Received February 17, 2021, accepted March 1, 2021, date of publication March 4, 2021, date of current version March 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3063723

TSMGWO: Optimizing Task Schedule Using Multi-Objectives Grey Wolf Optimizer for Cloud Data Centers

DEAFALLAH ALSADIE, (Graduate Student Member, IEEE)
Department of Information Systems, Umm Al-Qura University, Mecca 24381, Saudi Arabia
e-mail: dbsadie@uqu.edu.sa

ABSTRACT Cloud computing is considered to be the best solution for addressing the increasing computing requirements of high-performance applications. The efficient performance of the system requires optimal mapping of cloud tasks to resources. However, it is challenging to address computing and storage requirements of high-performance applications while achieving conflicting scheduling objectives like throughput, makespan, resource utilization. This work proposes a metaheuristic approach called task schedule using a multi-objective grey wolf optimizer (TSMGWO) to find near-optimal task scheduling solutions while handling conflicting objectives. The TSMGWO approach has been evaluated using three benchmark datasets, namely, GoCJ, HCSP and Synthetic dataset. The results are compared with heuristic FCFS and MT methods, and metaheuristic methods PSO, GA and WOA. The TSMGWO approach reduces makespan upto 67.52% over FCFS method, 60.93%, over PSO, 38.05% over GA, and 23.22% over WOA methods for 100 tasked cloud workload using GoCJ dataset. It reduces makespan upto 60.95% over FCFS method, 55.79% over MT method, 47.04% over PSO, 33.38% over GA and 19.91% over WOA method using synthetic dataset. Similarly, TSMGWO reduces makespan upto 27.03% over MT method, 18.95% over PSO, 11.90% over GA and 7.5% over WOA method using HSCP workload. The comparative analysis demonstrates that TSMGWO approach outperforms the earlier heuristic and metaheuristic methods using benchmark datasets in the cloud environment.

INDEX TERMS Optimization, cloud computing, task scheduling, meta-heuristics algorithm, multi-objective grey wolf optimizer.

I. INTRODUCTION

Cloud Computing has brought a revolutionary change in business by offering efficient sharing of computing resources. Cloud customers can provision and release Cloud Computing resources as per their requirement on pay per use basis through the public interface provided by the Cloud Service Provider [1], [2]. The recent development in cloud computing allows the number of geographically distributed and interconnected cloud data centres operate on-demand services to the cloud customer on the pay per use basis in more efficient manner [3]–[5]. As reported in [18], by 2021, 94% of computation workload is likely to shift to cloud data centers. The innovative idea of cloud computing as provided many advantages in terms of reduced infrastructure cost, execution time, maintenance and many more [6]. However, the increased workload of cloud computing for executing multiple applications also resulted in a decrease in resource utilization, leading to a decrease in return on investment [7]. One of the primary cause of the decrease in cloud computing resource utilization is incorrect scheduling of the task over virtual machines, leading to a reduction in their processing performance. Therefore, task scheduling plays a significant role in cloud computing to ensure maximum resource utilization by providing adequate performance under the different task constraints such as execution deadlines.

Task scheduling maps the user-submitted task to a suitable virtual machine in the cloud data centre [1]. To get required performance cloud users to enter into a service level agreement with the Cloud Service Provider that binds both the parties on the expected service quality, including the execution deadline of the task, security and budget. The user can

The associate editor coordinating the review of this manuscript and approving it for publication was Bijoy Chand Chatterjee.

demand computing resources required for executing his tasks as per SLA [8]. Task scheduling has a direct impact on the performance of cloud computing. An efficient task scheduling can generate more revenue to improve the performance and decrease the service level agreement violations. Task scheduling problem can be formulated as Bin packing problem. It is a non-deterministic polynomial time-based problem and considered as NP-hard problem [9], [10]. The increased complexity of cloud computing has also increased the complexity of solving the task scheduling problem. Therefore, it becomes more challenging to develop an efficient algorithm for solving the task scheduling problem in a cloud computing environment.

NP hard problems like task scheduling problem in cloud computing can be solved by using the enumeration method or heuristic-based methods [11]. Enumeration methods are not suitable in a cloud computing environment because they need to build all possible combinations of task schedules and select the best solution. This methodology makes it time-consuming, making it unsuitable for the cloud computing environment with a massive workload.

The second category of the method applied for solving the task scheduling problems is heuristic and metaheuristic methods. Several heuristic algorithms have been used for finding optimal task scheduling in the cloud computing environment [1]. Most powerful algorithms proposed in the heuristic category include Minimum Completion Time (MCT), Max-Min, Sufferage First Come First Serve (FCFS), Minimum Execution Time (MET), and Min–Min algorithms. However, these algorithms suffer from several limitations of trapping in local minima due to the multimodal nature of task scheduling in cloud computing environment [12]. Recently, metaheuristic algorithms have obtained considerable attention from the research community to obtain the near-optimal solution of the task scheduling problem in a reasonable period [13].

Metaheuristic methods have several advantages like flexibility, simplicity and ergodicity [12] over conventional methods. Metaheuristic methods can be easily applied and have low complexity in comparison to conventional methods. These methods are not problem dependent and can be applied to a wide range of optimization problems. Additionally, these algorithms can find multimodal search space in different ways to avoid local minima that address the limitation of heuristic methods in solving cloud computing's task scheduling.

Many metaheuristic approaches proposed in recent years have considered task scheduling problem with multiple objectives as a single objective. These approaches attempt to obtain a single optimal solution [1], [11]. A few researchers have focused on applying a metaheuristic algorithm to find near-optimal solutions and provide trade-off to the cloud service providers by considering multiple objectives simultaneously.

In this study, we focus on answering the research question: "can metaheuristic algorithms be applied to find an optimal or near-optimal solution to the task scheduling problem in a cloud computing environment by taking into account makespan, resource utilization, degree of imbalance, and throughput simultaneously".

To answer this research question, we propose applying a meta-heuristic grey wolf optimizer to find an optimal or near optimal solution to the task scheduling problem by considering multiple conflicting objectives of makespan, resource utilization, degree of imbalance, and throughput simultaneously. To evaluate the proposed approach's performance, we implement and compare it's performance with the existing heuristic and metaheuristic algorithms using benchmark datasets using identified performance metrics. The key deliverables of this work include.

- *Formulation of mathematical models for computing resource utilization, makespan, Degree of imbalance and Throughput of the cloud.*
- *Formulation of task scheduling problem as a multi-objective optimization problem having multiple conflicting objectives like resource utilization, makespan, Degree of imbalance and Throughput of the cloud.*
- *Performing multi-objective optimization for task scheduling problem using grey wolf optimizer to maximize resource utilization, minimizes makespan, minimize the degree of imbalance and maximize throughput of cloud simultaneously.*
- *Simulate and compare the proposed grey wolf optimizer-based approach with other meta-heuristic approaches.*
- *Evaluate the performance of the proposed grey wolf optimizer (TSMGWO) based approach in comparison to the identified existing heuristic methods, FCFS and MT; and meta-heuristic methods, namely, GA, PSO, and WOA using three benchmark data sets, namely, GoCJ, Synthetic and HSCP data sets in terms resource utilization, makespan, degree of imbalance and throughput of the cloud.*

The remainder of this article is structured as follows. Section II provides significant prior research on task scheduling problem using different optimization algorithms. Section III defines task scheduling problem in the cloud computing environment. Section IV provides salient features of grey wolf optimizer and its functional components. It highlights the motivation for using grey wolf optimizer for solving the task scheduling problem in cloud computing. Section VI describes the basic assumptions and the proposed TSMGWO approach for optimizing multiple objectives in task scheduling of the cloud computing environment. This section also formulates system models for makespan, resource utilization, Degree of imbalance and Throughput used in this work mathematically. Section VII describes experimental

Section II provides significant prior research on task scheduling problem using different optimization algorithms. Section III defines task scheduling problem in the cloud computing environment. Section IV provides salient features of grey wolf optimizer and its functional components.

It highlights the motivation for using grey wolf optimizer for solving the task scheduling problem in cloud computing. Section VI describes the basic assumptions and the proposed TSMGWO approach for optimizing multiple objectives in task scheduling of the cloud computing environment. This section also formulates system models for makespan, resource utilization, Degree of imbalance and Throughput used in this work mathematically. Section VI describes the experimental setup, results and provides a discussion on the obtained results. Finally, Section VII concludes the paper.

II. RELATED WORK

A large number of models and methods have been proposed for solving the task scheduling problem of cloud computing and grid computing [14], [15]. This section describes the significant research work of solving the task scheduling problem using metaheuristic methods. Metaheuristic methods can be further subdivided into two categories, bio-inspired methods and swarm optimization methods [12], [16].

Several algorithms have been proposed in bio-inspired method category, such as memetic algorithm, genetic algorithm, imperialism competitive algorithm, lion algorithm, etc. These algorithms have been used for finding the optimal solution to the task scheduling problem in cloud computing. For example, Sheng and Qiang used a genetic algorithm for task scheduling problems [17]. Whereas Keshanchi *et al.* [2] used the genetic algorithm with some heuristics for solving static task scheduling problem in the cloud. Similarly, Xiong *et al.* used Johnson rule method with a genetic algorithm to solve the task scheduling problem in cloud data centres [18]. They proposed a genetic algorithm for assigning tasks to the suitable virtual machine and used decoding approach for finding the proper sequence of the task to be managed on the virtual machine by using Johnson rule. Akbari *et al.* [19] improve the genetic algorithm's performance by proposing new operators to ensure solution diversity and reliability of the search space.

Task scheduling problem becomes more complicated in a heterogeneous cloud environment with large cloud workload. Therefore, Swarm intelligence-based method such as artificial Bee colony method, cuckoo search-based method, ant colony optimization method, bio geography-based optimization method, particle swarm optimization method is considered more suitable in solving task scheduling problem. Many researchers applied these Swarm intelligence-based methods to solve the task scheduling problem successfully. Li *et al.* [20] designed a task scheduling approached using ant colony optimization (ACO) algorithm for balancing the workload and minimizing the execution time of the cloud task. They demonstrated that the proposed approach provided better performance as compared to the first come first served approach. Similarly, Liu and Wang [21] also minimizes the execution time of cloud task using ant colony optimization method. Gupta and Garg [22] used the particle swarm optimization method for optimizing the execution time of the tasks and computing resource utilization of the cloud environment by solving the task scheduling problem.

Some authors also hybridized different metaheuristic methods [23]–[25]. For example, Zhan and Huo [23] proposed a hybrid approach based on particle swarm optimization and simulated annealing to obtain optimal task scheduling in cloud computing. The reported results indicate that the hybrid approach can reduce the execution time of the task and enhance resource utilization of the cloud. The authors of [26] used hill-climbing method with a stick for optimizing task schedule in the cloud computing environment. The proposed method outperformed the round-robin, and First Come First served basis approach. Abdullahi *et al.* [27] used a symbiotic organism search-based method for obtaining an optimal mapping of cloud tasks to the virtual machines. They demonstrated that the proposed method based upon symbiotic organism search method outperformed particle swarm optimization-based task scheduling in large search space.

Kim *et al.* [28] suggested biogeography-based optimization method for task scheduling problem. Their proposed method maintains a pool of solutions in different iterations and attempts to find the best possible solutions based upon specific criteria. However, biogeographical based methods are not very practical for a middle-sized task scheduling problem in the cloud computing environment [28]. Vasile *et al.* [29] suggested using hierarchical clustering of the computing resources in the cloud computing environment to find an optimal schedule of the task and virtual machines. Their method is effective in heterogeneous distributed computing such as high-performance computing systems that require modelling of application with data acquired from multimedia applications. Despite the effectiveness of the proposed method, this method ignored the dynamic behaviour of the computing resources. Zuo *et al.* [30] designed a model for optimizing resource cost based on the demand of the computing resources by the cloud tasks using ant colony optimization algorithm. The proposed model demonstrated the association between budget cost and the cost of computing resources. In their model, the authors attempted to optimize the users' makespan and budget cost using multi-objective optimization method.

On similar lines, the authors of [31] suggested the use of whale optimization method for solving the task scheduling problem in cloud data centres. They suggested a fitness function in terms of three parameters, resource utilization, energy consumption and service quality. The propose fitness function is optimized using a whale optimization method. The simulation results demonstrate that whale optimization method outperforms the other methods to minimise energy consumption and maximise resource utilization while maintaining the quality of service as per service level agreement. An approach called W-scheduler has been proposed in [32] by integrating multi-objective model with the whale optimization method. In this model, the authors used a fitness function based upon the cost of memory and CPU consumed in addition to makespan. They used wale optimizer to find the best task schedule that minimizes the cost of memory and CPU and minimises the makespan. A comparison of results with traditional algorithms such as PBACO, SPSO-SA, and SLPSO-SA shows that the performance of the integrated model is better than that of traditional algorithms in the cloud computing environment. An integrated approach using MapReduce, genetic algorithm and whale optimizer were proposed in [33]. Researchers attempted to extract task-specific features initially. MRQFLDA algorithm is further applied to reduce the number of features. The user-submitted task are classified into subtask using the MapReduce approach. It is followed by applying the genetic algorithm and whale optimizer for assigning sub-task to appropriate sized virtual machines in the cloud computing environment. Experimental results indicate that the proposed integrated approach outperforms the conventional algorithms in the cloud computing environment.

Mirjalili *et al.* [34] proposed a metaheuristic method waste on the behaviour of grey wolves called grey wolf optimizer. This method has been successfully implemented for solving classical engineering problems, namely, welded beam, pressure vessel designs, and tension/compression spring. This method has the capability of solving problems with and without constraints. In this method, exploration and exploitation of the search space are considered for finding an optimal solution.

Grey wolf optimizer has several advantages over traditional optimization method like particle swarm optimization method [33]. Particle swarm optimization method has the limitation of degraded performance in the task scheduling context due to low convergence rate in its iterations [35], [36]. Biogeographical based methods were found to be unsuitable for middle-sized touch during problems [28]. Whereas, ant colony optimization-based approaches have ignored the dynamic behaviour of computing resources.

Despite many advantages of grey wolf optimizer in solving different optimization problems, few researchers have used this method for task scheduling problem in a cloud computing environment with a large workload. Natesan and Chokkalingam [11] suggested task scheduling problem using mean grey wolf optimization method in the heterogeneous cloud environment. They demonstrated that the proposed method resulted in minimizing makespan and energy consumption based on effective task scheduling using grey wolf optimization method. Natesha *et al.* [14] proposed a multi-objective optimization method using grey wolf optimization for task scheduling in the cloud environment. The proposed method outperforms the performance of non-metaheuristic methods such as first come first serve (FCFS) and Modified Throttle (MT) as well as metaheuristic methods like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Cat Swarm Optimization (CSO) for optimizing resource utilization in terms of energy consumption and makespan.

Alzaqebah *et al.* [37] suggested the grey wolf optimization-based method for optimizing multi-objective task scheduling problem into a single fitness function. Their method optimized a single fitness function of makespan and cost to solve the task scheduling problem. The reported results demonstrated that their modified grey wolf Optimization method had outperformed traditional grey wolf optimization method and whale Optimization method in terms of makespan, degree of imbalance, and cost. Bacanin *et al.* [38] house adopted a grey wolf optimization method for solving the task scheduling problem in cloud computing. They showed that the grey wolf optimization-based method had provided good quality and robust task schedule over existing methods. Sanaj and Prathap [33] suggested an enhancement of the grey wolf optimizer-based approach for task scheduling for optimizing resource utilization, cost, execution time, communication time, and power usage. The authors employed epsilon constraint, penalty cost, and the fractional grey wolf optimizer with a modification in the position update in their approach. In this method, they used a combination of alpha and beta solutions in the optimization process.

III. TASK SCHEDULING IN THE CLOUD COMPUTING ENVIRONMENT

Cloud computing involves allocating computing resources to different applications as per their requirements [33]. Cloud data centers use several physical servers for providing computing services for various applications. Each physical server runs multiple virtual machines. Computing resources to the different applications are allocated from these virtual machines. Each user-submitted task is allocated and executed on a virtual machine-based upon its processing capability and cost of computing resources. Task scheduling methods aim to efficiently allocate computing resources for executing the user-submitted task as per requirement [39]. Efficient allocation of computing resources requires estimation of memory consumption, CPU, and bandwidth utilization for executing the specified task [40]. It is feasible to improve task scheduling performance with suitable values of multiple objectives like resource utilization, execution time, Degree of imbalance, makespan and Throughput off the cloud data center using a grey wolf optimization algorithm [40], [41]. This algorithm enables the allocation of virtual machines to the user-submitted task as per their requirement while meeting the cloud data center's multiple objectives like maximizing resource utilization, minimizing makespan, minimizing the degree of imbalance, and maximizing the Throughput of the cloud data center. The details of the grey wolf algorithm for optimizing the cloud computing environment's task scheduling process are described in the following section.

IV. GREY WOLF OPTIMIZATION (GWO) ALGORITHM AND MOTIVATION

This section describes salient features of the grey wolf optimizer and its functional components.

A. THE MOTIVATION OF GREY WOLF OPTIMIZER

The main benefit of using the grey wolf optimizer is that it can be applied to solve constrained and unconstrained problems in different fields. This optimizer begins with solutions that are initialized using random numbers without computing any derivative of search space for finding the optimal solution [33], [42]. This benefit enables grey wolf optimizer to solve real-world problems with unknown derivative information or expensive derivative information. This optimizer is also helpful in solving the problems with the unknown search space of real problems.

Grey wolf optimizer is a metaheuristic algorithm proposed by Mirjalili *et al.* in 2014 [34] and falls in the category of swarm intelligence algorithms. Grey wolf optimizer is one of the most commonly used algorithms for optimizing problems related to Engineering [43]. This algorithm is inspired by the social arrangement and hunting procedure of grey wolves. In 2016, the authors developed a multi-objective version of Grey wolf optimizer [44]. Grey wolf optimizer is known for using fewer parameters, fast convergence, scalability, flexibility, and capability to maintain the balance between exploitation and exploration during the subspace [45].

It has been shown that standard grey wolf optimizer outperforms many conventional optimizers like particle swarm optimization, differential evolution, Gravitational Search Algorithm, and free energy perturbation algorithm [34]. Mirjalili *et al.* [34] stated that grey wolf optimizer is easy to implement, exhibits a fast convergence rate, provides high accuracy for many optimization problems. The grey wolf optimizer characteristics make it suitable for time Limited task scheduling problems like task scheduling in the cloud computing environment. These features of grey wolf optimizer enabled it to be successfully used in different fields such as machine learning, bioinformatics, networking, medical, environment applications, and image processing applications [45].

B. FUNCTIONAL COMPONENTS OF GREY WOLF OPTIMIZER

Generally, grey wolves live in an organized structure called packs. In each pack, grey wolves can be divided into four types of wolves, namely, alpha, beta, delta, and omega wolves [34], [44], [45]. The alpha wolf is considered the strongest animal in the pack and behaves as a leader to navigate the hunting process. The beta wolf is the second level of the wolf in the hierarchy for making the decisions during the hunting process. Delta wolf has the next level of dominance in decision making after the beta wolf. Omega wolves are at the lowest level in the hierarchy of Grey wolves. Figure 1 represents the social hierarchy of the grey wolves.

In the absence of the powerful alpha wolf, decision-making capacity moves one step down in the social hierarchy of grey wolves. This social intelligence is the main inspiration of the grey wolf optimizer. During the hunting process, grey wolves also follow a systematic strategy, consisting of different iterative phases namely, search, encircle, and attack [46]. These phases are described below.

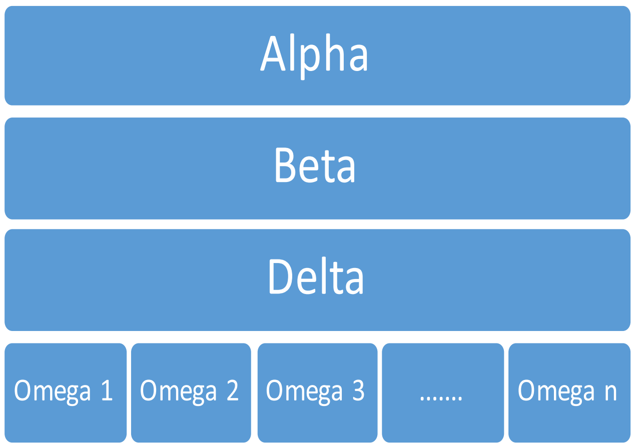

FIGURE 1. The social hierarchy of the grey wolves.

1) SEARCH PHASE

The search phase is an essential and first step in the hunting process of Grey wolves. In the mathematical model, it involves the computation of the distance between the Prey and Grave wolf. It is computed as per Eq. (1).

$$Dist = abs(coeff \times L_p(t) - L_{GW}(t)) \tag{1}$$

where *coeff* gives the coefficient vector; $L_{GW}(t)$ represents the location of the grey wolf at a given time interval t; L_p indicates the location of prey at a given time interval t. The value of *coeff* is computed using Eq. (2).

$$Coeff = 2 \times Ran(0, 1) \tag{2}$$

where $Ran(0, 1)$ is a function that returns a random number between 0 and 1. Table 1 is the notations used in the modelling.

TABLE 1. Notations used in the proposed TSMGWO task scheduling approach.

Notation	Definition
Avg_RUR	Average resource utilization ratio
$coeff$	Coefficient vector in grey wolf optimization method
CT_{xy}	The computation time of running a task T_x on virtual machine VM_y
$Dist$	Distance vector in grey wolf optimization method
I$Imbalance_Deg$	Degree of imbalance
L_{alpha}	Location of Alpha wolf
L_{beta}	Location of Beta wolf
L_{delta}	Location of Delta wolf
L_{GW}	Location of grey wolf
L_p	Location of prey
$MS(VM_x)$	Makespan of virtual machine x
n	Population size of grey wolf optimization method
T_i	Use submitted task $i, 1 < j < I$
t_{max}	Maximum number of iterations in the grey wolf optimization method
VM_j	Virtual machine $j, l < i < J$

2) ENCIRCLE PHASE

The second phase in the hunting process is encircling the prey by grey wolves. The encircle phase involves changing the location of different grey wolves concerning the prey to encircle it habitually. Mathematically, a change in position of nth grey wolves at a time interval of $t+1$ in this phase can be simulated as per Eq. (3) below.

$$L_{GW-n}(t+1) = L_p(t) \ - Mcoeff_n \times Dist_n \tag{3}$$

where $Dist_n$ is the distance of *nth* grey wolf from the *prey* and computed as per Eq. (4).

$$Dist_n = abs(coeff_n \times L_p(t) - L_{GW}(t)\,) \tag{4}$$

$Mcoeff_n$ gave the mean coefficient vector for grey wolves and computed as per Eq. (5).

$$Mcoeffn = 2 \times X \times ran(0, 1) \times X \tag{5}$$

X is a variable that gets decreased from 2 to 0 in the iterative process. The current value X in an iteration t in t_{max} as the maximum number of iterations is computed as per Eq. (6).

$$X = 2 - 2 \times \frac{t}{t_{max}} \tag{6}$$

3) ATTACK PHASE

Grey wolves change their positions as per the position of prey as simulated in Eq. (1) to Eq. (6). The social hierarchy of grey wolves is followed to attack the prey. The top three dominating wolves as alpha, beta and delta in the hierarchy are considered as best solutions. At a suitable distance after encircling the prey, it is assumed that alpha, beta and delta wolves have a better understanding of the position of prey. Therefore, the alpha wolf attacks the prey. The position of prey is updated as per the position of alpha, beta and delta wolves that is mathematically simulated as per Eq. (7). Omega wolves update their locations randomly around the prey as per Eq. (7).

$$\begin{aligned} Dist_{alpha} &= abs(coeff_1 \times L_{alpha}(t) - L(t)) \\ Dist_{beta} &= abs(coeff_2 \times L_{beta}(t) - L(t)) \\ Dist_{delta} &= abs(coeff_3 \times L_{delta}(t) - L(t)) \\ L_1 &= L_{alpha}(t) \ - coeff_1 \times Dist_{alpha} \\ L_2 &= L_{beta}(t) \ - coeff_2 \times Dist_{beta} \\ L_3 &= L_{delta}(t) \ - coeff_3 \times Dist_{delta} \\ L &= \frac{(L_1 + L_2 + L_3)}{3} \end{aligned} \tag{7}$$

Grey wolf optimizer works by using a set of random solutions as grey wolves. Set of random solutions are measured in terms of multiple objective functions [45], [46]. The values of multiple objective functions as fitness function indicates the quality of each solution. The top three quality solutions are denoted as alpha, beta, and delta wolves. Grey wolf optimizer regularly updates the location of wolves in each iteration. In any iteration, if any solution becomes better than alpha, beta, and delta wolves, then corresponding solutions are replaced with the identified solution. Grey wolf optimizer stops iterating the solutions till some stopping criteria are satisfied [46]. Figure 2 shows the pseudo-code of grey wolf optimize algorithm.

```
Input:
    1. Parameters of grey wolf optimizer.
    2. Task $(T_i)$ and virtual machines $(VM_j)$,
       for all $i \in \{1, 2, 3, - - - - -, I\}$ and $j \in \{1, 2, 3, - - - - -, J\}$.
Output:
    1. A set of optimal task schedules.
Parameter initialization:
    1. An initial population of size $n$ $(L_{GW})$
    2. Random initialization of coeff vector, Dist vector
    3. User defined $t_{max}$ as the maximum number of iterations
    4. Set $t = 0$ as initial counter
Start
Initialization phase:
    1. Set $c = 1$
    2. While $(c \leq n)$
          a. Initial $L_{GW}(t)$ randomly
          b. compute fitness
    3. end while
    4. Sort population in the order of fitness
    5. Top most fit grey wolf denoted as $L_{alpha}$
    6. Second most fit grey wolf denoted as $L_{beta}$
    7. Third most grey wolf denoted as $L_{delta}$
Updating phase:
    1. While $(t < t_{max})$    //Maximum number of iterations
    2. Loop for each element in the population
          a. Update location
    3. End loop
    4. Update value of $X$
    5. Update value of $coeff$ vector
    6. Update value of $Dist$
    7. Update value of $L_{alpha}$
    8. Update value of $L_{beta}$
    9. Update value of $L_{delta}$
    10. $t{+}{+}$          // Increment iteration count
    11. end while
Output phase:
    1. Return $L_{alpha}$ as best solution in the search space
End.
```

FIGURE 2. Pseudo-code of grey wolf optimizer.

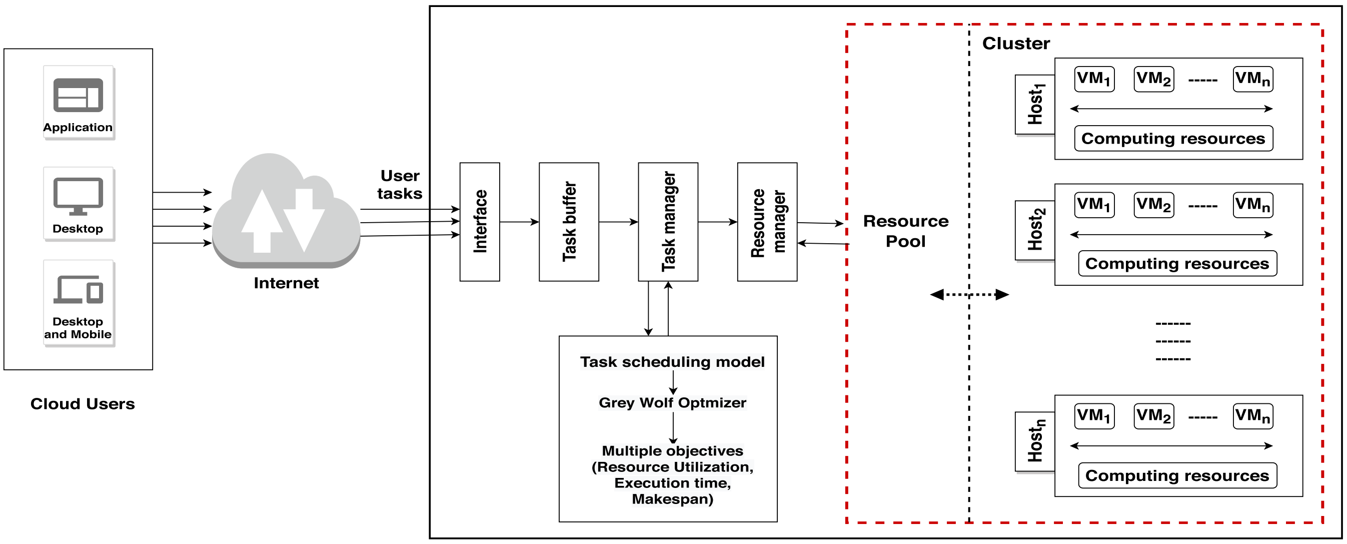

FIGURE 3. Proposed the TSMGWO approach.

V. TSMGWO - GREY WOLF OPTIMIZER BASED TASK SCHEDULING APPROACH

This article presents a new approach to optimize task resource mapping using the grey wolf optimization method in the cloud computing environment. To optimize task resource mapping, we attempt to optimize multiple objectives of makespan, resource utilization, Degree of imbalance, and Throughput using the proposed TSMGWO approach. The proposed grey wolf optimization method-based approach is shown in Figure 3. This work assumes that the underlying cloud environment is heterogeneous, where utilization information of physical machines is available. We attempt to optimize makespan, resource utilization, Degree of imbalance, and Throughput of the tasks simultaneously using grey wolf optimizer in the form of a Pareto front for providing the trade-off to the cloud service provider.

Cloud service providers offer virtual machines running on physical machines through a public interface of their proprietary software [47]. The cloud users submit their requests for processing of different tasks through the public interface. All the received user tasks are stored in a temporary storage called task buffer, as shown in Figure 3. The task manager takes user tasks from the task buffer and calls the task scheduling model. The task scheduling model applies grey wolf optimizer to obtain a set of solutions providing trade-off to the cloud service provider. This model optimizes the task schedule based on constrained performance requirements of virtual machines and different tasks' execution time. A suitable solution as an optimized task schedule is selected based on the cloud service provider's requirement. Accordingly, tasks are allocated computing resources in the different hosts in the form of virtual machines, as shown in Figure 3.

The flowchart for the TSMGWO based task scheduling approach is depicted in Figure 4. Initially the number of cloudlets and the list of VMs are given as an input for the algorithm. After inputting the values, the parameters such as a and tmax are initialized. In Step 2, the initial schedule is generated. Using this schedule, we calculated the fitness function of the VMs. In Step 3, VMs are grouped into alpha, beta, gamma group of wolves, where alpha will be the least F value, and beta is greater than alpha and less than other wolves and delta is greater than beta and less than other wolves and rest will be considered as omega wolves. Then the values of a and t max are varied in each loop to identify the better schedule. By updating the values of wolves group in each

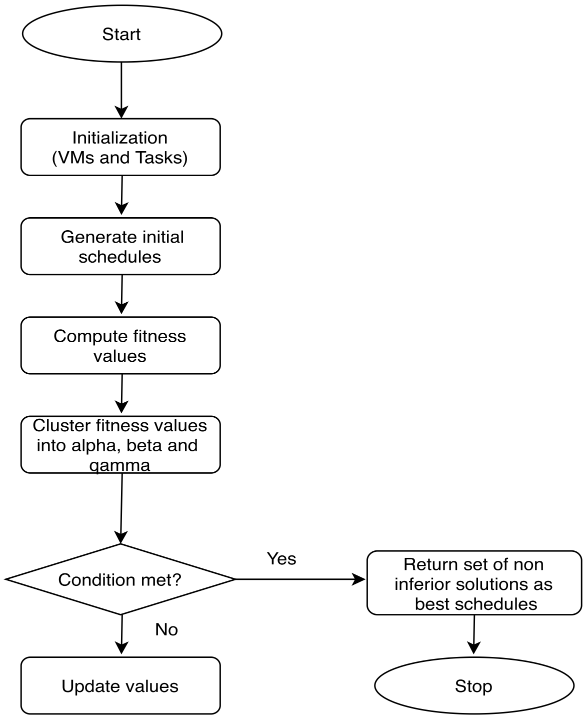
Start
Initialization (VMs and Tasks)
Generate initial schedules
Compute fitness values
Cluster fitness values into alpha, beta and gamma
Condition met?
Yes
No
Return set of non inferior solutions as best schedules
Update values
Stop

FIGURE 4. Flowchart of the TSMGWO approach.

iteration, the value of a is continuously decreasing from 2 to 0. When the value of a becomes zero or terminating condition is met, then a set of values of alpha will be returned as the best values and it will be considered as the non-inferior schedules for allocating tasks to the VMs.

A. SOLUTION REPRESENTATION

In this work, the solution to the task scheduling problem is represented as an array of tasks $T = \{T_1, T_2, T_3, \ldots\ldots, T_I\}$ that are mapped to virtual machines $V = \{V_1, V_2, V_3, \ldots \ldots, V_J\}$ using grey wolf population as shown in Figure 5. Initially, resources are allocated as per their cost and capacity. The proposed approach helps to obtain an optimal talks schedule for allocating resources to the task-based upon multiple objectives [1].

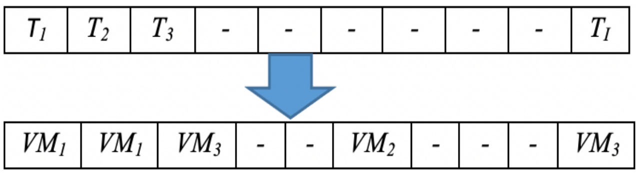

FIGURE 5. Task scheduling solution representation.

B. BASIC ASSUMPTIONS

In this work, we assume the following constraints for executing tasks on virtual machines in the cloud data center to simulate a realistic execution environment.

- *E*ach user-submitted task is assigned to only one virtual machine.
- *E*ach user-submitted task must be executed to its completion within the timeframe of the task on the assigned virtual machine.
- *T*he total capacity of virtual machines mounted on a physical machine exceeds the total computing requirements of the cloud tasks to be executed in that physical machine.
- *User-submitted* tasks are independent of each other.
- *E*ach task can be executed on any available virtual machine that fulfils its computing requirements.
- *M*ultiple tasks can be assigned to any virtual machine.
- *T*he execution time for different tasks depends upon the capacity of the virtual machine.

C. DEFINITION OF SYSTEM MODELS

This work consists of essential objectives of resource utilization, makespan and execution time for optimization. These objectives are conflicting with each other acquiring maximization of resource utilization and minimization of makespan and execution time.

1) MAKESPAN

Makespan defines the total time required from submitting a task to the completion of the task by the user. It is obtained by summing up the waiting time and processing time of the task. It signifies the minimization of execution time and hence meeting the execution constraints of the task [11]. Its lower value characterizes a good task schedule in terms of execution time. It is generally computed as the time taken by a task to complete its execution on various virtual machines. It is computed using from Eq. (8) to Eq. (11).

$$Makespan = max(MS(VM_x)) \tag{8}$$

where $1 \leq x \leq n$; n represents the number of virtual machines.

$$MS(VM_x) = \sum CT_{xy} \times Assigned(x, y) \tag{9}$$

where, CT gives the computation time of task T_x on VM_y; Assigned $(x, y) = 1$, if T_x is scheduled on VM_y, otherwise Assigned $(x, y) = 0$.
Computation time CT of a virtual machine VM_y for running a task T_x can be computed using Eq. (11).

$$CT_{xy} = \sum (T_x.MI \;/\; VM_y.MIPS) \tag{10}$$

where, $T_x.MI$ represents a million instructions of Task T_x. $VM_y.MIPS$ gives a million instructions per second of VM_y. Therefore, the first objective of this work is to minimize makespan as represented in Eq. (11).

$$\textbf{\textit{Objective I}} = Minimize(Makespan) \tag{11}$$

2) RESOURCE UTILIZATION

In this work, we evaluated the performance of the TSMGWO based approach using resource utilization in terms of an average resource utilization ratio. The average resource utilization ratio is the computed ratio of average makespan to the maximum makespan of the cloud system using Eq. (12) and Eq. (13) as follows [48], [49].

$$Avg\_RUR = (Avg\ Makespan \;/\; Cloud\ Makespan) \tag{12}$$

$$Avg\ Makespan = \sum MS(VM_x) \;/\; n, \tag{13}$$

where $1 \leq x \leq n$; n represents the number of virtual machines. Cloud Makespan = Maximum time for the completion of all the system workload. The value of Avg_RUR ranges from 0 to 1. The value 0 indicates the negligible utilization of computing resources and value 1 signifies full utilization of computing resources in the cloud environment.

Therefore, the second objective of this work is to maximize the average resource utilization ratio as represented in Eq. (14).

$$\textbf{\textit{Objective II}} = Maximize\ (Avg\_RUR) \tag{14}$$

3) DEGREE OF IMBALANCE

The degree of imbalance evaluates the imbalance of work load distribution of cloud among virtual machines as per their competencies [1]. It is generally computed as execution time of tasks on virtual machines. It can be computed using Eq. (15) as follows.

$$Imbalance\_Deg = \frac{(Max\_CTime_i - Min\_CTime_i)}{Avg\_CTime_i} \tag{15}$$

where Max_CTime_i gives the maximum completion time of $Task_i$ on all virtual machines; Min_CTim_i gives the minimum completion time of $Task_i$ on all virtual machines; Avg_CTime_i gives the average completion time of $Task_i$ on all virtual machines. The lower values of $Imbalance\_Deg$ indicates that cloud work load is balanced correctly, whereas higher values show that load balancing is inefficient. Therefore, the third objective of this work is to minimize the degree of imbalance as represented in Eq (16).

$$\boldsymbol{Objective\ III} = Minimize\ (Imbalance\_Deg) \tag{16}$$

4) THROUGHPUT

The Throughput measures the number of cloud tasks executed per unit time. It is computed in terms of makespan as per (17) [5].

$$Throughput = (Number\ of\ tasks)\ /\ (Makespan) \tag{17}$$

A higher value of the throughput metric is desired for a better performing task scheduling method. Therefore, the fourth objective of this work is to maximize Throughput as represented in Eq. (18).

$$\boldsymbol{Objective\ IV} = Maximize\ (Throughput) \tag{18}$$

5) TIME COMPLEXITY ANALYSIS

The computational costs of the proposed framework computed like complexity of standard GWO. The proposed framework involves the initialization (D_{ini}), position update (D_{update}), and fitness evaluations (D_{eval}) for the population. Given an N-wolf of pack and D-dimensional optimization problem, the computational complexity of the proposed framework can be calculated as:

$$\begin{aligned} D &= D_{ini} + (D_{update} + D_{eval}) \times F_{Evaluations} \\ &= N + (N \times D + N) \times F_{Evaluations} \\ &= N \times (1 + (D + 1) \times F_{Evaluations}) \end{aligned} \tag{19}$$

Therefore, the time complexity of the proposed framework is $O\ (N \times D \times F_{Evaluations})$.

VI. EXPERIMENT AND RESULTS

This section describes the experimental setup, performance metrics, evaluation data sets, and experimental results. The discussion of results includes an evaluation of the proposed TSGWO task scheduling approach along with identified heuristic and metaheuristic methods using three benchmark data sets. This evaluation uses benchmark data sets, namely, GoCJ [50] and HCSP [51] data sets and a realistic cloud workload-based data set.

A. EXPERIMENT CONFIGURATION

It is a very challenging task to evaluate the task scheduling approach using aerial cloud workload [48]. The real cloud workload put many constraints for conducting scalable experiments during the task scheduling approach's testing phase. Cloud computing works based on the pay per use model, making it very expensive to repeat the experiments using real cloud workload. A promising alternative is to simulate the cloud workload using cloud simulators such as cloudSim(version 5.0) [52].

Therefore, we evaluated the proposed TSGWO task scheduling approach in a simulated environment using cloudSim software based upon benchmark datasets. The experiments are simulated using a machine equipped with Intel (R) Core (TM) i5-4210 CPU @ 1.70 GHz, 8GBs RAM, 1TB HDD under Windows 10 64-bit operating system.

Our experimental setup assumes two data centers consisting of five physical machines with a total memory of 8 GB. The bandwidth of the physical machine is 2800 Mpbs. Twenty-five virtual machines were created over five physical machines. The experimental setup of the data center, physical machines, and virtual machines are presented in Table 2.

TABLE 2. Experimental setup for simulated evaluation.

Sr No	Item	Details	Value
1	Virtual machine	Bandwidth	1000
		CPU count	01
		Virtual machine count	25
		Main memory	1 GB
		MIPS	100 – 1000
		Operating system	Linux
		Policy type	Time Shared
		Size	10000
		VMM	Xen
2	Physical machine	Bandwidth	2800
		Hard disk drive	1 TB
		Hosts count	05
		Main memory	8 GB
		Policy	Time shared
		Power	4 Dual core (4000 MIPS), 26 Quad core (4000 MIPS)
3	Data center	Data center count	02

B. PERFORMANCE METRICS

We evaluated the proposed TSMGWO approach for task scheduling in the cloud computing environment in four performance metrics. These identified performance metrics are makespan, average resource utilization ratio, Degree of imbalance and Throughput of the cloud. These metrics are the most commonly used evaluation metrics for comparing and analyzing the performance of task scheduling approaches in cloud computing. The values of performance metrics, makespan, average resource utilization ratio, Degree of imbalance and Throughput are computed using equations (1), (5), (7), and (9) respectively as described in Section V-C.

C. EVALUATION DATA SETS

To evaluate and perform a comprehensive comparison of the proposed TSMGWO approach with the selected heuristic and metaheuristic approaches, we focused on the best choices for the selection of simulator, benchmark data set, and evaluation metrics.

We selected three benchmark datasets for evaluating the proposed approach, a realistic cloud workload, two benchmark data set, namely, the GoCJ data set proposed by Hussain and Aleem [50] and the HSCP data set proposed by Braun *et al.* [51]. The details are described below.

1) GoCJ DATA SET

Hussain and Aleem [50] have developed a realistic cloud workload data set using Google Cluster traces known as the GoCJ data set. The GoCJ data set is provided for evaluation purposes in the form of different text files in the Mendeley data repository. Data is arranged in the form of rows consisting of numeric values. The numeric value indicates the size of the cloud job in terms of million instructions (MI). This dataset is composed of five types of jobs with different proportions and is presented in Table 3. Each text file of the GoCJ data set contains a different number of jobs of different sized cloud workloads as shown in Table 4.

TABLE 3. Job types of GoCJ data set.

Sr no	Job type	MI range	Distribution
1	Small	15,000 - 55,000	20 %
2	Medium	59,000 - 99,000	40 %
3	Large	101,000 - 135,000	30 %
4	Extra large	150,000 - 337,500	6 %
5	Huge	525,000 - 900,000	4 %

2) HSCP DATA SET

Braun *et al.* [51] proposed a model for developing the HCSP cloud workload based data set. The HCSP data set is developed based on expected computation time consisting of a fixed number of virtual machines and number of tasks by considering four factors: distribution of workload, consistency, heterogeneity of task, and heterogeneity of the resources. The data instances are represented in the form *u_x_yy zz* [51].

u represents uniform distribution. *x* represents consistency type. Three Types of consistencies have been considered in this work as fully consistent, partially consistent, and inconsistent. *yy* represents the heterogeneity of the task. *zz* represents the heterogeneity of the resource. Heterogeneity of task and resource can take values of high and low. In this work, we focused on uniformly generated data instances. The data instances of the HCSP data set are denoted by the number of tasks multiplied by virtual machines. For example, (1024 x 32) indicates 1024 tasks in the workload to be mapped on to 32 virtual machines. Considering *u_x_yy zz* representation, the HCSP data set have twelve data formats corresponding to different consistency levels for uniform distribution of cloud workload as summarized in Table 5.

3) SYNTHETIC DATA SET

Many researchers have evaluated their approaches using synthetic data set having fixed-sized jobs. For example, Mehdi *et al.* [53] conducted many experiments using a genetic scheduler with the heterogeneous machine for executing about 100 cloud task workloads using a synthetic

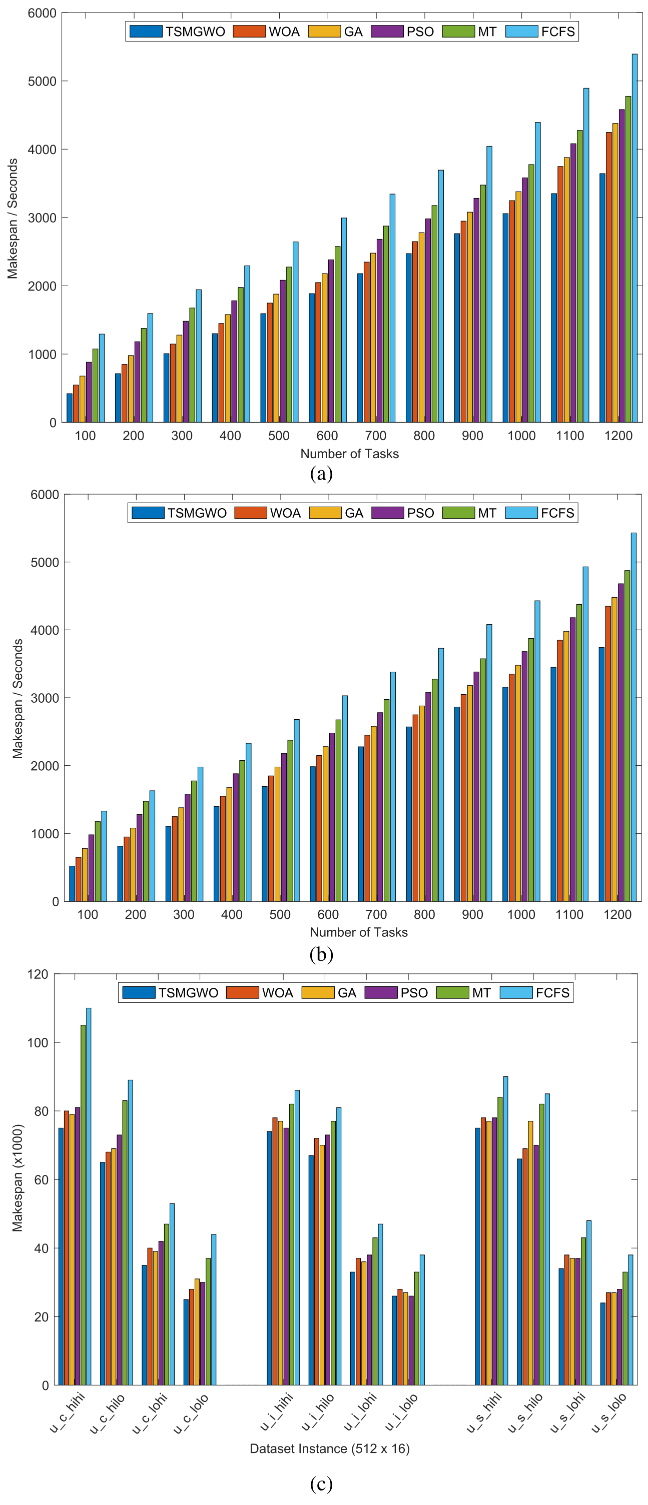

FIGURE 6. Makespan-based comparative results of (a) using the GoCJ data set, (b) using the synthetic data set, and (c) using the HSCP data set.

dataset. Similarly, Behzad *et al.* [54] also performed a comparative study of scheduling methods using a synthetic data set consisting of seven thousand jobs and 15000 jobs with many processors ranging from 4 to 64 processors. On similar lines, we also used a synthetic data set containing five

TABLE 4. **GoCJ data set job statistics.**

Sr No	File name	Job type	Number of jobs	Sr No	File name	Job type	Number of jobs	Sr No	File name	Job type	Number of jobs
1	GoCJ Dataset 100.txt	Small Jobs	19	2	GoCJ Dataset 150.txt	Small Jobs	22	3	GoCJ Dataset 200.txt	Small Jobs	42
		Medium Jobs	39			Medium Jobs	58			Medium Jobs	75
		Large Jobs	31			Large Jobs	51			Large Jobs	63
		Extra Large Jobs	3			Extra Large Jobs	6			Extra Large Jobs	4
		Huge Jobs	8			Huge Jobs	13			Huge Jobs	16
4	GoCJ Dataset 250.txt	Small Jobs	58	5	GoCJ Dataset 300.txt	Small Jobs	61	6	GoCJ Dataset 350.txt	Small Jobs	60
		Medium Jobs	93			Medium Jobs	114			Medium Jobs	131
		Large Jobs	79			Large Jobs	95			Large Jobs	119
		Extra Large Jobs	7			Extra Large Jobs	7			Extra Large Jobs	14
		Huge Jobs	13			Huge Jobs	23			Huge Jobs	26
7	GoCJ Dataset 400.txt	Small Jobs	55	8	GoCJ Dataset 450.txt	Small Jobs	85	9	GoCJ Dataset 500.txt	Small Jobs	96
		Medium Jobs	165			Medium Jobs	184			Medium Jobs	191
		Large Jobs	139			Large Jobs	141			Large Jobs	155
		Extra Large Jobs	18			Extra Large Jobs	17			Extra Large Jobs	25
		Huge Jobs	23			Huge Jobs	23			Huge Jobs	33
10	GoCJ Dataset 550.txt	Small Jobs	98	11	GoCJ Dataset 600.txt	Small Jobs	114	12	GoCJ Dataset 650.txt	Small Jobs	118
		Medium Jobs	226			Medium Jobs	229			Medium Jobs	243
		Large Jobs	176			Large Jobs	190			Large Jobs	210
		Extra Large Jobs	17			Extra Large Jobs	25			Extra Large Jobs	33
		Huge Jobs	33			Huge Jobs	42			Huge Jobs	46
13	GoCJ Dataset 700.txt	Small Jobs	116	14	GoCJ Dataset 750.txt	Small Jobs	145	15	GoCJ Dataset 800.txt	Small Jobs	160
		Medium Jobs	295			Medium Jobs	284			Medium Jobs	315
		Large Jobs	222			Large Jobs	240			Large Jobs	252
		Extra Large Jobs	29			Extra Large Jobs	28			Extra Large Jobs	27
		Huge Jobs	38			Huge Jobs	53			Huge Jobs	46
16	GoCJ Dataset 850.txt	Small Jobs	148	17	GoCJ Dataset 900.txt	Small Jobs	156	18	GoCJ Dataset 950.txt	Small Jobs	179
		Medium Jobs	363			Medium Jobs	344			Medium Jobs	347
		Large Jobs	259			Large Jobs	298			Large Jobs	337
		Extra Large Jobs	33			Extra Large Jobs	44			Extra Large Jobs	39
		Huge Jobs	47			Huge Jobs	58			Huge Jobs	48
				19	GoCJ Dataset 100.txt	Small Jobs	162				
						Medium Jobs	423				
						Large Jobs	322				
						Extra Large Jobs	33				
						Huge Jobs	60				

TABLE 5. **HCSP data set instances.**

HCSP Dataset (512 x 16) , (1024 x 32) , (2048 x 64) , (128 x 4096), (256 x 8192)												
Uniformity	u	u	u	u	u	u	u	u	u	u	u	u
Consistency (c/s/i)	c	c	c	c	s	s	s	s	i	i	i	i
Task Heterogeneity (hi/lo)	Hi	hi	Lo	lo	hi	hi	lo	lo	hi	hi	lo	lo
Resource Heterogeneity (hi/lo)	hi	lo	Hi	lo	hi	lo	hi	lo	hi	lo	hi	lo
data instance	u_c_ hi hi	u_c_ hi lo	u_c_ lo hi	u_c_ lo lo	u_s_ hi hi	u_s_ hi lo	u_s_ lo hi	u_s_ lo lo	u_i_ hi hi	u_i_ hi lo	u_i_ lo hi	u_i_ lo lo

fixed-size cloud task workloads of different sizes like tiny, small, medium, large, and extra-large cloud workloads in different proportions and are presented in Table 6.

D. RESULTS AND DISCUSSION

We evaluated the proposed TSMGWO task scheduling approach with the identified heuristic methods, FCFS and MT and metaheuristic methodsPSO, GA, and WOA. We used four evaluation metrics: makespan, Average Resource Utilization Ratio (*Avg_RUR*), Degree of imbalance, and Throughput. Performance of TSMGWO task scheduling approach is measured using equations (1), (5), (7) and (9), respectively as described in Section V-C. The results are presented for experimental evaluation using three benchmark data sets: GOCJ data set, HSCP data set, and Synthetic data set in the following sub-sections. We computed results using 512 X 16 data instances of the HSCP dataset containing 512 tasks allocated to 16 virtual machines.

TABLE 6. Job types of the synthetic data set.

Sr no	Job type	MI range	Distribution
1	Tiny	200	35 %
2	Small	1,000	40 %
3	Medium	5,000	5 %
4	Large	15,000	15 %
5	Extra large	45,000	5 %

1) MAKESPAN BASED EXPERIMENTAL RESULTS

Makespan represents the maximum execution time for all tasks in the cloud. It is the most commonly used metrics for Cloud customers in the cloud computing environment. A lower value of makespan metric indicates better performance. Values of makespan metric are computed using Eq. (1) for the identified benchmark datasets. Figure 6 represents the makespan based experimental results obtained and compared with identified heuristic and metaheuristic methods. It can be observed from Figure 6 (a) – (c) that the proposed TSGWO task scheduling approach exhibits better performance with lower values of makespan in comparison to the existing heuristic and metaheuristic methods.

Tables 7 shows a reduction in makespan using different benchmark data sets. It can be observed from Table 7 that the TSMGWO approach results in a maximum reduction in makespan 67.52% over FCFS method, 60.93%, 52.33% over PSO, 38.05% over GA, and 23.22% over WOA methods for 100 tasked cloud workload using GoCJ data set. It can also be observed that a minimum of 30.41%, 19.02%, 9.5%, 5.85% reduction in makespan has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method and WOA method respectively using all possible number of tasks considered in this work using GoCJ data set.

For the synthetic data set, Table 7 shows that the TSMGWO approach can result in a maximum reduction of makespan up to 60.95% over FCFS method, 55.79% over MT method, 47.04% over PSO, 33.38% over GA and 19.91% over WOA method for 100 tasked cloud workload. It can also be observed that a minimum of 28.74%, 18.53%,

TABLE 7. Reduction in makespan using the TSMGWO approach over heuristic and meta heuristic methods.

Dataset	Method / No.of tasks	Heuristic methods		Meta heuristic methods				Reduction in Makespan using TSMGWO				
								Heuristic methods		Meta heuristic methods		
		FCFS	MT	PSO	GA	WOA	TSMGWO	Over FCFS	Over MT	Over PSO	Over GA	Over WOA
GoCJ	100	1293	1075	881	678	547	420	67.52 %	60.93 %	52.33 %	38.05 %	23.22 %
	200	1593	1375	1181	978	847	713	55.24 %	48.15 %	39.63 %	27.10 %	15.82 %
	300	1943	1675	1481	1278	1147	1006	48.22 %	39.94 %	32.07 %	21.28 %	12.29 %
	400	2293	1975	1781	1578	1447	1299	43.35 %	34.23 %	27.06 %	17.68 %	10.23 %
	500	2643	2275	2081	1878	1747	1592	39.77 %	30.02 %	23.50 %	15.23 %	8.87 %
	600	2993	2575	2381	2178	2047	1885	37.02 %	26.80 %	20.83 %	13.45 %	7.91 %
	700	3343	2875	2681	2478	2347	2178	34.85 %	24.24 %	18.76 %	12.11 %	7.20 %
	800	3693	3175	2981	2778	2647	2471	33.09 %	22.17 %	17.11 %	11.05 %	6.65 %
	900	4043	3475	3281	3078	2947	2764	31.63 %	20.46 %	15.76 %	10.20 %	6.21 %
	1000	4393	3775	3581	3378	3247	3057	30.41 %	19.02 %	14.63 %	9.50 %	5.85 %
	1100	4893	4275	4081	3878	3747	3350	31.53 %	21.64 %	17.91 %	13.62 %	10.60 %
	1200	5393	4775	4581	4378	4247	3643	32.45 %	23.71 %	20.48 %	16.79 %	14.22 %
Synthetic	100	1329	1174	980	779	648	519	60.95 %	55.79 %	47.04 %	33.38 %	19.91 %
	200	1629	1474	1280	1079	948	812	50.15 %	44.91 %	36.56 %	24.75 %	14.35 %
	300	1979	1774	1580	1379	1248	1105	44.16 %	37.71 %	30.06 %	19.87 %	11.46 %
	400	2329	2074	1880	1679	1548	1398	39.97 %	32.59 %	25.64 %	16.74 %	9.69 %
	500	2679	2374	2180	1979	1848	1691	36.88 %	28.77 %	22.43 %	14.55 %	8.50 %
	600	3029	2674	2480	2279	2148	1984	34.50 %	25.80 %	20.00 %	12.94 %	7.64 %
	700	3379	2974	2780	2579	2448	2277	32.61 %	23.44 %	18.09 %	11.71 %	6.99 %
	800	3729	3274	3080	2879	2748	2570	31.08 %	21.50 %	16.56 %	10.73 %	6.48 %
	900	4079	3574	3380	3179	3048	2863	29.81 %	19.89 %	15.30 %	9.94 %	6.07 %
	1000	4429	3874	3680	3479	3348	3156	28.74 %	18.53 %	14.24 %	9.28 %	5.73 %
	1100	4929	4374	4180	3979	3848	3449	30.03 %	21.15 %	17.49 %	13.32 %	10.37 %
	1200	5429	4874	4680	4479	4348	3742	31.07 %	23.23 %	20.04 %	16.45 %	13.94 %
HSCP	u - c - hi hi	110	105	95	81	80	77	30.00 %	26.67 %	18.95 %	4.94 %	3.75 %
	u - c - hi lo	89	83	69	73	68	67	24.72 %	19.28 %	2.90 %	8.22 %	1.47 %
	u - c - lo hi	53	47	39	42	40	37	30.19 %	21.28 %	5.13 %	11.90 %	7.50 %
	u - c - lo lo	44	37	31	30	28	27	38.64 %	27.03 %	12.90 %	10.00 %	3.57 %
	u - i - hi hi	86	82	77	75	78	76	11.63 %	7.32 %	1.30 %	-1.33 %	2.56 %
	u - i - hi lo	83	79	70	73	72	69	16.87 %	12.66 %	1.43 %	5.48 %	4.17 %
	u - i - lo hi	47	43	36	38	37	36	23.40 %	16.28 %	0.00 %	5.26 %	2.70 %
	u - i - lo lo	38	33	27	26	28	26	31.58 %	21.21 %	3.70 %	0.00 %	7.14 %
	u - s - hi hi	90	84	77	78	78	76	15.56 %	9.52 %	1.30 %	2.56 %	2.56 %
	u - s - hi lo	85	82	77	70	69	67	21.18 %	18.29 %	12.99 %	4.29 %	2.90 %
	u - s - lo hi	48	43	37	37	38	36	25.00 %	16.28 %	2.70 %	2.70 %	5.26 %
	u - s - lo lo	38	33	27	28	27	26	31.58 %	21.21 %	3.70 %	7.14 %	3.70 %

14.24%, 9.28%, and 5.73% reduction in makespan has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method, and WOA method respectively using all possible number of tasks considered in this work using synthetic data set.

For the synthetic data set, Table 7 shows that TSMGWO approach can result into a maximum reduction of makespan up to 60.95 % over FCFS method, 55.79 % over MT method, 47.04 % over PSO, 33.38 % over GA and 19.91 % over WOA method for 100 tasked cloud work load respectively. It can also be observed that a minimum of 28.74 %, 18.53 %, 14.24 %, 9.28 % and 5.73 % reduction in makespan has been achieved using TSMGWO approach over FCFS method, MT method, PSO method, GA method and WOA method respectively using all possible number of tasks considered in this work using synthetic data set. Similarly, for 512 X 16 data instance of HSCP data set, Table 7 shows that the TSMGWO approach can result in a maximum reduction of makespan up to 38.64% over FCFS method for *u_c_lolo* data pattern, 27.03% over MT method for *u_c_lolo* data pattern, 18.95% over PSO for *u_c_hihi* data pattern, 11.90% over GA for *u_c_lohi* data pattern, and 7.5% over WOA method for *u_c_lohi* data pattern of HSCP cloud workload. It can also be observed that minimum 11.63%, 7.32%, 0.00%, -1.33%, and 1.47% reduction in makespan has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method and WOA method respectively using all possible number of tasks considered in this work using HSCP data set.

It can be concluded from Table 7 that there is a large reduction in makespan for workload having less number of tasks in comparison to more number of tasks using the TSMGWO approach over-all state of art methods considered in this work. The main reason behind the better performance of the TSMGWO approach is the improved optimizing ability of GWO over the other methods. Results delineate that the TSMGWO approach outperforms in reducing makespan by a considerable amount, leading to better performance of the cloud computing environment over the existing heuristic and metaheuristic methods using benchmark data sets.

2) AVERAGE RESOURCE UTILIZATION RATIO BASED EXPERIMENTAL RESULTS

Figure 7 represents the average resource utilization ratio based on experimental results obtained in this work and compared with identified heuristic methods and metaheuristic methods. The average resource utilization ratio for various benchmark data sets is computed using Eq 5. It can be observed from Figure 7 (a) – (c) that the TSMGWO approach has reported better performance than the identified heuristic and metaheuristic methods by increasing resource utilization.

Tables 8 shows the increase in resource utilization using different benchmark data sets. It can be observed from Table 8 that the TSMGWO approach can result in a maximum increase of up to 462.50% over FCFS method, 275% over

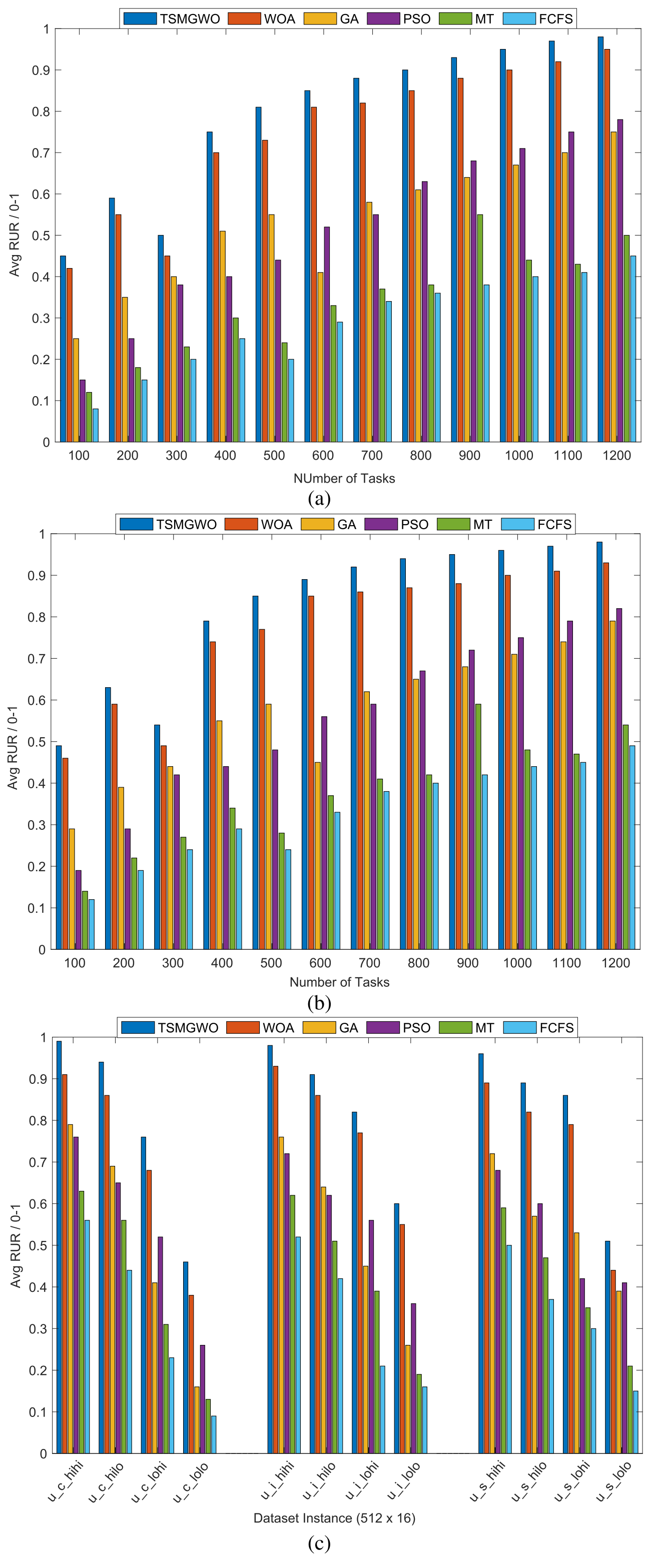

FIGURE 7. Average resource utilization ratio based results of (a) using the GoCJ data set, (b) using the synthetic data set, and (c) using the HSCP data set.

MT method, 200% over PSO, 107.32% over GA, 11.11% over WOA methods for 100 tasked cloud workload of GoCJ data set. It can also be observed that a minimum of 117.78%,

TABLE 8. Increase in Average RUR using the TSMGWO approach over heuristic and meta heuristic methods.

Dataset	Method / No.of tasks	Heuristic methods		Meta heuristic methods				Increase in Avg RUR using TSMGWO				
								Heuristic methods		Meta heuristic methods		
		FCFS	MT	PSO	GA	WOA	TSMGWO	Over FCFS	Over MT	Over PSO	Over GA	Over WOA
GoCJ	100	0.08	0.12	0.15	0.25	0.42	0.45	462.50 %	275.00 %	200.00 %	80.00 %	7.14 %
	200	0.15	0.18	0.25	0.35	0.55	0.59	293.33 %	227.78 %	136.00 %	68.57 %	7.27 %
	300	0.2	0.23	0.38	0.4	0.45	0.5	150.00 %	117.39 %	31.58 %	25.00 %	11.11 %
	400	0.25	0.3	0.4	0.51	0.7	0.75	200.00 %	150.00 %	87.50 %	47.06 %	7.14 %
	500	0.2	0.24	0.44	0.55	0.73	0.81	305.00 %	237.50 %	84.09 %	47.27 %	10.96 %
	600	0.29	0.33	0.52	0.41	0.81	0.85	193.10 %	157.58 %	63.46 %	107.32 %	4.94 %
	700	0.34	0.37	0.55	0.58	0.82	0.88	158.82 %	137.84 %	60.00 %	51.72 %	7.32 %
	800	0.36	0.38	0.63	0.61	0.85	0.9	150.00 %	136.84 %	42.86 %	47.54 %	5.88 %
	900	0.38	0.55	0.68	0.64	0.88	0.93	144.74 %	69.09 %	36.76 %	45.31 %	5.68 %
	1000	0.4	0.44	0.71	0.67	0.9	0.95	137.50 %	115.91 %	33.80 %	41.79 %	5.56 %
	1100	0.41	0.43	0.75	0.7	0.92	0.97	136.59 %	125.58 %	29.33 %	38.57 %	5.43 %
	1200	0.45	0.5	0.78	0.75	0.95	0.98	117.78 %	96.00 %	25.64 %	30.67 %	3.16 %
Synthetic	100	0.12	0.14	0.19	0.29	0.46	0.49	308.33 %	250.00 %	157.89 %	68.97 %	6.52 %
	200	0.19	0.22	0.29	0.39	0.59	0.63	231.58 %	186.36 %	117.24 %	61.54 %	6.78 %
	300	0.24	0.27	0.42	0.44	0.49	0.54	125.00 %	100.00 %	28.57 %	22.73 %	10.20 %
	400	0.29	0.34	0.44	0.55	0.74	0.79	172.41 %	132.35 %	79.55 %	43.64 %	6.76 %
	500	0.24	0.28	0.48	0.59	0.77	0.85	254.17 %	203.57 %	77.08 %	44.07 %	10.39 %
	600	0.33	0.37	0.56	0.45	0.85	0.89	169.70 %	140.54 %	58.93 %	97.78 %	4.71 %
	700	0.38	0.41	0.59	0.62	0.86	0.92	142.11 %	124.39 %	55.93 %	48.39 %	6.98 %
	800	0.4	0.42	0.67	0.65	0.87	0.94	135.00 %	123.81 %	40.30 %	44.62 %	8.05 %
	900	0.42	0.59	0.72	0.68	0.88	0.95	126.19 %	61.02 %	31.94 %	39.71 %	7.95 %
	1000	0.44	0.48	0.75	0.71	0.9	0.96	118.18 %	100.00 %	28.00 %	35.21 %	6.67 %
	1100	0.45	0.47	0.79	0.74	0.91	0.97	115.56 %	106.38 %	22.78 %	31.08 %	6.59 %
	1200	0.49	0.54	0.82	0.79	0.93	0.98	100.00 %	81.48 %	19.51 %	24.05 %	5.38 %
HSCP	u - c - hi hi	0.56	0.63	0.76	0.79	0.91	0.99	76.79 %	57.14 %	30.26 %	25.32 %	8.79 %
	u - c - hi lo	0.44	0.56	0.65	0.69	0.86	0.94	113.64 %	67.86 %	44.62 %	36.23 %	9.30 %
	u - c - lo hi	0.23	0.31	0.52	0.41	0.68	0.76	230.43 %	145.16 %	46.15 %	85.37 %	11.76 %
	u - c - lo lo	0.09	0.13	0.26	0.16	0.38	0.46	411.11 %	253.85 %	76.92 %	187.50 %	21.05 %
	u - i - hi hi	0.52	0.62	0.72	0.76	0.93	0.98	88.46 %	58.06 %	36.11 %	28.95 %	5.38 %
	u - i - hi lo	0.42	0.51	0.62	0.64	0.86	0.91	116.67 %	78.43 %	46.77 %	42.19 %	5.81 %
	u - i - lo hi	0.21	0.39	0.56	0.45	0.77	0.82	290.48 %	110.26 %	46.43 %	82.22 %	6.49 %
	u - i - lo lo	0.16	0.19	0.36	0.26	0.55	0.6	275.00 %	215.79 %	66.67 %	130.77 %	9.09 %
	u - s - hi hi	0.5	0.59	0.68	0.72	0.89	0.96	92.00 %	62.71 %	41.18 %	33.33 %	7.87 %
	u - s - hi lo	0.37	0.47	0.6	0.57	0.82	0.89	140.54 %	89.36 %	48.33 %	56.14 %	8.54 %
	u - s - lo hi	0.3	0.35	0.42	0.53	0.79	0.86	186.67 %	145.71 %	104.76 %	62.26 %	8.86 %
	u - s - lo lo	0.15	0.21	0.41	0.39	0.44	0.51	240.00 %	142.86 %	24.39 %	30.77 %	15.91 %

69.09%, 25.64%, 25%, and 3.16% increase in avg-ARUR has been achieved using the TSMGWO approach over the FCFS method, MT method, PSO method, GA method, and WOA method, respectively using the different number of tasks.

For the synthetic data set, Table 8 shows that the TSMGWO approach can result in a maximum increase in resource utilization up to 308.33% over FCFS method, 250% over MT method, 157.89% over PSO, 97.78% over GA, and 10.39% over WOA method for 100 tasked cloud workload. It can also be observed that a minimum 100%, 61.02%, 19.51%, 22.73%, and 4.71% increase in resource utilization has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method, and WOA method respectively using the different number of tasks using the synthetic data set. Similarly, for 512 X 16 data instance of HSCP data set, Table 8 shows that the TSMGWO approach can result in a maximum increase in resource utilization up to 411.11% over FCFS method for *u_c_lolo* data pattern, 253.85% over MT method for *u_c_lolo* data pattern, 104.76% over PSO for *u_c_lohi* data pattern, 187.5% over GA for *u_c_lolo* data pattern and 21.05% over WOA method for *u_c_lolo* data pattern of HSCP cloud workload. It can also be observed that a minimum 76.79%, 57.14%, 24.39%, 25.32%, and 5.38% increase in resource utilization has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method, and WOA method respectively using the different number of tasks based on HSCP data set.

It can be concluded from Table 8 that there is a large increase in average resource utilization for workload having less number of tasks in comparison to more number of tasks using the TSMGWO approach over the other methods. The main reason behind the better performance of the TSMGWO approach is its improved optimizing ability over the other methods. Results prove that TSMGWO approach increases resource utilization by a considerable amount, leading to better performance of the cloud computing environment over the existing heuristic and metaheuristic methods using benchmark data sets.

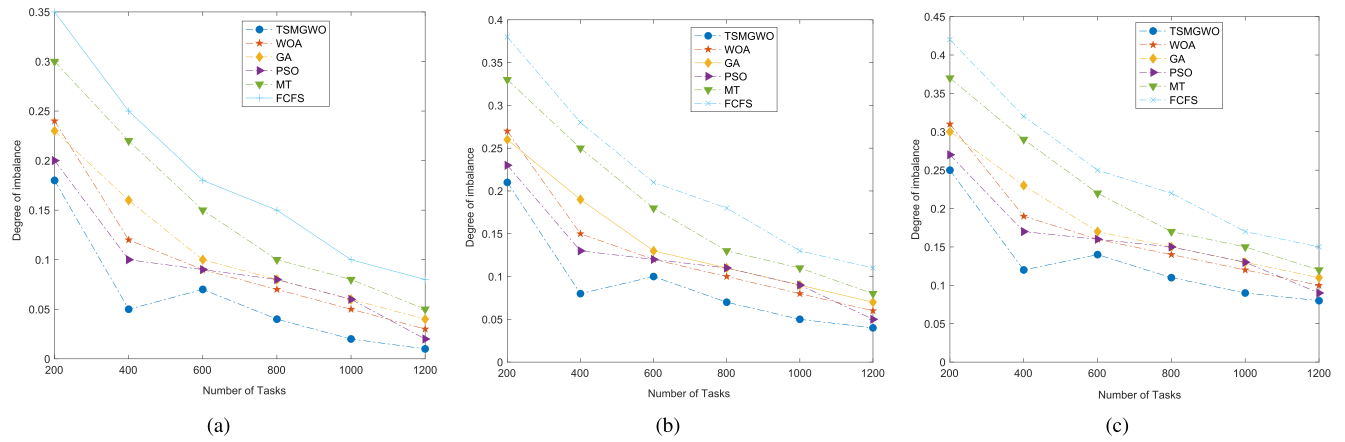

FIGURE 8. Degree of imbalance based comparative results of (a) using the GoCJ data set, (b) using the synthetic data set, and (c) using the HSCP data set.

TABLE 9. Reduction in the degree of imbalance using the TSMGWO approach over heuristic and meta heuristic methods.

Dataset	Method / No.of tasks	Heuristic methods		Meta heuristic methods				Reduction in degree of imbalance using TSMGWO				
								Heuristic methods		Meta heuristic methods		
		FCFS	MT	PSO	GA	WOA	TSMGWO	Over FCFS	Over MT	Over PSO	Over GA	Over WOA
GoCJ	200	0.35	0.3	0.2	0.23	0.24	0.18	48.57 %	40.00 %	10.00 %	21.74 %	25.00 %
	400	0.25	0.22	0.1	0.16	0.12	0.05	80.00 %	77.27 %	50.00 %	68.75 %	58.33 %
	600	0.18	0.15	0.09	0.1	0.09	0.07	61.11 %	53.33 %	22.22 %	30.00 %	22.22 %
	800	0.15	0.1	0.08	0.08	0.07	0.04	73.33 %	60.00 %	50.00 %	50.00 %	42.86 %
	1000	0.1	0.08	0.06	0.06	0.05	0.02	80.00 %	75.00 %	66.67 %	66.67 %	60.00 %
	1200	0.08	0.05	0.02	0.04	0.03	0.01	87.50 %	80.00 %	50.00 %	75.00 %	66.67 %
	200	0.38	0.33	0.23	0.26	0.27	0.21	44.74 %	36.36 %	8.70 %	19.23 %	22.22 %
	400	0.28	0.25	0.13	0.19	0.15	0.08	71.43 %	68.00 %	38.46 %	57.89 %	46.67 %
	600	0.21	0.18	0.12	0.13	0.12	0.1	52.38 %	44.44 %	16.67 %	23.08 %	16.67 %
	800	0.18	0.13	0.11	0.11	0.1	0.07	61.11 %	46.15 %	36.36 %	36.36 %	30.00 %
	1000	0.13	0.11	0.09	0.09	0.08	0.05	61.54 %	54.55 %	44.44 %	44.44 %	37.50 %
	1200	0.11	0.08	0.05	0.07	0.06	0.04	63.64 %	50.00 %	20.00 %	42.86 %	33.33 %
Synthetic	200	0.42	0.37	0.27	0.3	0.31	0.25	40.48 %	32.43 %	7.41 %	16.67 %	19.35 %
	400	0.32	0.29	0.17	0.23	0.19	0.12	62.50 %	58.62 %	29.41 %	47.83 %	36.84 %
	600	0.25	0.22	0.16	0.17	0.16	0.14	44.00 %	36.36 %	12.50 %	17.65 %	12.50 %
	800	0.22	0.17	0.15	0.15	0.14	0.11	50.00 %	35.29 %	26.67 %	26.67 %	21.43 %
	1000	0.17	0.15	0.13	0.13	0.12	0.09	47.06 %	40.00 %	30.77 %	30.77 %	25.00 %
	1200	0.15	0.12	0.09	0.11	0.1	0.08	47.06 %	40.00 %	30.77 %	30.77 %	25.00 %
	200	0.35	0.3	0.2	0.23	0.24	0.18	48.57 %	40.00 %	10.00 %	21.74 %	25.00 %
	400	0.25	0.22	0.1	0.16	0.12	0.05	80.00 %	77.27 %	50.00 %	68.75 %	58.33 %
	600	0.18	0.15	0.09	0.1	0.09	0.07	61.11 %	53.33 %	22.22 %	30.00 %	22.22 %
	800	0.15	0.1	0.08	0.08	0.07	0.04	73.33 %	60.00 %	50.00 %	50.00 %	42.86 %
	1000	0.1	0.08	0.06	0.06	0.05	0.02	80.00 %	75.00 %	66.67 %	66.67 %	60.00 %
	1200	0.08	0.05	0.02	0.04	0.03	0.01	87.50 %	80.00 %	50.00 %	75.00 %	66.67 %
HSCP	200	0.38	0.33	0.23	0.26	0.27	0.21	44.74 %	36.36 %	8.70 %	19.23 %	22.22 %
	400	0.28	0.25	0.13	0.19	0.15	0.08	71.43 %	68.00 %	38.46 %	57.89 %	46.67 %
	600	0.21	0.18	0.12	0.13	0.12	0.1	52.38 %	44.44 %	16.67 %	23.08 %	16.67 %
	800	0.18	0.13	0.11	0.11	0.1	0.07	61.11 %	46.15 %	36.36 %	36.36 %	30.00 %
	1000	0.13	0.11	0.09	0.09	0.08	0.05	61.54 %	54.55 %	44.44 %	44.44 %	37.50 %
	1200	0.11	0.08	0.05	0.07	0.06	0.04	63.64 %	50.00 %	20.00 %	42.86 %	33.33 %
	200	0.42	0.37	0.27	0.3	0.31	0.25	40.48 %	32.43 %	7.41 %	16.67 %	19.35 %
	400	0.32	0.29	0.17	0.23	0.19	0.12	62.50 %	58.62 %	29.41 %	47.83 %	36.84 %
	600	0.25	0.22	0.16	0.17	0.16	0.14	44.00 %	36.36 %	12.50 %	17.65 %	12.50 %
	800	0.22	0.17	0.15	0.15	0.14	0.11	50.00 %	35.29 %	26.67 %	26.67 %	21.43 %
	1000	0.17	0.15	0.13	0.13	0.12	0.09	47.06 %	40.00 %	30.77 %	30.77 %	25.00 %
	1200	0.15	0.12	0.09	0.11	0.1	0.08	47.06 %	40.00 %	30.77 %	30.77 %	25.00 %

3) DEGREE OF IMBALANCE BASED EXPERIMENTAL RESULTS

The degree of imbalance determines the imbalanced load distribution in context to its execution as per the capacities of virtual machines. It is computed using Eq. (5). A lower value of the degree of imbalance metric indicates the more balanced distribution of workload among virtual machines.

Figure 5 represents the degree of imbalance obtained and compared with identified heuristic methods and metaheuristic methods.

It can be observed from Figure 8 (a) – (c) that the proposed TSGWO task scheduling approach exhibits better performance by resulting in lower values of degree of imbalance in comparison to the existing heuristic and metaheuristic methods. Tables 9 shows the reduction in the degree of imbalance using different benchmark data sets. It can be observed from Table 9 (a) that the TSMGWO approach can result in a maximum reduction of up to 87.5% over FCFS method for 1200 tasked work load, 80.00% over MT method for 1200 tasked work load, 66.67% over PSO for 1000 tasked workload, 75% over GA for 1200 tasked work load, and 66.67% over WOA method for 1200 tasked cloud workload. It can also be observed that a minimum 48.57%, 10%, 10%, 21.74%, and 22.22% reduction in the degree of imbalance has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method, and WOA method respectively using all possible number of tasks considered in this work using GoCJ data set.

For synthetic data set, Table 9 shows that the TSMGWO approach can result in a maximum reduction of the degree of imbalance up to 71.43% over FCFS method for 400 tasked workload, 68.00% over MT method for 400 tasked workload, 44.44% over PSO for 1000 tasked workload, 57.89% over GA for 400 tasked workload and 46.67% over WOA method for 400 tasked cloud workload. It can also be observed that a minimum 44.74%, 36.36%, 8.7%, 19.23%, and 16.67% reduction in the degree of imbalance has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method, and WOA method respectively using all possible number of tasks considered in this work using synthetic data set.

Similarly, for 512 X16 data instance of HSCP data set, Table 9 shows that the TSMGWO approach can result into a maximum reduction of the degree of imbalance up to 62.50% over FCFS method for 400 tasked workload, 58.62% over MT method for 400 tasked workload, 30.77% over PSO for 1000 tasked workload, 47.83% over GA for 400 tasked workload and 36.84% over WOA method for 400 tasked cloud workload. It can also be observed that a minimum 40.48%, 32.43%, 7.41%, 16.67%, and 12.50% reduction in degree of imbalance has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method and WOA method respectively using all possible number of tasks considered in this work using HSCP data set.

It can be concluded from Table 9 that there is a considerable reduction in the degree of imbalance for workload having more number of tasks in comparison to work load with less number of tasks using the TSMGWO approach over other methods considered in this work. Results demonstrated that the TSMGWO approach outperforms in reducing degree of imbalance by a considerable amount, leading to better performance of cloud computing environment over the existing heuristic and meta heuristic methods using benchmark data sets.

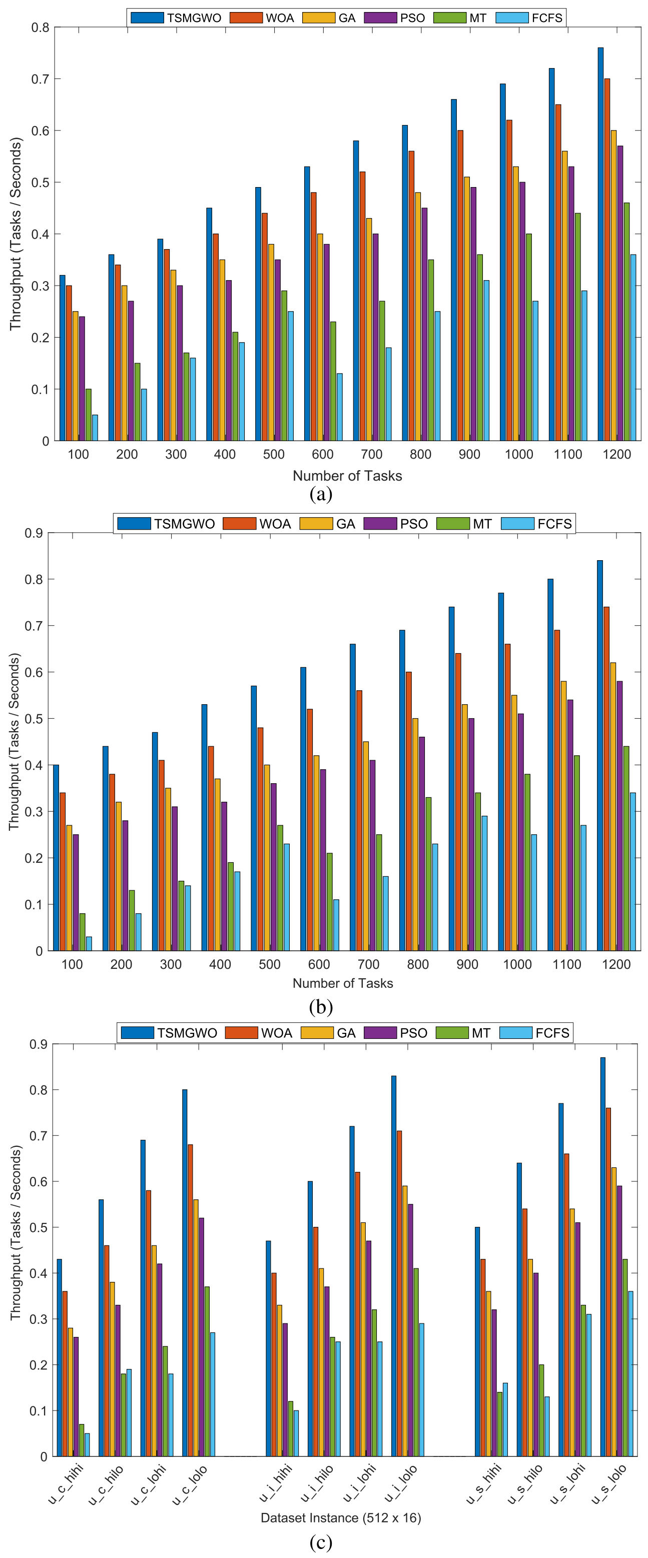

FIGURE 9. Throughput-based comparative results of (a) using the GoCJ data set, (b) using the synthetic data set, and (c) using the HSCP data set.

TABLE 10. Increase in Throughput using the TSMGWO approach over heuristic and meta heuristic methods.

Dataset	Method / No.of tasks	Heuristic methods		Meta heuristic methods				Increase in Throughput using TSMGWO				
								Heuristic methods		Meta heuristic methods		
		FCFS	MT	PSO	GA	WOA	TSMGWO	Over FCFS	Over MT	Over PSO	Over GA	Over WOA
GoCJ	100	0.05	0.1	0.24	0.25	0.3	0.32	540.00 %	220.00 %	33.33 %	28.00 %	6.67 %
	200	0.1	0.15	0.27	0.3	0.34	0.36	260.00 %	140.00 %	33.33 %	20.00 %	5.88 %
	300	0.16	0.17	0.3	0.33	0.37	0.39	143.75 %	129.41 %	30.00 %	18.18 %	5.41 %
	400	0.19	0.21	0.31	0.35	0.4	0.45	136.84 %	114.29 %	45.16 %	28.57 %	12.50 %
	500	0.25	0.29	0.35	0.38	0.44	0.49	96.00 %	68.97 %	40.00 %	28.95 %	11.36 %
	600	0.13	0.23	0.38	0.4	0.48	0.53	307.69 %	130.43 %	39.47 %	32.50 %	10.42 %
	700	0.18	0.27	0.4	0.43	0.52	0.58	222.22 %	114.81 %	45.00 %	34.88 %	11.54 %
	800	0.25	0.35	0.45	0.48	0.56	0.61	144.00 %	74.29 %	35.56 %	27.08 %	8.93 %
	900	0.31	0.36	0.49	0.51	0.6	0.66	112.90 %	83.33 %	34.69 %	29.41 %	10.00 %
	1000	0.27	0.4	0.5	0.53	0.62	0.69	155.56 %	72.50 %	38.00 %	30.19 %	11.29 %
	1100	0.29	0.44	0.53	0.56	0.65	0.72	148.28 %	63.64 %	35.85 %	28.57 %	10.77 %
	1200	0.36	0.46	0.57	0.6	0.7	0.76	111.11 %	65.22 %	33.33 %	26.67 %	8.57 %
Synthetic	100	0.03	0.08	0.25	0.27	0.34	0.4	1233.33 %	400.00 %	60.00 %	48.15 %	17.65 %
	200	0.08	0.13	0.28	0.32	0.38	0.44	450.00 %	238.46 %	57.14 %	37.50 %	15.79 %
	300	0.14	0.15	0.31	0.35	0.41	0.47	235.71 %	213.33 %	51.61 %	34.29 %	14.63 %
	400	0.17	0.19	0.32	0.37	0.44	0.53	211.76 %	178.95 %	65.63 %	43.24 %	20.45 %
	500	0.23	0.27	0.36	0.4	0.48	0.57	147.83 %	111.11 %	58.33 %	42.50 %	18.75 %
	600	0.11	0.21	0.39	0.42	0.52	0.61	454.55 %	190.48 %	56.41 %	45.24 %	17.31 %
	700	0.16	0.25	0.41	0.45	0.56	0.66	312.50 %	164.00 %	60.98 %	46.67 %	17.86 %
	800	0.23	0.33	0.46	0.5	0.6	0.69	200.00 %	109.09 %	50.00 %	38.00 %	15.00 %
	900	0.29	0.34	0.5	0.53	0.64	0.74	155.17 %	117.65 %	48.00 %	39.62 %	15.63 %
	1000	0.25	0.38	0.51	0.55	0.66	0.77	208.00 %	102.63 %	50.98 %	40.00 %	16.67 %
	1100	0.27	0.42	0.54	0.58	0.69	0.8	196.30 %	90.48 %	48.15 %	37.93 %	15.94 %
	1200	0.34	0.44	0.58	0.62	0.74	0.84	147.06 %	90.91 %	44.83 %	35.48 %	13.51 %
HSCP	u - c - hi hi	0.05	0.07	0.36	0.26	0.28	0.43	760.00 %	514.29 %	19.44 %	65.38 %	53.57 %
	u - c – hi lo	0.19	0.18	0.46	0.33	0.38	0.56	194.74 %	211.11 %	21.74 %	69.70 %	47.37 %
	u - c – lo hi	0.18	0.24	0.58	0.42	0.46	0.69	283.33 %	187.50 %	18.97 %	64.29 %	50.00 %
	u - c – lo lo	0.27	0.37	0.68	0.52	0.56	0.8	196.30 %	116.22 %	17.65 %	53.85 %	42.86 %
	u - i – hi hi	0.1	0.12	0.4	0.29	0.33	0.47	370.00 %	291.67 %	17.50 %	62.07 %	42.42 %
	u - i – hi lo	0.25	0.26	0.5	0.37	0.41	0.6	140.00 %	130.77 %	20.00 %	62.16 %	46.34 %
	u - i – lo hi	0.25	0.32	0.62	0.47	0.51	0.72	188.00 %	125.00 %	16.13 %	53.19 %	41.18 %
	u - i – lo lo	0.29	0.41	0.71	0.55	0.59	0.83	186.21 %	102.44 %	16.90 %	50.91 %	40.68 %
	u - s – hi hi	0.16	0.14	0.43	0.32	0.36	0.5	212.50 %	257.14 %	16.28 %	56.25 %	38.89 %
	u - s – hi lo	0.13	0.2	0.54	0.4	0.43	0.64	392.31 %	220.00 %	18.52 %	60.00 %	48.84 %
	u - s – lo hi	0.31	0.33	0.66	0.51	0.54	0.77	148.39 %	133.33 %	16.67 %	50.98 %	42.59 %
	u - s – lo lo	0.36	0.43	0.76	0.59	0.63	0.87	141.67 %	102.33 %	14.47 %	47.46 %	38.10 %

4) THROUGHPUT BASED EXPERIMENTAL RESULTS

Throughput refers to the number of tasks executed during a unit time in the cloud environment. It is computed using Eq. (9). The objective is to maximize the value of Throughput. Figure 9 represents the Throughput based experimental results obtained in this work and compared with identified heuristic methods and metaheuristic methods.

It can be observed from Figure 9 (a) – (c) that the proposed TSGWO task scheduling approach exhibits better performance by resulting in high values of throughput in comparison to the existing heuristic and metaheuristic methods.

Tables 10 shows the increase in throughput using different benchmark data sets. It can be observed from Table 10 that the TSMGWO approach can result in a maximum increase in throughput up to 540% over FCFS method for 100 tasked workload, 220% over MT method for 100 tasked workload, 45.16% over PSO for 400 tasked workload, 34.88% over GA for 700 tasked workload, and 12.50% over WOA methods for 400 tasked cloud workload of GoCJ data set. It can also be observed that a minimum 96.00%, 63.64%, 30.00%, 18.18%, and 5.41% increase in throughput has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method, and WOA method, respectively using a different number of tasks considered in this work. It can be concluded from Table 10 that there is a large increase in throughput for workload having a smaller number of tasks in comparison to workload having a greater number of tasks using the TSMGWO approach over other methods considered in this work. Results delineate that the TSMGWO approach outperforms in increasing throughput by a considerable amount, leading to better performance of the cloud computing environment over the existing heuristic and metaheuristic methods using benchmark GoCJ data set.

For synthetic data set, Table 10 shows that the TSMGWO approach can result in a maximum increase in throughput up to 1233.33% over FCFS method for 100 tasked workload, 400% over MT method for 100 tasked workload, 65.63% over PSO for 400 tasked workload, 48.15% over GA for 100 tasked workload, and 20.45% over WOA method for 400 tasked cloud work load. It can also be noticed that a minimum 147.06%, 90.48%, 44.83%, 34.29%, and 13.51% increase in throughput has been achieved using the TSMGWO approach over FCFS method, MT method, PSO method, GA method, and WOA method, respectively using a different number of tasks considered in this work using synthetic data set.

Similarly, for 512 X16 data instance of HSCP data set, Table 10 shows that the TSMGWO approach can result in a maximum increase in throughput up to 760% over FCFS method for *u_c_hihi* data pattern, 514.29% over MT method for *u_c_hihi* data pattern, 21.74% over PSO for *u_c_hilo* data pattern, 69.70% over GA for *u_c_hilo* data pattern and 53.57% over WOA method for *u_c_hihi* data pattern of HSCP data set. It can also be noticed that a minimum 140%, 102.33%, 14.47%, 47.46%, and 38.10% increase in throughput has been achieved using THE TSMGWO approach over FCFS method, MT method, PSO method, GA method, and WOA method respectively using a different number of tasks considered in this work using HSCP data set.

It can be concluded from Table 10 that there is a considerable increase in throughput for workload having a smaller number of tasks in comparison to workload with a high number of tasks using the TSMGWO approach over other methods considered in this work. Results demonstrated that the TSMGWO approach increases the throughput by a considerable amount, leading to better performance of cloud computing environment over the existing heuristic and metaheuristic methods using benchmark data sets.

VII. CONCLUSION

Optimal mapping of user-submitted tasks to the virtual machines in the cloud computing environment is considered one of the most challenging tasks. This mapping is critical in fulfilling the computing and resource requirements of high-performance applications while achieving the scheduling objectives like maximizing Throughput, minimizing makespan, maximizing resource utilization, and balancing the cloud workload distribution among virtual machines in a cloud data centre.

This article proposes a metaheuristic Grey wolf optimizer-based approach called the TSMGWO for finding an optimal or near-optimal solution to the task scheduling problem by considering multiple conflicting objectives of makespan, resource utilization, Degree of imbalance, and Throughput simultaneously. The primary motivation for using the grey wolf optimization method is the to exploit social hierarchy of grey wolves and their organized structure while hunting the prey. Grey wolf optimizer considers exploitation and exploration equally while determining the best solution in the search space. The performance analysis of the TSMGWO approach has been done using three sets of experiments using different benchmark data sets GOCJ data set, Synthetic data set, and HSCP data set. The results were compared with two heuristic methods, FCFS and MT; and three metaheuristic methods, PSO, GA, and WOA. From the results of these experiments, it can be concluded that the proposed TSMGWO approach outperforms the identified heuristic and metaheuristic methods to a considerable level. The obtained results indicate that the TSMGWO approach can lead to a maximum reduction of makespan up to 67.52% over FCFS method, 60.93%, 52.33% over PSO, 38.05% over GA, and 23.22% over WOA methods for 100 tasked cloud workload using GoCJ data set. Whereas, a maximum reduction of makespan up to 60.95% over FCFS method, 55.79% over MT method, 47.04% over PSO, 33.38% over GA and 19.91% over WOA method can be obtained using the TSMGWO approach based on synthetic data set. Similarly, the TSMGWO approach can reduce makespan up to 27.03% over MT method, 18.95% over PSO, 11.90% over GA and 7.5% over WOA method based on HSCP cloud workload. A notable improvement of results in resource utilization, Degree of imbalance and Throughput of the cloud has been observed for the identified benchmark data sets. The experimental results demonstrate that the proposed TSMGWO approach is a useful and practical task scheduler that maximises resource utilization and Throughput of the cloud while minimizing the makespan and Degree of imbalance in the cloud.

In our future work, we will focus on enhancing the TSMGWO approach's performance by using parallel programming and concentrating on other parameters of the cloud environment like memory usage during peak loads etc.

REFERENCES

[1] Z. Zhong, K. Chen, X. Zhai, and S. Zhou, "Virtual machine-based task scheduling algorithm in a cloud computing environment," *Tsinghua Sci. Technol.*, vol. 21, no. 6, pp. 660–667, Dec. 2016.

[2] B. Keshanchi, A. Souri, and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing," *J. Syst. Softw.*, vol. 124, pp. 1–21, Feb. 2017.

[3] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019.

[4] K. Wu, P. Lu, and Z. Zhu, "Distributed online scheduling and routing of multicast-oriented tasks for profit-driven cloud computing," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 684–687, Apr. 2016.

[5] X. Zhu, C. Chen, L. T. Yang, and Y. Xiang, "ANGEL: Agent-based scheduling for real-time tasks in virtualized clouds," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3389–3403, Dec. 2015.

[6] G. Lovász, F. Niedermeier, and H. de Meer, "Performance tradeoffs of energy-aware virtual machine consolidation," *Cluster Comput.*, vol. 16, no. 3, pp. 481–496, Sep. 2013.

[7] H. He, G. Xu, S. Pang, and Z. Zhao, "AMTS: Adaptive multi-objective task scheduling strategy in cloud computing," *China Commun.*, vol. 13, no. 4, pp. 162–171, Apr. 2016.

[8] D. Alsadie, Z. Tari, E. J. Alzahrani, and A. Y. Zomaya, "Dynamic resource allocation for an energy efficient VM architecture for cloud computing," in *Proc. Australas. Comput. Sci. Week Multiconference*, Jan. 2018, pp. 1–8.

[9] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Comput.*, pp. 1–19, Mar. 2020.

[10] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Comput. Methods Appl. Mech. Eng.*, vol. 376, Apr. 2021, Art. no. 113609.

[11] G. Natesan and A. Chokkalingam, "Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm," *ICT Exp.*, vol. 5, no. 2, pp. 110–114, Jun. 2019.

[12] M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowl.-Based Syst.*, vol. 169, pp. 39–52, Apr. 2019.

[13] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manage.*, vol. 23, no. 3, pp. 567–619, Jul. 2015.

[14] B. V. Natesha, N. Kumar Sharma, S. Domanal, and R. M. R. Guddeti, "GWOTS: Grey wolf optimization based task scheduling at the green cloud data center," in *Proc. 14th Int. Conf. Semantics, Knowl. Grids (SKG)*, Sep. 2018, pp. 181–187.

[15] L. Abualigah, M. Shehab, A. Diabat, and A. Abraham, "Selection scheme sensitivity for a hybrid salp swarm algorithm: Analysis and applications," *Eng. Comput.*, vol. 7, pp. 1–27, Jul. 2020.

[16] L. Abualigah, "Multi-verse optimizer algorithm: A comprehensive survey of its results, variants, and applications," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12381–12401, Aug. 2020.

[17] X. Sheng and Q. Li, "Template-based genetic algorithm for QoS-aware task scheduling in cloud computing," in *Proc. Int. Conf. Adv. Cloud Big Data (CBD)*, Aug. 2016, pp. 25–30.

[18] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, "A Johnson's-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 597–610, Sep. 2019.

[19] M. Akbari, H. Rashidi, and S. H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems," *Eng. Appl. Artif. Intell.*, vol. 61, pp. 35–46, May 2017.

[20] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *Proc. 6th Annu. Chinagrid Conf.*, Aug. 2011, pp. 3–9.

[21] Z. Liu and X. Wang, "A PSO-based algorithm for load balancing in virtual machines of cloud computing environment," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2012, pp. 142–147.

[22] A. Gupta and R. Garg, "Load balancing based task scheduling with ACO in cloud computing," in *Proc. Int. Conf. Comput. Appl. (ICCA)*, Sep. 2017, pp. 174–179.

[23] S. Zhan and H. Huo, "Improved PSO-based task scheduling algorithm in cloud computing," *J. Inf. Comput. Sci.*, vol. 9, no. 13, pp. 3821–3829, 2012.

[24] H. S. Al-Olimat, M. Alam, R. Green, and J. Kwan Lee, "Cloudlet scheduling with particle swarm optimization," in *Proc. 5th Int. Conf. Commun. Syst. Netw. Technol.*, Apr. 2015, pp. 991–995.

[25] L. Abualigah, "Group search optimizer: A nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications," *Neural Comput. Appl.*, vol. 13, pp. 1–24, Jul. 2020.

[26] B. Mondal, K. Dasgupta, and P. Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach," *Procedia Technol.*, vol. 4, pp. 783–789, Jun. 2012.

[27] M. Abdullahi, M. A. Ngadi, and S. M. Abdulhamid, "Symbiotic organism search optimization based task scheduling in cloud computing environment," *Future Gener. Comput. Syst.*, vol. 56, pp. 640–650, Mar. 2016.

[28] S.-S. Kim, J.-H. Byeon, H. Yu, and H. Liu, "Biogeography-based optimization for optimal job scheduling in cloud computing," *Appl. Math. Comput.*, vol. 247, pp. 266–280, Nov. 2014.

[29] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Gener. Comput. Syst.*, vol. 51, pp. 61–71, Oct. 2015.

[30] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *IEEE Access*, vol. 3, pp. 2687–2699, 2015.

[31] G. N. Reddy and S. P. Kumar, "Multi objective task scheduling algorithm for cloud computing using whale optimization technique," in *Proc. Int. Conf. Next Gener. Comput. Technol.* Cham, Switzerland: Springer, 2017, pp. 286–297.

[32] K. Sreenu and M. Sreelatha, "W-scheduler: Whale optimization for task scheduling in cloud computing," *Cluster Comput.*, vol. 22, no. 1, pp. 1087–1098, Jan. 2019.

[33] M. Sanaj and P. J. Prathap, "An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment," *Mater. Today, Proc.*, vol. 37, pp. 3199–3208, Oct. 2020.

[34] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[35] Y. H. Zweiri, J. F. Whidborne, and L. D. Seneviratne, "A three-term backpropagation algorithm," *Neurocomputing*, vol. 50, pp. 305–318, Jan. 2003.

[36] M. Hamada and M. Hassan, "Artificial neural networks and particle swarm optimization algorithms for preference prediction in multi-criteria recommender systems," *Informatics*, vol. 5, no. 2, p. 25, 2018.

[37] A. Alzaqebah, R. Al-Sayyed, and R. Masadeh, "Task scheduling based on modified grey wolf optimizer in cloud computing environment," in *Proc. 2nd Int. Conf. new Trends Comput. Sci. (ICTCS)*, Oct. 2019, pp. 1–6.

[38] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, and M. Zivkovic, "Task scheduling in cloud computing environment by grey wolf optimizer," in *Proc. 27th Telecommun. Forum (TELFOR)*, Nov. 2019, pp. 1–4.

[39] G. Natesan and A. Chokkalingam, "Opposition learning-based grey wolf optimizer algorithm for parallel machine scheduling in cloud environment," *Int. J. Intell. Eng. Syst.*, vol. 10, no. 1, pp. 186–195, 2017.

[40] M. Abdel-Basset, D. El-Shahat, K. Deb, and M. Abouhawwash, "Energy-aware whale optimization algorithm for real-time task scheduling in multiprocessor systems," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106349.

[41] Y. Yang, B. Yang, S. Wang, T. Jin, and S. Li, "An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing," *Appl. Soft Comput.*, vol. 87, Feb. 2020, Art. no. 106003.

[42] R. Jena, "Multi objective task scheduling in cloud environment using nested PSO framework," *Procedia Comput. Sci.*, vol. 57, pp. 1219–1227, Jan. 2015.

[43] K. Jiang, H. Ni, P. Sun, and R. Han, "An improved binary grey wolf optimizer for dependent task scheduling in edge computing," in *Proc. 21st Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2019, pp. 182–186.

[44] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. D. S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Syst. Appl.*, vol. 47, pp. 106–119, Apr. 2016.

[45] H. Faris, I. Aljarah, M. A. Al-Betar, and S. Mirjalili, "Grey wolf optimizer: A review of recent variants and applications," *Neural Comput. Appl.*, vol. 30, no. 2, pp. 413–435, Jul. 2018.

[46] Y. Yang, B. Yang, S. Wang, W. Liu, and T. Jin, "An improved grey wolf optimizer algorithm for energy-aware service composition in cloud manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 105, nos. 7–8, pp. 3079–3091, Dec. 2019.

[47] N. Gobalakrishnan and C. Arun, "A new multi-objective optimal programming model for task scheduling using genetic gray wolf optimization in cloud computing," *Comput. J.*, vol. 61, no. 10, pp. 1523–1536, Oct. 2018.

[48] A. Hussain, M. Aleem, M. A. Iqbal, and M. A. Islam, "Investigation of cloud scheduling algorithms for resource utilization using CloudSim," *Comput. Informat.*, vol. 38, no. 3, pp. 525–554, 2019.

[49] T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2014, pp. 658–664.

[50] A. Hussain and M. Aleem, "GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures," *Data*, vol. 3, no. 4, p. 38, Sep. 2018.

[51] Braun. *Data Set*. Accessed: Mar. 31, 2017. [Online]. Available: https://github.com/chgogos/hcsp/tree/master/512x16

[52] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[53] N. A. Mehdi, A. Mamat, H. Ibrahim, and S. K. Subramaniam, "Impatient task mapping in elastic cloud using genetic algorithm," *J. Comput. Sci.*, vol. 7, no. 6, p. 877, 2011.

[54] S. Behzad, R. Fotohi, and M. Effatparvar, "Queue based job scheduling algorithm for cloud computing," *Int. Res. J. Appl. Basic Sci.*, vol. 4, no. 11, pp. 3785–3790, 2011.

DEAFALLAH ALSADIE (Graduate Student Member, IEEE) received the B.Sc. degree in computer science from Taibah University, in 2007, the M.A.Sc. degree in computer science from Latrobe University, Australia, in 2011, and the Ph.D. degree in computer science from RMIT University, Melbourne, Australia, in 2019. He is currently an Assistant Professor with the Department of Information Systems, College of Computers and Information Systems, Umm Al-Qura University, Saudi Arabia. His research interests include scheduling and resource allocation for parallel and distributed computing systems, data centers, edge computing, and the Internet of Things.

• • •