energies

MDPI

Article

Energy-Aware Bag-of-Tasks Scheduling in the Cloud Computing System Using Hybrid Oppositional Differential Evolution-Enabled Whale Optimization Algorithm

Amit Chhabra [1,*], Sudip Kumar Sahana [2], Nor Samsiah Sani [3,*], Ali Mohammadzadeh [4] and Hasmila Amirah Omar [3]

1 Department of Computer Engineering & Technology, Guru Nanak Dev University, Amritsar 143005, India
2 Department of Computer Science & Engineering, Birla Institute of Technology, Mesra, Ranchi 835215, India; sudipsahana@bitmesra.ac.in
3 Center for Artificial Intelligence Technology, Faculty of Information Science & Technology, Universiti Kebangsaan Malaysia, Bangi 43600, Selangor, Malaysia; p91214@siswa.ukm.edu.my
4 Department of Computer Engineering, Shahindezh Branch, Islamic Azad University, Shahindezh 5981693695, Iran; ali.mohammadzadeh@iau.ac.ir
* Correspondence: amit.cse@gndu.ac.in (A.C.); norsamsiahsani@ukm.edu.my (N.S.S.)

Abstract: Bag-of-Tasks (BoT) scheduling over cloud computing resources called Cloud Bag-of-Tasks Scheduling (CBS) problem, which is a well-known NP-hard optimization problem. Whale Optimization Algorithm (WOA) is an effective method for CBS problems, which still requires further improvement in exploration ability, solution diversity, convergence speed, and ensuring adequate exploration–exploitation tradeoff to produce superior scheduling solutions. In order to remove WOA limitations, a hybrid oppositional differential evolution-enabled WOA (called h-DEWOA) approach is introduced to tackle CBS problems to minimize workload makespan and energy consumption. The proposed h-DEWOA incorporates chaotic maps, opposition-based learning (OBL), differential evolution (DE), and a fitness-based balancing mechanism into the standard WOA method, resulting in enhanced exploration, faster convergence, and adequate exploration–exploitation tradeoff throughout the algorithm execution. Besides this, an efficient allocation heuristic is added to the h-DEWOA method to improve resource assignment. CEA-Curie and HPC2N real cloud workloads are used for performance evaluation of scheduling algorithms using the CloudSim simulator. Two series of experiments have been conducted for performance comparison: one with WOA-based heuristics and another with non-WOA-based metaheuristics. Experimental results of the first series of experiments reveal that the h-DEWOA approach results in makespan improvement in the range of 5.79–13.38% (for CEA-Curie workloads), 5.03–13.80% (for HPC2N workloads), and energy consumption in the range of 3.21–14.70% (for CEA-Curie workloads) and 10.84–19.30% (for HPC2N workloads) over well-known WOA-based metaheuristics. Similarly, h-DEWOA also resulted in significant performance in comparison with recent state-of-the-art non-WOA-based metaheuristics in the second series of experiments. Statistical tests and box plots also revealed the robustness of the proposed h-DEWOA algorithm.

Keywords: cloud computing; Bag-of-Tasks scheduling; metaheuristics; energy efficiency; simulation; optimization

check for updates

Citation: Chhabra, A.; Sahana, S.K.; Sani, N.S.; Mohammadzadeh, A.; Omar, H.A. Energy-Aware Bag-of-Tasks Scheduling in the Cloud Computing System Using Hybrid Oppositional Differential Evolution-Enabled Whale Optimization Algorithm. *Energies* **2022**, *15*, 4571. https://doi.org/10.3390/en15134571

Academic Editor: Ravinesh Deo

Received: 28 May 2022
Accepted: 11 June 2022
Published: 22 June 2022

1. Introduction

Modern cloud computing setups offer incredible seamless, ubiquitous, reliable, cost-effective, and scalable resources, e.g., databases, platforms, software, computers, and servers as services to execute a variety of user-oriented applications [1,2]. Infrastructure as a service (IaaS) clouds provide customized anytime-anywhere computing resources to customers with ease. Due to these features, IaaS clouds become an attractive platform for executing parallel and high-performance computing (HPC) applications [2]. IaaS clouds usually contain a number of CDCs containing thousands of pre-configured virtual machines (VMs) to execute parallel applications, satisfying users or service providers' requirements [3]. Examples of different IaaS cloud infrastructures are Amazon EC2, Windows Azure, Rackspace, and Google Compute Engine [3].

1.1. Research Motivations

Cloud computing enables the remote execution of diverse applications of different types of customers. Such applications span a variety of fields such as IoT, Big-data analytics, HPC engineering, biomedical, etc. For example, global CDC traffic increased from 6 ZB in 2016 to around 20 ZB in the year 2021 [4,5]. Additionally, the recent COVID-19 pandemic has further triggered the heavy use of cloud computing resources since most of the essential day-to-day services associated with healthcare, work, food, and education are accessed online [6]. Consequently, CDC infrastructures are expanding in terms of size and scale to accommodate growing needs of user-oriented traffic. Such expansion resulted in a huge rise in energy consumption due to cloud resources, which further enhances carbon emissions and energy costs. As per a recent report, CDCs are among the world's largest energy consumers, and will be consuming 4.5% of total world energy consumption in 2025, and their energy costs are likely to be doubled almost every 5 years [7,8]. Predictively by 2025, data centers will contribute to emitting nearly 3.5% of carbon emissions (equivalent to 100 million metric tons of carbon pollution) globally per year in the environment [8]. As a result, it is essential to address the ever-growing issue of cloud energy consumption.

Along with the issue of cloud energy consumption, another research dimension focuses on cloud users demanding the execution of their applications as fast as possible with reduced makespan [6–8]. Therefore, multi-objective task scheduling strategy which meets the performance-energy tradeoff is the need of the hour. In this paper, a performance- and energy-aware scheduling solution for executing famous BoT applications over IaaS clouds is proposed. It is worth mentioning here that BoTs belong to the class of most executed workloads on clouds and other cloud-like parallel computing platforms [7,8].

The CBS problem for scheduling independent BoT applications over cloud resources is a popular NP-hard problem due to resource and workload heterogeneity, problem size, and cloud dynamicity [9–11]. In the past, many researchers addressed the CBS problem using queuing-, heuristic-, and metaheuristic (MH)-based solutions. Out of these, both queuing- and heuristic-based solutions did not produce global optimum results especially for bigger scheduling problems [8–12]. Contrarily, metaheuristics have produced effective scheduling solutions in polynomial time as compared to existing heuristics and queuing approaches [11–15]. WOA is one such prominent swarm-intelligence-based metaheuristic, and has proven its excellence in solving a wide range of real-life optimization problems including from wireless sensor networks, fluid mechanics, applied sciences, engineering, and academia [16–26]. However, besides the effectiveness in obtaining competitive solutions, the WOA still struggles to maintain robust exploration and has chances of trapping in local optima, resulting int inadequate exploration–exploitation tradeoff [23–25]. These deficiencies affect the scheduling solution quality thereafter. In addition to this, "No free lunch theorem (NFL)" [9] proves that there is no single metaheuristic algorithm fit enough to optimize all optimization problems. These facts act as strong motivations to undertake the current research work to propose a hybrid WOA-based scheduling approach to overcome the standard WOA limitations by hybridizing it with DE, chaotic maps, and OBL techniques. A list of acronyms used in this paper is presented in Table 1.

Table 1. List of acronyms.

Acronyms	Full Form	Acronyms	Full Form
ACO	Ant colony optimization	HS	Harmony search
ABC	Artificial bee colony	ICA	Imperialist colony algorithm
BAT	Bat algorithm	LWOA	Levy flight whale optimization algorithm
BFO	Bacterial foraging optimization	MCT	Minimum completion time
CS	Cuckoo search	MET	Minimum execution time
CSOA	Cat swarm optimization algorithm	MS	Moth search algorithm
CSOS	Coexistence symbiotic organism search	MFO	Moth flame optimization
DE	Differential evolution	MRFO	Manta-ray foraging optimizer
DSOS	Discrete symbiotic organism search	PSO	Particle swarm optimization
FA	Firefly algorithm	SA	Simulated annealing
FCFS	First come first serve	SGO	Social group optimization
GA	Genetic algorithm	SJF	Shortest job first
GD	Gradient descent	SOS	Symbiotic organism search
GWO	Grey wolf optimization	SSA	Salp swarm algorithm
GSA	Gravitational search algorithm	WPCO	Water pressure change optimization

1.2. Research Contributions

Inspired by the aforesaid issues, a hybrid metaheuristic for optimizing the CBS problem is proposed to schedule concurrent BoT applications. Research contributions are highlighted as follows:

(1) A hybrid metaheuristic called h-DEWOA is suggested, which uses the search mechanisms of chaotic map and OBL techniques to generate diverse initial population, and integrates the evolutionary DE method with the basic WOA approach to improve search exploration and convergence rate.

(2) The DE metaheuristic is further augmented with the OBL method to improve convergence rate and solution diversity throughout the h-DEWOA approach.

(3) A fitness-aware tradeoff scheme is integrated in the h-DEWOA approach to provide sufficient tradeoff between exploration and exploitation phases, which further helps to improve the quality of scheduling solutions.

(4) Finally, real-workloads of CEA-Curie and HPC2N supercomputing sites are selected for performance evaluation. Simulation results of the proposed h-DEWOA are compared with WOA-based and other state-of-the-art scheduling approaches using extensive experiments. Results and observations clearly show the supremacy of h-DEWOA over baseline algorithms.

The remaining part of the paper is outlined as follows. Section 2 highlights the related CBS problem research works. In Section 3, details of the system model, BoT application model, problem definition, and proposed fitness function are presented. Section 4 describes the encoding of the scheduling solution and methodology of the suggested h-DEWOA approach. Section 5 presents the simulation results, followed by the discussion and observations. In the end, the paper concludes by listing conclusions and future directions in Section 6.

2. Related Works

Existing research works for solving CBS problems involve the use of heuristics (e.g., FCFS, SJF, Min-min, Max-min, MCT, MET, and Suffrage [11]) and metaheuristics (e.g., GA, CS, ACO, PSO, WOA, BFO, etc. [10]). Heuristics offer problem-specific solutions and are suitable for small-sized problems, whereas metaheuristics (MHs) are simple, flexible, derivative-free, repetitive algorithms which guide a subordinate heuristic by an intelligent mechanism [27–29]. Out of these, metaheuristics-based scheduling solutions have produced better results than problem-specific heuristics, especially for complex and bigger scheduling problems [12,13,30]. However, MHs generally experience certain deficiencies, e.g., premature convergence, being caught in local optima, lack of diversity, and imbalance between exploration–exploitation phases [31,32]. These deficiencies may result in unacceptable solutions when applied over task scheduling problems. Hybrid metaheuristics have also been proposed in the literature to overcome standalone metaheuristics limitations by combining these with other MHs to produce better solutions [14,15].

A number of WOA-based scheduling solutions have already been suggested which are inspired by the hunting strategy of humpback whales, for scheduling BoT applications to obtain near-optimal results. These include solutions using standard, modified, and hybrid WOA approaches [8,22,23]. A recent cloud scheduling solution named GCWOAS2 [22] combines a Gaussian model, standard WOA, and OBL methods to generate efficient task–resource pairs. In another recent research work [8], a hybrid metaheuristic solution called OWPSO is proposed to remove the WOA deficiencies by combining OBL and PSO algorithms with the original WOA. In [24], authors suggested random double adaptive WOA (RDWOA) by utilizing the mutation operators of a Bee optimization algorithm for scheduling cloud tasks to minimize execution time and cost. Authors in [25] suggested an improved WOA approach by the using two advanced optimization strategies and incorporating the increase and decrease operators in the standard WOA to enhance search ability. In [26], authors suggested an IWC algorithm, which uses the inertial weight strategy to improve the local search efficiency and avoid the premature convergence of the basic WOA. Authors in [33] introduced WHOA by hybridizing WOA with HS to improve execution time, cost, and energy consumption. Sharma and Garg [34] suggested the WOA-based cloud task scheduling method, which results in simultaneous optimization of makespan and energy consumption. The whale-Scheduler approach was suggested to schedule BoT application over clouds, producing optimal makespan and execution cost [35].

Various GA-based scheduling solutions have been proposed in the past, either by applying standard GA method [36], or using improved GA approaches by modifying traditional mutation and crossover operations, and hybrid GA solutions, which integrate traditional GA with other methods [37,38]. Basic versions of SOS and certain modifications to SOS algorithms using chaotic maps and opposition-based learning have been employed by many researchers to achieve significant scheduling performance over state-of-the-art heuristics [39,40]. Standard ACO and other modified ACO-based scheduling solutions have been suggested by researchers to solve CBS problems to achieve different QoS objectives [41,42].

Standard PSO algorithms, modified PSO, and hybrid PSO variants have been used in many studies for solving CBS problems to optimize different objectives such as makespan, execution cost, degree of imbalance, and throughput [43,44]. An adaptive PSO approach is suggested for the CBS problem, by introducing an adaptive inertia weight scheme to tradeoff between exploration and exploitation [45]. Chen and Long [46] suggested a hybrid scheduling solution combining PSO and ACO methods to maintain population diversity and improve solution quality. In [37], authors applied a bi-objective PSO scheduler to enhance system performance and reduce the execution cost. Two deadline-constrained scheduling techniques employ a multi-objective PSO strategy to improve QoS metrics parameters [47,48]. In Ref. [49], authors applied a multi-objective PSO to reduce the workload makespan time.

Various CS-based task scheduling solutions have been proposed by using a standard version of CS [50] as well as by modifying the basic structure of CS, and hybridizing it with other metaheuristics [13,51,52]. In [13], Chhabra et al. proposed a hybrid CS metaheuristic by integrating CS and DE algorithms to improve the exploration of the original CS metaheuristic, resulting in better scheduling performance over the state-of-the-art heuristics. Another robust metaheuristic viz. GWO has been used in the past for producing near-optimal scheduling solutions to optimize different QoS metrics. For example, an MO-GWO approach was suggested to optimize both makespan and energy [53]. Modified versions of GWO, named MGWO and mean GWO, were proposed, which resulted in better performance over the baseline algorithms [54,55].

In another research work, Elaziz et al. in [56] have combined the efficient local searching feature of the DE algorithm in the MS algorithm to improve the scheduling solution. Milan et al. [57] have suggested the BFO-based scheduling approach to optimize the idle time, degree of imbalance, and overall runtime. The WPCO algorithm mimicking the occurrence of change in the water density is proposed to solve the CBS problem involving a single BoT application, resulting in optimal solution quality as compared to the baseline metaheuristics [58]. In Ref. [59], authors proposed an SGO-based scheduling solution, which mimics social group interactions to solve CBS problems, resulting in maximal throughput and reduced makespan. Authors in [6] tried to optimize performance and energy by generating the initial population of the used GA approach with the help of a Modified Worst Fit Decreasing heuristic. Some other recent task scheduling research works are presented in [60–71].

In a recent scheduling solution [72], a hybrid of multi-verse optimizer (MVO) and GA is combined to construct the MVO–GA approach to reduce total execution time of independent tasks of CBS problems. Similarly, we found that many recent research works involving multiple effective scheduling algorithms for executing a single BoT application are suggested, using improved ACO in [73], a hybrid of MRFO and SSA in [74], and deep-reinforcement learning (DRL) scheduler [75] to optimize different QoS parameters.

Limitations of existing recent scheduling techniques are shown in Table 2. General issues or limitations of the existing metaheuristics-based research works for scheduling BoT applications over cloud systems are insufficient balancing between exploration and exploitation phases, slow convergence, no focus on optimization of schedule order, lack of performance benchmarking using standard workloads, insufficient or absence of convergence analysis to tune metaheuristic parameters, and concurrent optimization of performance and energy consumption objectives. These deficiencies create the sufficient space to improve existing metaheuristics or design new metaheuristics to improve the efficiency of the CBS problem.

Table 2. Limitations and issues of recent cloud scheduling techniques.

Ref. & Year	**Technique**	**Single/Multiple BoTs**	**Architecture**	**Simulator**	**Limitations/Issues ***					
					L1	**L2**	**L3**	**L4**	**L5**	**L6**
[45]; 2022	AdPSO	Single	Private Cloud	CloudSim	√		√	√		√
[76]; 2022	IABS	Multiple	Hybrid Cloud	JAVA	√		√			√
[72]; 2022	MVO-GA	Single	Hybrid Cloud	MATLAB	√		√	√		√
[73]; 2022	IACO	Single	Private Cloud	MATLAB	√	√	√	√		√
[74]; 2022	MRFOSSA	Single	Private Cloud	CloudSim	√		√			√
[75]; 2022	DRL	Single	Private Cloud	Python						
[8]; 2022	OWPSO	Multiple	Private Cloud	CloudSim	√				√	
[6]; 2022	GA_LC-MLR	Single	Private Cloud	CloudSim	√		√		√	√
[24]; 2022	RDWOA	Single	Private Cloud	CloudSim	√		√			√
[70]; 2022	FLD-DGSTS	Single	Private Cloud	CloudSim	√		√		√	√
[22]; 2021	GCWOAS2	Single	Private Cloud	MATLAB	√		√	√		√
[26]; 2021	IWC	Single	Private Cloud	MATLAB	√		√	√	√	√

Table 2. *Cont.*

Ref. & Year	Technique	Single/Multiple BoTs	Architecture	Simulator	Limitations/Issues * L1	L2	L3	L4	L5	L6
[33]; 2021	WHOA	Single	Private Cloud	CloudSim	√		√	√		
[71]; 2021	IHHO	Single	Private Cloud	CloudSim	√				√	√
[13]; 2021	CSDEO	Multiple	Private Cloud	CloudSim	√				√	
[68]; 2020	CSPSO	Multiple	Private Cloud	CloudSim	√	√			√	
[49]; 2020	OBL-TP-PSO	Single	Private Cloud	CloudSim	√	√	√	√		√
[38]; 2019	BLEMO	Multiple	Private Cloud	CloudSim	√			√	√	
[51]; 2019	HCSGD	Single	Private Cloud	CloudSim			√	√		√
[53]; 2019	Mean GWO	Single	Private Cloud	CloudSim		√	√	√	√	√
[54]; 2019	MGWO	Single	Private Cloud	CloudSim		√	√	√	√	√
[77]; 2019	ICA-FA	Single	Private Cloud	MATLAB			√	√	√	√
[56]; 2019	MSDE	Single	Private Cloud	CloudSim		√	√	√		√
[23]; 2021	OppoCWOA	Single	Fog	Python		√	√	√		√
[35]; 2019	W-Scheduler	Single	Private Cloud	CloudSim	√	√	√	√		√
[50]; 2019	MOCSO	Single	Private Cloud	MATLAB		√	√	√	√	√
[47]; 2019	COGENT	Single	Private Cloud	CloudSim	√	√	√	√		√
[78]; 2019	CPSO	Single	Private Cloud	CloudSim		√	√	√		√
[52]; 2018	CGSA	Single	Private Cloud	CloudSim		√	√	√	√	√
[79]; 2018	PSOGA	Single	Private Cloud	CloudSim			√	√	√	√

* **L1**: Insufficient balance between exploitation and exploration, **L2**: Insufficient or absence of convergence investigation, **L3**: No optimization of BoT application ordering, **L4**: Benchmarking with real-cloud workloads, **L5**: Slow convergence, **L6:** Lack of/no attention on performance and energy together.

The current paper suggests a unique h-DEWOA approach, which takes into account both maximum completion time of tasks and energy saving aspects, adopts real supercomputing workloads for benchmarking, conducts systematic convergence examination to tune the proposed hybrid metaheuristic, optimizes both *task-execution order* and *allocation* of VMs sub-problems, and removes the inherent WOA deficiencies by incorporating chaotic maps, OBL, and DE techniques.

3. System Model and Problem Definition

This section explains the modeling of the BoT application, cloud data center, problem objectives, and the fitness function used in the proposed scheduling approach. Various notations are presented in Table 3.

Table 3. System notations.

Notations	Meaning
PM	Physical machine
VM	Virtual machine
Npm	Total physical machines
Nvm	Total virtual machines
m	Index of pre-configured VM
$VMID_m$	Unique identification number of mth VM
NC_m	Total CPU cores in mth VM
$MIPS_m$	VM capacity in million-instructions-per-second (MIPS)
C_{mk}	CPU core k at mth VM
CID_{mk}	Unique identification number of core
$CMIPS_{mk}$	Computation capacity of the core in MIPS
ECC_{mk}	Energy consumption of C_{mk} in execution mode (watts/hour)
EIC_{mk}	Energy consumption of C_{mk} in idle mode (watts/hour)
T_j	BoT application with index j
CBA	Number of concurrent BoT applications
TID_j	Unique identification number of BoT application
$TSize_j$	Number of cores needed to execute BoT application
$Tlength_j$	BoT application task length in million-instructions format

3.1. Cloud Data Center Model

The IaaS cloud infrastructure considered in this paper is composed of a single data center containing *Nvm* virtual machines associated with *Npm* physical machines for assignment to the queued BoT applications using the proposed h-DEWOA scheduling algorithm.

The cloud data center (*CDC*) is represented as follows:

$$CDC = \left\{PM_1, PM_2, \ldots, PM_i, \ldots, PM_{N_{pm}}\right\}, \tag{1}$$

Each PM_i is characterized as set of unique virtual machines as follows:

$$PM_i = \{VM_1, VM_2, \ldots, VM_m, \ldots, VM_{N_{vm}}\}, \tag{2}$$

Each VM_m is denoted using three attributes:

$$VM_m = \{VMID_m,\ NC_m,\ MIPS_m\}, \tag{3}$$

Each CPU core k of VM_m, i.e., $CORE_{mk}$ is represented using four tuples:

$$CORE_{mk} = \{COREID_{mk}, COREMIPS_{mk}, ECCORE_{mk}, EICORE_{mk}\}, \tag{4}$$

3.2. BoT Application, Execution Time, and Scheduling Model

Each BoT application is a set of non-communicating tasks, where each task requires a single CPU core for execution [8,12]. Due to the nature of BoT applications, it is a common practice to submit such applications in static mode to the cloud for execution. Concurrent *BoT* applications are submitted to the CDC broker for execution, which in turn invokes the h-DEWOA scheduling algorithm to decide the execution order of submitted BoTs and allocation of demanded number of VMs to BoT applications subject to the satisfaction of the fitness function as shown in Figure 1.

CBA is a set of concurrent BoT applications denoted as follows:

$$CBA = \{T_1, T_2, T_3, \ldots, T_j, \ldots, T_{N_{BTS}}\}, \tag{5}$$

Every BoT application (T_j) is defined as follows:

$$T_j = \{TID_j, TSize_j,\ Tlength_j, ET_j\}, \tag{6}$$

Execution time to execute a BoT application is calculated as follows:

$$ET_j = \frac{Tlength_j}{\sum_{\substack{k=1 \\ m \in VM}}^{TSize_j} C_{mk} \times CMIPS_{mk}} \tag{7}$$

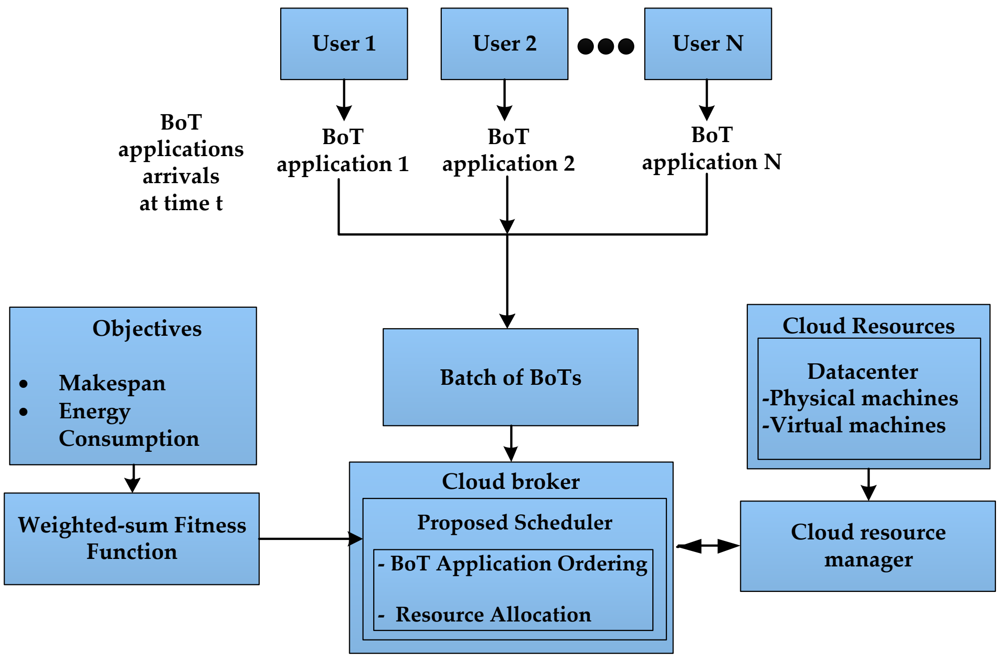


Figure 1. Proposed scheduling model using h-DEWOA.

3.3. Problem Objectives and Fitness Function

3.3.1. Makespan Model

Makespan (MS) is last task completion time while executing all tasks of submitted BoT applications. Completion time is calculated by adding the task wait time to task execution time. *MS* is calculated as follows:

$$MS = max_{j \in CBA}\left(CompletionTime_j\right) \tag{8}$$

3.3.2. Energy Model

The energy consumption (EC) of an individual CPU core can be expressed as:

$$EC(C_k) = \int_0^{MS} ECC(C_k, t) + ECI(C_k\,,\ t)dt, \tag{9}$$

The total energy consumption of the cloud data center can be calculated as:

$$EC = \sum_{C_k}^{Nvm} EC(C_k) \tag{10}$$

3.3.3. Fitness Function

In this research paper, the CBS problem is represented as a constrained combinatorial optimization problem to reduce both makespan and total energy consumption objectives. The fitness function, F(X), is represented using a *Weighted-sum-method* [8] as follows:

$$\text{F(X)} = min\ (MS_{weight} \times \text{MS} + EC_{weight} \times \text{EC}) \tag{11}$$

where MS_{weight} and EC_{weight} are weights of Makespan and Energy consumption objectives, respectively. After conducting a few independent pilot experiments by varying weights, we found that optimal values of these weights are $MS_{weight} = 0.5$ and $EC_{weight} = 0.5$. Therefore, Equation (12) can be rewritten with the obtained weight values as follows:

$$F(X) = min\ (0.5 \times MS + 0.5 \times EC) \quad (12)$$

Constraints of the CBS problem are given as per below:

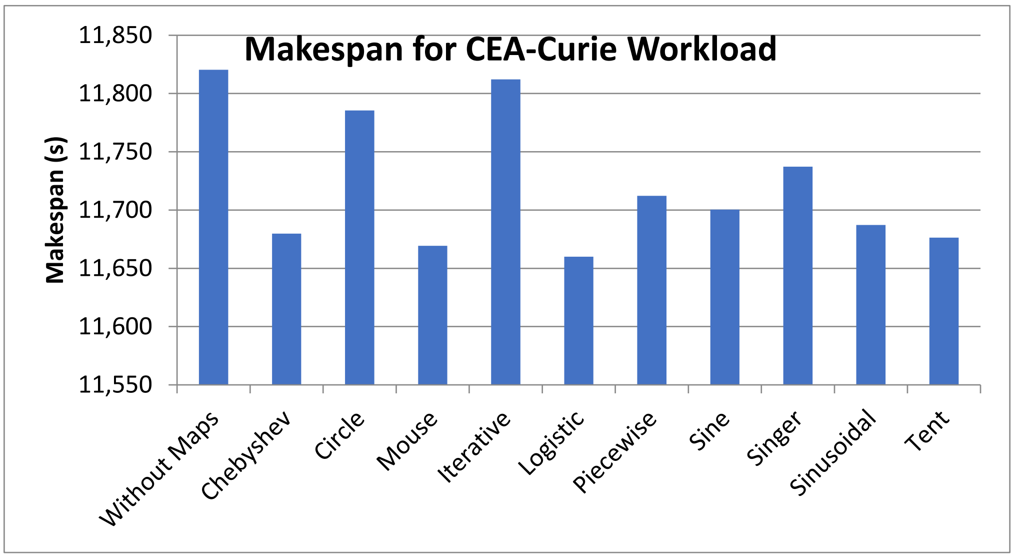
1. Every BoT application demands a static number of VMs for processing.
2. Every VM only processes a single task of the BoT application at a time.
3. The numbers of BoT applications and VMs in cloud data center are fixed.
4. Deadlines are not associated with the execution of BoT applications.

4. Proposed Scheduling Methodology

This sub-section briefly explains methods, e.g., chaotic maps, opposition-based learning, basic WOA algorithm, and traditional DE algorithm, which form the basis of the proposed h-DEWOA strategy.

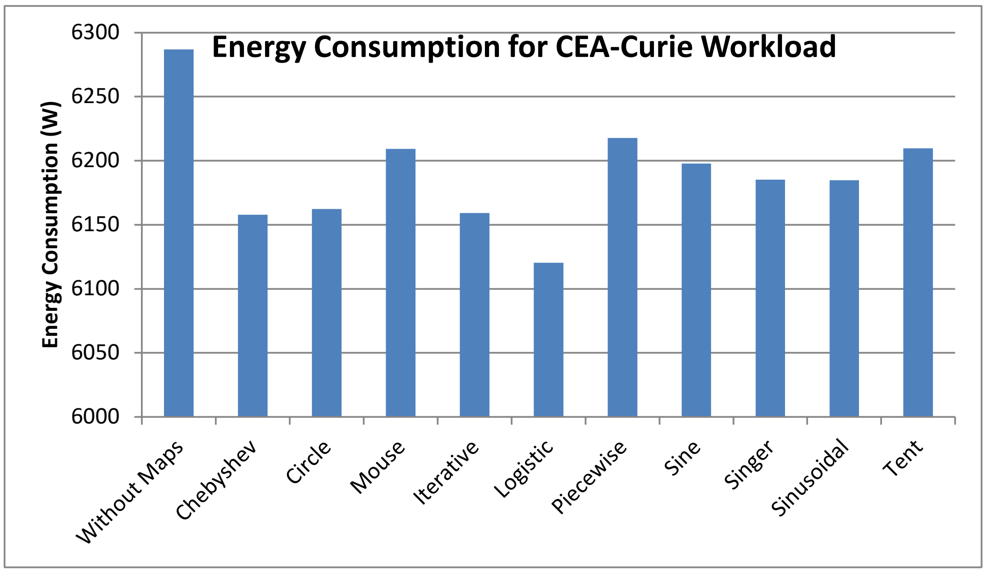
4.1. Chaotic Maps (CM)

Chaos is a bounded nonlinear arrangement having both ergodic and stochastic properties. Generally, for initial population generation, metaheuristics (MHs) use random solutions having sub-standard population diversity, leading to less-fitted final solutions. To overcome such situations in MHs, many recent research works have employed chaotic maps to generate an initial population of well-distributed non-repetitive chaotic sequences leading to more population diversity, better exploration, and improved convergence rate. Figure 2 shows the performance of pilot experiments of h-DEWOA with ten different chaotic maps in the case of CEA-Curie workload. It is evident that the logistic map improves the efficiency of the proposed h-DEWOA approach among all tested chaotic maps and overall chaotic maps resulted in better scheduling performance as compared to the situation when the initial population is randomly generated.



(**a**)

Figure 2. *Cont.*



(**b**)

Figure 2. Performance of pilot experiments of h-DEWOA with chaotic maps for CEA-Curie workload: (**a**) Makespan; (**b**) Energy consumption.

4.2. Opposition-Based Learning (OBL)

OBL is another useful strategy for metaheuristics, and has been proved to be useful for helping MHs avoid trapping in local optima and hence improve their convergence rate [80]. The OBL method is straightforward: it selects candidate solutions from the original population at random and generates their inverse solutions. Following that, the best-fitting solutions among OBL solutions and random solutions are chosen as the beginning population. In this approach, the OBL method aids the suggested h-DEWOA strategy by adding optimal diversified initial populations rather than solely random solutions, resulting in increased population variety and speed of convergence.

4.3. Whale Optimization Algorithm (WOA)

The WOA is a population-based MH mimicking the foraging and hunting behavior of humpback whales [81]. It consists of a population of search agents (called whales or solutions) capable of conducting independent searches and having indirect concurrent communications between each other during exploration and exploitation phases, which results in achieving global optima in the solution space [25]. The WOA search mechanism is described mathematically in the following sub-sections obtained from the original WOA research paper [81]. WOA notations and symbols are defined in Table 4.

4.3.1. Encircling Prey (When $|A| < 1$ and $p < 0.5$)

Encircling behavior of humpback whales is represented using the following equations [81]:

$$\vec{X}(G+1) = \vec{X}^*(G) - \vec{A}\cdot\vec{D} \tag{13}$$

$$\vec{D} = |\vec{C}\cdot\vec{X}^*(G) - \vec{X}(G)| \tag{14}$$

$$\vec{A} = 2\vec{a}\cdot\vec{r} - \vec{a} \tag{15}$$

$$\vec{C} = 2\vec{r} \tag{16}$$

$$a = 2 - \frac{2G}{G_{max}} \tag{17}$$

Table 4. WOA notations and symbols.

Notations	Meaning
G	Generation index
$\vec{X}(G)$	Present solution position at generation G
$\vec{X}^*\ (G)$	Position of present best solution
$\vec{D}$	Distance between any solution and current best solution
$\vec{A}$ and $\vec{C}$	Coefficient vectors
a	Variable that linearly decreases from 2 to 0
∣ ∣	Absolute value
dot (·)	Element-by-element multiplication
$\vec{r}$	Random vector in [0, 1]
$Gmax$	Maximum number of generations
$\vec{D'}$	Absolute distance between any solution and present best solution
b	Constant representing the shape of the logarithmic spiral
l	Random number in $[-1, 1]$
$\vec{X}_{rand}(G)$	Random solution during exploration phase for generation G

4.3.2. Bubble-Net Attacking (Exploitation Process)

In this phase, whale solutions swim around the target solution in a shrinking circle and along a spiral path simultaneously with 50% chances of performing each activity [81].

Shrinking Encircling Mechanism

This mechanism is achieved by decreasing the fluctuation range of vector A in Equation (15), which uses the variable a, using Equation (17) to update the position of the candidate solution anywhere between $\vec{X}^*\ (G)$ and $\vec{X}(G)$.

Spiral updating position (When ∣A∣ < 1 and p < 0.5)

A spiral path between the present whale position and target prey solution, which simulates the helix-shaped movement of whales, can be defined as follows:

$$\vec{X}(G+1) = \vec{D'}\cdot e^{bl}\cdot\cos(2\pi l) + \vec{X}^*(G) \tag{18}$$

$$\vec{D'} = |\vec{X}^*(G) - \vec{X}(G)| \tag{19}$$

4.3.3. Search for Prey

This exploration mechanism represented by Equation (20) is called when the value of $p \geq 0.5$.

$$\vec{X}(G+1) = \vec{X}_{rand}(G) - \vec{A}\cdot\vec{D} \tag{20}$$

$$\vec{D} = |\vec{C}\cdot\vec{X}_{rand}(G) - \vec{X}(G)| \tag{21}$$

4.4. Differential Evolution (DE)

DE is an evolutionary MH possessing faster convergence, low computational complexity, and satisfactory exploration ability [56]. These qualities make it popular among researchers for solving a variety of optimization problems [56,82]. DE evolves a population of possible solutions by using mutation, crossover, and selection operations. The DE algorithm can be mathematically formulated as follows:

DE initial population consisting of p random solutions is denoted as $P_{X,G}$ as follows:

$$X_L \leqslant X_{i,0} \leqslant X_U\ \forall i \in \{1,2,\ldots,P\},\ P_{X,G} = (X_{i,G}),\ i \in \{1,2,\ldots,P\},\ G \in \{0,1,\ldots,G_{\max}\},\ X_{i,G} = (X_{j,i,G}),\ j \in \{1,2,\ldots,d\}.$$

where $\mathbf{P}_{\mathbf{X},\mathrm{G}}$ represents an array of p vector solutions, $\mathrm{X}_{i,G}$ denotes d—dimensional vector representing a solution, i is a solution vector index, j is a problem dimension index, and the parentheses indicate an array.

4.4.1. DE Mutation Operator

For every generation, the following mutation operations are applied on the current population $\mathbf{P}_{\mathbf{X},\mathrm{G}}$ to obtain a mutated population $\mathbf{P}_{\mathrm{M,G}}$ [82].

$$\mathrm{M}_{i,\mathrm{G}} = \begin{cases} X_{best,\mathrm{G}} + F\cdot(X_{r1,\mathrm{G}} - X_{r2,\mathrm{G}}), & if\ (rand\ (0,1) \leqslant r_b) \\ X_{i,\mathrm{G}} + F\cdot(X_{best} - X_{i,\mathrm{G}}) + F\cdot(X_{r1,\mathrm{G}} - X_{r2,\mathrm{G}}), & otherwsie \end{cases} \tag{22}$$

where r_1, and r_2 are random solutions in {0,1, . . . ,$p - 1$} and F is a scalar parameter. A high value of F is desirable for exploration. Switching factor r_b allows the mutation to switch between *DE/best/1/bin* and *DE/rand-to-best/1/bin* mutation mechanisms.

4.4.2. DE Crossover Operator

Next, a trial population $\mathbf{P}_{\mathrm{C,G}}$ is generated as follows:

$$\mathrm{C}_{i,\mathrm{G}+1} = \begin{cases} M_{i,j,\mathrm{G}}, & if\ (rand\ (0,1) \leqslant CR\ or\ \ j = j_{rand}) \\ \mathrm{X}_{i,j,\mathrm{G}}, & otherwsie \end{cases} \tag{23}$$

where j_{rand} is a random index in {0,1, . . . ,$p - 1$} to ensure that at least one design variable originates from the mutant vector, and C_r is a scalar DE algorithm parameter, which is normally in the range (0,1] in the classical DE. CR controls the mutation; a larger CR ensures that more solution vectors are selected from mutant population [82].

4.4.3. DE Selection Operator

The final step is applying a greedy selection method [82] to obtain the minimum fitness solution as follows:

$$\mathrm{X}_{i,\mathrm{G}+1} = \begin{cases} \mathrm{C}_{i,\mathrm{G}}, & if\ Fitness(\mathrm{U}_{i,\mathrm{G}}) < Fitness(\mathrm{X}_{i,\mathrm{G}}) \\ \mathrm{X}_{i,\mathrm{G}}, & otherwise \end{cases} \tag{24}$$

Note that, in this research work, DE operators have been incorporated in the h-DEWOA approach to improve the exploration ability of the basic WOA method.

4.5. Limitations of the Standard WOA Approach

Existing research works [16–25] have exposed certain deficiencies of WOA viz. weak exploration, improper exploration–exploitation tradeoff, and insufficient solution diversity. WOA performs spiral updating exploitation activity with 50% chances, while whales tend to perform exploitation by encircling the prey and search new solutions (i.e., conduct exploration) activities in the rest of 50% chances [20]. This clearly confirms that the original WOA method conducts more exploitation than exploration, resulting in serious imbalance between exploitation and exploration as well as causing insufficient solution diversity over the generations of execution of the WOA method. Thus, the WOA may become stuck in local optima, forcing premature convergence and causing weak solutions. Moreover, the exploration–exploitation imbalance also weakens the performance during initial phases because early stages require more exploration than exploitation [21].

Another limitation of the WOA method is that its exploration activity is more dependent on the random search, which fails to incorporate any knowledge about the current position of the best solution found thus far [25]. This random exploration produces sub-standard solutions in the successive generations, affecting WOA solution diversity and final solution quality thereafter.

Lastly, the exploration–exploitation tradeoff in the traditional WOA is purely decided using a randomization mechanism without incorporating the fitness of current solution, which subsequently results in degradation of WOA performance [8].

4.6. Proposed h-DEWOA Algorithm

The proposed h-DEWOA eliminates aforesaid deficiencies of the standard WOA method by boosting the exploration ability, improving solution diversity, and enhancing balance in the exploration–exploitation tradeoff. h-DEWOA hybridizes DE, OBL, and chaotic map techniques with the standard WOA to schedule a set of *BoTs* over different types of cloud resources. Before explaining the detailed methodology of h-DEWOA, it is important to discuss the encoding of the proposed CBS solution, which plays a great role in the implementation of final solution.

4.6.1. Solution Encoding

The solution (X) to the CBS problem involves encoding of *BoT application ordering* (X^O) and *Resource allocation* (X^A) dimensions as shown in Figure 3.

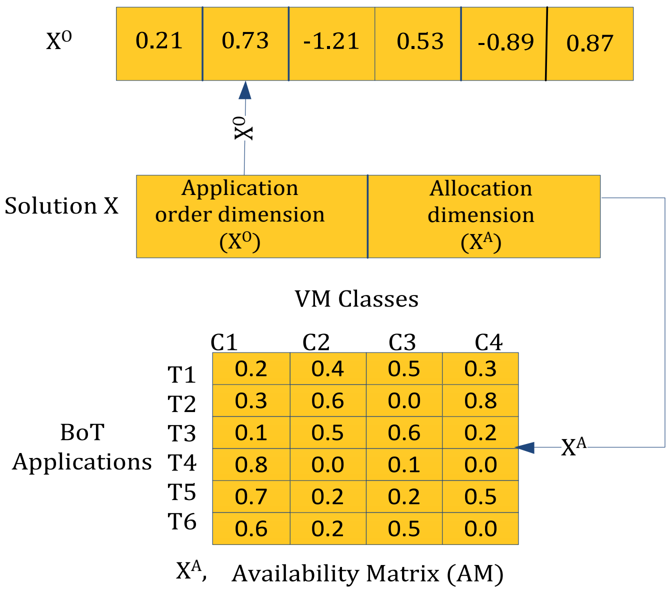


Figure 3. Encoding of the initial solution.

The basic WOA is only applicable for continuous-value problems. To apply the WOA approach over discrete-nature CBS problems, a discretization mechanism is suggested to convert real-value solutions to equivalent discrete ones in the proposed h-DEWOA approach. The discretization process is shown in Figure 4 and explained as follows:

Discretization of BoT Application-Order Vector

In the beginning, the BoT application-order dimension of scheduling solution is encoded by a continuous solution vector, X^O. Discrete-value execution order (X^{*O}) is obtained by applying the smallest position value (SPV) method over X^O using Algorithm 1 [8,13]. The process is elaborated in Figure 5a.

Algorithm 1. SPV method.

```
Input: Real-value BoT application-order vector $X^o$
Output: Discrete BoT application-order vector $X^{*o}$
1. Begin
2. Sort $X^o$ in the increasing order
3. Calculate $X^{*o}$ with discrete-values where $X^{*o} = dimension(sorted X^o)$
4. End
```

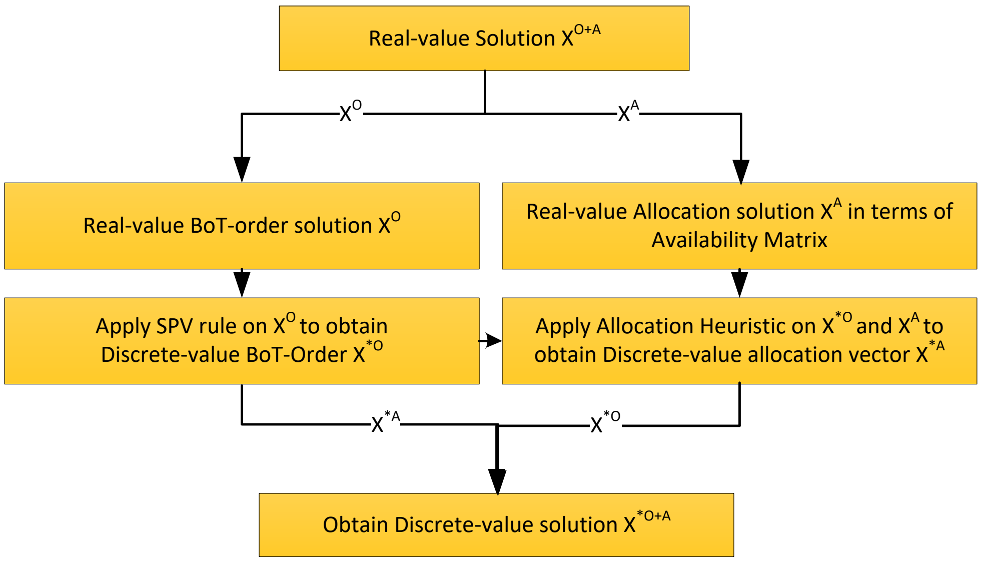

Real-value Solution X^O+A
X^O
X^A
Real-value BoT-order solution X^O
Real-value Allocation solution X^A in terms of Availability Matrix
Apply SPV rule on X^O to obtain Discrete-value BoT-Order X^*O
Apply Allocation Heuristic on X^*O and X^A to obtain Discrete-value allocation vector X^*A
X^*A
X^*O
Obtain Discrete-value solution X^*O+A

Figure 4. Discretization mechanism for producing discrete-value solution [13].

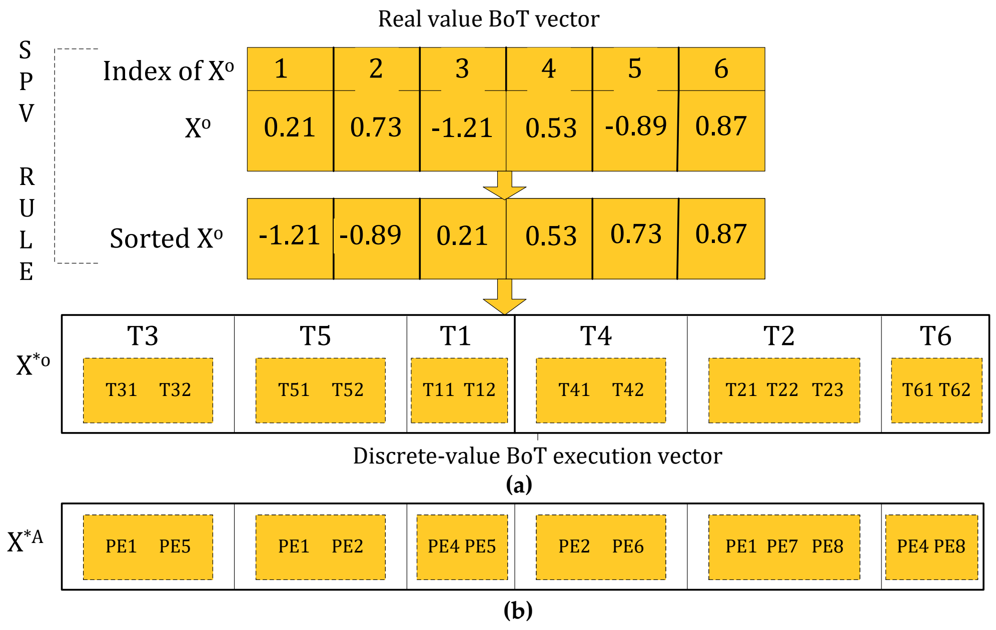


Figure 5. (**a**) Example showing procedure to obtain X^{*O} vector applying SPV method, (**b**) Example showing procedure to obtain X^{*A} vector applying *MDeRA* heuristic.

Discretization of Resource Allocation Vector

Initially, the resource allocation dimension (X^A) is represented by VM availability matrix (AM) [38,57,58] consisting of real-value entries representing a *BoT application-vmType* pair. Each pair entry indicates the fixed percentage of availability of VMs of a particular *vmType* for allocation to the BoT application. X^{*A} vector, i.e., the actual discrete allocation of VMs to tasks of BoT application, is obtained by applying the *MDeRA* (Modified Discrete and Efficient Resource Allocation) heuristic on AM matrix and X^{*O} vector, as shown in the Figures 4 and 5b. *MDeRA* heuristic (pseudo-code in the Algorithm 2) is an improved version of the published work *DeRA* heuristic [13] and allocation heuristic [38]. Instead of random BoTs execution-order in the *DeRA* heuristic, *MDeRA* initially sorts the BoTs in the increasing order of their expected execution time (Line 1). The rest of the *MDeRA* mechanism is the same as the *DeRA* heuristic. Readers are further encouraged to read [13] for discussion about the rest of the allocation mechanism in detail. In a nutshell, the *MDeRA* heuristic provides an efficient discrete resource allocation of VMs to BoT applications, optimizing both makespan and energy consumption.

Algorithm 2. MDeRA heuristic.

```
Input: PM: Total physical machines (pm), VM: Total virtual machines (vm),
vms.PM: Virtual machines (vm) in each pm, CBA: Set of concurrent BoT applications
Require: AM matrix representing VMs availability
Result: Allocation list (AL): a pair of (BoT application ∈ CBA, vm ∈ VM)
1. Sort available BoTs in CBA in the ascending order of their expected execution time
2. for each BoT application ∈ CBA do
3.     Apply DeRA heuristic [13].
4. end for
```

4.6.2. Pseudo-Code of the h-DEWOA Algorithm

Pseudo-code of the h-DEWOA is provided in Algorithm 3 as follows:

Algorithm 3. h-DEWOA Algorithm.

```
Input: Tasks, Virtual machines, fitness function: ObjFn(X),
Variables: $P_{\mathbf{X},G}$: Real-value population, $P_{\mathbf{XD},G}$: Equivalent discrete-value population, G: Generation index; N: Population size, $G_{max}$: Total generations
Output: $\vec{XD}^*$: Best solution containing discrete-value BoT-order and resource allocation
1: Initialization of parameters of scheduling problem, WOA, and DE metaheuristics
2: G = 1//First generation for producing initial population
3: $P_{\mathbf{XChaotic},G} \leftarrow \frac{1}{2}$ ChaoticInitialPopulation( )//First half of initial real-value population using logistic map
4: $P_{\mathbf{XOBL},G} \leftarrow \frac{1}{2}$ OBLbasedInitialPopulation( )//Second half of initial real-value population using OBL
5: $P_{\mathbf{X},G} \leftarrow P_{\mathbf{XChaotic},G} + P_{\mathbf{XOBL},G}$    //Merge both halves of initial real-value population
6: $P_{\mathbf{XD},G} \leftarrow MDeRA(SPV(P_{\mathbf{X},G}))$    //Generate discrete-form population using SPV and MDeRA heuristics
7: $Fitness\ (P_{\mathbf{XD},G}) \leftarrow ObjFn(P_{\mathbf{XD},G})$    //Evaluate the fitness of population
8: $\vec{XD}^* = argmin\ (Fitness\ (P_{\mathbf{XD},G}))$    //Global best discrete-value solution
9: $\vec{X}^* \leftarrow$ getrealvaluesolution $(\vec{XD})$    //Global best real-value solution
10. G = G + 1
11: while (G < MaxGeneration)
12:   for each solution $X_{i,G} \in$ population $P_{\mathbf{X},G}$ do
13:   if (Fitness (solution $X_{i,G}$) <= Fitness($gBest_{G-1}$)) //Exploration–exploitation tradeoff condition
14:     for both task-order and allocation dimensions of solution $X_{i,G}$ do//WOA Exploitation
15:       Apply WOA on solution $X_{i,G}$ using Equation (13) and Equation (18) to obtain $X_{i,G+1}$
16:       Store position if $X_{i,G+1}$ is better than $X_{i,G}$
17:     end for    //WOA phase ends
18:     else (Fitness (solution $X_{i,G}$) > Fitness($gBest_{G-1}$))
19:     for both task-order and allocation dimensions of solution $X_{i,G}$ do//DE Exploration
20:         Generate Mutant of $X_{i,G}$ using Equation (22)
21:             Crossover of Mutant with the original solution using Equation (23)
22:             Obtain solution $X_{i,G+1}$ using selection operator using Equation (24)
23:                 Apply OBL on $X_{i,G+1}$//Apply OBL on DE solution to improve exploration
24:                 Retain best solution among OBL- and non-OBL-based DE solution
25:         end for    //DE phase ends
26:   end if
27:   end for
28:     $P_{\mathbf{XD},G} \leftarrow MDeRA(SPV(P_{\mathbf{X},G}))$//Generate discrete-form population
29:     $Fitness\ (P_{\mathbf{XD},G}) \leftarrow ObjFn(P_{\mathbf{XD},G})$ //Fitness evaluation Evaluate the fitness of population
30:     Update the current integer and real-value global best solution
31: end while
32: $\vec{XD}^* = argmin\ (Fitness\ (P_{\mathbf{XD},G}))$//Record the global best scheduling solution found
```

The line-by-line explanation of the proposed h-DEWOA scheduling approach is provided as follows:

Line 1: Initialize parameters of the scheduling problem, WOA algorithm, and DE algorithm
Line 2: Initialize G equal to 1
Line 3: Generate first random half of initial real-value population using logistic chaotic maps.
Line 4: Generate second random half of initial real-value population using opposition-based learning method.
Line 5: Merge both initial population halves generated in Line 2 and 3 to generate full initial real-value population.
Line 6: Convert real-form population into an equivalent discrete-form population using SPV and *MDeRA* heuristic.
Line 7: Evaluate the fitness of the population using objective function.
Line 8: Find the global best discrete-value solution from whole population
Line 9: Generate the real-value population
Line 10: Update G value
Line 11: Iterations of the h-DEWOA using *while* loop till ($G < MaxGeneration$) condition is true
Line 12: Start of the *for* loop to update the position of solutions
Line 13–17: Exploration–exploitation tradeoff condition is tested on the basis of fitness of current solution (Line 13). When the current solution's fitness is equal to or better than the previous generation population's global best solution (Line 13), then WOA exploitation mechanism is applied on the current solution position to search nearby better solutions (Line 14–17).
Line 18–25: When the current solution's fitness is less than the previous generation population's global best solution (Line 18), the h-DEWOA algorithm performs DE exploration activity for searching better far-away solutions (Line 19–22). After that, OBL is applied on the real-value solution and select the best solution between the OBL solution and non-OBL solution (Line 23–25).
Line 26: End of if condition for switching between exploration and exploitation.
Line 27: End of for loop for updating solution position.
Line 28: Obtain the discrete-form population by applying *SPV* and *MDeRA* heuristic.
Line 29: Evaluate the fitness of new population using fitness function.
Line 30: This step results in updating of the current integer and real-value global best solution.
Line 31: End of while loop since algorithm iteration condition is met.
Line 32: Output the best solution found with minimum fitness, which represents the optimal schedule consisting of BoT application-order and allocation of VMs to BoT applications.

Summarizing the discussion, the following substantial amendments are added to improve the standard WOA, resulting in the design of the proposed h-DEWOA approach:

- Initially, both logistic map and OBL are incorporated to produce an optimal initial population, leading to improved exploration and convergence rate thereafter in the later stages of the proposed h-DEWOA approach.
- Thereafter, the DE metaheuristic is added to the traditional WOA to boost the exploration ability and convergence rate. The OBL method is also incorporated in the DE to further maintain population diversity throughout consecutive generations, producing efficient final solutions and improving convergence rate further.
- Further, a fitness-based exploration–exploitation tradeoff mechanism is incorporated in the proposed h-DEWOA method to adequately balance the DE exploration and WOA exploitation phases, which leads to generation of better schedules.

4.7. Time Complexity Analysis

Overall time complexity of the h-DEWOA is dependent on the structure and phases of the algorithm involving WOA, DE, OBL, CM, the number of iterations (G_{max}), population size (N), and dimension (*dim*) of the CBS problem. The time complexity of DE is $O((N \times dim + N \times dim + N) \times G_{max})$, which can be reduced to $O(N \times dim \times G_{max})$, whereas the time complexities of the logistic chaotic map and OBL (line 3–4) are $O(N \times dim)$ and $O(N \times dim)$, respectively. Similarly, the time complexity of WOA is $O(N \times dim \times G_{max})$. Combining all these time complexities of the h-DEWOA can be written as follows:

$$O(h\text{-}DEWOA) = O(CM) + O(OBL) + O(WOA) + O(DE) \tag{25}$$

$$O(h\text{-}DEWOA) = O(N \times dim) + O(N \times dim) + O(\alpha \times (N \times dim) + (N - \alpha) \times (N \times dim + N \times dim)\ G_{max}) \tag{26}$$

where α and $(N - \alpha)$ are the solutions processed by the WOA exploitation and DE exploration operators, respectively. It can be seen that Equation (26) can be easily reduced to $O(N \times dim \times G_{max})$, which is the same as that of complexity of the traditional WOA method.

5. Experimentation and Results

Testing and implementation of algorithms involved in this paper were conducted on the CloudSim 3.0.3 simulator [83] using Java language and JMetal 5.4 [84] installed on a computer system with i7-8550U @ 1.80–2.0 GHz (8 Cores) processor, 16 GB RAM, and Windows 10 operating system. CloudSim is a standard simulator, which provides testing of synthetic and real workloads in a repeatable and controllable environment with different infrastructure configurations.

5.1. Workloads and Cloud Configuration

In the current research work, experiments are conducted using 20 different real-workloads, which are obtained from CEA-Curie and HPC2N workload logs of supercomputing sites (refer to http://www.cs.huji.ac.il/labs/parallel/workload accessed on 23 January 2022) as shown in Table 5.

Table 5. Sample real experimental workloads.

Workload Log	Workloads	Number	BoTs per Workload	Max. CPUs
CEA-Curie	WL0–WL9	10	200	32
HPC2N	WL10–WL19	10	200	16

This paper assumes a cloud computing system comprising of a single data center which further contains five different categories of pre-configured VMs, as shown in Table 6 [19].

Table 6. Description of the VM configuration.

VM Instance Type	VM Instance id	# VMs	# CPU Cores per VM	MIPS per Core	CPU Model	Energy during Idle Time (W/h)	Energy during Comp. Time (W/h)
Category 1	T2.nano	20	1	3400	Xeon E5-2637 V4	23.625	33.75
Category 2	T2.xlarge	10	4	2600	Xeon E5-2623 V4	59.5	85
Category 3	T2.2xlarge	8	8	2100	Xeon E5-2620 V4	59.5	85
Category 4	M5.4xlarge	6	16	2500	Xeon Plat. 8180 M	82	117.14
Category 5	M4.10xlarge	4	40	2400	Xeon E5-2686 V4	225.55	322.22

5.2. Comparative Results

In this paper, two series of experiments are conducted; one with WOA-based metaheuristics and another with non-WOA-based metaheuristics as shown in Table 7.

Table 7. Comparative experimental plan.

Experiments	Details	Workloads
First series	Comparing h-DEWOA with popular WOA-based metaheuristics	CEA-Curie and HPC2N
Second series	Comparing h-DEWOA with well-known non-WOA-based metaheuristics	CEA-Curie and HPC2N

5.2.1. Comparison with WOA Variants—First Series of Experiments

This sub-section deals with the performance evaluation of the h-DEWOA against popular WOA-based metaheuristics using CEA-Curie and HPC2N benchmarks.

5.2.2. Baseline WOA Variants

In this section, the theoretical characteristics of all comparative scheduling algorithms and the proposed h-DEWOA approach are presented as per Table 8.

Table 8. Characteristics of WOA-based scheduling algorithms.

Algorithm	Algorithm Type	Algorithm Description	BoT Application Ordering Method	Scheduling Problem Optimization
WOA [81]	MH	Aggregation-based Multi-objective WOA	FCFS	AL
IWOA [16]	h-MH	Pareto front-based Multi-objective WOA + DE	FCFS	AL
HSWOA [85]	h-MH	Aggregation-based Multi-objective HS + WOA	FCFS	AL
GCWOAS2 [22]	h-MH	Pareto front-based Multi-objective WOA	FCFS	AL
h-DEWOA (proposed)	h-MH	Aggregation-based Multi-objective WOA + DE	h-DEWOA	AO and AL

MH: Metaheuristic; h-MH: Hybrid Metaheuristic; AL: Allocation; AO: Application ordering.

This section is dedicated towards presenting comparative results of the h-DEWOA and baseline scheduling algorithms using benchmarking workloads. To remove any biasing and to provide statistical analysis, each scheduling algorithm runs 30 times with fixed workload and VM configuration.

5.2.3. Convergence Examination and Final Values of Parameters

Convergence investigation is carried out to fix optimal values of the population size (*PopSize*) and number of generations (*NGen*) parameters of undertaken metaheuristics, as these values affect the final solution quality. Some pilot experiments are conducted using CEA-Curie and HPC2N workloads to find optimal values by alternatively varying the *PopSize (10-100)* and *NGen (1-100)* so that an adequate tradeoff point between solution quality and convergence rate can be determined. Initial convergence investigation with workload of 200 BoT applications reveals that almost every scheduling approach showed good performance when the *PopSize* is in the range of 60 to 70 for both HPC2N and CEA-Curie workloads. The convergence investigation of tested scheduling algorithms for CEA-Curie workloads with *PopSize* = 60 is shown in Figure 6. Values of parameters other than *PopSize* and *NGen* of different metaheuristics are taken from original research papers [16,22,81,85].

After performing parameter tuning, final parameter settings of different metaheuristics with *PopSize* = 60 are shown in Table 9 for the first series of experiments.

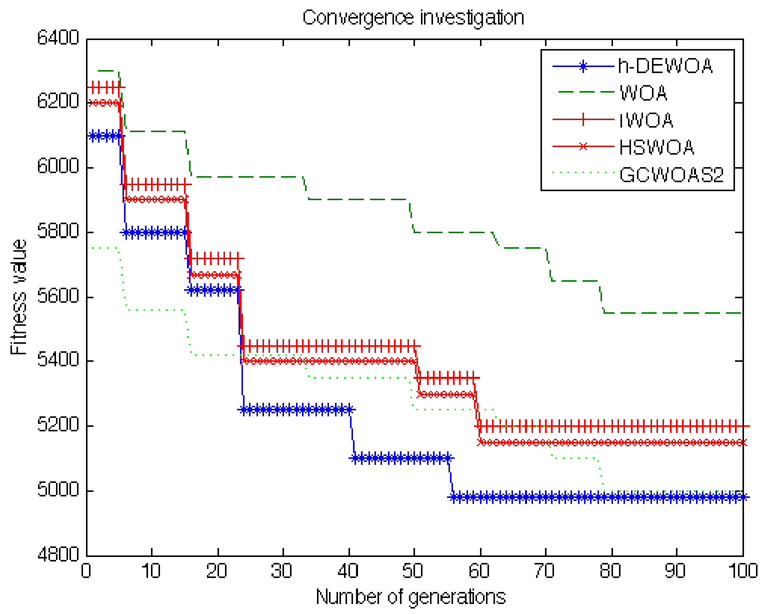


Figure 6. Convergence examination of algorithms with *PopSize* = 60 for CEA-Curie workload for first series of experiments.

Table 9. Optimal parameter settings of algorithms for the first series of experiments for 30 independent evaluations.

Algorithm	**Generations**	**Parameters**
WOA	60	$a = 2$
IWOA	60	$a = 2$, F = [0.1–0.9], and CR = 0.75
HSWOA	60	$a = 2$
GCWOAS2	70	$a = 2$, $lb = -5$, $ub = 10$
h-DEWOA	50	$a = 2$, F = [0.5–0.9], and CR = 0.75

5.2.4. Performance Evaluation of First Set of Experiments

This sub-section analyzes the h-DEWOA algorithm's efficiency and robustness over WOA-based baseline scheduling approaches in terms of *Best*, *Average*, and *Worst* metrics, which are minimum, average, and maximum value, respectively, among 30 executions using individual workloads from the set of CEA-Curie and HPC2N workloads.

Workload-Wise Makespan Results for CEA-Curie Workloads

Figure 7 presents the makespan results of scheduling algorithms obtained from executing WL0–WL9 workloads of CEA-Curie traces. It is observed that the h-DEWOA algorithm produces the minimum makespan values of best, average, and worst case metrics in most CEA-Curie workloads.

Workload-Wise Energy Consumption Results for CEA-Curie Workloads

Figure 8 indicates that the h-DEWOA algorithm produces the minimum best, average, and worst values of the energy consumption in a majority of CEA-Curie workloads.

Workload-Wise Makespan Results for HPC2N Workloads

The h-DEWOA algorithm also produces very good results against WOA-based baseline techniques for HPC2N workloads as shown in Figure 9.

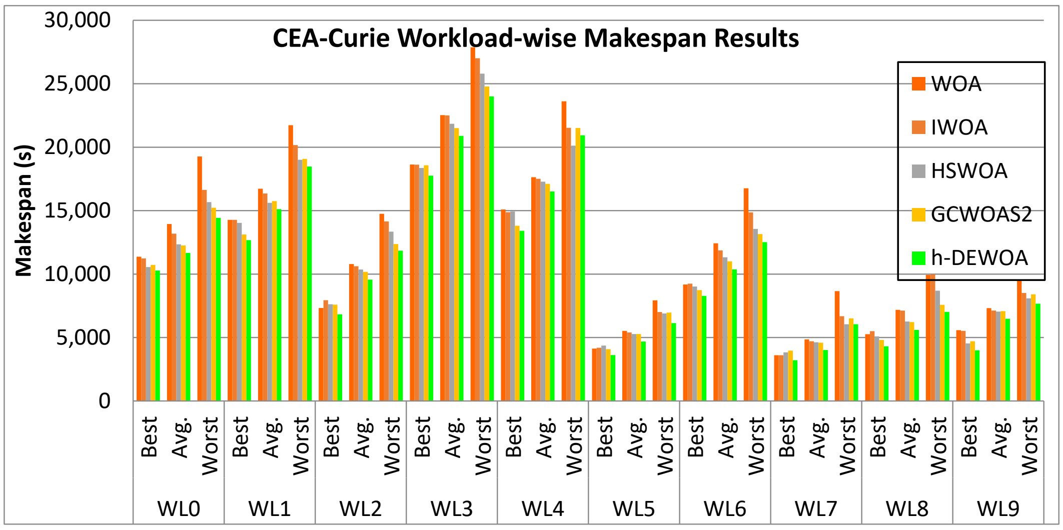


Figure 7. Statistical results of the makespan metric for WL0–WL9 workloads.

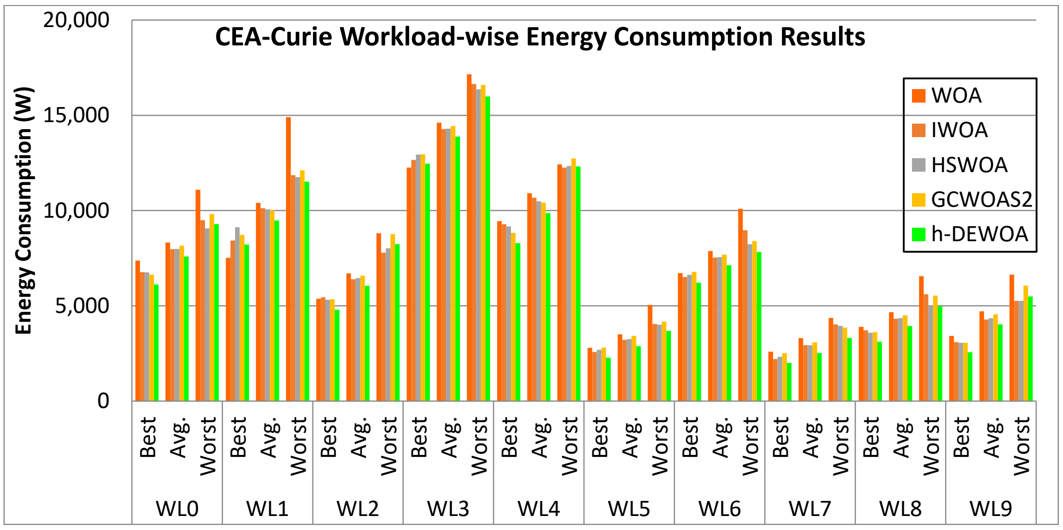


Figure 8. Statistical results of the energy consumption metric for WL0–WL9 workloads.

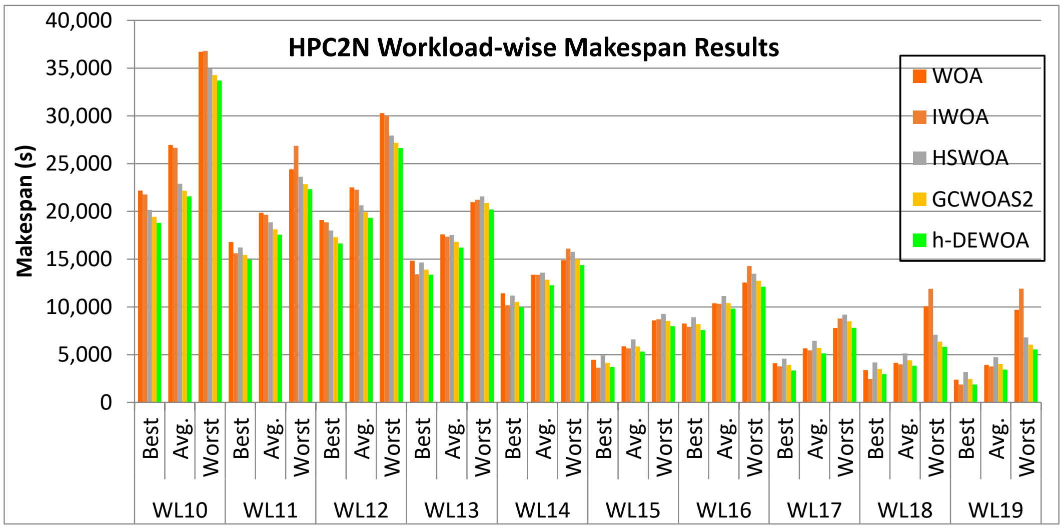


Figure 9. Statistical results of the makespan metric for WL10–WL19 workloads.

Workload-Wise Energy Consumption Results for HPC2N Workloads

Figure 10 indicates that the h-DEWOA algorithm produces the minimum best, average, and worst values of the energy consumption for almost every HPC2N workload.

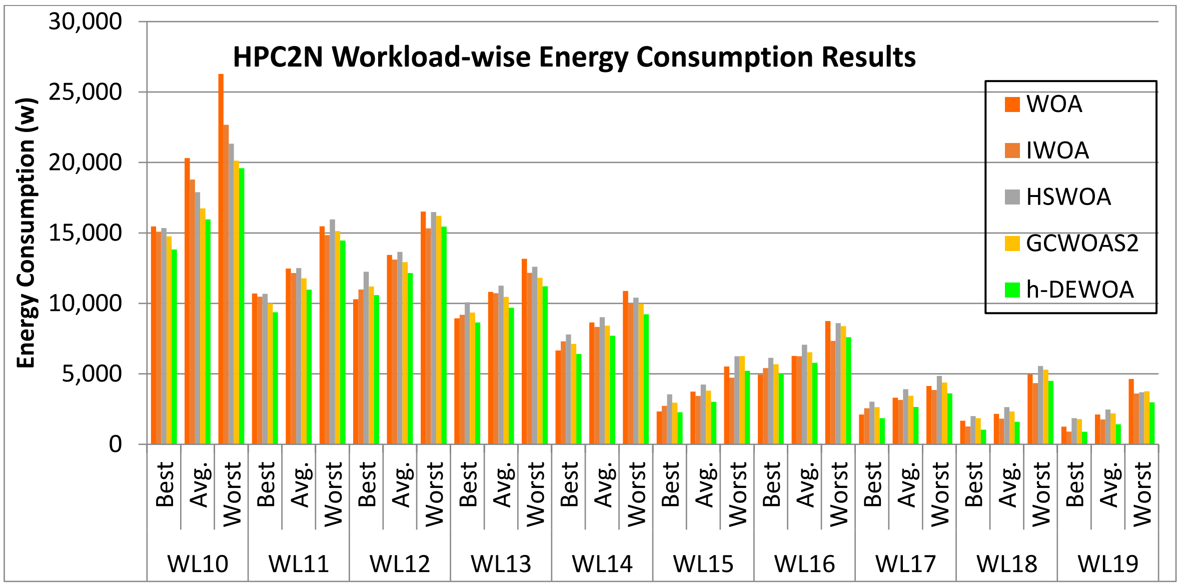


Figure 10. Statistical results of the energy consumption objective for WL10–WL19 workloads.

5.2.5. Overall Collective Results—First Set of Experiments

In the previous section, a comparison of the workload-wise performance of proposed h-DEWOA was presented with respect to WOA-based metaheuristics. This section presents the overall collective results from all workloads together associated with all the scheduling experiments in the form of box plots. Box plots help us to characterize the large number of experimental results in the form of overall minimum, maximum, mean, and median values of makespan and energy consumption.

Figures 11–14 present the box plots of makespan and energy consumption of all WOA-based scheduling approaches for CEA-Curie and HPC2N workloads. Readers are encouraged to refer to [86] to better understand the various characteristics and parameters of box plots. These box plots clearly indicate that the h-DEWOA produces the overall lowest minimum, maximum, median, and mean makespan among all scheduling approaches for both workloads. The performance of IWOA and GCWOAS2 is found to be closer to the h-DEWOA approach.

Summarizing the Overall Results—First Series of Experiments

Finally, the performance of overall experimental results is summarized with the help of mean, median, and standard deviation values of the makespan and energy consumption results associated with all workloads as shown in Table 10. The proposed h-DEWOA method has clearly obtained the lowest values of different performance measures amongst all tested scheduling metaheuristics, which clearly shows the dominance and stability of h-DEWOA over comparative algorithms.

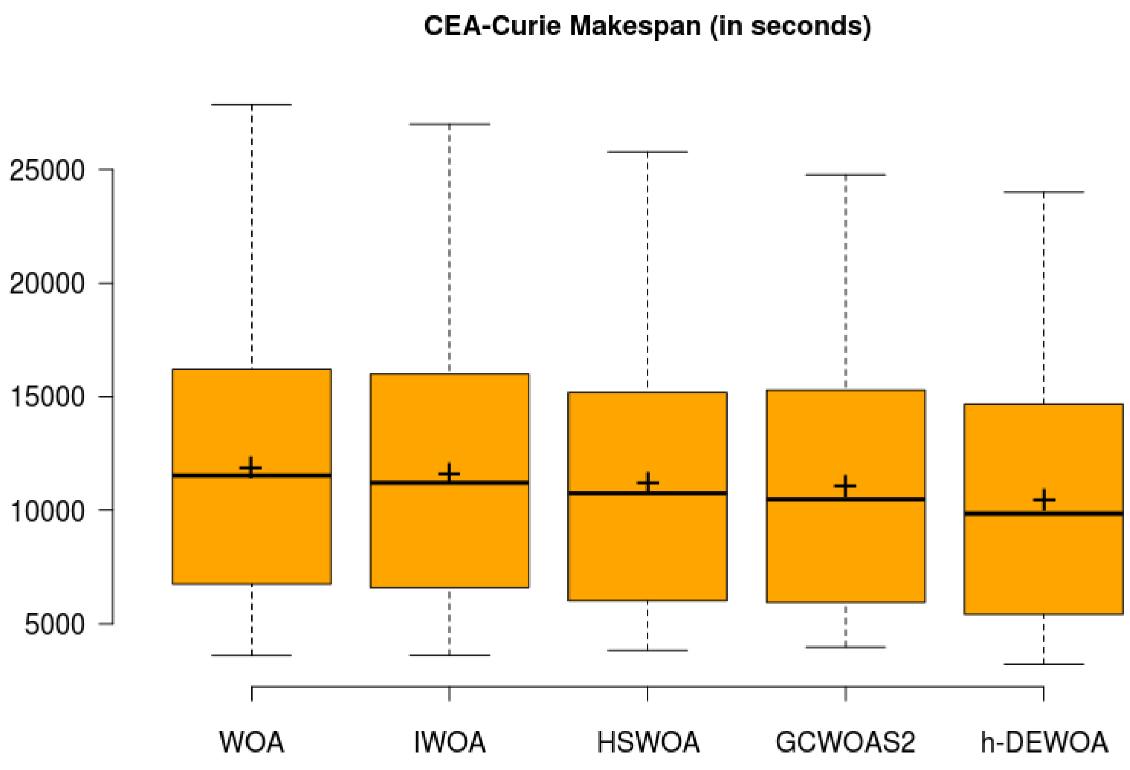


Figure 11. Comparative boxplot analysis of the overall makespan of h-DEWOA- and WOA-based scheduling algorithms for CEA-Curie workloads.

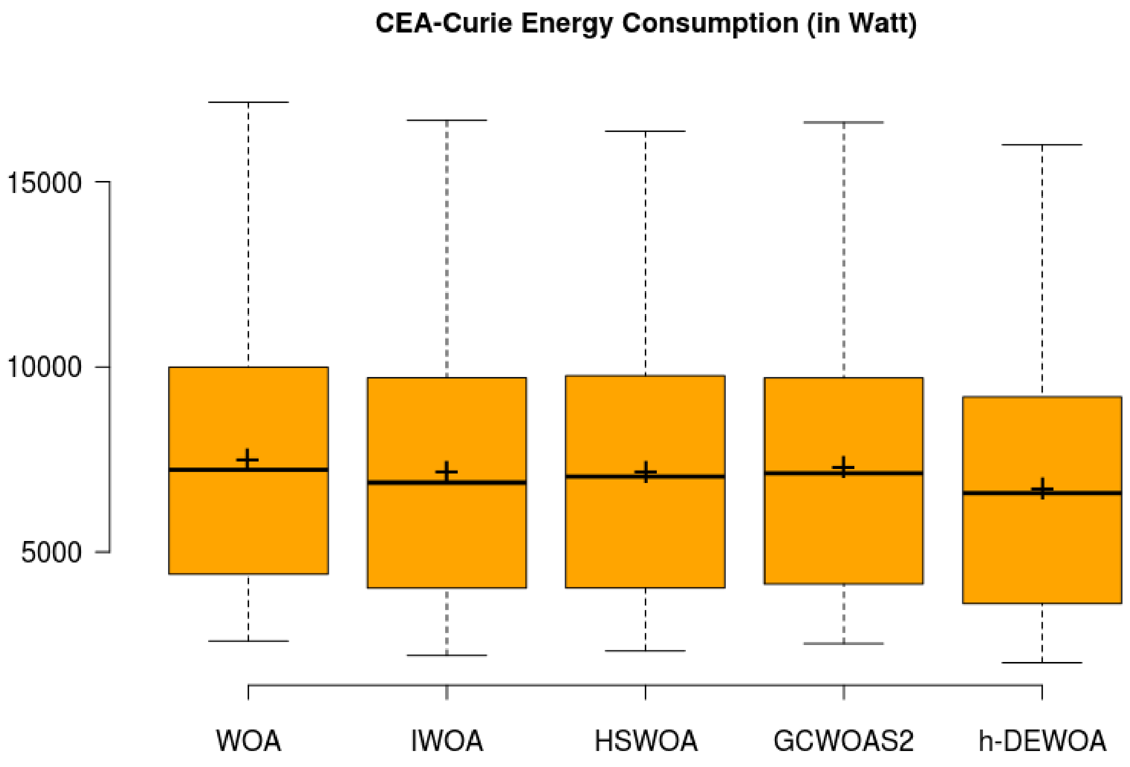


Figure 12. Comparative boxplot analysis of the overall energy consumption of h-DEWOA- and WOA-based scheduling algorithms for CEA-Curie workloads.

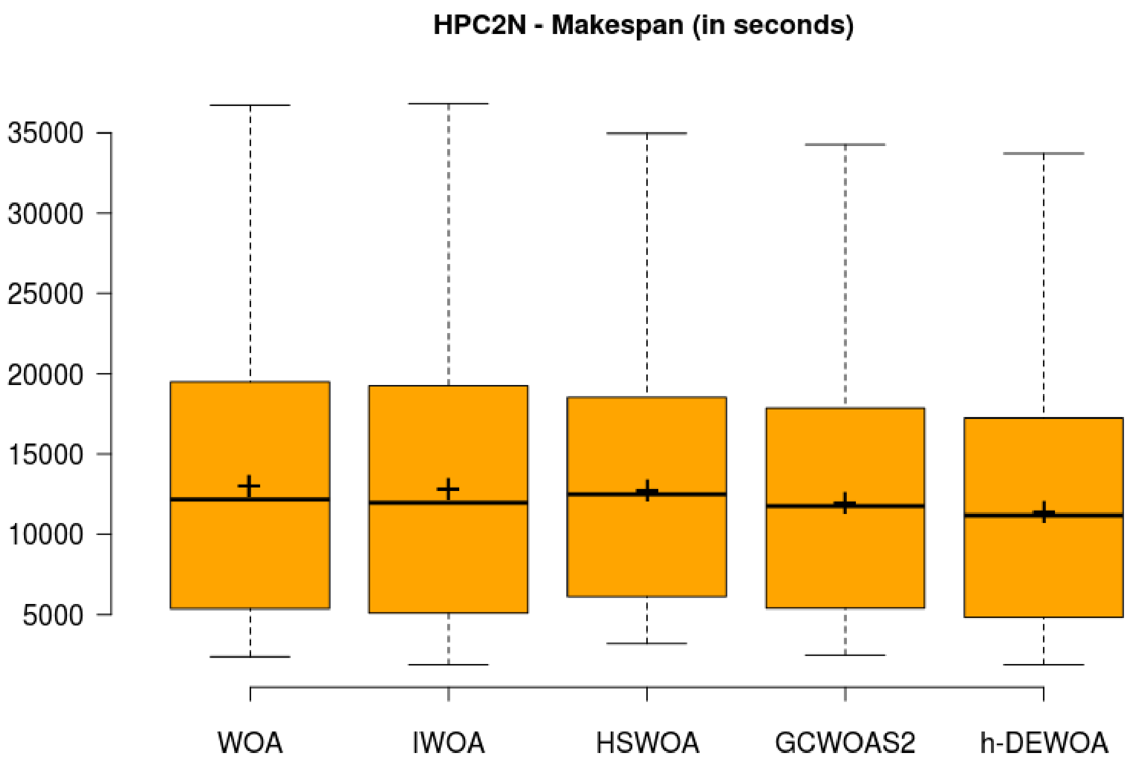


Figure 13. Comparative boxplot analysis of the overall makespan of h-DEWOA- and WOA-based scheduling algorithms for HPC2N workloads.

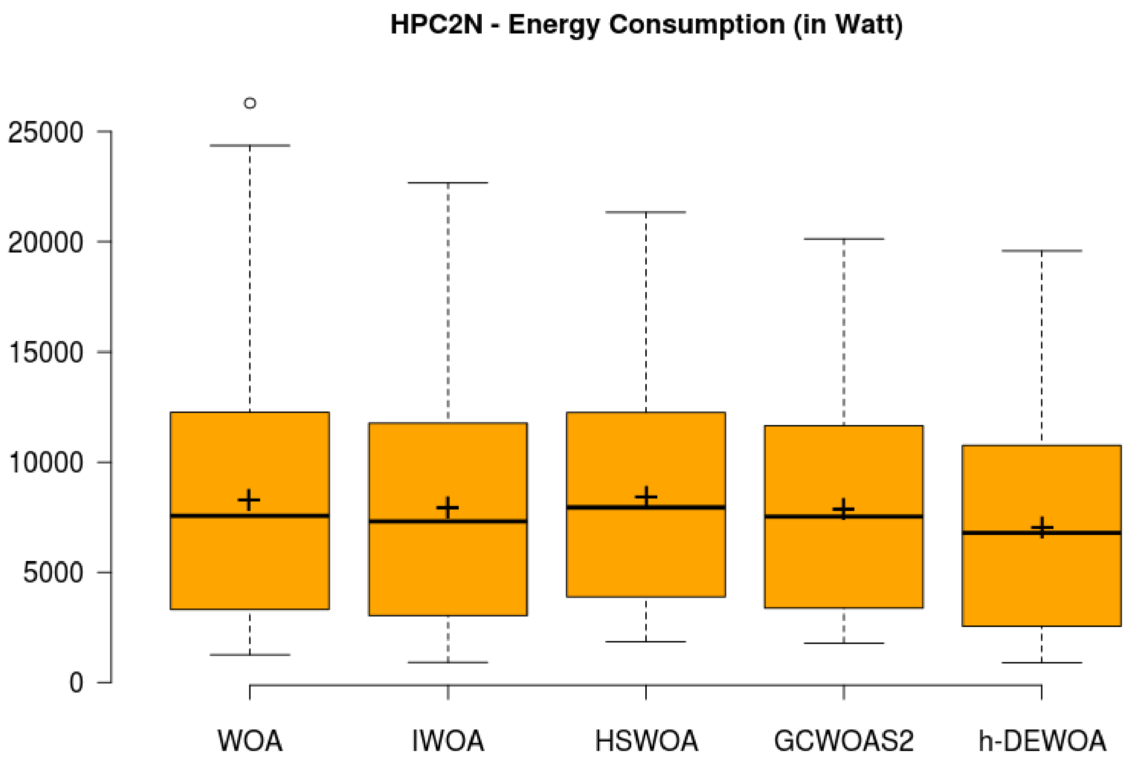


Figure 14. Comparative boxplot analysis of the overall energy consumption of h-DEWOA- and WOA-based scheduling algorithms for HPC2N workloads.

Table 10. Summary of overall results of scheduling algorithms for CEA-Curie and HPC2N workloads for the first set of experiments.

Algorithm		**WOA**	**IWOA**	**HSWOA**	**GCWOAS2**	**h-DEWOA**
CEA-Curie						
Makespan (s)	**Mean**	11,890.95	11,639.63	11,198.02	11,095.44	**10,488.06**
	Median	11,522.28	11,199.14	10,751.37	10,475.4	**9848.7**
	Std. dev.	5773.51	5682.59	5520.11	5463.85	**5453.62**
Energy Consumption (W)	**Mean**	7502.48	7173.63	7175.54	7287.76	**6739.39**
	Median	7218.97	6872.21	7032.31	7122.41	**6588.44**
	Std. dev.	3577.75	3572.11	3528.71	3493.12	**3494.09**
HPC2N						
Makespan (s)	**Mean**	13,030.95	12,846.14	12,755.14	12,026.48	**11,450.91**
	Median	12,165.13	11,958.84	12,494.82	11,748.11	**11,172.98**
	Std. dev.	8089.23	8099.17	6725.83	6726.24	**6722.55**
Energy Consumption (W)	**Mean**	8328.99	7953.16	8467.48	7866.75	**7097.58**
	Median	7568.49	7311.03	7946.03	7533.84	**6779.76**
	Std. dev.	5793.52	5484.77	5083.24	4851.47	**4844.79**

Bold values are better.

In addition to this, performance improvement rate percentage (PIR %) is also calculated to derive the amount of percentage reduction achieved by the h-DEWOA approach over baseline algorithms. PIR % is calculated as follows and results are shown in Table 11.

$$PIR(\%) = \frac{Metric\ (Baseline\ Algorithm) - Metric\ (h - DEWOA)}{Metric\ (h - DEWOA)} \times 100\%$$

Table 11. PIR% of the h-DEWOA over other WOA-based scheduling algorithms.

Policy	**WOA**	**IWOA**	**HSWOA**	**GCWOAS2**
CEA-Curie				
Makespan (s)				
PIR% of h-DEWOA over	+13.38	+10.98	+6.77	+5.79
Energy Consumption (W)				
PIR% of h-DEWOA over	+11.32	+6.44	+6.47	+8.14
HPC2N				
Makespan (s)				
PIR% of h-DEWOA over	+13.80	+12.18	+11.39	+5.03
Energy consumption (W)				
PIR% of h-DEWOA over	+17.35	+12.05	+19.30	+10.84

+ sign shows improved performance of the h-DEWOA approach over compared algorithms.

PIR% results achieved by the h-DEWOA are promising for both CEA-Curie and HPC2N workloads for makespan and energy consumption metrics. For example, for CEA-Curie workloads, the h-DEWOA achieves 5.79, 6.77, 10.98, and 13.38% makespan reduction over WOA, IWOA, HSWOA, and GCWOAS2 algorithms, respectively. Similarly in the case of energy consumption for CEA-Curie workloads, h-DEWOA results in 8.14, 6.47, 6.44, and 11.32% performance improvement over compared algorithms, respectively. For HPC2N workloads, h-DEWOA approach attains 5.03–13.80% makespan improvement and 10.84–19.30% energy consumption improvement over baseline algorithms.

5.3. Comparative Results with State-of-the-Art Algorithms—Second Series of Experiments

5.3.1. Baseline State-of-the-Art Scheduling Algorithms

The theoretical characteristics of all comparative scheduling algorithms and the proposed h-DEWOA approach are presented as per Table 12.

Table 12. Characteristics of tested scheduling algorithms.

Algorithm	Algorithm Type	Algorithm Description	BoT Application Ordering Method	Scheduling Problem Optimization
MOPSO [49]	MH	Aggregation-based Multi-objective PSO	FCFS	AL
MOCS [50]	MH	Aggregation-based Multi-objective CS	FCFS	AL
BLEMO [38]	MH	Pareto front-based Multi-objective GWASF-GA	FCFS	AL
CSDEO [13]	h-MH	Aggregation-based Multi-objective CS + DE	FCFS	AL
BOA [69]	h-MH	Aggregation-based Multi-objective BOA	FCFS	AL
MFO [87]	h-MH	Pareto front-based Multi-objective MFO	FCFS	AL
h-DEWOA (proposed)	h-MH	Aggregation-based Multi-objective WOA + DE	h-DEWOA	AO and AL

MH: Metaheuristic; h-MH: Hybrid Metaheuristic; AL: Allocation; AO: Application ordering.

5.3.2. Convergence Examination and Final Values of Parameters

Convergence investigation is again carried out for the second series of experiments by conducting some pilot experiments. *PopSize* is fixed to 60 for both workloads for all scheduling approaches and the number of generations (*NGen*) is varied from 1 to 100 as shown in Figure 15. Parameters settings of different metaheuristics are taken from original research papers [13,38,49,50,69,87]. Final parameters after parameter tuning are shown in Table 13.

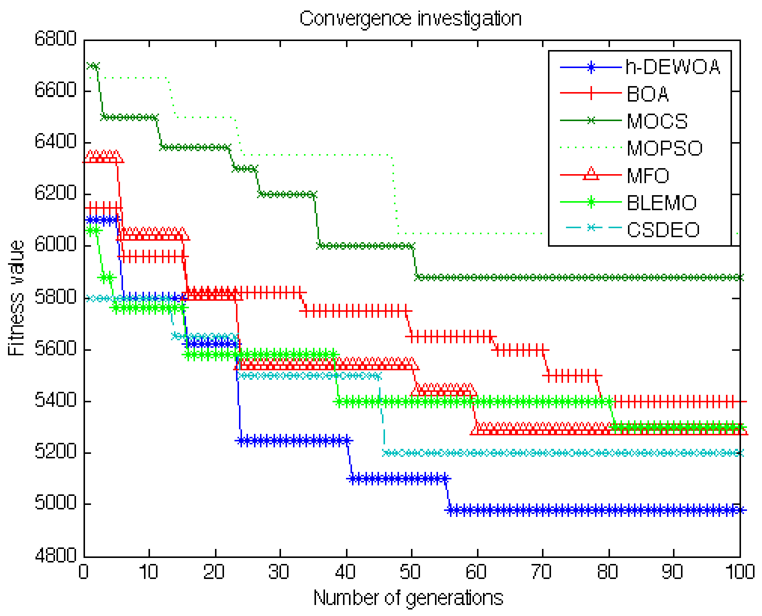


Figure 15. Convergence examination of scheduling algorithms with *PopSize* = 60 for CEA-Curie workload for the second series of experiments.

Table 13. Parameter settings of algorithms for the second series of experiments for 30 independent executions of each algorithm.

Algorithm	Generations	Parameters
MOPSO	60	Initial value of variable a: 2
MOCS	60	WOA parameters: Same as WOA algorithm DE parameters: Scaling factor (F): 0.1–0.9, and Crossover rate (CR): 0.75
BLEMO	40	Crossover rate: 0.33 and Mutation rate: 0.10
CSDEO	50	CS parameters: step size (α_0): 0.01, β: 1.5 DE parameters: Scaling factor (F): 0.1–0.3, and Crossover rate (CR): 0.75
BOA	60	Modular modality c is 0.01, p = 0.8, and power exponent a is increased from 0.1 to 0.3
MFO	60	b = 1
h-DEWOA	50	WOA parameters: Same as WOA algorithm DE parameters: Scaling factor (F): 0.5–0.9, and Crossover rate (CR): 0.75

5.3.3. Performance Evaluation of Second Set of Experiments

The following sub-section presents the best, average, and worst case readings of each scheduling algorithm for 30 independent evaluations for the tested workloads.

Workload-Wise Makespan Results for CEA-Curie Workloads

Table 14 shows that the h-DEWOA algorithm produces the minimum best, average, and worst values for the makespan objective for most CEA-Curie workloads.

Table 14. Makespan statistical results for WL0–WL9 workloads.

CEA-Curie Workloads	Indicator	MOPSO	BLEMO	MOCS	CSDEO	BOA	MFO	h-DEWOA
WL0	Best	11,960.26	**10,200.10**	11,547.45	10,969.92	11,545.10	10,651.75	10,275.9
	Avg.	14,454.11	13,165.61	14,237.23	12,482.34	13,491.24	12,449.01	**11,660.04**
	Worst	21,599.99	20,732.08	19,633.46	15,499.38	16,945.47	15,779.38	**14,435.67**
WL1	Best	14,443.43	**11,700.21**	14,696.73	13,850.81	14,580.85	14,130.81	12,679.97
	Avg.	17,483.91	16,597.74	17,035.45	15,889.18	16,663.68	15,725.85	**15,109.39**
	Worst	25,807.67	24,840.16	22,049.84	18,831.50	20,470.21	19,111.50	**18,477.35**
WL2	Best	8811.59	7200.21	7820.90	7442.08	8247.814	7722.08	**6829.378**
	Avg.	12,080.79	10,413.30	11,095.74	10,186.98	10,931.14	10,466.98	**9560.564**
	Worst	16,524.91	16,350.10	14,946.78	13,173.06	14,463.57	13,453.06	**11,850.62**
WL3	Best	19,848.23	**16,650.21**	19,095.59	18,177.46	18,929.71	18,457.46	17,764.81
	Avg.	24,679.88	22,875.85	22,845.36	21,676.68	22,815.07	21,956.68	**20,896.7**
	Worst	29,426.71	32,400.10	28,146.30	25,609.87	27,323.36	25,889.87	**24,003.9**
WL4	Best	15,174.35	**13,005.17**	15,353.75	14,758.71	15,190.5	15,038.71	13,416.31
	Avg.	18,752.94	17,552.51	17,939.16	17,108.33	17,821.62	17,388.33	16,508.44
	Worst	25,005.48	24,444.68	24,029.27	**19,959.83**	21,840.89	20,239.83	20,937.46
WL5	Best	4517.03	3798.96	4230.69	4203.09	4505.652	4483.09	**3620.954**
	Avg.	6274.81	5483.11	5782.48	5109.55	5704.507	5389.55	**4684.121**
	Worst	9284.97	8535.21	8351.09	6733.47	7316.663	7013.47	**6140.261**
WL6	Best	9058.90	8771.46	9402.54	8843.01	9560.026	9123.01	**8277.555**
	Avg.	12,689.67	12,950.13	12,704.20	11,153.84	12,176.79	11,433.84	**10,371.22**
	Worst	19,732.64	20,400.21	17,142.14	13,390.13	15,173.11	13,670.13	**12,517.62**
WL7	Best	3746.42	**3180.21**	4001.28	3653.73	3915.613	3933.73	3208.096
	Avg.	5614.67	4967.06	5158.19	4451.31	5018.274	4731.31	**4008.443**
	Worst	8717.20	8325.21	8766.96	**5876.96**	6996.048	6156.96	6041.973
WL8	Best	5975.56	5400.10	5644.80	4929.97	5815.429	5209.97	**4308.133**
	Avg.	8785.39	7770.17	7507.07	6093.44	7435.082	6373.44	**5599.787**
	Worst	12,786.17	14,850.21	10,272.95	8509.48	10,293.07	8789.48	**7020**
WL9	Best	5883.46	4500.21	5855.55	4368.56	5833.676	4648.56	**3988.775**
	Avg.	7564.67	7282.67	7594.58	6876.61	7438.893	7156.61	**6481.927**
	Worst	10,204.26	11,250.10	10,011.50	7924.87	8814.705	8204.87	**7672.408**

Bold values are lowest.

Workload-Wise Energy Consumption Results for WL0–WL9 Workloads

Results in Table 15 show that the h-DEWOA approach produces the lowest statistical values of energy consumption for most of the WL0–WL9 workloads. The average metric of h-DEWOA is the minimum among all comparative algorithms for each tested workload.

Workload-Wise Makespan Results for HPC2N Workloads

Statistical data in Table 16 clearly show that the h-DEWOA approach results in the lowest best, average, and worst makespan values over compared algorithms in most of the WL10–WL19 workloads.

Table 15. Energy consumption statistical results for WL0–WL9 workloads.

CEA-Curie Workloads	Indicator	MOPSO	BLEMO	MOCS	CSDEO	BOA	MFO	h-DEWOA
WL0	Best	7206.21	6588.22	7408.80	6529.32	6945.26	6880.32	**6120.27**
	Avg.	8721.13	8229.59	8432.26	7824.27	8340.70	8198.12	**7591.65**
	Worst	11,899.30	12,070.62	11,263.31	**8849.12**	9674.02	9180.12	9289.22
WL1	Best	8633.28	**7783.98**	7719.42	8868.56	8739.48	9229.56	8220.83
	Avg.	11,053.11	10,479.61	10,504.87	9872.62	10,456.31	10,238.58	**9480.57**
	Worst	17,715.73	15,133.85	15,054.43	11,603.77	12,087.60	11,859.77	**11,514.59**
WL2	Best	5582.60	**4733.53**	5414.94	5144.26	5737.73	5393.17	4793.72
	Avg.	7165.61	6524.43	6805.25	6289.01	6608.40	6525.21	**6052.35**
	Worst	8970.75	9882.84	8988.50	7918.24	7997.96	8102.24	8237.94
WL3	Best	13,506.06	**10,812.66**	12,322.39	12,872.81	12,874.75	13,026.76	12,464.88
	Avg.	15,468.06	14,200.27	14,715.03	14,140.03	14,594.40	14,481.83	**13,887.57**
	Worst	17,426.96	19,538.90	17,242.44	16,095.68	16,758.80	16,491.68	**15,988.8**
WL4	Best	9899.06	8569.14	9563.15	8968.66	9601.81	9286.66	**8296.68**
	Avg.	12,001.09	11,099.04	11,016.98	10,326.07	10,935.66	10,611.28	**9870.25**
	Worst	14,625.70	14,141.90	12,556.49	12,301.57	12,509.75	12,427.57	12,316.24
WL5	Best	3007.40	2504.03	2980.97	2410.98	2824.03	2772.98	**2278.04**
	Avg.	3809.02	3398.61	3619.83	3101.46	3511.13	3355.32	**2883.59**
	Worst	4468.13	5128.42	5211.90	3780.75	4333.96	4111.37	**3684.22**
WL6	Best	6815.78	**5812.50**	6796.31	6512.76	6643.86	6696.47	6212.68
	Avg.	7954.14	8046.64	7982.58	7402.84	7774.60	7627.41	**7137.29**
	Worst	9842.82	12,332.43	10,135.10	8123.47	9248.67	8327.51	**7824.83**
WL7	Best	2561.79	2074.51	2702.79	2295.04	2420.52	2422.04	**2006.16**
	Avg.	3797.62	3082.18	3415.45	2791.81	3193.88	3048.35	**2525.13**
	Worst	5173.74	4962.87	4473.37	3714.33	4157.40	4040.33	**3317.04**
WL8	Best	3970.40	3462.00	4076.17	3512.84	3919.93	3698.84	**3113.85**
	Avg.	4905.71	4771.51	4761.41	4187.68	4624.10	4487.93	**3942.42**
	Worst	6194.94	8650.75	6604.29	**4901.30**	5779.50	5084.30	4989.44
WL9	Best	2868.46	2937.93	3519.60	2891.83	3458.43	3106.15	**2561.73**
	Avg.	4849.14	4481.61	4817.17	4217.99	4651.67	4573.47	**4023.08**
	Worst	7715.02	6644.12	6823.52	**5178.55**	5642.32	5375.55	5496.83

Bold values are lowest.

Table 16. Makespan statistical results for WL10–WL19 workloads.

HPC2N Workloads	Indicator	MOPSO	BLEMO	MOCS	CSDEO	BOA	MFO	h-DEWOA
WL10	Best	22,935.48	21,000.10	22,450.11	20,036.63	22,066.17	20,136.63	**18,801.74**
	Avg.	31,659.31	27,397.33	27,258.24	22,258.71	26,959.94	22,358.71	**21,579.70**
	Worst	47,578.36	41,259.62	37,070.36	35,291.42	37,109.26	35,391.42	**33,700.98**
WL11	Best	18,980.14	**14,644.01**	16,359.12	16,522.34	15,919.99	16,622.34	14,953.21
	Avg.	22,903.96	19,552.95	20,227.04	18,941.76	19,948.61	19,041.76	**17,567.38**
	Worst	28,885.86	28,848.91	27,708.46	22,696.08	27,174.21	22,796.08	**22,324.56**
WL12	Best	20,212.26	16,650.10	19,382.40	17,810.82	19,164.36	17,910.82	**16,649.19**
	Avg.	25,260.03	22,961.31	22,812.50	20,308.97	22,572.30	20,408.97	**19,329.17**
	Worst	32,357.04	33,907.60	30,560.06	29,312.09	30,301.89	29,412.09	**26,644.20**
WL13	Best	16,746.90	13,522.67	14,416.59	14,515.07	13,719.31	14,615.07	**13,366.98**
	Avg.	19,585.00	17,200.76	17,893.86	17,188.48	17,648.52	17,288.48	**16,207.58**
	Worst	22,251.07	22,965.20	21,316.48	20,332.90	21,513.13	20,432.90	**20,203.84**
WL14	Best	11,622.14	10,650.25	11,106.19	11,616.94	10,476.61	11,716.94	**99,72.00**
	Avg.	13,851.12	13,953.94	13,868.76	13,067.45	13,654.34	13,167.45	**12,274.21**
	Worst	18,141.63	18,150.31	16,804.36	14,915.63	16,400.40	15,015.63	**14,380.36**
WL15	Best	4524.74	4200.10	4212.11	4017.70	3922.51	4117.70	**3704.70**
	Avg.	7052.93	5930.53	6206.88	5576.38	5957.67	5676.38	**5320.46**
	Worst	12,901.66	11,625.10	9690.71	**7468.60**	9005.34	7568.60	7980.15

Table 16. *Cont.*

HPC2N Workloads	Indicator	MOPSO	BLEMO	MOCS	CSDEO	BOA	MFO	h-DEWOA
WL16	Best	9200.91	8287.76	8751.31	7828.44	8218.58	7928.44	**7581.52**
	Avg.	11,268.08	10,553.76	10,893.29	10,110.14	10,626.45	10,210.14	**9821.84**
	Worst	15,773.25	15,232.71	14,651.73	11,991.02	14,576.44	**12,091.02**	12,118.86
WL17	Best	4738.35	3825.10	4012.78	3778.44	4087.09	3878.44	**3338.51**
	Avg.	6559.59	5910.08	6007.81	5380.60	5747.06	5480.60	**5137.35**
	Worst	9001.94	9450.17	9569.37	7980.22	9081.96	8080.22	**7803.77**
WL18	Best	3047.82	3600.10	3347.34	3006.86	**2760.23**	3206.86	2969.94
	Avg.	4386.93	4738.25	4507.69	3964.54	4284.68	4064.54	**3834.73**
	Worst	6202.69	21,937.59	12,418.78	7878.97	12,194.68	7978.97	**5808.41**
WL19	Best	2804.61	2250.10	2787.46	2130.94	2175.34	2230.94	**1875.20**
	Avg.	4179.46	4657.41	4367.47	3697.48	4061.85	3797.48	**3436.67**
	Worst	7970.18	20,970.09	12,303.99	7232.97	12,209.61	7332.97	**5548.10**

Bold values are lowest.

Workload-Wise Energy Consumption Results for HPC2N Workloads

As can been seen from Table 17, the h-DEWOA obtains significantly better best, average, and worst statistical values than the well-known baseline scheduling algorithms for almost each workload.

Table 17. Energy consumption statistical results for WL10–WL19 workloads.

HPC2N Workloads	Indicator	MOPSO	BLEMO	MOCS	CSDEO	BOA	MFO	h-DEWOA
WL10	Best	17,723.35	**13,584.58**	15,982.21	14,511.92	15,342.49	14,711.92	13,823.81
	Avg.	23,907.99	16,937.46	20,533.79	16,970.77	19,040.17	17,170.77	**15,966.82**
	Worst	31,626.77	23,420.63	26,343.40	20,251.14	22,918.38	20,451.14	**19,593.61**
WL11	Best	11,258.19	9776.93	11,048.08	9855.60	10,731.61	10,055.60	**9369.74**
	Avg.	13,991.44	12,258.83	12,694.22	11,550.63	12,402.66	11,750.63	**10,978.52**
	Worst	18,560.88	16,909.84	15,501.51	14,855.03	15,098.86	15,055.03	**14,459.70**
WL12	Best	11,558.36	10,789.13	10,670.37	11,086.86	11,234.09	11,286.86	**10,588.87**
	Avg.	14,067.10	13,704.00	13,670.77	12,698.74	13,364.66	12,898.74	**12,161.39**
	Worst	17,854.48	17,730.69	17,097.92	15,458.78	15,571.12	15,658.78	**15,449.40**
WL13	Best	10,186.48	8964.51	9135.88	9024.07	9444.43	9224.07	**8644.88**
	Avg.	12,652.43	10,894.61	10,985.65	10,256.97	10,979.42	10,456.97	**9687.76**
	Worst	15,289.81	13,618.89	13,268.02	11,479.04	12,416.42	11,679.04	**11,209.28**
WL14	Best	7506.78	6926.25	7082.51	6774.34	7555.62	6974.34	**6411.12**
	Avg.	9044.92	8819.79	8892.65	8085.65	8577.42	8285.65	**7700.74**
	Worst	11,732.53	11,121.16	11,114.54	9365.08	10,243.36	9565.08	**9227.94**
WL15	Best	3340.41	2854.22	2669.03	2645.06	2985.37	2845.06	**2272.91**
	Avg.	4210.07	3838.32	4041.86	3340.14	3679.90	3540.14	**3009.24**
	Worst	5197.82	7066.33	5700.02	5205.52	4984.47	5405.52	**5219.22**
WL16	Best	5576.80	5418.26	5207.54	5382.38	5656.58	5582.38	**5019.96**
	Avg.	7145.42	6688.17	6579.13	6128.16	6503.26	6328.16	**5790.69**
	Worst	10,248.08	9209.20	9065.99	7711.92	7590.18	7911.92	**7603.42**
WL17	Best	2984.80	2586.33	2497.43	2227.06	2814.00	2427.06	**1852.30**
	Avg.	3888.74	3477.84	3652.42	2990.15	3399.06	3190.15	**2654.09**
	Worst	5374.67	5140.45	4522.45	3810.12	4106.29	4010.12	**3612.28**
WL18	Best	1767.25	1776.39	1967.11	1222.13	1510.62	1422.13	**1036.32**
	Avg.	2517.81	2183.11	2445.86	1714.90	2070.95	1914.90	**1596.70**
	Worst	6170.04	7264.35	5009.49	4565.55	4590.59	4765.55	**4500.49**
WL19	Best	1518.28	1278.92	1495.37	938.99	1155.77	1138.99	**888.67**
	Avg.	2644.26	2079.39	2427.79	1501.94	2014.08	1701.94	**1429.86**
	Worst	5604.93	6403.11	4905.22	**2568.03**	3852.78	2768.03	2978.92

Bold values are lowest.

5.3.4. Overall Collective Results—Second Series of Experiments

Further, a box plot tool is utilized to explain the makespan and energy consumption results of the entire second series of scheduling experiments. It is evident from Figures 16–19 that the h-DEWOA method has shown significant improvements, represented by box plot parameters such as minimum, maximum, mean, and median for both makespan and energy consumption objectives when compared to recent non-WOA scheduling metaheuristics. MFO, CSDEO, and BLEMO algorithms have also performed competitively well. Makespan dispersion and standard deviation of the h-DEWOA algorithm in all box plots is also found to be least among all scheduling algorithms, indicating its strength, dominance, and stability.

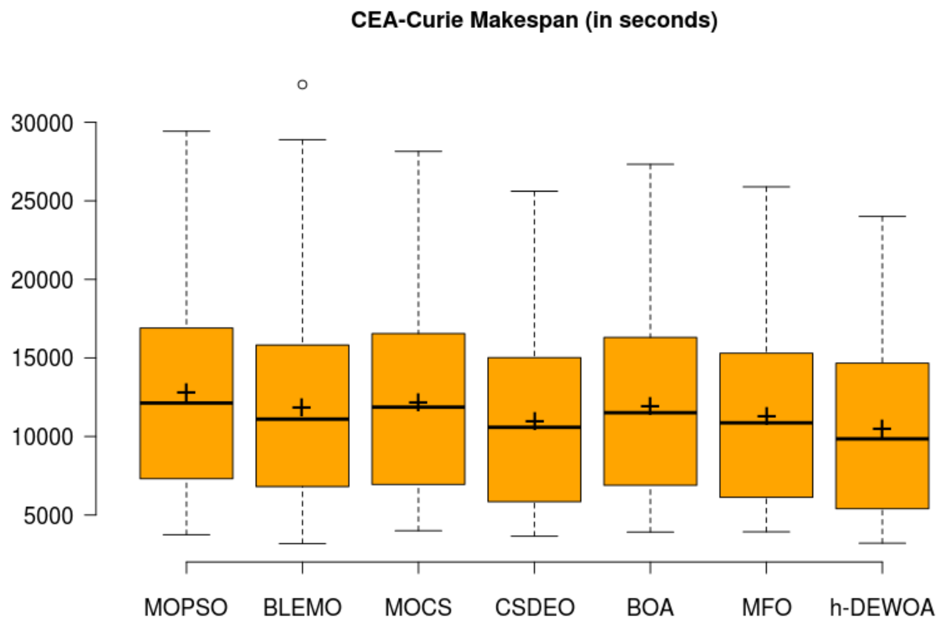


Figure 16. Comparative boxplot analysis of the overall makespan of h-DEWOA and recent state-of-the-art algorithms for CEA-Curie workloads.

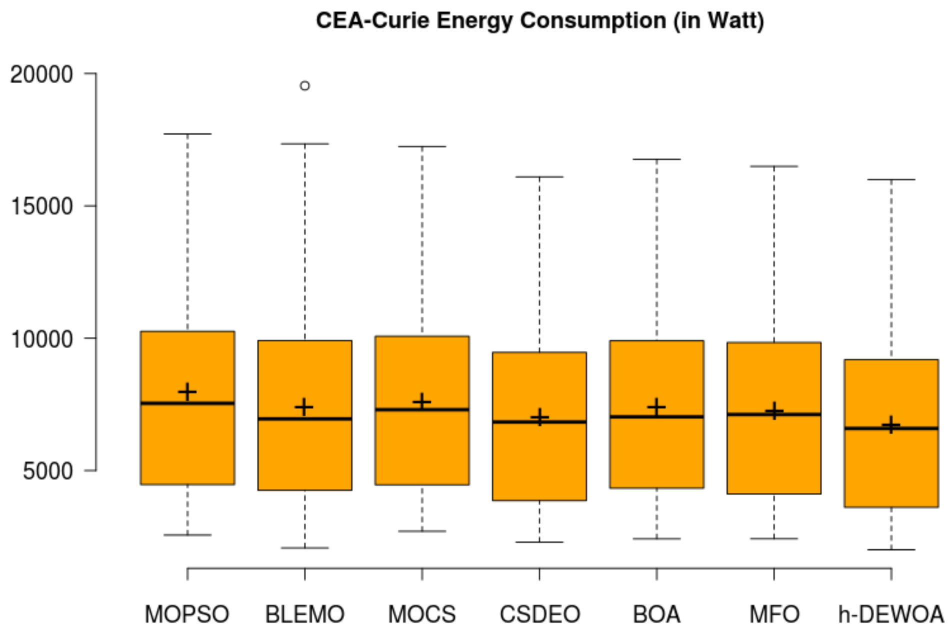


Figure 17. Comparative boxplot analysis of the overall energy consumption of h-DEWOA and recent state-of-the-art algorithms for CEA-Curie workloads.

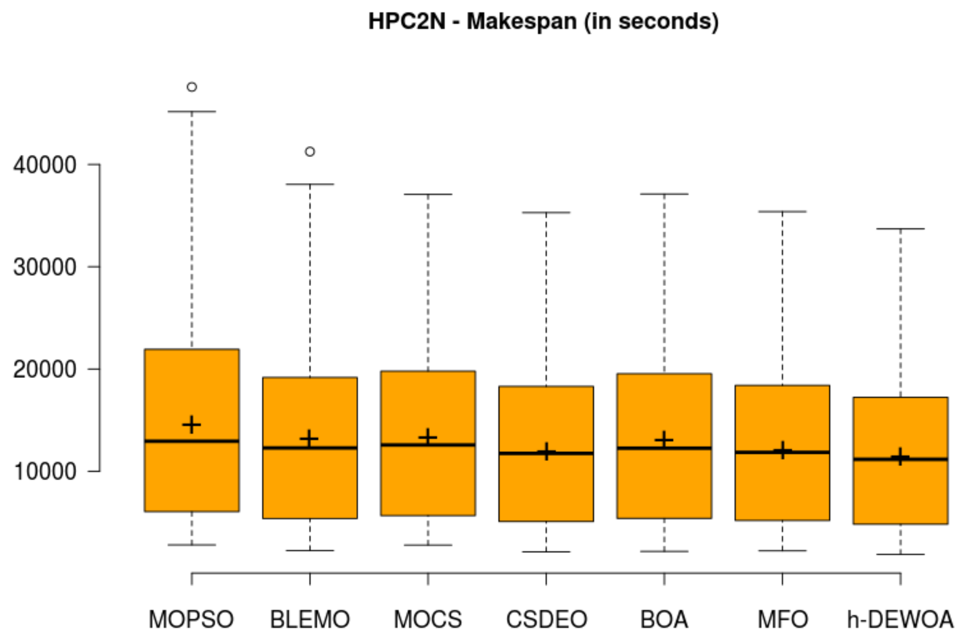


Figure 18. Comparative boxplot analysis of the overall makespan of h-DEWOA and recent state-of-the-art algorithms for HPC2N workloads.

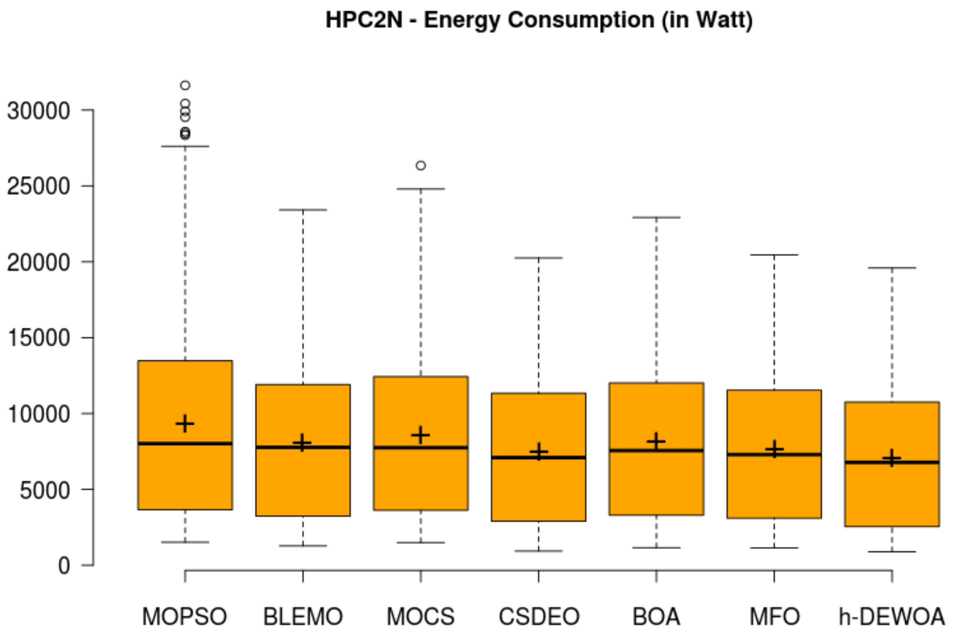


Figure 19. Comparative boxplot of the overall energy consumption of h-DEWOA and recent state-of-the-art algorithms for HPC2N workloads.

Summarizing the Overall Results—Second Series of Experiments

Finally, Table 18 shows the summary of performance of overall experimental results of the second series of experiments in terms of mean, median, and standard deviation values of the makespan and energy consumption objectives. It is evident that the h-DEWOA achieved better overall results as compared to other classical scheduling metaheuristics.

Table 18. Summary of overall results of scheduling algorithms for CEA-Curie and HPC2N workloads for the second set of experiments.

Algorithm		**MOPSO**	**BLEMO**	**MOCS**	**CSDEO**	**BOA**	**MFO**	**h-DEWOA**
CEA-Curie								
Makespan (s)	**Mean**	12,838.08	11,905.82	12,189.95	11,027.16	11,949.63	11,307.16	**10,488.06**
	Median	12,130.13	11,103.94	11,867.99	10,586.13	11,509.14	10,866.13	**9848.7**
	Std. dev.	6223.92	6034.79	5782.63	5519.54	5682.59	5519.54	**5453.62**
Energy Cons. (W)	**Mean**	7972.46	7431.35	7602.63	7015.38	7418.61	7265.13	**6739.39**
	Median	7539.61	6951.81	7298.25	6831.89	7028.03	7117.54	**6588.44**
	Std. dev.	3873.96	3714.71	3571.39	3521.47	3569.56	3529.74	**3494.09**
HPC2N								
Makespan (s)	**Mean**	14,670.64	13,285.63	13,404.35	12,049.45	13,146.14	12,149.45	**11,450.91**
	Median	12,945.09	12,281.47	12,576	11,751.78	12,258.84	11,851.78	**11,172.98**
	Std. dev.	9569.72	8476.04	8098.13	7046.38	8099.17	7046.38	**6722.55**
Energy Cons. (W)	**Mean**	9407.02	8088.15	8592.42	7523.8	8203.16	7723.8	**7097.58**
	Median	8019.48	7772.85	7749.99	7097.58	7561.03	7297.58	**6779.76**
	Std. dev.	6717.04	5200.80	5752.87	5071.34	5484.77	5071.34	**4844.79**

Bold values are better.

Percentage reduction, i.e., PIR% achieved by the h-DEWOA approach over baseline algorithms for the second series of experiments is shown in Table 19.

Table 19. PIR% of the h-DEWOA over comparative algorithms for the second set of experiments.

Policy	**MOPSO**	**MOCS**	**BLEMO**	**IWOA**	**CSDEO**	**BOA**	**MFO**
CEA-Curie							
Makespan (s)							
PIR% of h-DEWOA over	+22.41	+13.52	+16.23	+10.98	+5.14	+13.94	+7.81
Energy Consumption (W)							
PIR% of h-DEWOA over	+18.30	+10.27	+12.81	+6.44	+4.10	+10.08	+7.80
HPC2N							
Makespan (s)							
PIR% of h-DEWOA over	+28.12	+16.02	+17.06	+12.18	+5.23	+14.80	+6.10
Energy consumption (W)							
PIR% of h-DEWOA over	+32.54	+13.96	+21.06	+12.05	+6.01	+15.58	+8.82

+ sign shows improved performance of the h-DEWOA approach over compared algorithms.

Table 19 shows that the proposed h-DEWOA algorithm produces significant reduction in both makespan and energy consumption objectives for CEA-Curie and HPC2N workloads over compared scheduling approaches. For CEA-Curie workloads, makespan and energy consumption reduction in h-DEWOA are in the range of 5.14–22.41% and 4.10–18.30%, respectively. In the case of HPC2N workloads, h-DEWOA achieved 5.23–28.12% makespan reduction and 6.01–32.54% energy consumption reduction over the recent scheduling metaheuristics.

In a nutshell, the h-DEWOA algorithm showed extraordinary performance for both makespan and energy consumption objectives for CEA-Curie and HPC2N workloads as compared to WOA-based metaheuristics and other state-of-the-art scheduling techniques. In addition to this, the run-time to simulate the proposed hybrid h-DEWOA approach is also in acceptable limits when compared to baseline algorithms as shown in Tables 20 and 21.

Table 20. Mean simulation time of scheduling algorithms for the first series of experiments.

Policy	**WOA**	**IWOA**	**HSWOA**	**GCWOAS2**	**h-DEWOA**
CEA-Curie					
Simulation Time (s)	17.64	15.98	17.06	16.76	**14.87**
HPC2N					
Simulation Time (s)	28.98	27.01	28.89	28.19	**15.59**

Bold values are better.

Table 21. Mean simulation time of scheduling algorithms for the second series of experiments.

Policy	MOPSO	MOCS	BLEMO	IWOA	CSDEO	BOA	MFO	h-DEWOA
CEA-Curie								
Simulation Time (s)	**10.29**	14.35	15.75	15.98	11.63	16.89	16.02	14.87
HPC2N								
Simulation Time (s)	**14.24**	21.58	26.32	27.01	15.97	25.31	18.91	15.59

Bold values are better.

5.4. Main Reasons for Excellent Performance of h-DEWOA

Main reasons for excellent scheduling performance of the proposed h-DEWOA approach are the use of chaotic maps and OBL strategies in generating optimal initial population, use of the OBL-enabled DE method to carry out global search, employing excellent exploitation properties of the standard WOA, fusion of fitness-based exploration–exploitation tradeoff scheme, and application of the efficient *MDeRA* heuristic for mapping of resources to BoT applications. Moreover, the h-DEWOA approach optimizes both BoT application ordering and resource allocation sub-problems of the CBS problem, whereas most existing scheduling heuristics focus on only optimizing resource allocation sub-problems and ignore optimizing scheduling order problems.

6. Conclusions

In the last decade, tremendous escalation in the energy consumption of cloud computing systems has been observed due to unprecedented increase in the cloud-user base. Consequently, CSPs always hunt to incorporate energy saving solutions to cut-down their hefty energy bills and reduce carbon emissions. Alongside this trend, the priority of cloud users is to execute their applications as fast as possible with shorter makespan. To address these issues, this paper proposed a hybrid metaheuristic h-DEWOA to schedule BoT applications over IaaS clouds to optimize both makespan and energy consumption. The h-DEWOA approach eliminates the shortcomings of standard WOA by combining the advantages of WOA, DE, OBL, and chaotic maps, finally producing efficient scheduling solutions. Additionally, an effective fitness-enabled balancing mechanism and an efficient resource allocation heuristic are also incorporated in the h-DEWOA approach to ensure sufficient exploration–exploitation tradeoff and optimal resource allocation, respectively, which further strengthens the resource allocation and BoT application scheduling performance. Each scheduling algorithm is evaluated on the CloudSim simulator using supercomputing workloads. In comparison with WOA-based approaches, the h-DEWOA approach achieved 5.79−13.38% makespan improvement and 8.14−11.32% energy saving for CEA-Curie workloads, and 5.03−13.80% makespan improvement and 10.84−17.35% energy consumption improvement for HPC2N workloads. Similar substantial performance improvement of h-DEWOA has also been observed against MOPSO, BELMO, MOCS, CSDEO, BOA, and MFO metaheuristics for both CEA-Curie and HPC2N workloads. In addition to this, the convergence rate of the proposed h-DEWOA approach is also found to be reasonably good. In the future, the h-DEWOA algorithm will be evaluated against other well-known scheduling algorithms with different objective functions and with different types of workloads as well.

Author Contributions: Conceptualization, A.C.; Data curation, A.C.; Formal analysis, A.C.; Funding acquisition, N.S.S. and H.A.O.; Investigation, A.C.; Methodology, A.C.; Project administration, A.C.; Resources, A.C.; Software, A.C.; Supervision, A.C.; Validation, A.M., S.K.S., N.S.S. and H.A.O.; Visualization, A.C.; Writing—original draft, A.C.; Writing—review & editing, S.K.S., N.S.S. and H.A.O. All authors have read and agreed to the published version of the manuscript.

Funding: The APC of this research paper is funded by Universiti Kebangsaan Malaysia (UKM) and Ministry of Education (MOE), Malaysia under the grant number FRGS/1/2018/ICT02/UKM/02/6.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data used and produced is available in the article.

Acknowledgments: The authors would like to thank Universiti Kebangsaan Malaysia (UKM) and Ministry of Education (MOE), Malaysia under the grant FRGS/1/2018/ICT02/UKM/02/6 for funding this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Moghaddam, S.K.; Buyya, R.; Ramamohanarao, K. Performance-Aware Management of Cloud Resources: A Taxonomy and Future Directions. *ACM Comput. Surv.* **2019**, *52*, 1–37. [CrossRef]
2. Netto, M.A.S.; Calheiros, R.N.; Rodrigues, E.R.; Cunha, R.L.F.; Buyya, R. HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges. *ACM Comput. Surv.* **2018**, *51*, 1–29. [CrossRef]
3. Amazon EC2 Instance Types-Amazon Web Services. 2022. Available online: https://aws.amazon.com/ec2/instance-types/ (accessed on 6 March 2022).
4. Ilager, S.; Ramamohanarao, K.; Buyya, R. ETAS: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation. *Concurr. Computat. Pract. Exper.* **2019**, *31*, e5221. [CrossRef]
5. Khattar, N.; Sidhu, J.; Singh, J. Toward energy-efficient cloud computing: A survey of dynamic power management and heuristics-based optimization techniques. *J. Supercomput.* **2019**, *75*, 4750–4810. [CrossRef]
6. Materwala, H.; Ismail, L. Performance and energy-aware bi-objective tasks scheduling for cloud data centers. *Procedia Comput. Sci.* **2022**, *197*, 238–246. [CrossRef]
7. Brochard, L.; Kamath, V.; Corbalán, J.; Holland, S.; Mittelbach, W.; Ott, M. *Energy-Efficient Computing and Data Centers*; Wiley: Hoboken, NJ, USA, 2019.
8. Chhabra, A.; Huang, K.-C.; Bacanin, N.; Rashid, T.A. Optimizing Bag-of-Tasks scheduling on cloud data centers using hybrid swarm-intelligence meta-heuristic. *J. Supercomput.* **2022**, *78*, 9121–9183. [CrossRef]
9. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Computat.* **1997**, *1*, 67–82. [CrossRef]
10. Mohamed, A.W.; Hadi, A.A.; Mohamed, A.K. Gaining-sharing knowledge based algorithm for solving optimization problems: A novel nature-inspired algorithm. *Int. J. Mach. Learn. Cyber.* **2019**, *11*, 1501–1529. [CrossRef]
11. Madni, S.H.H.; Abd Latiff, M.S.; Abdullahi, M.; Abdulhamid, S.M.; Usman, M.J. Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. *PLoS ONE* **2017**, *12*, e0176321. [CrossRef]
12. Sukhoroslov, O.; Nazarenko, A.; Aleksandrov, R. An experimental study of scheduling algorithms for many-task applications. *J. Supercomput.* **2019**, *75*, 7857–7871. [CrossRef]
13. Chhabra, A.; Singh, G.; Kahlon, K.S. Multi-criteria HPC task scheduling on IaaS cloud infrastructures using meta-heuristics. *Clust. Comput.* **2021**, *24*, 885–918. [CrossRef]
14. Kumar, M.; Sharma, S.C.; Goel, A.; Singh, S.P. A comprehensive survey for scheduling techniques in cloud computing. *J. Netw. Comput. Appl.* **2019**, *143*, 1–33. [CrossRef]
15. Amini Motlagh, A.; Movaghar, A.; Rahmani, A.M. Task scheduling mechanisms in cloud computing: A systematic review. *Int J. Commun. Syst.* **2020**, *33*, e4302. [CrossRef]
16. Mostafa Bozorgi, S.; Yazdani, S. IWOA: An improved whale optimization algorithm for optimization problems. *J. Comput. Des. Eng.* **2019**, *6*, 243–259. [CrossRef]
17. Abdel-Basset, M.; Abdle-Fatah, L.; Sangaiah, A.K. An improved Lévy based whale optimization algorithm for bandwidth-efficient virtual machine placement in cloud computing environment. *Clust. Comput.* **2019**, *22*, 8319–8334. [CrossRef]
18. Luan, F.; Cai, Z.; Wu, S.; Jiang, T.; Li, F.; Yang, J. Improved Whale Algorithm for Solving the Flexible Job Shop Scheduling Problem. *Mathematics* **2019**, *7*, 384. [CrossRef]
19. Kaur, G.; Arora, S. Chaotic whale optimization algorithm. *J. Comput. Des. Eng.* **2018**, *5*, 275–284. [CrossRef]
20. Mohammed, H.M.; Umar, S.U.; Rashid, T.A. A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm. *Comput. Intell. Neurosci.* **2019**, *2019*, 8718571. [CrossRef]
21. Lee, K.-C.; Lu, P.-T. Application of Whale Optimization Algorithm to Inverse Scattering of an Imperfect Conductor with Corners. *Int. J. Antennas Propag.* **2020**, *2020*, 8205797. [CrossRef]
22. Ni, L.; Sun, X.; Li, X.; Zhang, J. GCWOAS2: Multiobjective task scheduling strategy based on gaussian cloud-whale optimization in cloud computing. *Comput. Intell. Neurosci.* **2021**, *2021*, 5546758. [CrossRef]
23. Movahedi, Z.; Defude, B.; Mohammad, H.A. An efficient population-based multi-objective task scheduling approach in fog computing systems. *J. Cloud Comput.* **2021**, *10*, 53. [CrossRef]
24. Manikandan, N.; Gopalakrishnan, N.; Pradeep, K. Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment. *Comput. Commun.* **2022**, *187*, 35–44. [CrossRef]
25. Chen, X.; Cheng, L.; Liu, C.; Liu, Q.; Liu, J.; Mao, Y.; Murphy, J. A WOA-based optimization approach for task scheduling in cloud computing systems. *IEEE Syst. J.* **2020**, *14*, 3117–3128. [CrossRef]

26. Jia, L.; Li, K.; Shi, X. Cloud computing task scheduling model based on improved whale optimization algorithm. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 4888154. [CrossRef]
27. Bezdan, T.; Zivkovic, M.; Bacanin, N.; Strumberger, I.; Tuba, E.; Tuba, M. Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. *IFS* **2021**, *42*, 411–423. [CrossRef]
28. Muhammad, A.; Abdullah, S.; Samsiah Sani, N. Optimization of Sentiment Analysis Using Teaching-Learning Based Algorithm. *Comput. Mater. Contin.* **2021**, *69*, 1783–1799. [CrossRef]
29. Aldulaimi, M.H.; Zainudin, S.; Bakar, A.A. An improved method to enhance protein structural class prediction using their secondary structure sequences and genetic algorithm. *Int. J. Bioinform. Res. Appl.* **2018**, *14*, 376–400. [CrossRef]
30. Shreem, S.S.; Ahmad Nazri, M.Z.; Abdullah, S.; Sani, N.S. Hybrid Symmetrical Uncertainty and Reference Set Harmony Search Algorithm for Gene Selection Problem. *Mathematics* **2022**, *10*, 374. [CrossRef]
31. Buang, N.; Hanawi, S.A.; Mohamed, H.; Jenal, R. B-Spline Curve Modelling Based on Nature Inspired Algorithms. *APJITM* **2016**, *5*. [CrossRef]
32. Alathamneh, G.M.; Abdullah, S.; Sani, N.S. Genetic Algorithm Selection Strategies based Rough Set for Attribute Reduction. *IJCSNS* **2019**, *19*, 187–194.
33. Albert, P.; Nanjappan, M. WHOA: Hybrid based task scheduling in cloud computing environment. *Wirel. Pers. Commun.* **2021**, *121*, 2327–2345. [CrossRef]
34. Sharma, M.; Garg, R. Energy-aware whale-optimized task scheduler in cloud computing. In Proceedings of the 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 7–8 December 2017; pp. 121–126.
35. Sreenu, K.; Sreelatha, M. W-Scheduler: Whale optimization for task scheduling in cloud computing. *Cluster Comput.* **2019**, *22*, 1087–1098. [CrossRef]
36. Rekha, P.M.; Dakshayini, M. Efficient task allocation approach using genetic algorithm for cloud environment. *Cluster Comput.* **2019**, *22*, 1241–1251. [CrossRef]
37. Sun, Y.; Li, J.; Fu, X.; Wang, H.; Li, H. Application research based on improved genetic algorithm in cloud task scheduling. *J. Intell. Fuzzy Syst.* **2020**, *38*, 239–246. [CrossRef]
38. Vila, S.; Guirado, F.; Lerida, J.L.; Cores, F. Energy-saving scheduling on IaaS HPC cloud environments based on a multi-objective genetic algorithm. *J. Supercomput.* **2019**, *75*, 1483–1495. [CrossRef]
39. Abdullahi, M.; Ngadi, M.A.; Abdulhamid, S.M. Symbiotic Organism Search optimization based task scheduling in cloud computing environment. *Future Gener. Comput. Syst.* **2016**, *56*, 640–650. [CrossRef]
40. Abdullahi, M.; Ngadi, M.A.; Dishing, S.I. Chaotic symbiotic organisms search for task scheduling optimization on cloud computing environment. In Proceedings of the 2017 6th ICT International Student Project Conference (ICT-ISPC), Johor, Malaysia, 23–24 May 2017; pp. 1–4.
41. Li, G.; Wu, Z. Ant Colony Optimization Task Scheduling Algorithm for SWIM Based on Load Balancing. *Future Internet* **2019**, *11*, 90. [CrossRef]
42. Zuo, L.; Shu, L.; Dong, S.; Zhu, C.; Hara, T. A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing. *IEEE Access* **2015**, *3*, 2687–2699. [CrossRef]
43. Huang, X.; Li, C.; Chen, H.; An, D. Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. *Clust. Comput.* **2020**, *23*, 1137–1147. [CrossRef]
44. Gill, S.S.; Buyya, R.; Chana, I.; Singh, M.; Abraham, A. BULLET: Particle Swarm Optimization Based Scheduling Technique for Provisioned Cloud Resources. *J. Netw. Syst. Manag.* **2018**, *26*, 361–400. [CrossRef]
45. Nabi, S.; Ahmad, M.; Ibrahim, M.; Hamam, H. AdPSO: Adaptive pso-based task scheduling approach for cloud computing. *Sensors* **2022**, *22*, 920. [CrossRef] [PubMed]
46. Chen, X.; Long, D. Task scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm. *Clust. Comput.* **2019**, *22*, 2761–2769. [CrossRef]
47. Kumar, M.; Sharma, S.C. PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint. *Sustain. Comput. Inform. Syst.* **2018**, *19*, 147–164. [CrossRef]
48. Kumar, M.; Sharma, S.C. PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. *Neural Comput Applic.* **2019**, *32*, 12103–12126. [CrossRef]
49. Zhou, Z.; Li, F.; Abawajy, J.H.; Gao, C. Improved PSO Algorithm Integrated with Opposition-Based Learning and Tentative Perception in Networked Data Centers. *IEEE Access* **2020**, *8*, 55872–55880. [CrossRef]
50. Madni, S.H.H.; Latiff, M.S.A.; Ali, J.; Abdulhamid, S.M. Multi-objective-Oriented Cuckoo Search Optimization-Based Resource Scheduling Algorithm for Clouds. *Arab. J. Sci. Eng.* **2019**, *44*, 3585–3602. [CrossRef]
51. Madni, S.H.H.; Abd Latiff, M.S.; Abdulhamid, S.M.; Ali, J. Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment. *Cluster Comput.* **2019**, *22*, 301–334. [CrossRef]
52. Pradeep, K.; Jacob, T.P. CGSA scheduler: A multi-objective-based hybrid approach for task scheduling in cloud environment. *Inf. Secur. J. A Glob. Perspect.* **2018**, *27*, 77–91. [CrossRef]
53. Natesha, B.V.; Kumar Sharma, N.; Domanal, S.; Reddy Guddeti, R.M. GWOTS: Grey Wolf Optimization Based Task Scheduling at the Green Cloud Data Center. In Proceedings of the 2018 14th International Conference on Semantics, Knowledge and Grids (SKG), Guangzhou, China, 12–14 September 2018; pp. 181–187.

54. Alzaqebah, A.; Al-Sayyed, R.; Masadeh, R. Task Scheduling based on Modified Grey Wolf Optimizer in Cloud Computing Environment. In Proceedings of the 2nd International Conference on new Trends in Computing Sciences (ICTCS), Amman, Jordan, 9–11 October 2019; pp. 1–6.
55. Natesan, G.; Chokkalingam, A. Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm. *ICT Express* **2019**, *5*, 110–114. [CrossRef]
56. Elaziz, M.A.; Xiong, S.; Jayasena, K.P.N.; Li, L. Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. *Knowl.-Based Syst.* **2019**, *169*, 39–52. [CrossRef]
57. Srichandan, S.; Ashok Kumar, T.; Bibhudatta, S. Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. *Future Comput. Inform. J.* **2018**, *3*, 210–230. [CrossRef]
58. Nasr, A.A.; Chronopoulos, A.T.; El-Bahnasawy, N.A.; Attiya, G.; El-Sayed, A. A novel water pressure change optimization technique for solving scheduling problem in cloud computing. *Clust. Comput.* **2019**, *22*, 601–617. [CrossRef]
59. Praveen, S.P.; Rao, K.T.; Janakiramaiah, B. Effective Allocation of Resources and Task Scheduling in Cloud Environment using Social Group Optimization. *Arab. J. Sci. Eng.* **2018**, *43*, 4265–4272. [CrossRef]
60. Domanal, S.G.; Guddeti, R.M.R.; Buyya, R. A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment. *IEEE Trans. Serv. Comput.* **2020**, *13*, 3–15. [CrossRef]
61. Shirani, M.R.; Safi-Esfahani, F. Dynamic scheduling of tasks in cloud computing applying dragonfly algorithm, biogeography-based optimization algorithm and Mexican hat wavelet. *J. Supercomput.* **2020**, *77*, 1214–1272. [CrossRef]
62. Gill, S.S.; Buyya, R. A Taxonomy and Future Directions for Sustainable Cloud Computing: 360 Degree View. *ACM Comput. Surv.* **2019**, *51*, 1–33. [CrossRef]
63. Lu, Y.; Sun, N. An effective task scheduling algorithm based on dynamic energy management and efficient resource utilization in green cloud computing environment. *Clust. Comput.* **2019**, *22*, 513–520. [CrossRef]
64. Haris, M.; Khan, R.Z. A Systematic Review on Load Balancing Issues in Cloud Computing. In *Sustainable Communication Networks and Application*; Karrupusamy, P., Chen, J., Shi, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 297–303.
65. Wei, J.; Zeng, X. Optimal computing resource allocation algorithm in cloud computing based on hybrid differential parallel scheduling. *Clust. Comput.* **2019**, *22*, 7577–7583. [CrossRef]
66. Milan, S.T.; Rajabion, L.; Darwesh, A.; Hosseinzadeh, M.; Navimipour, N.J. Priority-based task scheduling method over cloudlet using a swarm intelligence algorithm. *Clust. Comput.* **2020**, *23*, 663–671. [CrossRef]
67. Assiri, A.S.; Hussien, A.G.; Amin, M. Ant Lion Optim ization: Variants, Hybrids, and Applications. *IEEE Access* **2020**, *8*, 77746–77764. [CrossRef]
68. Chhabra, A.; Singh, G.; Kahlon, K.S. QoS-aware energy-efficient task scheduling on HPC cloud infrastructures using swarm-intelligence meta-heuristics. *Comput. Mater. Contin.* **2020**, *64*, 813–834. [CrossRef]
69. Assiri, A.S. On the performance improvement of Butterfly Optimization approaches for global optimization and Feature Selection. *PLoS ONE* **2021**, *16*, e0242612. [CrossRef]
70. Ajitha, K.M.; Indra, N.C. Fisher linear discriminant and discrete global swarm based task scheduling in cloud environment. *Clust. Comput.* **2022**. [CrossRef]
71. Abd Elaziz, M.; Attiya, I. An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing. *Artif. Intell. Rev.* **2021**, *54*, 3599–3637. [CrossRef]
72. Abualigah, L.; Alkhrabsheh, M. Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. *J. Supercomput.* **2022**, *78*, 740–765. [CrossRef]
73. Natarajan, Y.; Kannan, S.; Dhiman, G. Task Scheduling in Cloud Using ACO. *RACSC* **2022**, *15*, 348–353. [CrossRef]
74. Attiya, I.; Elaziz, M.A.; Abualigah, L.; Nguyen, T.N.; Abd El-Latif, A.A. An Improved Hybrid Swarm Intelligence for Scheduling IoT Application Tasks in the Cloud. *IEEE Trans. Ind. Inf.* **2022**, *18*, 6264–6272. [CrossRef]
75. Cheng, F.; Huang, Y.; Tanpure, B.; Sawalani, P.; Cheng, L.; Liu, C. Cost-aware job scheduling for cloud instances using deep reinforcement learning. *Clust. Comput.* **2022**, *25*, 619–631. [CrossRef]
76. Yin, L.; Zhou, J.; Sun, J. A stochastic algorithm for scheduling Bag-of-Tasks applications on hybrid clouds under task duration variations. *J. Syst. Softw.* **2022**, *184*, 111123. [CrossRef]
77. Kashikolaei, S.M.G.; Hosseinabadi, A.A.R.; Saemi, B.; Shareh, M.B.; Sangaiah, A.K.; Bian, G.-B. An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm. *J. Supercomput.* **2019**, *76*, 6302–6329. [CrossRef]
78. Prem Jacob, T.; Pradeep, K. A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization. *Wirel. Pers. Commun.* **2019**, *109*, 315–331. [CrossRef]
79. Agarwal, M.; Srivastava, G.M.S. Genetic Algorithm-Enabled Particle Swarm Optimization (PSOGA)-Based Task Scheduling in Cloud Computing Environment. *Int. J. Info. Tech. Dec. Mak.* **2018**, *17*, 1237–1267. [CrossRef]
80. Tizhoosh, H.R. Opposition-Based Learning: A New Scheme for Machine Intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005; pp. 695–701.
81. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
82. Eltaeib, T.; Mahmood, A. Differential Evolution: A Survey and Analysis. *Appl. Sci.* **2018**, *8*, 1945. [CrossRef]
83. Available online: https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3 (accessed on 20 March 2022).

84. jMetal 5 Web Site. Available online: http://jmetal.github.io/jMetal/ (accessed on 1 March 2022).
85. Chakraborty, S.; Saha, A.K.; Chakraborty, R.; Saha, M.; Nama, S. HSWOA: An ensemble of hunger games search and whale optimization algorithm for global optimization. *Int. J. Intell. Syst.* **2022**, *37*, 52–104. [CrossRef]
86. Available online: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51 (accessed on 13 May 2022).
87. Hussien, A.G.; Amin, M.; Abd El Aziz, M. A comprehensive review of moth-flame optimisation: Variants, hybrids, and applications. *J. Exp. Theor. Artif. Intell.* **2020**, *32*, 705–725. [CrossRef]