Peer-to-Peer Networking and Applications (2023) 16:2560–2573
https://doi.org/10.1007/s12083-023-01536-3

Task scheduling based on minimization of makespan and energy consumption using binary GWO algorithm in cloud environment

Gobalakrishnan Natesan[1] · N. Manikandan[2] · K. Pradeep[3] · L. Sherly Puspha Annabel[4]

Received: 4 April 2022 / Accepted: 26 July 2023 / Published online: 25 August 2023

Abstract
The Cloud environment had been the go-to for many users recently. Once request from users get submitted, cloud resources are put into action to fulfill the request. Scheduling is the primary task in cloud that needs to be up-to the mark for completing the requests swiftly. Multiple dynamic requests are submitted simultaneously by cloud users that necessitates precise and prompt scheduling in cloud. Scheduling in cloud may be hampered by various constraints, take for example the various QoS parameters that needs to be upheld. Though many researchers had proposed solutions for scheduling in cloud, improvisations can still be made by combining several QoS parameters that help attain optimized scheduling in cloud to boost the overall cloud performance. In this paper, we had proposed a Binary Grey Wolf Optimization (BGWO) algorithm to optimize the scheduling activity in cloud computing environment. The BGWO is a multi-heuristic algorithm where tasks are scheduled based on a fitness function, explicitly designed for achieving optimization goal. The fitness function that had been designed comprises of three prime parameters namely, the degree of imbalance (DoI), energy consumption and makespan. The performance efficiency of the proposed BGWO had been ascertained by comparing it with Oppositional based Grey Wolf Optimization algorithm (OGWO) and Mean Grey Wolf Optimization algorithm (Mean GWO) with respect to imbalance, energy and makespan parameters. The proposed algorithm had produced a cumulative improvement of 10.13% and 17.4% for makespan, 30.18% and 41.96% for DoI, 8.94% and 14.95% for energy consumption parameters. Detailed comparative results obtained had been described in the Results part of this research article.

Keywords Cloud Computing · Optimization · Meta-heuristics · Scheduling · QoS · Binary-GWO · Degree of Imbalance · Makespan

1 Introduction

Parallel computing and Distributed Computing techniques had been employed in cloud computing for servicing the users through the Internet platform. Ubiquitous-ness had been brought in cloud computing services that adopts the "pay as you use" model [1–3]. Virtualization technology had been exploited by the cloud providers for providing the resources requested by the users through virtual machines (VMs). These virtual machines are in turn put to appropriate use by the service providers for providing application level services to their users. Efficient task scheduling techniques are needed by the service providers to assign the received user tasks to appropriate VM, minimize the response time taken, make optimal use of available resources and satisfy the various quality level constraints as demanded by the users [4, 5]. Hence task scheduling is deemed to be one of the primary activities in the cloud computing infrastructure.

✉ Gobalakrishnan Natesan
gobalakrishnanse@gmail.com

N. Manikandan
macs2005ciet@gmail.com

K. Pradeep
pradeepkrishnadoss@gmail.com

L. Sherly Puspha Annabel
sherlyannabel@gmail.com

[1] Department of Information Technology, Sri Venkateswara College of Engineering, Sriperumpudur, India

[2] Department of Data Science and Business Systems, SRM Institute of Science and Technology, Kattankulathur, India

[3] School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, India

[4] Department of Information Technology, St. Joseph's College of Engineering, Chennai, India

Tasks that are scheduled with respect to real time approximations on a distributed platform had been categorized as NP-complete problems [6]. The problem gets compounded when numerous tasks are executed in the cloud environment. Obtaining solutions to such large problems are computationally intractable [7]. Appropriate solution through heuristics could be obtained for such cases. Several related studies had been carried out by researchers using meta-heuristic algorithms with respect to scheduling in the cloud computing environment [8–10].

In this article, we have proposed an efficient scheduling method that is based on the GWO algorithm. The original GWO algorithm had been improvised using a scheduling technique resulting in the attainment of faster convergence of the optimal solution.

The primary objectives of this proposed research had been enlisted herewith:

1. Designing a fitness function based on multiple QoS parameters to optimize the scheduling performance in heterogenous cloud environment.
2. A multi-heuristic Binary Grey Wolf Optimization (BGWO) had been proposed.
3. The BGWO had been designed to optimize the cloud scheduling performance with respect to degree of imbalance, energy consumption and makespanQoS parameters. The fitness function designed comprises of these parameters.
4. The proposed Binary-GWO performance efficiency had been evaluated by comparing it with OGWO and Mean GWO algorithms.
5. Experiments had been carried out using Cloudsim toolkit. Normal as well as HPC2N datasets of varying instance sizes were used for obtaining the results.

The remainder of this research article had been ordered as follows. A formal background analysis and literature review had been presented in Section 2. The proposed system architecture and definition of the same had been illustrated in Section 3. In Section 4, a detailed description about the proposed BGWO technique had been presented. Simulation results and comparative analysis had been presented in Section 5. Finally in Section 6, conclusion and scope for carrying out the future work had been discussed.

2 Literature work

Golchi et al. [11] had presented an algorithm which was the fusion of the Firefly and Improved particle swarm optimization algorithms. The major focus of the algorithm presented by the authors was to attain enhanced average load for making in addition to the jobs like refining the utilization of the resources and improving the retort time of the tasks.

To provide superlative class of facility, the overall accomplishment time had to be reduced. Li [12] had presented a task scheduling scheme for reducing the overall accomplishment time and vitality depletion. The scheme proposed by the author was controlled by both vitality and time. The author had demonstrated that there existed algorithm for discovering a non-preemptive schedule with the least accomplishment time for an assumed overall vitality depletion limitation and vice versa.

Ismayilov and Topcuoglu [13] had proposed an algorithm namely NN-DNSGA-II to resolve the same. The algorithm proposed is a blend of artificial neural network and NSGA-II. Experimental analysis performed on the Pegasus workflow management system indicated that the proposed scheme outpaced other similar schemes.

Mansouri et al. [14] had proposed a novel task scheduling algorithm which was the fusion of Fuzzy system and the modified Particle Swarm Optimization scheme. The proposed algorithm's major objective was to augment the load balancing and throughput in the cloud. The overall quest ability was improved by four altered swiftness updating schemes and roulette wheel selection scheme. The weakness of the particle swarm optimization technique was overwhelmed by the utilization of crossover and mutation and the fitness was determined by the fuzzy inference system.

In [15], selection of a suitable CPU frequency configuration that maintains equilibrium between pricing and accomplishment time enactment is a major issue. Pietri and Sakellariou had proposed an algorithm to professionally search alternate CPU configurations for an agreed number of resources and to find Pareto-efficient keys for cost and accomplishment of time balance. The proposed algorithm was assessed by simulation by using three diverse valuing models to charge for CPU provisioning based on the allotted CPU frequency and the usage of the scientific workflow applications.

In [16], delivery of cloud services can be improvised by unifying the scheduling of resources and balancing the load. A solution in this regard had been proposed by Priya et al. The authors had proposed a fuzzy based solution in a multi-dimensional perspective with respect to the resource scheduling and load optimization. Experimental result analysis had shown that the proposed technique had attained superior enactment with respect to average triumph rate, resource scheduling efficacy and retort time.

Providing a better Quality of Service in a lucrative manner to the users and efficiently conserve the energy expended for the providers was a tough task. To resolve the same, Stavrinides and Karatza [17] had presented a scheduling scheme for real-time workflow requests. The objective was to deliver appropriateness and vitality efficacy by

compromising the accuracy of the result while maintaining the result excellency of the concluded jobs at an adequate standard and the fiscal need for the accomplishment of the jobs at a realistic level.

The issue of scheduling Bag-of-tasks (BoT) application on hybrid cloud under due date restrictions was articulated as an integer program. To resolve the issue Zhang et al. [18] had presented various heuristics by commissioning an effective task scheduling strategy. In order to enhance the efficiency, two hastening methods were developed. The success and the competence of the presented heuristics were justified by simulation experiments.

Quality parameters like makespan and cost (workflow planning cost) were taken up for optimization by Zhou et al. [19] in the cloud workflow scheduling activity. The proposed solution had been based upon the fuzzy dominance sort based heterogeneous earliest-finish-time (FDHEFT) algorithm. The authors had also integrated the list scheduling heuristic information along with their proposed algorithm. The efficiency of the authors' scheme was demonstrated by experiments conducted using practical and artificial workflows.

A novel approach by combining Imperialist Competitive Algorithm and Firefly algorithm (ICAFA) had been proposed by Kashikolaei et al. [20] for enhancing the cloud scheduling performance. The local search capability of Firefly algorithm had been applied for reinforcing the ICA algorithm. The authors had evaluated the performance of the proposed ICAFA algorithm by comparing it with other existing algorithms with respect to makespan, load balancing, CPU time, planning speed and stability parameters.

A hybrid approach for task scheduling named Modified Genetic and Greedy Algorithm (MGGA) had been proposed by Zhou et al. [21] for increasing the cloud scheduling performance. The authors had hybridized their approach by integrating the Genetic Algorithm and Greedy Algorithm. The authors had evaluated the performance of the proposed ICAFA algorithm by comparing it with other existing algorithms with respect to total completion time, degree of workload balance, QoS delivered and response time.

Ababneh [22] had proposed the novel meta-heuristic approach named hybrid Grey Wolf and Whale Optimization (HGWWO) by amalgamated the Grey Wolf Optimizer (GWO) and the Whale Optimization Algorithm (WOA) for improvising the task scheduling in cloud. The HGWWO approach had been assess by compare with standard GWO and classical WOA algorithms based on makespan, resource utilization, cost and degree of imbalance QoS parameters.

An Energy-Efficient and Reliability based task scheduling (EERS) had been proposed by Medara and Singh [23] for optimizing the scheduling activity in cloud. The authors had integrated the five sub algorithms namely rank calculation method, clustering approach, time distribution method, VM-mapping and slack for providing optimal results for designing their EERS. The performance improvement of the EERS had been evaluated with respect to Energy and reliability.

A hybrid Multi-Verse Optimizer and Genetic Algorithm (MVO-GA) had been introduced by Abualigah and Alkhrabsheh [24] for improving the scheduling action in cloud. The MVO-GA had been evaluated by comparing it with other existing approaches with respect to total expected completion time with crossover and without crossover and expected completion time parameters.

An Elite Learning Harris Hawks Optimizer (ELLHO) had been proposed by Amer et al. [25] for optimizing the cloud scheduling activity by integrating the elite Opposition-based Learning and Harris Hawks Optimizer. The authors had used a host of parameters namely resource utilization, scheduling length, execution cost, throughput, balance degree for comparative purposes while carrying out performance evaluation of their proposed ELLHO algorithm.

Jain and Sharma [26] had hybridized the Quantum-inspired Salp Swarm algorithm (SSA) and Grey Wolf Optimizer (GWO) algorithm to propose a Quantum-inspired Salp Swarm Grey Wolf Algorithm (QSSGWA) for task scheduling in cloud. The quantum operator had been used for initializing the population and an improvised global optimum solution is realized using the hybrid version of SSGWA. The authors had evaluated their proposed QSSGWA with other existing algorithms with respect to provider profit, makespan, task rejection rate, response time, resource utilization and SLA violation rate parameters and had found that the results obtained were highly encouraging.

A novel generative adversarial reinforcement learning scheduling (GARLSched) algorithm had been proposed by Li et al. [27] to enhance the task scheduling activity in cloud. The proposed approach effectively provides a platform to study the reinforcement of deep learning in a large-scale dynamic scheduling. The discriminator architecture embedded in the proposed model not only improvises the learning process but also lends greater stability.

A novel Evolutionary Game Fuzzy Improved Red Fox Optimization (EGFIRFO) algorithm for enhancing the task scheduling in cloud had been proposed by Zade and Mansouri [28]. The authors had generated the initial population for their work based on the Quasi-Opposition Based Learning method. The Leavy flight method is applied in generating new foxes. Balance between exploration and exploitation phases had been achieved using two fuzzy control systems. The cornu-spiral movement had been considered for improvising the local search activity. The performance of the proposed EGFIRFO algorithm had been evaluated by comparing it with other existing algorithms with respect to makespan, execution time, load balancing and resource utilization.

3 Solution framework

Users submit their tasks to the cloud that are placed in the task repository. Queue of tasks is built, with the tasks that are lined up to be scheduled in an optimized way. A fitness function is formalized such that the QoS demands placed by the user should be met with precise data values. The Task Queue (TQ) stores the incoming task requests. These requests are sorted in ascending order inside the queue based on their completion time criteria and the same has been shown in the diagram given below. The Task Scheduler component has the responsibility of retrieving the tasks from the task queue and assigning it to a precise infrastructure resource. Scheduling activity is carried out with the help of information provided by the Cloud Resource Information DB (CRIDB) component. Details regarding the resource availability in the cloud infrastructure is gathered by the CRIDB component and the same is provided by it to the Task Scheduler component. The available resources are adjusted and assigned to the received workload by the CRIDB component. When there is any imbalance detected between the received tasks and available resources, the CRDIB takes a note of it and selects an appropriate cloud provider that has the necessary resources required for any specific application in an economical manner. The tasks are fetched from the Task Queue by the Scheduler which submits the same to any available or identified resource. The tasks that assigned to any specific resource are tracked until their completion by the task manager. The proposed scheduling architecture is shown in Fig. 1.

3.1 Problem description

Information on currently presented resources in the Data Center (DC) is aggregated by the Cloud Resource Information DB (CRIDB) for mapping them on to complete the received task requests. User submitted tasks are aggregated at the task repository. Let us assume, for example there to be a set of tasks like $Task_1$, $Task_2$, $Task_3$,……,$Task_n$ that make up the task repository at any given interval of time, and each processing element (Virtual Machine) have their own processing rates (possibly varying) and memory size. Incoming task requests are queued up in the Task Queue (TQ). Tasks inside task queue are sorted based on their stated completion time criteria. A point to ponder at this stage is that these tasks consume different time and cost while being executed at different virtual machines. Let VM_1, VM_2, ….VM_m be the virtual machine set available, to which the scheduler assigns the task from the task queue appropriately. The VMs that exhibit minimal makespan and less energy consumption criteria are chosen to stay in line with our objective. The Expected Time to Complete (ETC) parameter of each VM helps in identifying the ideal VM to whom the task could be scheduled. Assessment of ETC could be made by computing the ratio between Million Instructions per Second (MIPS) of the VM to the length of the task to be executed. The assessed ETC values are decked up in matrix form where each row depicts the tasks that are to be scheduled and each column indicates the number of available VMs. To be more precise, the row and column of the matrix indicate the execution time of any given task on each VM and execution time of each task on any given VM, respectively.

4 GWO algorithm overview

The GWO algorithm [29] replicates the grey wolves' lifestyle of hunting and adhering to a social hierarchy. In addition to this, the characteristic of hunting in pack that is exhibited by grey wolves has been inherited in the GWO. Encircling, hunting and attacking the target are the main sections of GWO. The steps of GWO algorithm are presented below:

The algorithm's parameters namely maximum number of iterations ($iter_{max}$), Search Agents (G_{search}), vectors a, A, C and Design Variable Size (G_{dim}) are initialized.

$$\vec{A} = 2 \cdot \vec{a} \cdot random_1 - \vec{a} \tag{1}$$

$$\vec{C} = 2 \cdot random_2 \tag{2}$$

In each iteration performed, the value of $\vec{a}$ is linearly reduced starting from 2 to 0. The random vectors are denoted as $random_1$ and $random_2$ and are in the range [0,1].

Wolves are generated in a random manner according the size of pack chosen. This sequence is given in Eq. (3):

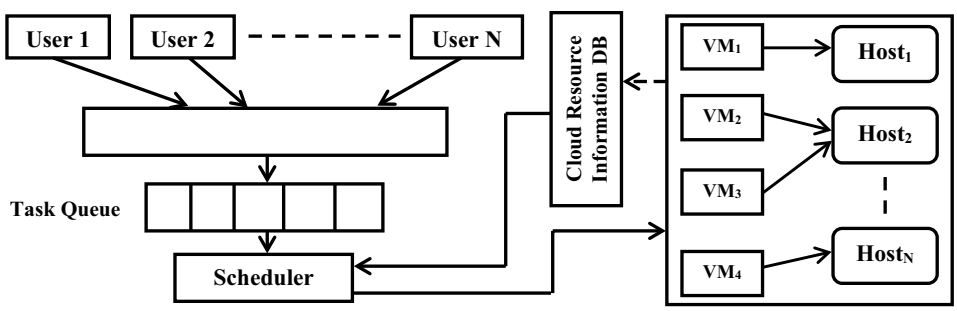

Fig. 1 Proposed cloud scheduling architecture

$$Wolves = \begin{bmatrix} GW_1^1 GW_2^1 GW_3^1 \cdots \cdots GW_{Gd-1}^1 GW_{Gd}^1 \\ GW_1^2 GW_2^2 GW_3^2 \cdots \cdots GW_{Gd-1}^2 GW_{Gd}^2 \\ \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \\ \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \\ GW_1^{Gs} GW_2^{Gs} GW_3^{Gs} \cdots \cdots GW_{Gd-1}^{Gs} GW_{Gd}^{Gs} \end{bmatrix} \tag{3}$$

where, GW_{ij} denotes the value of the j[th] pack of the i[th] wolves.

The coefficient vectors are $\vec{C}$ and $\vec{A}$. $\overrightarrow{G_{Prey}}$ denotes the prey position and the wolf position is denoted by $\overrightarrow{G_W}$. The distance vector is denoted as $\overrightarrow{D_{Dis}}$ and 't' is the iteration number. The Eqs. (4)-(5) represents the each hunt agent fitness value is deduced from:

$$\overrightarrow{D_{Dis}} = \left| \vec{C} \cdot \overrightarrow{G_{Prey}}(t) - \overrightarrow{G_W}(t) \right| \tag{4}$$

$$\overrightarrow{G_W}(t+1) = \left| \overrightarrow{G_{Prey}}(t) - \vec{A} \cdot \overrightarrow{D_{Dis}} \right| \tag{5}$$

The first best $\overrightarrow{G_\alpha}$, the second best $\overrightarrow{G_\beta}$ and the third best $\overrightarrow{G_\delta}$ hunt agents are extracted at a given iteration 't'. The values for $\overrightarrow{A_1}$, $\overrightarrow{A_2}$ and $\overrightarrow{A_3}$ and $\overrightarrow{C_1}$, $\overrightarrow{C_2}$ and $\overrightarrow{C_3}$ are defined in Eqs. (1) and (2) respectively. $\overrightarrow{D_{Dis}^\alpha}$, $\overrightarrow{D_{Dis}^\beta}$ and $\overrightarrow{D_{Dis}^\delta}$ represent the distance vectors between α, β and δ respectively.

$$\overrightarrow{D_{Dis}^\alpha} = \left| \overrightarrow{C_1} \cdot \overrightarrow{G_\alpha} - \overrightarrow{G} \right| \tag{6}$$

$$\overrightarrow{D_{Dis}^\beta} = \left| \overrightarrow{C_2} \cdot \overrightarrow{G_\beta} - \overrightarrow{G} \right| \tag{7}$$

$$\overrightarrow{D_{Dis}^\delta} = \left| \overrightarrow{C_3} \cdot \overrightarrow{G_\delta} - \overrightarrow{G} \right| \tag{8}$$

$$\overrightarrow{G_1} = \overrightarrow{G_\alpha} - \overrightarrow{A_1} \cdot \left(\overrightarrow{D_{Dis}^\alpha} \right) \tag{9}$$

$$\overrightarrow{G_2} = \overrightarrow{G_\beta} - \overrightarrow{A_2} \cdot \left(\overrightarrow{D_{Dis}^\beta} \right) \tag{10}$$

$$\overrightarrow{G_3} = \overrightarrow{G_\delta} - \overrightarrow{A_3} \cdot \left(\overrightarrow{D_{Dis}^\delta} \right) \tag{11}$$

In Eq. (12) the current hunt agent position is updated:

$$\overrightarrow{G_W}(t+1) = \frac{\overrightarrow{G_1} + \overrightarrow{G_2} + \overrightarrow{G_3}}{3} \tag{12}$$

The tradeoff between exploration and exploitation is controlled by updating the variable 'a'. In each iteration the variable 'a' is updated linearly and the variable range is from 2 to 0 based on the Eq. (13).

$$a = 2 - \frac{(2 \cdot t)}{iter_{max}} \tag{13}$$

where, the maximum number of iteration is denoted as $iter_{max}$ and iteration number is represented as 't' for the optimization.

GWO algorithm:

Initialize the associated parameter of GWO
The wolf's positions are generated randomly
Fitness of an each wolf is calculated by Eq. (28)
Find $\overrightarrow{G_\alpha}$, $\overrightarrow{G_\beta}$ and $\overrightarrow{G_\delta}$
While $\left(t < iter_{max}\right)$

Update $a, A and C$ using Eqs. (1), (2) and (13)
Each wolf position is computed using Eqs. (6)-(12)
Fitness of each wolf is calculated using Eq. (28)
Update $\overrightarrow{G_\alpha}$, $\overrightarrow{G_\beta}$ and $\overrightarrow{G_\delta}$

end while
Output $\overrightarrow{G_\alpha}$

In GWO, the position updating equations had been implemented. In the Binary GWO [30], the wolves' positions values are fixed at 0 or 1, with the assumption that the wolves search area is hypercube. The location update in binary GWO is done by tracking the wolves which change their locations to be near to the hypercube or somewhere by altering the variable values. Therefore in binary GWO, the update is not performed by using the old update equations.

BGWO acquires the values of α, β and δ by using the same old scheme, and values for $\overrightarrow{D_{Dis}^\alpha}$, $\overrightarrow{D_{Dis}^\beta}$ and $\overrightarrow{D_{Dis}^\delta}$ are calculated using the Eqs. (6)-(8). Then using sigmoid equation (called S1), s1, s2 and s3 are deduced as shown below.

$$Sig1_\alpha^d = \frac{1}{\left(1 + e^{-10\left(A^d \cdot D_\alpha^d - 0.5\right)}\right)} \tag{14}$$

$$Sig2_\beta^d = \frac{1}{\left(1 + e^{-10\left(A^d \cdot D_\beta^d - 0.5\right)}\right)} \tag{15}$$

$$Sig3_{\delta}^{d} = \frac{1}{\left(1 + e^{-10\left(A^{d} \cdot D_{\delta}^{d} - 0.5\right)}\right)} \tag{16}$$

where, the d[th] dimension of a wolf (agent) is represented as 'd'. Equations (17)-(19) had been applied for evaluating bstep1, bstep2 and bstep3. This results in the generation of a binary value. Then the transfer function had been applied for switching purposes as shown in Eqs. (14)-(16). The binary values of 0 and 1 are needed for comparing with random numbers.

$$bstep1_{\alpha}^{d} = \begin{cases} 1 & if\left(Sig1_{\alpha}^{d} \geq rand\right) \\ 0 & otherwise \end{cases} \tag{17}$$

$$bstep2_{\beta}^{d} = \begin{cases} 1 & if\left(Sig2_{\beta}^{d} \geq rand\right) \\ 0 & otherwise \end{cases} \tag{18}$$

$$bstep3_{\delta}^{d} = \begin{cases} 1 & if\left(Sig2_{\delta}^{d} \geq rand\right) \\ 0 & otherwise \end{cases} \tag{19}$$

where rand is a random number between [0, 1]. bstep1, bstep2 and bstep3 are the distances that are moved relative to α, β and δ respectively. The values of X1, X2 and X3 are then deduced from the following equations.

$$X1_{\alpha}^{d} = \begin{cases} 1 & if\left(X_{\alpha}^{d} + bstep1_{\alpha}^{d} \geq 1\right) \\ 0 & otherwise \end{cases} \tag{20}$$

$$X2_{\beta}^{d} = \begin{cases} 1 & if\left(X_{\beta}^{d} + bstep2_{\beta}^{d} \geq 1\right) \\ 0 & otherwise \end{cases} \tag{21}$$

$$X3_{\delta}^{d} = \begin{cases} 1 & if\left(X_{\delta}^{d} + bstep3_{\delta}^{d} \geq 1\right) \\ 0 & otherwise \end{cases} \tag{22}$$

Finally, a simple stochastic crossover had been adopted for updating the position of Xi in the subsequent iteration, as shown in Eq. (23).

$$X(i)^{d}(t+1) = \begin{cases} X1_{\alpha}^{d} & if\left(rand \leq 1/3\right) \\ X2_{\beta}^{d} & elseif(1/3 \leq rand \leq 2/3) \\ X3_{\delta}^{d} & otherwise \end{cases} \tag{23}$$

The BGWO algorithm in the form of pseudocode is presented below.

BGWO algorithm:

Initialize the associated parameter of GWO
The wolf's positions are generated randomly
Fitness of each wolf is calculated using Eq. (28)
Find $\overrightarrow{G_{\alpha}}$, $\overrightarrow{G_{\beta}}$ and $\overrightarrow{G_{\delta}}$
While $\left(t < iter_{max}\right)$

Update $a, AandC$ using Eqs. (1), (2) and (13)
Each wolf position is computed using Eqs. (16)-(23)
Fitness of each wolf is calculated using Eq. (28)
Update $\overrightarrow{G_{\alpha}}$, $\overrightarrow{G_{\beta}}$ and $\overrightarrow{G_{\delta}}$

end while
Output $\overrightarrow{G_{\alpha}}$

4.1 Performance metrics

4.1.1 Execution time

The Execution Time (ET) for i^{th} node can be deduced by accounting the time taken to execute all the assigned tasks on it. This can be given as:

$$ET_{i} = \sum_{k=1}^{M} \sum_{j=1}^{N} ET_{ijk} \tag{24}$$

where, ET_i is the measure of time taken by 'j' VMs for executing the 'k' tasks in the 'i' node. Then the total Execution Time (ET) for the system could be deduced as

$$ET = \sum_{i=1}^{L} ET_{i} \tag{25}$$

where, ET_i is the time taken by each node 'i' present in the system to complete their assigned tasks.

4.1.2 Energy consumption

The second objective, energy consumptionisdirectly proportional to the number of DC present in the cloud environment. Enlarge in DC results in the increase of the energy factor. The energy consumed by each virtual machine could be multiplied with the distribution matrix element for deducing the energy consumption parameter as shown:

$$Energy_{Consum} = 1/(NT \times VM)\left(\sum_{a=1}^{X} \sum_{b=1}^{Y} DM_{xy} \times EU_{xy}\right) \tag{26}$$

where, 'NT' denotes the number of tasks and 'VM' indicates the virtual machine. Item in the distribution matrix is denoted as 'DM_{xy}' and 'EU_{xy}' indicates the energy utilized by each resource while executing the tasks.

4.1.3 Degree of imbalance

Degree of Imbalance (DoI) defines the amount of load sharing among VMs according to their computing capacities. DoI measures the imbalance among VMs. By taking into

consideration of the DoI measure during load distribution, results in the avoiding of unbalanced load being allocated among virtual machines. DoI can be deuced as:

$$DoI = \frac{\left[\left(ET^{Max}\right) - \left(ET^{Min}\right)\right]}{\left(ET^{Avg}\right)} \tag{27}$$

Let, ET^{Max} is maximum task execution time, ET^{Min} is minimum task execution time and ET^{Avg} is average task execution time among all VMs.

4.1.4 Fitness function

The near optimal solution is deduced by validating each of the solution obtained against the fitness function. Hence quality of an obtained solution is ascertained by the fitness function. An equation describing the fitness is required for all optimization problems to deduce an optimal solution. The main aim of this research work is centered onminimizing the execution time, energy consumption and degree of imbalance parameters. For the given search agent, the objective values of time, energy and degree of imbalance are deduced in the proposed technique using Eqs. (25), (26) and (27) respectively. The fitness can be computed using Eq. (28) as shown below:

$$Fitness\ Function_i^t\left(FF_i^t\right) = \alpha \cdot ET + \beta \cdot Energy_{Consum} + \delta \cdot DoI \tag{28}$$

where, the range values of $\alpha, \beta\ and\ \delta$ between $1 > \alpha, \beta, \delta \geq 0$ and FF_i^t is the formulated fitness function.

4.1.5 Performance improvement rate (PIR in %)

The percentage change in improvement of the proposed BGWO algorithm can be deduced by compare it with other algorithms from the related work in accordance to the Eq. (29) depicted below:

$$Performance_{Prop}(\%) = \left(X_{Exist} - X_{Prop}\right) \times \left(\frac{100}{X_{Prop}}\right) \tag{29}$$

where X_{Prop} indicates the performance measure produced by the proposed BGWO and X_{Exist} is the measure obtained from other algorithms considered for comparative purpose.

4.2 Proposed binary grey wolf optimization algorithm (BGWO)

The proposed Binary-GWO algorithm flowchart is shown in Fig. 2. The Binary-GWO algorithm has been presented here.

Input:

Binary-GWO Algorithm parameters.
Task and Resource Tasks $\in \{1,2,\ldots m\}$ *and* *Resources* $\in \{1, 2,\ldots n\}$.

Output:

The near optimal solution for the tasks scheduling.

Parameters initialization:

1. Let $\overrightarrow{X_i} = (i = 1, 2, \ldots \ldots .N)$ be the population, with 'N' denoting the actual size of such population, factor 'a', C & A are the coefficient vectors and *ITR_MAX* denotes the variable that holds the maximum iteration value.
2. Let the preliminary counter value *t,* be initialized to 0.

Initialization of Population:

3. Define $i = 1$
4. While $(i \leq n)$ do
5. Initial population is generated randomly $\overrightarrow{X_i(t)}$
6. Using Eq. (28), compute the fitness value of individual wolf present in the given search space.
7. End While
8. Let Alpha $\overrightarrow{G_\alpha}$ indicate the wolf with high fitness value.
9. Let Beta $\overrightarrow{G_\beta}$ indicate the wolf with second highest fitness value
10. Let Delta $\overrightarrow{G_\delta}$ indicate the grey wolf with third highest fitness value

Updating (Solution)

11. While(t < ITR_MAX)
12. For individual search wolf
13. Individual wolf position is updated using Eqs. (16)-(23).
14. End for
15. Start decrementing the value of 'a' from 2 to 0.
16. The coefficient values of "A" and "C" are updated using Eqs. (1) and (2) respectively.
17. Using Eq. (28), calculate the fitness value of all wolves.
18. The Location of First (Alpha) ($\overrightarrow{G_\alpha}$), Second (Beta) ($\overrightarrow{G_\beta}$) and Third (Delta) ($\overrightarrow{G_\delta}$) are updated using Eqs. (20), (21) and (22) respectively.
19. Increment the iteration counter t = t + 1
20. Stop the While Loop

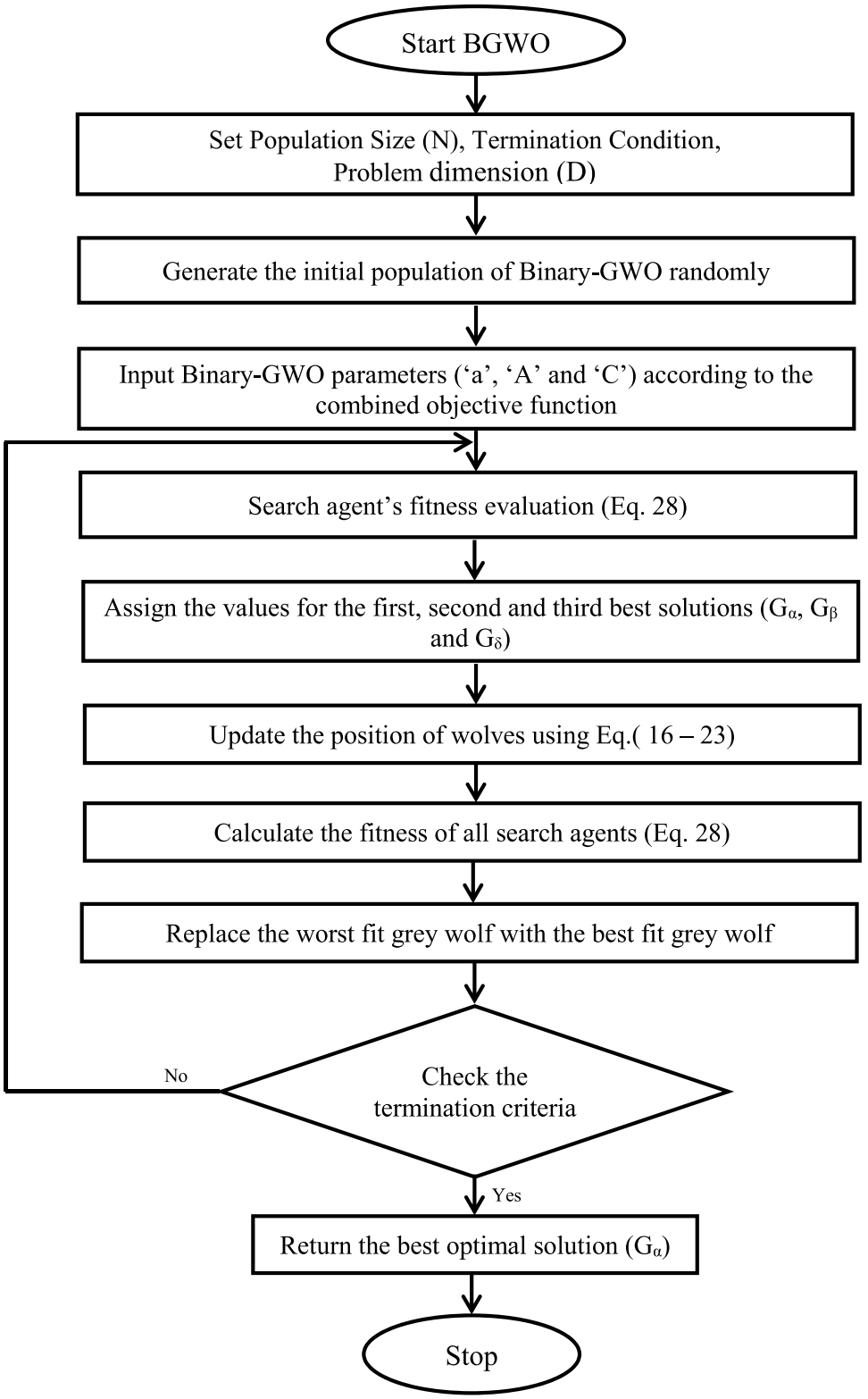

Fig. 2 Flowchart of proposed Binary-GWO Algorithm

Best Optimal Solution

21. The value of First (Alpha) ($\overrightarrow{G_\alpha}$) is returned as the near optimal solution from the given search area.

5 Result and discussion

In this section, the experimental simulation results of the proposed algorithms' performance are compared. The cloud environment requirements and the parameters representation for evaluation are given in Table 1. The cloudsim 3.0 toolkit has been used to simulate the proposed BGWO algorithm. The experiment analysis had been carried out using desktopwith a desired configuration comprising of IntelCore i7 processor with quad core at 2.0 GHz, 8GB of main memory and OS of Windows 64-bit configuration. The experimental results were achieved for the proposed BGWO approachwith respect to degree of imbalance and makespan parameters and had been compared with results obtained using OGWO and Mean GWO algorithms. The performance of the proposed methodology is evaluated using normal and HPC2N datasets [31] comprising of 100to 1000 tasks. A normal dataset includes medium sized tasks as well as a smaller quantity of large and small sized tasks. The HPC2N set log work are independent and it's preemptive. Researchers had generally used the HPC2N dataset as a benchmark workload while evaluating the performance of distributed systems.

5.1 Makespan

The performance efficiency of the proposed BGWO in task scheduling had been analyzed with respect to the makespan parameter. The obtained results of the proposed BGWO are compared with the results that were obtained using the OGWO and Mean GWO algorithms. The number of tasks were varied from 200 to 1000. While performing simulation, tasks were run 100 times and an average had been deduced. For both normal as well as HPC2N datasets, 100 VMs had been allotted for executing the tasks. The plots in Figs. 3 and 4 shows the performance results obtained using the proposed BGWO, OGWO and Mean GWO with respect to the makespan parameter for normal and HPC2N datasets respectively.

In Fig. 3, the normal dataset had been applied. For 200 number of cloud tasks the obtained makespan values are 42.86, 38.96 and 48.70 for OGWO, BGWO and Mean GWO respectively. Similarly, when the task numbers are increased to 600, the makespan values obtained are 123.06, 110.20 and 132.47 for OGWO, BGWO and Mean GWO respectively. It could be inferred from the below plot that the PIR of the proposed BGWO for normal dataset with respect to makespan parameter is 16.76% and 9.19% when compared to mean-GWO and OGWO respectively.

In Fig. 4, the HPC2N dataset had been applied. For 400 number of cloud tasks the obtained makespan values are 129.87, 119.39 and 140.28 for OGWO, BGWO and Mean GWO respectively. Similarly, when the task numbers are increased to 800, the makespan values obtained are 231.04, 206.32 and 243.04 for OGWO, BGWO and Mean GWO respectively. It could be inferred from the above plot that the PIR of the proposed BGWO for HPC2N dataset with respect to makespan parameter is 18.04% and 11.07% when compared to Mean-GWO and OGWO respectively.

Table 1 Simulation Settings

Entity Type	Parameter	Value
Datacenter	Number of Data Center	1
	Number of Host	5
User	Number of User	10
Task	Number of Tasks	200
	Task Length	100–1500 MIPS
	Host Memory	70GB
Host	Host Bandwidth	2000
	Host Storage	1000 GB
	Number of VMs	100
	vRAM	256 – 1024 MB
Virtual Machine	Bandwidth	250 – 1000
	Virtual Machine Monitor	XEN
	OS	Linux

5.2 Degree of imbalance

The performance efficiency of the proposed BGWO in task scheduling had been analyzed with respect to the degree of imbalance parameter. The obtained results of the proposed BGWO are compared with the results that were obtained using the OGWO and Mean GWO algorithms. The number of tasks were varied from 200 to 1000. While performing simulation, tasks were run 100 times and an average had

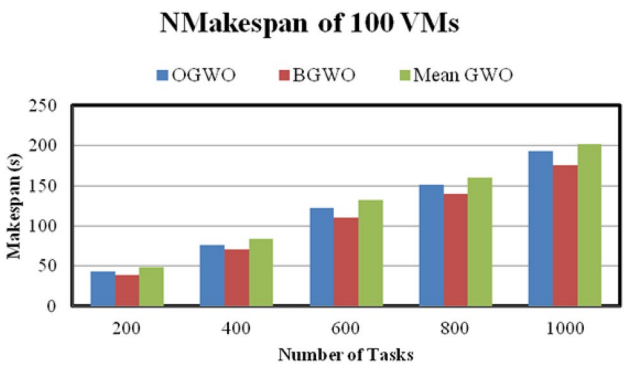

Fig. 3 Performance analysis of Makespan for Normal dataset

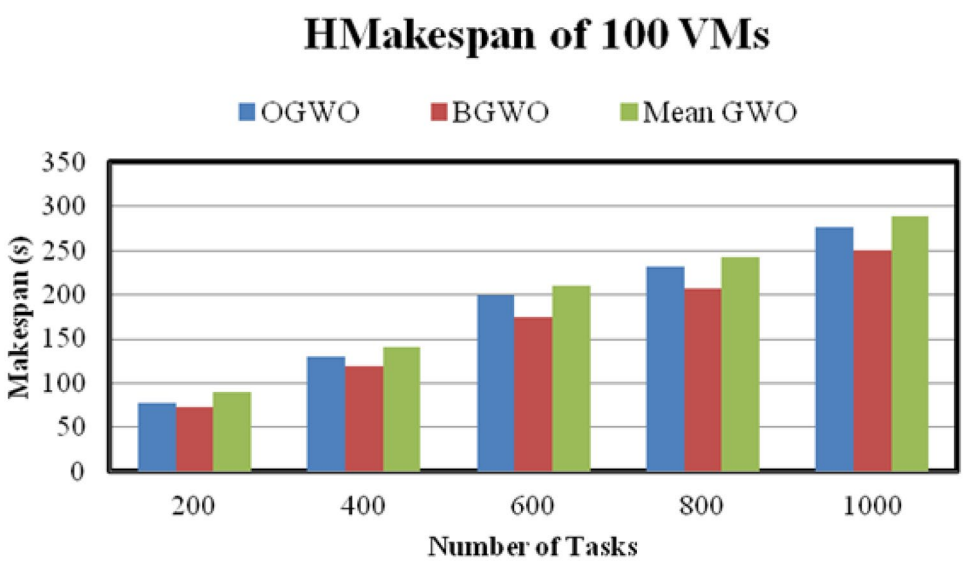

Fig. 4 Performance analysis of Makespan for HPC2N dataset

been deduced. For both normal as well as HPC2N datasets, 100 VMs had been allotted for executing the tasks. The plots in Figs. 5 and 6 shows the performance results obtained using the proposed BGWO, OGWO and Mean GWO with respect to the degree of imbalance parameter for normal and HPC2N datasets respectively.

In Fig. 5, the performance of the proposed BGWO using normal dataset has been assessed by varying the number of tasks. It could be seen that for 200, 400, 600, 800 and 1000 tasks, the DoI of the proposed BGWO technique comes out to be 50.98, 80.39100.06, 120.42 and 140.97, for OGWO the DoI is 80.54, 140.52, 170.07, 190.23 and 230.56, for mean GWO the DoI is 120.19, 160.59, 190.20, 210.25 and 260.18 respectively. The proposed BGWO technique gives an improvement of 37.5% and 45.69% than OGWO and mean GWO techniques respectively.

Similarly, in Fig. 6 the performance of the proposed BGWO using HPC2N dataset has been assessed by varying the number of tasks. It could be seen that for 200, 400, 600, 800 and 1000 tasks, the DoI of the proposed BGWO technique comes out to be 140.95, 200.14, 240.14, 290.06 and 350.93, for OGWO the DoI is 190.63, 300.49, 320.44, 360.34 and 440.77, for Mean- GWO the DoI is 260.83, 340.35, 400.32, 440.63 and 540.98 respectively. The proposed technique gives an improvement of 24.1% and 38.23% than OGWO and means GWO respectively.

5.3 Energy consumption

The Figs. 7 and 8 given below depicts the value for energy consumption deduced by applying the proposed BGWO, OGWO and mean GWO using normal and HPC2N datasets. Upon observing, it could be seen that the proposed

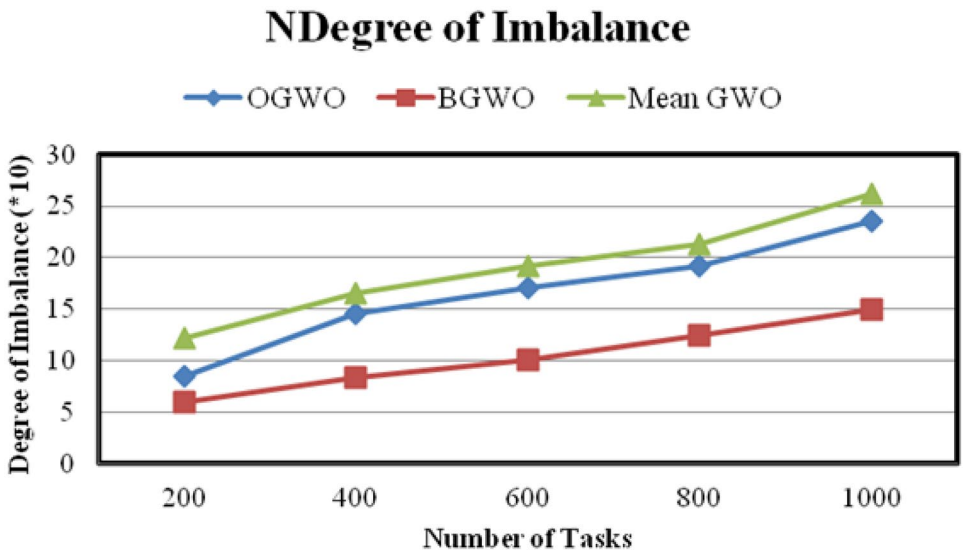

Fig. 5 Performance analysis of DoI for Normal dataset

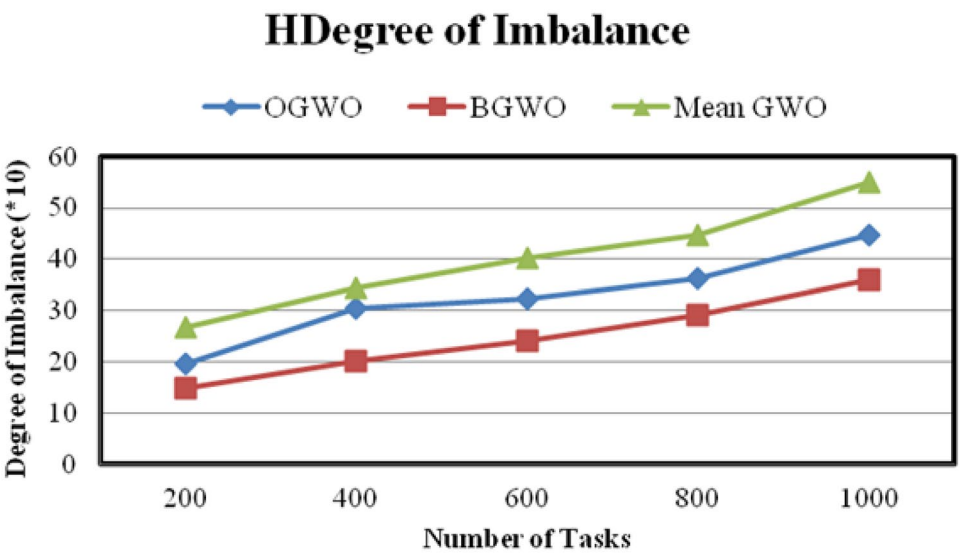

Fig. 6 Performance analysis of DoI for HPC2N dataset

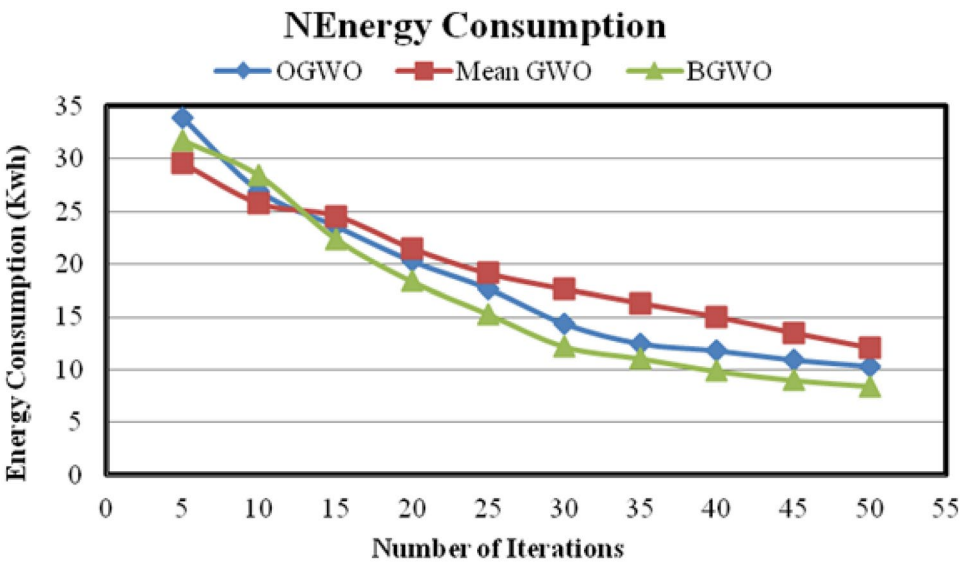

Fig. 7 Energy Consumption of Normal dataset for 500 Tasks

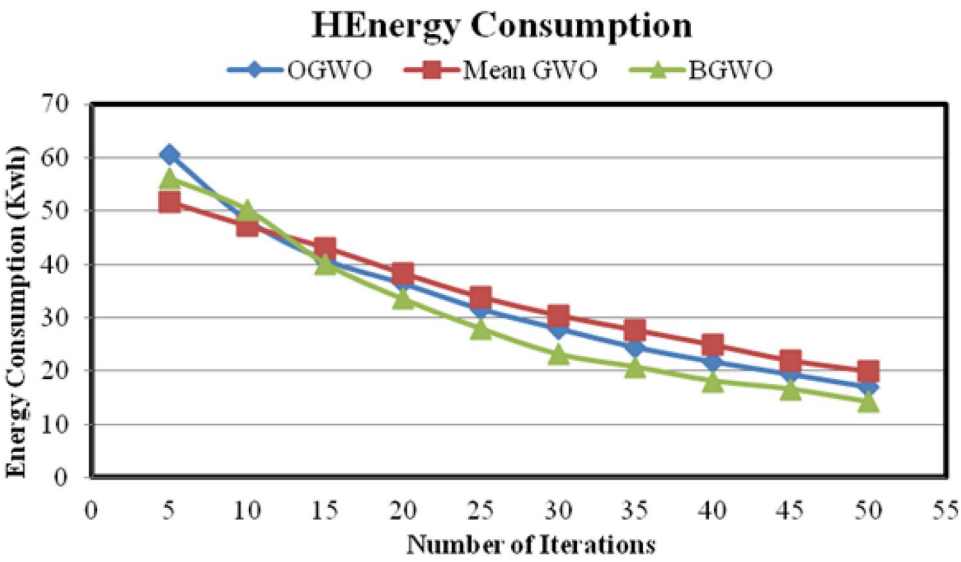

Fig. 8 Energy Consumption of HPC2N dataset for 800 Tasks

BGWO reduces the energy consumption value, resulting in improving the performance of the task scheduling process. In the simulation conducted, 50 iterations were performed for evaluating the energy consumption value. When the iterations were set to a minimum, like for instance (5, 10), the Mean GWO had produced a better energy consumption value compared to the other two techniques. As the iterations are increased, it could be observed that the proposed BGWO approach produces better results when compared to the OGWO and mean GWO approaches.

In Fig. 7, the normal dataset had been applied for evaluation and comparison purposes. It could be seen that the proposed BGWO technique accounts for minimum energy consumption as the iterations are increased. In the iteration 10, the energy consumption valued for the proposed BGWO, OGWO and MGWO are 28.38, 26.94 and 25.74. Similarly for 20, 30, 40 and 50 iterations, the proposed BGWO produces energy consumption values of 18.37, 12.204, 9.87 and 8.46, the OGWO produces energy consumption values of 20.29, 14.36, 11.82 and 10.34 and the energy consumption values for Mean GWO are 21.49, 17.65, 15.04 and 12.12 respectively. It could be inferred that the proposed BGWO technique produces an improvement of 8.51% and 16.97%, compared to OGWO and Mean GWO techniques respectively.

In Fig. 8, the energy consumption values obtained using the proposed BGWO, OGWO and Mean GWO techniques by applying the HPC2N dataset for 800 tasks had been shown. For 5, 15, 25, 35 and 45 iterations, the proposed BGWO produces energy consumption values of 56.08, 40.12, 27.9, 20.69, the OGWO produces energy consumption values of 60.57, 40.87, 31.65, 24.45 and 19.39 and the energy consumption values for Mean GWO are 51.59, 43.12, 33.89, 27.68 and 21.98 respectively. It could be inferred that the proposed BGWO technique produces an improvement of 9.37% and 12.92%, compared to OGWO and Mean GWO techniques respectively.

6 Conclusion

A multi-objective Binary Grey Wolf Optimization (BGWO) algorithm had been presented in this paper for optimizing the scheduling action in the cloud computing environment based on QoS metrics such as, degree of imbalance, energy consumption and makespan. It is a known fact the cloud performance through scheduling can increase manifold when a multi-objective approach is followed. The BGWO algorithm improvises the scheduling activity where each task gets updated by adaptively selecting the binary updating strategies. Experiments were carried using Cloudsim toolkit where normal as well as HPC2CN datasets were utilized. The improvement of the BGWO algorithm had been measured by comparing it with OGWO and Mean GWO algorithms with respect to Degree of Imbalance, energy consumption and makespan QoS parameters. The results obtained justify the superior performance of the proposed BGWO technique compared to the other techniques. The effectiveness of the proposed algorithm could be further substantiated by comparing it with other available meta-heuristic algorithms by considering other sets of QoS parameters. The proposed algorithm had produced a cumulative improvement of 10.13% and 17.4% for makespan, 30.18% and 41.96% for DoI, 8.94% and 14.95% for energy consumption parameters. This could highly enhance and further improvise the solutions to task scheduling and other impending research issues in the cloud computing environment. In future, QOS parameters like fault tolerance and reliability could be incorporated along with the parameters considered in this work to further expand the scope of this research.

Data availability The data can be shared on valid request made to corresponding author.

Declarations

Conflict of interest All authors declare that they have no conflicts of interest.

References

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616. https://doi.org/10.1016/j.future.2008.12.001
2. Zhang Q, Cheng L, Boutaba R (2010) Cloud computing: State-of-the-art and research challenges. J Internet Serv Appl 1(1):7–18. https://doi.org/10.1007/s13174-010-0007-6
3. Houssein EH, Gad AG, Wazery YM, Suganthan PN (2021) Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm Evol Comput 62:100841. https://doi.org/10.1016/j.swevo.2021.100841
4. Ibrahim IM (2021) Task scheduling algorithms in cloud computing: a review. Turk J Comput Math Educ 12(4):1041–1053. https://doi.org/10.17762/turcomat.v12i4.612
5. Pradhan A, Bisoy SK, ADas, (2021) A Survey on PSO Based Meta-Heuristic Scheduling Mechanism in Cloud Computing Environment, J. King Saud Univ, Comput. Info. Scie. https://doi.org/10.1016/j.jksuci.2021.01.003
6. Jacob TP, Pradeep K (2019) A Multi-objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle Swarm Optimization. Wirel Pers Commun 109(1):315–331. https://doi.org/10.1007/s11277-019-06566-w
7. Gobalakrishnan N, Arun C (2018) A new multi-objective optimal programming model for task scheduling using genetic gray wolf optimization in cloud computing. Comput J 61(10):1523–1536. https://doi.org/10.1093/comjnl/bxy009
8. Prasanna Kumar KR, Kousalya K (2020) Amelioration of task scheduling in cloud computing using crow search algorithm. Neural Comput Appl 32(10):5901–5907. https://doi.org/10.1007/s00521-019-04067-2

9. Natesan G, Chokkalingam A (2020) Multi-objective task scheduling using hybrid whale genetic optimization algorithm in heterogeneous computing environment. Wirel Pers Commun 110(4):1887–1913. https://doi.org/10.1007/s11277-019-06817-w
10. Pradeep K, Javid Ali L, Gobalakrishnan N, Raman CJ, Manikandan N (2021) CWOA: hybrid approach for task scheduling in cloud environment. Comput J. https://doi.org/10.1093/comjnl/bxab028
11. Golchi MM, Saraeian S, Heydari M (2019) A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: performance evaluation. Comput Netw 162:106860. https://doi.org/10.1016/j.comnet.2019.106860
12. Li K (2019) Energy and time constrained scheduling for optimized quality of service. Sustain Comput Informatics Syst 22:134–138. https://doi.org/10.1016/j.suscom.2019.04.001
13. Ismayilov G, Topcuoglu HR (2020) Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. Future Gener Comput Syst 102:307–322. https://doi.org/10.1016/j.future.2019.08.012
14. Mansouri N, Zade BMH, Javidi MM (2019) Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory. Comput Ind Eng 130:597–633. https://doi.org/10.1016/j.cie.2019.03.006
15. Pietri I, Sakellariou R (2019) A Pareto-based approach for CPU provisioning of scientific workflows on clouds. Future Gener Comput Syst 94:479–487. https://doi.org/10.1016/j.future.2018.12.004
16. Priya V, Kumar CS, Kannan R (2019) Resource scheduling algorithm with load balancing for cloud service provisioning. Appl Soft Comput 76:416–424. https://doi.org/10.1016/j.asoc.2018.12.021
17. Stavrinides GL, Karatza HD (2019) An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. Future Gener Comput Syst 96:216–226. https://doi.org/10.1016/j.future.2019.02.019
18. Zhang Y, Zhou J, Sun J (2019) Scheduling bag-of-tasks applications on hybrid clouds under due date constraints. J Syst Archit 101:101654. https://doi.org/10.1016/j.sysarc.2019.101654
19. Zhou X, Zhang G, Sun J, Zhou J, Wei T, Hu S (2019) Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT, Future Gener. Comput Syst 93:278–289. https://doi.org/10.1016/j.future.2018.10.046
20. Kashikolaei SMG, Hosseinabadi AAR, Saemi B, Shareh MB, Sangaiah AK, Bian GB (2020) An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm. J Supercomput 76(8):6302–6329. https://doi.org/10.1007/s11227-019-02816-7
21. Zhou Z, Li F, Zhu H, Xie H, Abawajy JH, Chowdhury MU (2020) An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Comput Appl 32(6):1531–1541. https://doi.org/10.1007/s00521-019-04119-7
22. Ababneh J (2021) A hybrid approach based on grey wolf and whale optimization algorithms for solving cloud task scheduling problem. Math Prob Eng 2021:3517145. https://doi.org/10.1155/2021/3517145
23. Medara R, Singh RS (2021) Energy efficient and reliability aware workflow task scheduling in cloud environment. Wirel Pers Commun 119(2):1301–1320. https://doi.org/10.1007/s11277-021-08263-z
24. Abualigah L, Alkhrabsheh M (2022) Amended Hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. J Super Comput 78(1):740–765. https://doi.org/10.1007/s11227-021-03915-0
25. Amer DA, Attiya G, Zeidan I, Nasr AA (2022) Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. J Super Comput 78(2):2793–2818. https://doi.org/10.1007/s11227-021-03977-0
26. Jain R, Sharma N (2022) A quantum inspired hybrid SSA–GWO algorithm for SLA based task scheduling to improve QoS parameter in cloud computing. Cluster Comput. https://doi.org/10.1007/s10586-022-03740-x
27. Li J, Zhang X, Wei J, Ji Z, Wei Z (2022) GARLSched: generative adversarial deep reinforcement learning task scheduling optimization for large-scale high performance computing systems. Future Gener Comput Syst 135:259–269. https://doi.org/10.1016/j.future.2022.04.032
28. Zade BMH, Mansouri N (2022) Improved red fox optimizer with fuzzy theory and game theory for task scheduling in cloud environment. J Comput Sci 63:101805. https://doi.org/10.1016/j.jocs.2022.101805
29. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
30. Hu P, Pan JS, Chu SC (2020) Improved binary grey wolf optimizer and its application for feature selection. Knowl Based Syst 195:105746. https://doi.org/10.1016/j.knosys.2020.105746
31. Abdullahi M, Ngadi MA (2016) Symbiotic Organism Search optimization based task scheduling in cloud computing environment. Future Gener Comput Syst 56:640–650. https://doi.org/10.1016/j.future.2015.08.006

Dr. Gobalakrishnan Natesan pursued his Bachelor's degree in Information Technology at Anna University, Tamilnadu, India in 2005. He then obtained his Master's degree in Software Engineering from Bharathidhasan University, Tamilnadu, India in 2008. He completed his Ph.D in Sathyabama University, Chennai, India and currently working as Associate Professor in the Department of Information Technology, Sri Venkateswara college of Engineering, Tamilnadu, India. His current research interests are Cloud computing, image processing and Big Data.

Dr. N. Manikandan obtained his Bachelors of Engineering degree in Computer Science Engineering from Anna University 2005. Then he obtained his Master's of Engineering degree in in Computer Science Engineering from Anna University 2011. He completed his PhD in Sathyabama Institute of Science and Technology 2020, he is an Assistant Professor of SRM Institute of Science and Technology, Kattankulathur, Chennai. His specializations include cloud computing, Operating systems, Networking.

Dr. K. Pradeep pursued his Bachelor's degree in Computer Science and Engineering at Madras University, Tamilnadu, India in 2004. He then obtained his Master's degree in Computer Science and Engineering from Anna University, Tamilnadu, India in 2011. He completed his Ph.D in Sathyabama Institute of Science and Technology, Chennai, India and working an Assistant Professor in the School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, Tamilnadu, India. His current research interests are Cloud computing, Virtualization and Big Data.

Dr. L. Sherly Puspha Annabel received her PhD degree in information and communication engineering from Anna University, Chennai, India, in 2016. She received her BE degree in computer science and engineering from the Karunya Institute of Technology, Coimbatore, India, in 2000 and her ME degree in computer science and engineering from Jaya Engineering College, Chennai, India, in 2006. Currently she is working as a professor at St. Joseph' s College of Engineering, Chennai. Her areas of interest include wireless ad hoc networking, sensor networks, and energy-efficient MAC protocols.