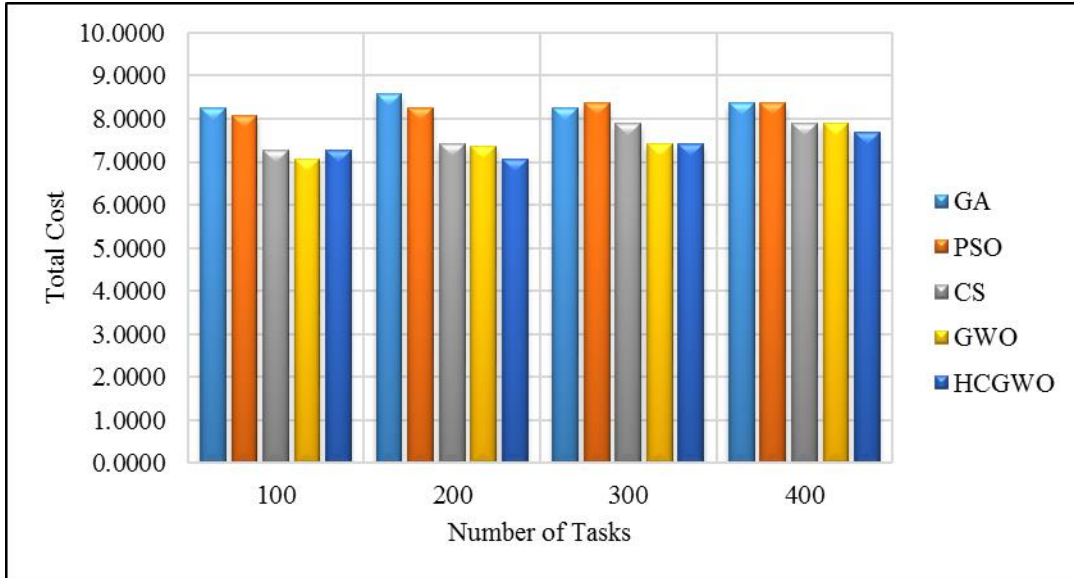
5.3.2 Performance Analysis on cost

Fig. 5 shows the performance analysis of proposed method based on the total cost. Here, we analyzed how much amount of cost needed for task scheduling. Fig. 5a represents the total cost of the proposed HCGWO and other methods for the population size of 10 on iteration 25. Our proposed approach achieves the minimum cost for the population size of 10 on iteration 25 in all cases except when the number of tasks is 100. When the number of tasks is 100, the total cost of proposed HCGWO is 7.2633 while it is 8.2633 for GA, 8.0667 for PSO, 7.2633 for CS, and 7.07 for GWO. The average cost improvement of proposed HCGWO is 4.83% in comparison with GA, PSO, CS, and GWO when the number of tasks is 100. When the number of tasks is 200, the total cost of proposed HCGWO is 7.0667 while it is 8.5733 for GA, 8.2633 for PSO, 7.4167 for CS, and 7.3670 for GWO. The average cost improvement of proposed HCGWO is 10.21% in comparison with GA, PSO, CS, and GWO when the number of tasks is 200. When the number of tasks is 300, the total cost of proposed HCGWO is 7.4167 while it is 8.2633 for GA, 8.3670 for PSO, 7.8850 for CS, and 7.4167 for GWO. The average cost improvement of proposed HCGWO is 6.88% in comparison with GA, PSO, CS, and GWO when the number of tasks is 300. When the number of tasks is 400, the total cost of proposed HCGWO is 7.6950 while it is 8.3667 for GA, 8.3667 for PSO, 7.8850 for CS, and 7.8850 for GWO. The average cost improvement of proposed HCGWO is 9.26% in comparison with GA, PSO, CS, and GWO when the number of tasks is 400. Fig. 5b represents the total cost of the proposed method and other methods on 50th iteration with the population size of 20. In this experiment, the performance of the proposed method based on total cost is the minimum in all cases, and the average improvement is more than the last experiment. When the number of tasks is 100, the total cost of the proposed HCGWO 7.07 while it is 8.2633 for using GA, 8.2633 for using PSO, 7.2633 for using CS, and 7.07 for using GWO. The cost improvement of proposed HCGWO is 7.88% in comparison with GA, PSO, CS, and GWO when the number of tasks is 100. When the number of tasks is 200, the total cost of the proposed HCGWO 6.885 while it is 8.367 for using GA, 8.367 for using PSO, 7.367 for using CS, and 7.167 for using GWO. The cost improvement of proposed HCGWO is 11.47% in comparison with GA, PSO, CS, and GWO when the number of tasks is 200. When the number of tasks is 300, the total cost of the proposed HCGWO 7.07 while it is 8.5733 for using GA, 8.4167 for using PSO, 7.6667 for using CS, and 7.367 for using GWO. The cost improvement of proposed HCGWO is 11.33% in comparison with GA, PSO, CS, and GWO when the number of tasks is 300. When the number of tasks is 400, the total cost of the proposed HCGWO 7.2633 while it is 8.5733 for using GA, 8.0067 for using PSO, 7.5732 for using CS, and 7.4167 for using GWO. The cost improvement of proposed HCGWO is 7.68% in comparison with GA, PSO, CS, and GWO when the number of

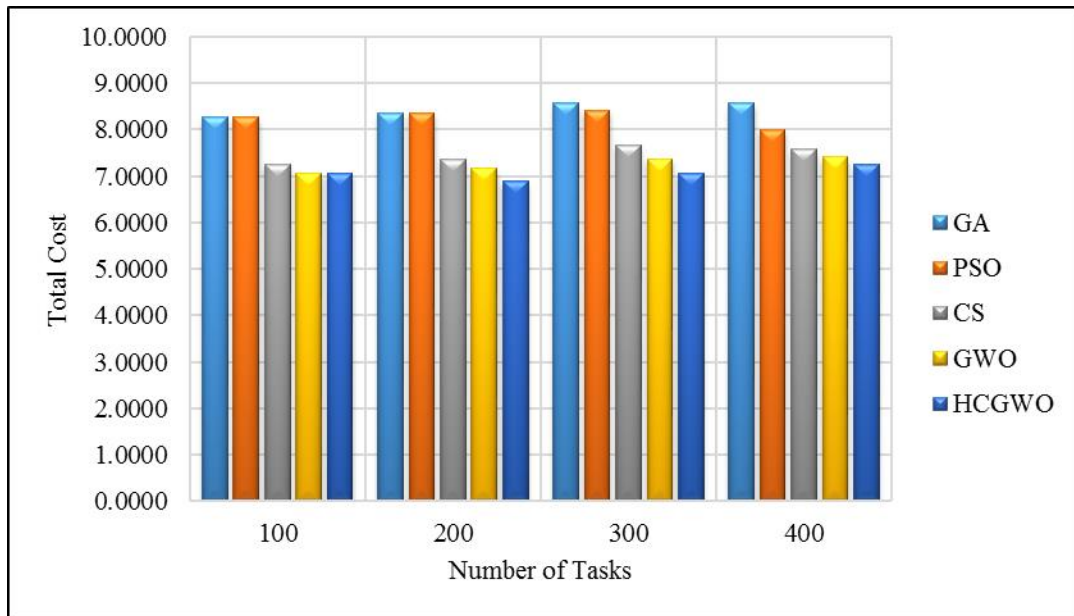
Hybrid Chaotic Grey Wolf Optimizer for Task Scheduling in Cloud Computing

tasks is 400. In overall, the proposed HCGWO can achieve 8.69% improvement for minimizing the total cost, compared to GA, PSO, CS, and GWO.

In addition, Fig. 6 shows the convergence of the proposed task scheduling method based on fitness function against other optimization methods for the population size of 10 and 20. The number of tasks used in Fig. 6a and Fig. 6b are 100 and 400, respectively. Here we obtain minimum fitness value, compared with existing task scheduling methods. Minimum fitness function gives the high profit. When analyzing Figs. 4, 5, and 6, it is clear that our proposed approach achieves the better results, compared to other approaches.

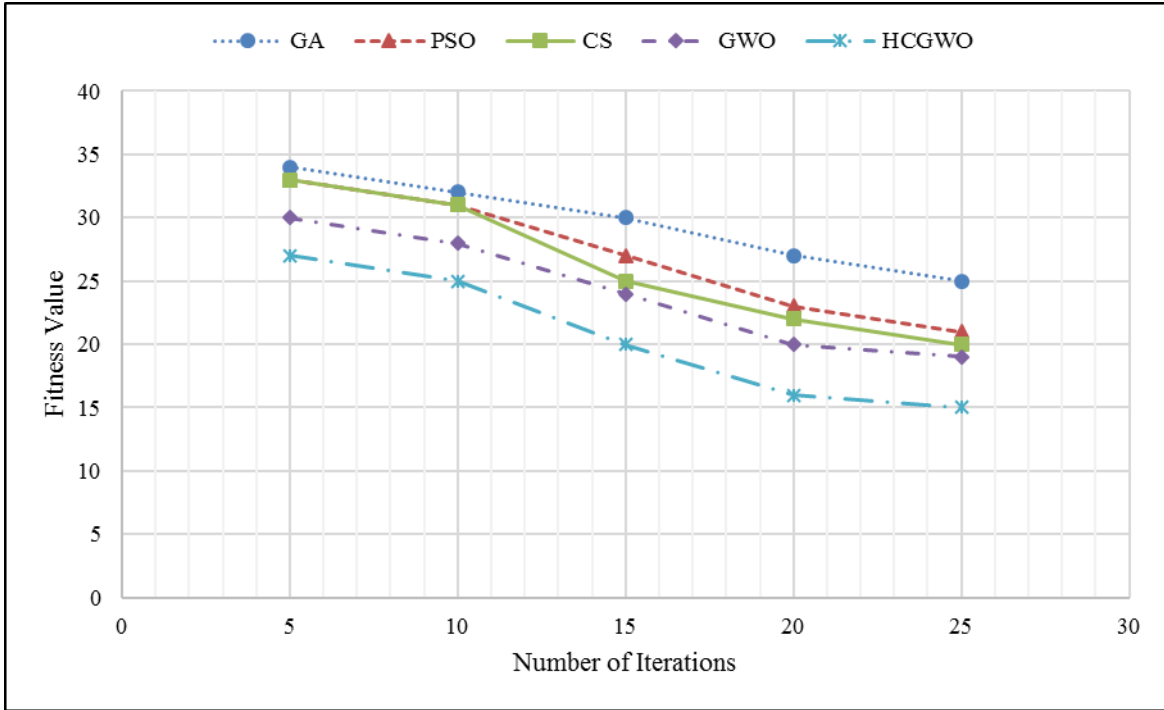


(a)

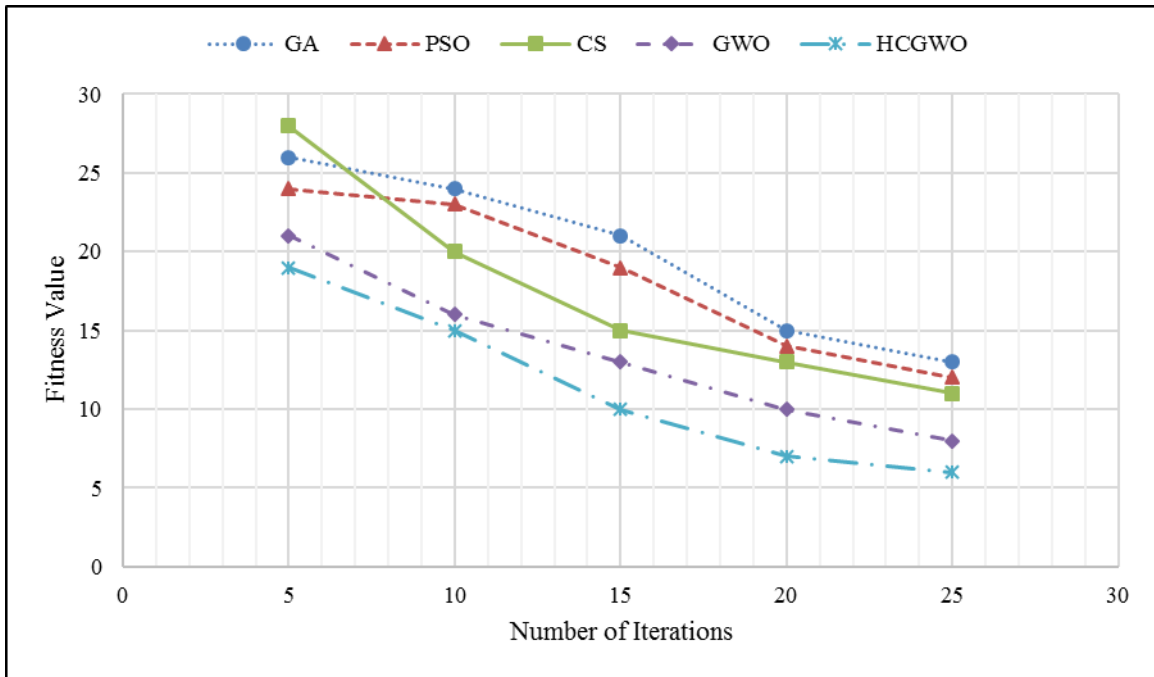


(b)

Fig. 5. Illustration of the total cost of the proposed HCGWO and other algorithms. (a) Total cost of the proposed HCGWO and other algorithms for the population size of 10 on iteration 25. (b) Total cost of the proposed HCGWO and other algorithms for the population size of 20 on iteration 50.



(a)



(b)

Fig. 6. Illustration of the fitness of the proposed HCGWO and other algorithms. **(a)** Fitness of the proposed HCGWO and other algorithms for the population size of 10 (Tasks=100) **(b)** Fitness of the proposed HCGWO and other algorithms for the population size of 20 (Tasks=400).

6. Conclusion

In this paper, a multi-objective task scheduling method was proposed based on the Hybrid Chaotic grey wolf optimizer (HCGWO). In order to improve the performance of grey wolf optimizer, crossover and mutation operators from evolutionary algorithm and differential evolution are added to GWO. In addition, a greedy strategy and chaos theory are used to improve our hybrid GWO. Then, the multi-objective optimization approach is used to improve the scheduling performance, compared to single objective function. The multi-objective function consists of two important performance metrics such as makespan and cost. Finally, the fitness value is calculated by adding the makespan and the cost function. In HCGWO, each dimension of a solution represents a task and a solution as a whole representation of all tasks priorities. Two configurations with different simulation setups are used to evaluate the proposed method. The experimental results show that our proposed multiobjective-based task scheduling is better than other approaches. The results obtained from HCGWO are better than the existing GA, PSO, CS, and GWO based task scheduling methods, and experimental results demonstrated that the HCGWO based scheduling approach was well suited for task scheduling problem.

References:

1. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems* **25**(6), 599-616 (2009).
2. Lim, J., Suh, T., Gil, J., Yu, H.: Scalable and leaderless Byzantine consensus in cloud computing environments. *Information Systems Frontiers* **16**(1), 19-34 (2014).
3. Hameed, A., Khoshkbarforousha, A., Ranjan, R., Jayaraman, P.P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q.M., Tziritas, N., Vishnu, A.: A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing* **98**(7), 751-774 (2016).
4. Muthulakshmi, B., Somasundaram, K.: A hybrid ABC-SA based optimized scheduling and resource allocation for cloud environment. *Cluster Computing*, 1-9 (2017).
5. Valarmathi, R., Sheela, T.: Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing. *Cluster Computing*, 1-14 (2017).
6. Singh, P., Dutta, M., Aggarwal, N.: A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowledge and Information Systems* **52**(1), 1-51 (2017).
7. Kansal, N.J., Chana, I.: Existing load balancing techniques in cloud computing: A systematic review. *Journal of Information Systems and Communication* **3**(1), 87 (2012).
8. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. *Advances in engineering software* **69**, 46-61 (2014).
9. Nasr, A.A., Chronopoulos, A.T., El-Bahnasawy, N.A., Attiya, G., El-Sayed, A.: A novel water pressure change optimization technique for solving scheduling problem in cloud computing. *Cluster Computing*, 1-17.
10. Ramezani, F., Lu, J., Taheri, J., Hussain, F.K.: Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments. *World Wide Web* **18**(6), 1737-1757 (2015).
11. Zhao, Q., Xiong, C., Yu, C., Zhang, C., Zhao, X.: A new energy-aware task scheduling method for data-intensive applications in the cloud. *Journal of Network and Computer Applications* **59**, 14-27 (2016).
12. Moon, Y., Yu, H., Gil, J.-M., Lim, J.: A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. *Human-centric Computing and Information Sciences* **7**(1), 28 (2017).
13. Alla, H.B., Alla, S.B., Ezzati, A., Mouhsen, A.: A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing. In: *Advances in Ubiquitous Networking 2*. pp. 205-217. Springer, (2017)
14. Keshanchi, B., Souri, A., Navimipour, N.J.: An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. *Journal of Systems and Software* **124**, 1-21 (2017).
15. Lin, W., Wang, W., Wu, W., Pang, X., Liu, B., Zhang, Y.: A heuristic task scheduling algorithm based on server power efficiency model in cloud environments. *Sustainable Computing: Informatics and Systems* (2017).
16. Yang, J., Jiang, B., Lv, Z., Choo, K.-K.R.: A task scheduling algorithm considering game theory designed for energy management in cloud computing. *Future Generation Computer Systems* (2017).
17. Jena, R.: Task scheduling in cloud environment: A multi-objective ABC framework. *Journal of Information and Optimization Sciences* **38**(1), 1-19 (2017).
18. Gopalakrishnan, N., Arun, C.: A New Multi-Objective Optimal Programming Model for Task Scheduling using Genetic Gray Wolf Optimization in Cloud Computing. *The Computer Journal* (2018).

19. Sobhanayak, S., Turuk, A.K., Sahoo, B.: Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. *Future Computing and Informatics Journal* (2018).
20. Alla, H.B., Alla, S.B., Touhafi, A., Ezzati, A.: A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. *Cluster Computing* **21**(4), 1797-1820 (2018).
21. Emary, E., Zawbaa, H.M., Hassanien, A.E.: Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **172**, 371-381 (2016).
22. Tawhid, M.A., Ali, A.F.: A Hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function. *Memetic Computing* **9**(4), 347-359 (2017).
23. Jayabarathi, T., Raghunathan, T., Adarsh, B., Suganthan, P.N.: Economic dispatch using hybrid grey wolf optimizer. *Energy* **111**, 630-641 (2016).
24. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation* **3**(4), 257-271 (1999).
25. Price, K., Storn, R.M., Lampinen, J.A.: *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, (2006)
26. Michalewicz, Z.: *Evolution strategies and other methods*. In: *Genetic Algorithms+ Data Structures= Evolution Programs*. pp. 159-177. Springer, (1996)
27. Gandomi, A.H., Alavi, A.H.: Krill herd: a new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation* **17**(12), 4831-4845 (2012).
28. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* **11**(4), 341-359 (1997).
29. Xavier, V.A., Annadurai, S.: Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. *Cluster Computing*, 1-11 (2018).
30. Martinez, G., Zeadally, S., Chao, H.-C.: Cloud computing service and architecture models. *Information Sciences*(258), 353-354 (2014).
31. Sreenu, K., Sreelatha, M.: W-Scheduler: whale optimization for task scheduling in cloud computing. *Cluster Computing*, 1-12 (2017).
32. Pradeep, K., Jacob, T.P.: A Hybrid Approach for Task Scheduling Using the Cuckoo and Harmony Search in Cloud Computing Environment. *Wireless Personal Communications* **101**(4), 2287-2311 (2018).
33. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* **41**(1), 23-50 (2011).