Received: January 9, 2024. Revised: February 8, 2024.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Multi Objective Prairie Dog Optimization Algorithm for Task Scheduling and Load Balancing

Amith Shekhar Chandrashekhar[1]* Niranjan Murthy Chandrashekarappa[2]
Puneetha Bandalli Hanumanthagowda[3] Anupkumar Manohara Bongale[4]

[1]*Department of Computer Science and Engineering, B N M Institute of Technology, Bengaluru, India*
[2]*Department of Computer Science and Engineering, Jain Institute of Technology, Davanagere, India*
[3]*Department of Computer Science & Business Systems, Bapuji Institute of Engineering & Technology, Davanagere, India*
[4]*Department of Artificial Intelligence and Machine Learning, Symbiosis Institute of Technology, Pune, India*
* Corresponding author's Email: amith_shekhar@bnmit.in

Abstract: An efficient task scheduling plays an important role in facilitating the virtual resource in a cloud computing environment by minimalizing make span and enhancing the allocation of resources. Requests for resources are treated as tasks, and appropriate resources are allocated based on user requirements. But, due to high demand and requests, the cloud has difficulty allocating resources. To overcome the issues, this research introduced an optimization-based task scheduling approach. The Multi-Objective Prairie Dog Optimization (MOPDO) algorithm is introduced which considers the makespan time and the execution time as the major objective while allocating resources in IoT. The proposed MOPDOA effectively allocates the resource to the Virtual Machines (VMs) by choosing the host with maximized resources. The search mechanism with the help of MOPDO helps to detect a suitable VM for resource allocation will be continued. After the process of allotting the resources to VMs, the load balancing process must be initiated to schedule the tasks for VMs. When the task count is assigned as 100, the makespan time of MOPDOA is 12s while Particle Swarm Gray Wolf Optimization (PSGWO) obtains a makespan time of the 20s. Similarly, for different VMs, the proposed approach is 175.45s for execution whereas the existing Improved Multi-Objective Multi-Verse Optimizer obtains 186.33s to execute for 10 VMs.

Keywords: Cloud computing, Task scheduling, Multi-objective prairie dog optimization, Make span time, Execution time, Load balancing.

1. Introduction

Cloud computing emerged as a revolutionized computing technology where the computing resources are evaluated globally via the internet. Moreover, it offers a set of services and resources through Cloud Service Providers (CSP) [1, 2]. The term "Cloud Computing" is derived from two unique words such as cloud which is related to network and computing refers to calculation. So, cloud computing is referred to as the process of computing data using a computer [3]. The available resources are utilized by physically available machines. While scheduling the tasks among virtual machines, the load imbalance is created due to overutilization and unutilization of resources [4]. Containerizing acts as a relevant solution to fulfil the needs in a cloud environment [5]. In comparison to virtual machines, containerization is a cloud-based technology that is becoming more and more popular because of its advantages in terms of lightweight, scalability, and availability [6]. Workflows for continuous integration and delivery are best suited for containers. To balance resources among the virtual machines and to have positive feedback in balancing resources, a precise load-balancing system acts as a key method. In addition, the load balancing is maintained by spreading the load by allotting the specified load to every virtual machine [7, 8]. The resources can be increased in two

different ways: vertically by adding additional resources to the deployed virtual machines, or horizontally by adding new virtual machines. Both approaches take more time, have latency problems, and could be more expensive [9]. These issues affect the processing of the data and create a critical situation in maintaining the equalized data distribution among the virtual machines.

The optimized load balancer acts as an important component that considers all the aforementioned issues by using efficient balancing strategies [10]. In general, load balancing is classified into two types such as static load balancing and dynamic load balancing. In static load balancing, the information is obtained in a prior state regarding the characteristics, computing nodes, and network. For dynamic load balancing, the efficacy of the processor is distinguished at the initial stage of the execution process [11, 12]. Every service transaction is performed based on allocating and scheduling the resources. At the time of scheduling the tasks, some of the short-time tasks are dismissed due to time delay. In the process of task scheduling, the scheduler deliberates the task based on resource availability [13]. Moreover, the virtual machine must be suitable to balance the resources among the hosts which tends to diminish the computational cost. The scheduling of virtual machines in a big data environment must act as an identifier to detect the optimal pairs of virtual machines [14]. Therefore, a discrete optimization technique should be utilized to overcome the issues regarding optimized load balancing [15,16]. This research introduced an improved optimization-based load balancing technique based on a cloud environment for the applications related to the allocation of resources and scheduling the tasks.

The significant findings of this research are,

1. This research presented an optimization-based task scheduling scheme using MOPDOA to maintain load balance in a cloud environment.
2. The proposed MOPDOA is based on multi-objective functions such as makespan time and execution time. These two functionalities are used to allocate the resources in the VMs with better scalability.

The remaining research paper is structured as follows: Section 2 describes related works and Section 3 describes the proposed methodology. The results and analysis of this research are described in Section 4 and finally, the conclusion of the research is described in Section 5.

2. Related works

Here, the various research based on task scheduling and load balancing in cloud computing environments are discussed.

Mohd Sha Alam Khan and R. Santhosh [17] have introduced a hybrid optimization algorithm that is comprised of Particle Swarm Optimization (PSO) and Gray Wolf Optimization (GWO) referred to as PSGWO for an optimistic resource allocation in the cloud platform. At the initial stage, the VMs are categorized on the basis of Support Vector Machines (SVM). The suggested method detects the optimal VM and helps in the optimal allocation of resources. Moreover, the suggested approach effectively minimizes the waiting time and enhances the QoS. However, the suggested approach offers diminished results when the VMs are placed randomly. Sudheer Mangalampalli [18] have introduced a task scheduling approach using Cat Swarm Optimization (CSO) algorithm. The CSO addresses the makespan, time taken for migration, energy consumption, and power cost. The task scheduling was performed by calculating the priorities at the task level and VM level to provide a mapping of tasks for the respective VMs. However, the suggested approach was not suitable for scheduling tasks based on a real-time cloud environment.

Mohammed Otair [19] have introduced an Improved Multi-Objective Multi-Verse Optimizer (IMOMVO) to solve the task scheduling problems which occur in a cloud environment. The suggested approach rectifies the issues related to average positioning by improving the position based on the best solution. The IMOMVO can perform various tasks comprised of different tasks and VM to evaluate the ability of the scheduling tasks. However, the search space of the proposed approach does not consider any fitness functionalities. Mohammed I. Alghamdi [20] have introduced Artificial Neural Network Binary Particle Swarm Optimization (ANN-BPSO) to perform load balancing and task scheduling in the cloud computing environment. The ANN-BPSO allots reference for each particle and helps to accelerate the search for an optimal solution for the allocation of resources and scheduling the tasks. Moreover, the suggested approach updates the position of every individual particle and helps to prohibit the overload or underload of VMs. However, the resource allocation using ANN-BPSO does not consider energy consumption which is an essential parameter that needs to be considered while allocating the resources.

Chirag Chandrasekhar [21] have introduced a Hybrid Weighted Ant Colony Optimization

(HWACO) algorithm to perform ideal task scheduling. The suggested HWACO considered two functions such as the Construct-ANT solution and the Update pheromone function to schedule the tasks with better makespan and parameters based on cost. The HWACO algorithm achieved integration in minimal time but the algorithm lacked its efficiency when the task count was enhanced. Mehak Sharma [22] have introduced an optimistic approach for scheduling tasks in a cloud computing environment. The suggested approach plays an effective role in positioning the VMs and helps to enhance the QoS parameters. Moreover, the suggested approach minimizes the configuration overhead in a cloud environment through effective scheduling and improvising the QoS with fewer hosts.

The results obtained from the overall methods discussed in the related works show that the existing approaches had taken higher makespan time and execution time while allocating the resources over a cloud environment. So, the proposed optimization approach effectively balances the load and allocates resources to the cloud environment.

3. Optimization-based load balancing and task scheduling in a cloud environment

The cloud service providers have designed the cloud environment based on Virtual Machines (VMs) and Physical Machines (PMs), which also rely as a boundary for public accessibility. The user executes tasks based on their requirement and it will be aggregated by the resource manager to retain the records effectively. Moreover, the tasks which are performed based on the user's request are known as system loads. Following this, a task scheduler is used to schedule tasks for the VMs by maintaining the load balance by allocation of resources. The task scheduling is performed after the scheduler receives the task from request manager. The process of scheduling the task is improvised by obtaining the exact location of VMs along with the user-requested tasks. The process involved in task scheduling helps to minimize the migration cost, utilization of load, time, and computational cost. The proposed task scheduling approach is effective in balancing the loads to the virtual machines and scheduling the tasks. The incoming tasks from the user get dispersed among the cloud and the task scheduling is performed with load balance. The entire process involved in task scheduling is diagrammatically represented in Fig. 1 as follows:

The list of tasks that need to be executed is represented as $T = \{t_1, t_2, \ldots, t_m\}$ and the list of virtual machines is represented as $V = \{V_{M1}, V_{M2}, \ldots, V_{Mn}\}$. Every individual VM can process and execute the task that was assigned to it. The main objective of this research is to build an effective task scheduler algorithm for the allocation of resources and maintain the load balance.

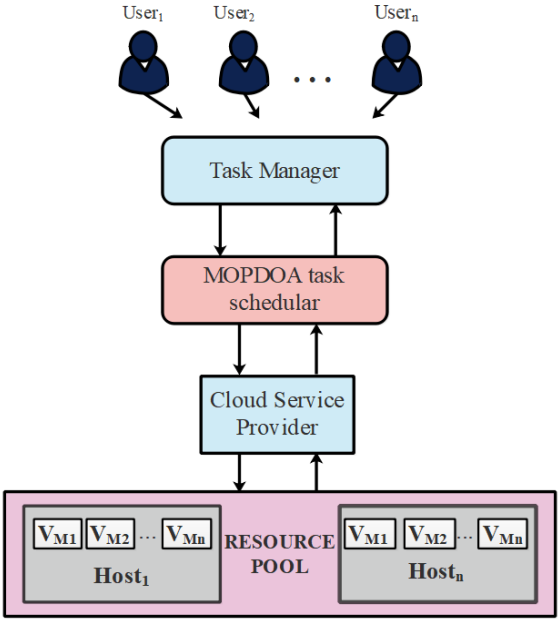

Figure. 1 The architecture of the proposed MOPDOA task scheduler for allocation of resources in a cloud environment

3.1 Allocating resources to Virtual Machines (VMs)

The problem related to the allocation of resources is framed as an optimization problem (P) which is represented as Eq. (1).

$$P = Maximize\left(\frac{P_j^{CPU}}{V_{M_i}^{CPU}} + \frac{P_j^{MEM}}{V_{M_i}^{MEM}} + \frac{P_j^{bw}}{V_{M_i}^{bw}}\right) \quad (1)$$

Where the variables that are required in allocating the resources in the VM model are represented as $V_{M_i}^{CPU}$ (Variable for processing unit), $V_{M_i}^{MEM}$ (Variable for memory), and $V_{M_i}^{bw}$ (Variable for bandwidth). The variables used by the physical machines are represented as P_j^{CPU} (Variable for processing unit), P_j^{MEM} (Variable for memory), P_j^{bw} (Variable for bandwidth) . Since the host allows additional resources based on the limit of VMs, the proposed model worked in detecting a host with the highest units based on the Eq. (2) to Eq. (5) as follows:

$$\forall i \sum_{j=1}^{m} y_{ij} = 1 \quad (2)$$

$$\forall i \sum_{j=1}^{n} y_{ij} V_{M_i}^{CPU} \leq P_j^{CPU} \quad (3)$$

$$\forall i \sum_{j=1}^{n} y_{ij} V_{M_i}^{MEM} \leq P_j^{MEM} \quad (4)$$

$$\forall i \sum_{j=1}^{n} y_{ij} V_{M_i}^{bw} \leq P_j^{bw} \quad (5)$$

The proposed method focuses on detecting the host with a maximal count of available resources. From Eq. (2) to Eq. (5), the count of tasks and the count of VMs are represented as M and n respectively. Moreover, y_{ij} is known as a binary variable which shows that the VMs can be placed on the jth layer of the physical machine. Eq. (2) shows that every individual VM is allotted to one physical machine. Eq. (3), Eq. (4), and Eq. (5) verify the available resources with the existing ones which include CPU, memory and the bandwidth of jth layer of the physical machine. This bandwidth of the jth layer is capable of being selected as a replacement for ith VMs. If the conditions defined in Eq. (2) to Eq. (5) are satisfied, then the VMs will be positioned on the required physical machine.

3.2 Task Scheduling to VMs

Task scheduling is defined as a process of mapping tasks to suitable resources to optimize their usage. Tasks are allotted to the VM which has more effort and less communication delay. The VM host on the physical machines is used to allocate the resources and gets isolated from the varying users of cloud infrastructure. The provider follows up on the tasks and confirms the effective Quality of Service (QoS) based on the user requirements. In the stage of task scheduling, the response time of suggested approach is minimized. The following Eq. (6) to Eq. (8) are the conditions that are satisfied by the proposed method at the time of task scheduling.

$$\forall i \sum_{j=1}^{m} T_{ij} = 1 \quad (6)$$

$$F_i \leq A_i + D_i \quad (7)$$

$$et_{ij} + D_i \leq ct_{ij} \quad (8)$$

Where T_{ij} is denoted as the task i which is allotted for j the virtual machine, estimated period to accomplish task is denoted as F_i and A_i denotes time of arrival of the i_{th} task. Moreover, the time of the deadline of the i_{th} the task is denoted as D_i. The time of execution and the time of completion of ith task in jth VM is represented as et_{ij} and ct_{ij} respectively. Eq. (6) denotes that every individual task should be allotted to one VM only and Eq. (7) is utilized to verify the time of deadline and the time of executing the ith task. The final Eq. (8) shows the period taken to accomplish the ith task on jth VM in a minimal time by considering the deadline. The evaluation of runtime and the work completion is acquired based on Eq. (9) and Eq. (10) mentioned as follows:

$$et_{ij} = \frac{Length\ of\ the\ task}{Mips\ of\ V_{m_j} \times number\ of\ processors} \quad (9)$$

$$ct_{ij} = \mathrm{et_{ij}} + wt_i \quad (10)$$

Where $\mathrm{V_{m_j}}$ represents the jth VM and the time of waiting to accomplish the ith task is denoted as wt_i.

3.3 Evaluation of load status

The load status of the VMs is computed based on the provided load which takes place in the prior time of scheduling and assigning the task to the preferred machine. The VM load is based on parameters such as processor load, memory usage, and bandwidth. Moreover, these parameters take the significant responsibility to pre-determine the load state of the VM. Eq. (11) shows the mathematical expression for the evaluation of load status.

$$L = \{L_1, L_2, L_3\} \quad (11)$$

Where L_1, L_2 and L_3 represents CPU usage, memory space and bandwidth respectively. The evaluation of L_1, L_2 and L_3 takes place based on equation (12) as follows:

$$\left.\begin{array}{c} L_1 = \text{CPU usage of} \frac{V_{M_i}}{P_j^{CPU}} \\ L_2 = \text{Memory usage of} \frac{V_{M_i}}{P_j^{MEM}} \\ L_3 = \text{Bandwidth usage of } V_{M_i} \end{array}\right\} \quad (12)$$

$L_i(t) = \sum_{j=1}^{n} \frac{L_j}{n}$ is the degree of the load and the conditions to be satisfied are represented in Eq. (13) Eq. (14).

$$\begin{cases} \text{idle}, & L_i(t) = L_i^{idle}(t) = 0 \\ \text{under load}, & L_i^{idle}(t) < L_i(t) < L_i^{min}(t) \end{cases} \quad (13)$$

$$\begin{cases} \text{normal}, & L_i^{min}(t) < L_i(t) < L_i^{max}(t) \\ \text{overload}, & L_i(t) > L_i^{max}(t) \end{cases} \quad (14)$$

Where the maximal and minimal load of the host is represented as $L_i^{max}(t)$ and $L_i^{min}(t)$ respectively, the status of the load is at the final stage and is considered as the final decision.

3.4 Overview of PDOA

Generally, the load balancing is performed by the optimization models which detect an optimistic solution to balance the load and allocate the tasks to the respective VMs. In this research, MOPDOA is used to allocate resources considering the make span and the execution time while allocating the resources. The proposed algorithm is obtained from PDOA [23] to minimize the make span and the execution time while allocating the resources. The prairie dog's population relies as a search agent in d-dimensional vector space to evaluate an individual's position. The activities based on foraging and burrow building come under the stage of exploration and they produce a squeaky sound to designate the occurrence of food which is under the phase of exploitation.

3.4.1. Exploration stage

In the stage of exploration, the prairie dogs undergo two processes such as foraging and the construction of burrows which are represented in Eq. (15) and Eq. (16) respectively.

$$PD_{i+1.j+1} = GBest_{i,j} - eCBest_{i,j} \times \rho - CPD_{i,j} \times Levy(n) \forall iter < \frac{Max_{iter}}{4} \quad (15)$$

Where $GBest_{i,j}$ is the global best solution and best solution is denoted as $eCBest_{i,j}$. The food source is ρ and randomized outcome is denoted as $CPD_{i,j}$. The position will be based on constructing a burrow which is denoted in Eq. (16).

$$PD_{i+1.j+1} = GBest_{i,j} \times rPD \times DS \times Levy(n) \forall \frac{Max_{iter}}{4} \leq iter < \frac{Max_{iter}}{2} \quad (16)$$

Where the random solution is denoted as rPD and the digging strength is denoted as DS. $Levy(n)$ is the Levy distribution function in the exploration stage.

3.4.2. Exploitation stage

The communication ability of prairie dogs has a major role in sustaining food requirements and safeguarding them from enemies. The two strategies followed by them are denoted in Eq. (17) and Eq. (18).

$$PD_{i+1.j+1} = GBest_{i,j} - eCBest_{i,j} \times \varepsilon - CPD_{i,j} \times rand \; \forall \frac{Max_{iter}}{2} \leq iter < 3\frac{Max_{iter}}{4} \quad (17)$$

$$PD_{i+1.j+1} = GBest_{i,j} \times PE \times r \; and \; \forall \; 3\frac{Max_{iter}}{4} \leq iter < Max_{iter} \quad (18)$$

Where ε is the quality of the food source and random number in the range 0 to 1 is denoted as $rand$ and predator effect is PE that is represented in Eq. (19).

$$PE = 1.5 \times \left(1 - \frac{iter}{Max_{iter}}\right)^{\left(2\frac{iter}{Max_{iter}}\right)} \quad (19)$$

Where iteration in current state and best iteration is denoted as $iter$ and Max_{iter} correspondingly.

3.5 Load balancing and task scheduling using MOPDOA

After acquiring the information related to load status, the load balancing is performed. In this research, load balancing is performed by resource allocation to VM and task scheduling. The MOPDOA considered the multiobjective functions such as makespan and the execution time rather than the traditional fitness function in PDOA. The proposed MOPDOA effectively allocates the resource to the VMs by choosing the host with maximized resources which should satisfy the conditions mentioned previously in equations (1-4). Whenever the computed VM has enough number of resources, these resources are allotted to the target VMs. Otherwise, the search mechanism to detect a suitable VM for resource allocation will be continued. After the process of allotting the resources to VMs, the load balancing process must be initiated to schedule the tasks for VMs. The scheduler selects a VM to proceed the values based on execution and completion time which was stored in the memory to make decisions accordingly. The objective function of the proposed MOPDOA is updated by considering the fitness like make span and execution time which is mathematically represented in Eq. (20) as follows:

$$Objective\ function = \sum_{i=1}^{n} T_i\ (\alpha.Makespan + \beta.Execution\ time) \quad (20)$$

Where the task which is selected from the list of tasks is represented as T_i and value of i varies from [1-n] (n denotes total count of tasks). The make span constant is represented as α and the constant which is related to execution time is represented as β. The steps involved in the process of scheduling the tasks are listed as follows:

i. The request from the user is transmitted to the cloud server and the evaluation is done based on various VMs. The tasks are stored based on time matrices on execution and completion period.
ii. The set of requests is received and the set with the minimum deadline is selected. Moreover, the VMs are selected based on execution and completion time matrices and the model's constraints.
iii. The status of the load for the selected machine is computed, if the load status of the VM is in the abnormal state then the search is continued to find a better machine. The overall process involved in task scheduling using the proposed MOPDOA is represented in Fig. 2 as follows:

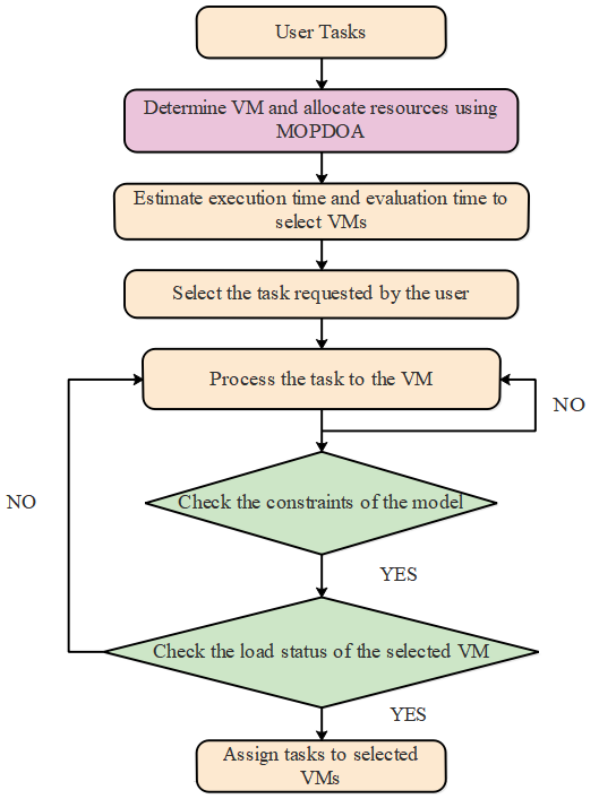

Figure. 2 Flowchart for the process involved in resource allocation and task scheduling using MOPDOA

4. Results and analysis

Here, the experimental outcome of the suggested framework is evaluated based on efficiency in makespan, execution time, and fitness. The proposed approach is implemented on a VM with 10GB Random Access Memory, a 2.4 processor, 36GB virtual storage, and Windows 10 operating system. The makespan and the execution time are evaluated using the Eq. (21) and Eq. (22) respectively.

$$Make\ span = \sum_{i=1}^{n}\sum_{j=1}^{m} E_{ij}X_{ij} \quad (21)$$

Where the makespan denotes the waiting time and the completion time of the tasks. The execution time of the task i on the VM is represented as E_{ij} and the Boolean representation in VM is represented in Eq. (23) X_{ij}.

$$Execution\ time = \sum_{t=0}^{m} End\ of\ execution\ time - \frac{start\ time\ (t)}{n} \quad (23)$$

Where the number of tasks executed in the VM is represented as n.

4.1. Performance analysis

Here, efficiency of MOPDOA is assessed on the basis of makespan time, execution time, and based on fitness functions. The effectiveness of MOPDOA is assessed with existing metaheuristic optimization techniques such as the Particle Swarm Optimization algorithm (PSO), Grey Wolf Optimization algorithm (GWO), Whale Optimization Algorithm(WOA), and Grasshopper Optimization Algorithm (GOA). The efficacy of MOPDOA is assessed for different task count ranges from 100 to 1000. Table 1 exhibits the experimental outcome based on makespan time.

The outcome through Table 1 exhibits MOPDOA has taken minimum makespan time when evaluated with other existing optimization techniques. For example, makespan time of the proposed MOPDOA for 100 tasks is 12s whereas the existing optimization techniques like PSO, GWO, WOA, and GOA have the makespan time of 37s, 33s, 28s, and 23s respectively. The optimal outcome of MOPDOA is due to its ability to maintain balance among exploration and exploitation that assists in maintaining load balance while scheduling the tasks to VMs. Fig. 3 presents the graph for comparison of makespan time for different task counts.

Table 1. Evaluation of makespan time

Methods	Make span time (sec)									
	100	200	300	400	500	600	700	800	900	1000
PSO	37	76	132	158	189	213	249	281	368	392
GWO	33	64	121	142	187	204	240	274	359	387
WOA	28	60	117	138	175	198	235	277	341	381
GOA	23	54	108	121	172	192	221	260	329	377
MOPDOA	12	48	93	115	168	188	210	255	300	370

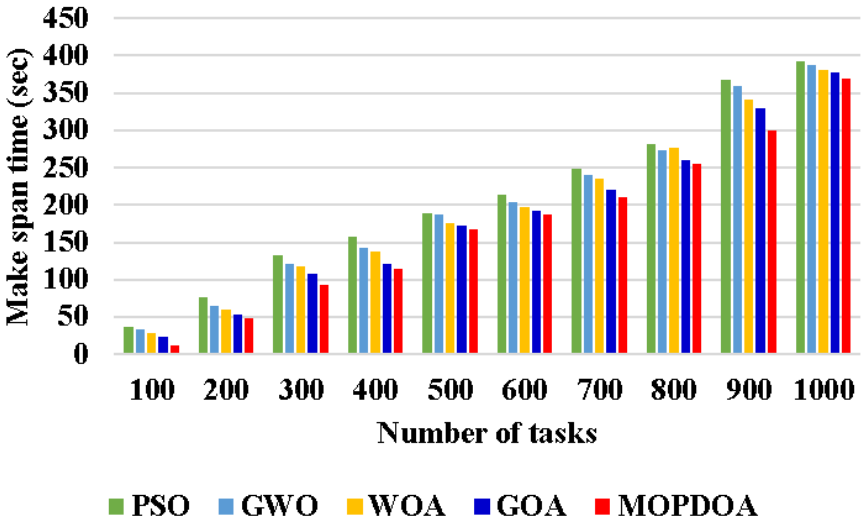

Figure. 3 Graphical representation for comparison of makespan time

Table 2. Evaluation of execution time to allocate resources

Methods	Execution time (sec)									
	100	200	300	400	500	600	700	800	900	1000
PSO	1450	1600	1700	1750	1800	1900	2200	2600	3400	3800
GWO	1200	1350	1500	1400	1650	1750	2150	2550	3250	3650
WOA	1050	1300	1250	1350	1500	1700	2000	2500	3150	3400
GOA	900	1150	1100	1150	1400	1600	1900	2450	3100	3150
MOPDOA	750	900	950	1100	1300	1550	1800	2400	2950	3200

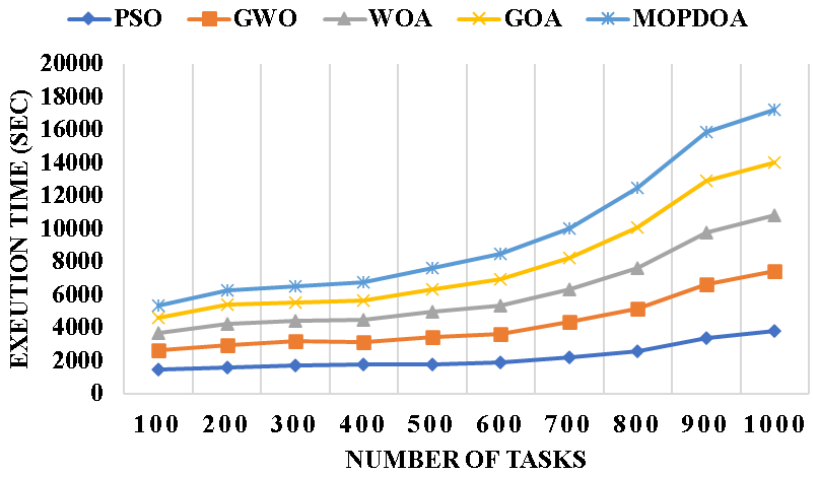

Figure. 4 Graphical representation for comparison of execution time

Secondly, the efficiency of MOPDOA is assessed on the basis of execution time. Table 2 show the time taken by proposed optimization approach to implement the tasks when compared with existing optimization techniques.

Table 2 and Fig. 4 exhibits that MOPDOA took minimal execution time when it is compared with the

existing optimization algorithm. For a task count of 1000, the proposed approach took 3200 seconds whereas the existing PSO, GWO, WOA, and GOA had taken execution times of 3800s, 3650s, 3400s and 3150s respectively. The better result of MOPDOA is due to the minimal response time which helps to execute the tasks to the VM in minimal time.

4.2. Comparative analysis

Here, the efficiency of MOPDOA is assessed on the basis of makespan time and the execution time for different task counts. The performance of MOPDOA is assessed with PSGWO [17], IMOMVO [19] and ANN-BPSO [20]. The simulation setup of the proposed approach is based on three cases. The PSGWO is evaluated based on case 1, the IMOMVO is evaluated based on case 2 and ANN-BPSO is evaluated based on case 3.

Case 1: There are 10 data centers considered for evaluation for task count from 100-1000. The number of hosts and total VMs are 20 and 100 respectively. The evaluation of results based on case 1 is presented in table 3.

Case 2: The evaluation is performed for VMs from 10-60 and the number of cores from each host is 2. When VM is 10, the number of tasks is 100 and vice versa. The evaluation of results based on case 2 is presented in table 4.

Case 3: The evaluation is performed from task count from 1000-5000. The cloud sim simulator is used to assess the proposed method's efficiency and 150 VMs are spread in an even manner across 15 hosts. The evaluation of results based on case 3 is presented in table 5.

In Table 3, the performance of the proposed MOPDOA is evaluated with PSGWO for different task counts ranging from 100 to 1000.

Secondly, the efficiency of MOPDOA is assessed with IMOMVO based on execution time and throughput for variable number of VMs. Table 4 presents the outcome achieved while evaluating MOPDOA with IMOMVO. Thirdly, the performance is evaluated based on MOPDOA's efficiency with ANN-BPSO based on average resource utilization and make span time. The table 5 presents the results achieved while evaluating MOPDOA with ANN-BPSO.The experimental outcome through Table 3 and Table 4 exhibits MOPDOA acquires better results in terms of make span and execution time for different numbers of tasks and different numbers of VMs. When the task count is assigned as 100, the make span time of MOPDOA is 12 s whereas PSGWO obtains a make span time of 20 s. Similarly, for different VMs, the proposed approach is 175.45s for execution whereas the existing IMOMVO obtains 186.33s to execute for 10 VMs. The experimental results from Table 5 shows that MPDOA performs better in terms of average resource utilization and make span time. The average resource utilization of MOPDOA for 1000 tasks is 98.21% which is comparably higher than the existing ANN-BPSO

Table 3. Comparison of makespan and execution time based on number of tasks

Methods	**Metrics (sec)**	**Number of tasks**									
		100	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
PSGWO [17]	Make span time	20	60	100	130	180	200	230	270	310	390
MOPDOA		12	48	93	115	168	188	210	255	300	370
PSGWO [17]	Execution time	900	1000	1100	1300	1700	1850	2400	3000	3300	3900
MOPDOA		750	900	950	1100	1300	1550	1800	2400	2950	3200

Table 4. Comparison of execution time based on number of VMs

Methods	**Metrics**	**Number of VMs**					
		10	**20**	**30**	**40**	**50**	**60**
IMOMVO [19]	Execution time (sec)	186.33	385.34	502.56	743.11	851.89	934.92
MOPDOA		175.45	370.67	400.37	690.87	802.19	883.58
IMOMVO [19]	Throughput (Kbps)	0.19	0.43	0.441	0.525	0.78	0.797
MOPDOA		0.23	0.46	0.47	0.54	0.81	0.83

Table 5. Comparison of average resource utilization and make span time

Methods	**Methods**	**Number of tasks**				
		1000	**2000**	**3000**	**4000**	**5000**
ANN-BPSO [20]	Average resource utilization (%)	96.84	95.21	94.54	94.09	93.56
MOPDOA		98.21	97.63	95.98	95.13	94.89
ANN-BPSO [20]	Make span time (Sec)	90	96	100	120	150
MODPOA		86	92	97	108	135

Notation list

Parameter	Description
$V_{M_i}^{CPU}$	Variable for processing unit
$V_{M_i}^{MEM}$	Variable for memory
$V_{M_i}^{bw}$	Variable for bandwidth
P_j^{CPU}	Physical machine for processing unit
P_j^{MEM}	Physical machine for memory
P_j^{bw}	Physical machine for bandwidth
M	Number of tasks
n	Number of VM
y_{ij}	Binary variable
T_{ij}	The task i which is allotted for j the virtual machine
F_i	Estimated period to complete the task
A_i	The time of arrival of the i_{th} task
D_i	The time of the deadline of the i_{th} task
et_{ij}	Time of execution
ct_{ij}	Time of completion
$L_i^{max}(t)$	Maximal load of the host
$L_i^{min}(t)$	Minimal load of the host
$GBest_{i,j}$	Global best solution
$eCBest_{i,j}$	Best solution among the population
ρ	Source of food
$CPD_{i,j}$	Randomized cumulative outcome
rPD	Random solution
DS	Digging strength
$Levy(n)$	Levy distribution function
ε	Quality of the food source
$rand$	Random number in the range 0 to 1
PE	Predator effect
$iter$	Iteration in the current state
Max_{iter}	Maximum iteration
T_i	The task which is selected from the list of tasks
β	The execution time
E_{ij}	The execution time of the task i on the VM
X_{ij}	The Boolean representation in VM

with 96.84% accuracy. The better result of MOPDOA is due to the ability in balancing load among exploration and exploitation to maintain the load balance while scheduling the tasks to VMs.

5. Conclusion

This research introduced an optimization-based task scheduler that effectively schedule tasks to the corresponding VMs with minimal make-span time and execution time. The MOPDO prioritizes make span time and execution time in resource allocation with load balancing and task scheduling. When the job count is set to 100, the suggested approach has a manufacture span time of 12 s, whereas the existing Particle Swarm Gray Wolf Optimization (PSGWO) has a make span time of 20 s. Similarly, the suggested technique takes 175.45s to run for different VMs, whereas the existing Improved Multi-Objective Multi-Verse Optimizer takes 186.33s to perform for 10 VMs. The better result of MOPDOA is due to the ability in balancing load among exploration and exploitation to maintain the load balance while scheduling the tasks to VMs that helps in balancing the load and aids in better allocation of resources. In the future, hybridized optimization approaches can be utilized to minimize the makespan time and execution time.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

The paper conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, have been done by 1st author. The supervision and project administration, have been done by 2nd , 3rd and 4th author.

References

[1] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment", *Journal of King Saud University - Computer and Information Sciences*, Vol. 34, No. 6, pp. 2332–2342, 2022.

[2] C. A. Shekhar and G. S. Sharvani, "MTLBP: A Novel Framework to Assess Multi-Tenant Load Balance in Cloud Computing for Cost-Effective Resource Allocation", *Wireless Communications and Mobile Computing*, Vol. 120, No. 2, pp. 1873–1893, 2021.

[3] P. Rani, P. N. Singh, S. Verma, N. Ali, P. K. Shukla, and M. Alhassan, "An Implementation of Modified Blowfish Technique with Honey Bee Behavior Optimization for Load Balancing in Cloud System Environment", *Wireless Communications and Mobile Computing*, Vol. 2022, pp. 1–14, 2022.

[4] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment", *Journal of King Saud University - Computer and Information Sciences*, Vol. 34, No. 3, pp. 813–824, 2022.

[5] A. Thakur and M. S. Goraya, "RAFL: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment", *Simulation Modelling Practice and Theory*, Vol. 116, p. 102485, 2022

[6] A. Narwal and S. Dhingra, "A novel approach for Credit-Based Resource Aware Load Balancing algorithm (CB-RALB-SA) for scheduling jobs in cloud computing", *Data & Knowledge Engineering*, Vol. 145, p. 102138, 2023.

[7] N. Iqbal, A.N. Khan, A. Rizwan, F. Qayyum, S. Malik, R. Ahmad and D. H. Kim, "Enhanced time-constraint aware tasks scheduling mechanism based on predictive optimization for efficient load balancing in smart manufacturing", *Journal of Manufacturing Systems*, Vol. 64, pp. 19–39, 2022,

[8] N. Singh, Y. Hamid, S. Juneja, G. Srivastava, G. Dhiman, T. R. Gadekallu, and M. A. Shah, "Load balancing and service discovery using Docker Swarm for microservice based big data applications", *Journal of Cloud Computing*, Vol. 12, No. 1, p. 4, 2023.

[9] S. Meera and C. Sundar, "A hybrid metaheuristic approach for efficient feature selection methods in big data", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 12, No. 3, pp. 3743–3751, 2021.

[10] K. Sridevi and M. A. Saifulla, "LBABC: Distributed controller load balancing using artificial bee colony optimization in an SDN", *Peer-to-Peer Networking and Applications*, Vol. 16, No. 2, pp. 947–957, 2023.

[11] S. L. Mirtaheri and L. Grandinetti, "Optimized load balancing in high-performance computing for big data analytics", *Concurrency and Computation: Practice and Experience*, Vol. 33, No. 16, p. e6265, 2021

[12] S. Bagui, K. Devulapalli, and J. Coffey, "A heuristic approach for load balancing the FP-growth algorithm on MapReduce", *Array*, Vol. 7, p. 100035, 2020

[13] F. N. Al-Wesabi, M. Obayya, M. A. Hamza, J. S. Alzahrani, D. Gupta, and S. Kumar, "Energy Aware Resource Optimization using Unified Metaheuristic Optimization Algorithm Allocation for Cloud Computing Environment", *Sustainable Computing: Informatics and Systems*, Vol. 35, p. 100686, 2022

[14] A. Javadpour, A. M. H. Abadi, S. Rezaei, M. Zomorodian, and A. S. Rostami, "Improving load balancing for data-duplication in big data cloud computing networks", *Cluster Computing*, Vol. 25, No. 4, pp. 2613–2631, 2022

[15] Z. Miao, P. Yong, Y. Mei, Y. Quanjun, and X. Xu, "A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment", *Future Generation Computer Systems*, Vol. 115, pp. 497–516, 2021.

[16] M. Beshley, N. Kryvinska, O. Yaremko, and H. Beshley, "A Self-Optimizing Technique Based on Vertical Handover for Load Balancing in Heterogeneous Wireless Networks Using Big Data Analytics", *Applied Sciences*, Vol. 11, No. 11, p. 4737, 2021.

[17] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm", *Soft Computing*, Vol. 26, No. 23, pp. 13069–13079, 2022.

[18] S. Mangalampalli, S. K. Swain, and V. K. Mangalampalli, "Multi Objective Task Scheduling in Cloud Computing Using Cat Swarm Optimization Algorithm", *Arabian Journal for Science and Engineering*, Vol. 47, No. 2, pp. 1821–1830, 2022

[19] M. Otair, A. Alhmoud, H. Jia, M. Altalhi, A. M. Hussein, and L. Abualigah, "Optimized task scheduling in cloud computing using improved multi-verse optimizer", *Cluster Computing*, Vol. 25, No. 6, pp. 4221–4232, 2022.

[20] M. I. Alghamdi, "Optimization of Load Balancing and Task Scheduling in Cloud Computing Environments Using Artificial Neural Networks-Based Binary Particle Swarm Optimization (BPSO)", *Sustainability*, Vol. 14, No. 19, p. 11982, 2022

[21] C. Chandrashekar, P. Krishnadoss, V. Kedalu Poornachary, B. Ananthakrishnan, and K. Rangasamy, "HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing", *Applied Sciences*, Vol. 13, No. 6, p. 3433, 2023

[22] M. Sharma, M. Kumar, and J. K. Samriya, "An optimistic approach for task scheduling in cloud computing", *International Journal of Information Technology*, Vol. 14, No. 6, pp. 2951–2961, 2022.

[23] A. E. Ezugwu, J. O. Agushaka, L. Abualigah, S. Mirjalili, and A. H. Gandomi, "Prairie Dog Optimization Algorithm", *Neural Computing and Applications*, Vol. 34, No. 22, pp. 20017–20065, 2022