International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Task Scheduling Mechanism for MapReduce Framework using Inertia Weight based Grey Wolf Optimization

Vasantha.M [1*], Dr. Chandramouli.H [2]

Submitted: 27/01/2024 **Revised**: 05/03/2024 **Accepted**: 13/03/2024

Abstract: MapReduce is an admirable Hadoop technique for processing enormous information in cloud computing which runs programs and instructions in equivalent employing computers or processors. It simplifies complex data processing tasks by breaking them down into two basic operations mapping and reducing. Numerous scheduling algorithms are established for the Hadoop-MR model which varies in behavior and design, managing various issues like user share impartiality, and data locality along with resource consciousness. This paper proposed an innovative optimization algorithm such as Inertia Weight-based Grey Wolf Optimization (IW-GWO) Algorithm. The inertia weight strategy is utilized for controlling the exploration and population growth capability which effectively enhances the algorithm search capability and balance the connection between local development and global exploration. The obtained result shows that the IW-GWO attains better results in terms of makespan, cost, execution time, and throughput of 2.82s, 59.75 tasks/s, 52$ and 381.62ms for 200 nodes compared with existing algorithms like MOABCQ_LJF, HWACO, and IMOMVO.

Keywords*: *Cloud Computing, Grey Wolf Optimization, Inertia Weight, MapReduce Framework, Task Scheduling

1. Introduction

Task scheduling is a significant aspect of cloud computing which plays a significant role in optimizing resource consumption and ensuring effective task execution within a cloud framework [1]. Task scheduling is the process of allocating resources to various tasks, minimizing cost and improving the overall system effectiveness [2]. The virtual machines (VMs) deployment, data processing, handling of user requests, and effective task scheduling approaches are important for connecting the full potential cloud environment [3]. Cloud computing is utilized to generate user requirements for retrieving computing resources or allowing users to buy cloud services as essential in on-demand resource-sharing concept over internet-based applications [4] [5]. It is a payment method for the capability to use customizable computing sources across shared networks like servers, applications, storage space, and services [6]. This method is according to number of existing requests and accessed network levels that were generated rapidly through the smallest number of management and interference from user service providers [7] [8]. Cloud computing is a technique for allocating data and computing through huge network nodes like cloud services, computers, and data centers [9]. The cloud system is a procedure of distributed and parallel systems that contains a collection of VMs that contribute computing resources according to the service-level agreement (SLA) and through an agreement among clients and service providers [10]. Task scheduling provides a computing environment where different sources are distributed to customers through the Internet. In general, cloud architecture has an intranet or internet-related back and front end [11] [12]. It simplifies complex data processing tasks by breaking them down into two basic operations mapping and reducing [13]. In the mapping phase, the data was divided into smaller chunks and processed in parallel across multiple nodes [14].

The reducing phase aggregates and integrates the outputs, generating a comprehensive solution to complex problems.

Wang et al. [16] introduced a Heterogeneous Throughput Driven (HTD) task scheduling algorithm in MapReduce. This model initially provided a formal description of TD task scheduling in a heterogeneous network. Then, build scheduling HTD rapidly attained the completion sequence and enhanced scheduling information in a heterogeneous network. Lastly, the sequence was taken to a heterogeneous network to optimize the information of task assignment and attain task execution system. This developed model had some advantages like strong obtainability, huge reliability, and minimized processing time. However, this model was unable to resolve the idle resource problems.

Sharma et al. [17] developed an Ant Colony Optimization (ACO) model for QoS-based task scheduling in a cloud computing environment. The developed model functioned in two stages, the primary stage utilized an event-based scheduler to address the difficult planning problems. An updated ACO method for optimal global search utilizing a neural network to schedule numerous activities for overcoming the complexity of multiple objectives was suggested. The ACO model delivered high statistics mean

[1] *Research Scholar, East Point College of Engineering and Technology, Bengaluru-560049, Karnataka, India.*
[2] *Computer Science and Engineering, East Point College of Engineering and Technology, Bengaluru-560049, Karnataka, India.*
** Corresponding Author Email: vasantha.nhce@gmail.com, vasanthavijendra@gmail.com*

access times and reliable job assignments. However, this model led to imbalance issues over the devices.

Jeyaraj and Paul [18] implemented an Optimized MapReduce task scheduling on virtualized heterogeneous environments using Ant Colony Optimization (ACO). It was utilized to enhance exploitation by an accurate integration of maps and minimizing tasks in every VM. The MR task scheduling was transformed into a 2D-bin packing technique and attained the optimum schedule by the ACO algorithm. Each task combined the result in a single file, sorted, and grouped according to key. The developed ACO-BP model reduced the makespan for the batch of jobs. However, this model was not able to reduce the makespan.

Mangalampalli et al. [19] presented a multi-objective task scheduling algorithm in cloud computing through Grey Wolf Optimization (MOTSGWO). This model made runtime scheduling decisions according to cloud resources and task priorities. This model was applied to the Cloudsim toolkit and the distributions. The workload production was finished by making a dataset with various disseminations like the normal, uniform, right, and left skewed distributions. The MOTSGWO model minimized makespan and energy consumption. Nonetheless, this model had local optima issues and low convergence speed.

Praveen et al. [20] suggested a hybrid Genetic Algorithm (GA) and Gravitational Emulation Local Search (GELS) for task scheduling. This model integrated local search ability of GELS by GA that resulted in high effectiveness to obtain better solutions. The performance was estimated by comparing the results of GA and PSO algorithms. This analysis found that the model was effective in addressing task scheduling problems and delivered better results. The GAGELS model reduced response time and enhanced processor utilization. However, this model required a high running time.

Kruekaew and Kimpan [21] introduced a Multi-Objective task scheduling optimization for load balancing through Artificial Bee Colony with a Q-learning algorithm (MOABCQ). The developed model aimed to enhance VM throughput, optimize scheduling and resource exploitation, and make load balancing among VMs based on makespan, source exploitation and cost. The MOABCQ model minimized the makespan, degree imbalance and cost, thereby enhancing source exploitation and throughput. But, this model had slow convergence in complex problems and large solution spaces.

Chandrashekar et al. [22] developed a Hybrid Weighted Ant Colony Optimization Algorithm (HWACOA) for task scheduling in cloud computing. It was tested and related by existing algorithms in terms of makespan, parameter cost, and proficiency. This model improved the shortcomings encountered with various algorithms which enhanced the task scheduling performance. This developed HWACOA model reduced execution time, makespan, and energy consumption. However, this model considered only a smaller number of tasks and the performance was affected when increasing number of tasks.

Otair et al. [23] implemented an improved multi-objective multi-verse optimizer (IMOMVO) to resolve task scheduling issues. It was utilized to resolve the problems of average positioning (AP) by dynamically improving the equation of AP according to a better solution. The developed model was estimated according to three different objectives - throughput, makespan and execution time. The developed IMOMVO model performed tasks in less time, obtained minimum makespan and better throughput. Yet, this model had limitations like local optima issues and poor search scalability.

The purpose of this study is discussed below:

- This paper proposes an innovative optimization algorithm like Inertia Weight based Grey Wolf Optimization (IW-GWO) Algorithm. By adjusting inertia weight, the IW-GWO adapts its search behavior, leading to faster convergence towards optimal solutions.
- The FCGK-KMA is employed to improve the mapping reliability along with reducing the complications and the map-reduce function is accomplished over OB-BOA.
- The inertia weight strategy is utilized for controlling the exploration and population growth capability which effectively enhances the algorithms search capability and balance the connection between local development and global exploration.

The rest of the paper is arranged as follows: The complete description of the proposed method is given in Section 2, while the results and discussion are demonstrated in Section 3. At last, the conclusion is presented in Section 4.

2. Proposed Methodology

The Hadoop has become a standard technique for examining and storing data in a cost-efficient way which is a MapReduce (MR) open-source framework. Numerous scheduling algorithms are established for the Hadoop-MR model which vary in behavior and design, managing various issues like user share impartiality, and data locality along through resource consciousness. Therefore, an IW-GWO-based secured and optimized scheduling technique is proposed for MR by considering execution time. The inertia weight strategy is utilized for managing the exploration growth capability which effectively enhances the algorithm search capability and balances the connection between local development and global exploration. The MR job scheduling framework is presented in Fig. 1.

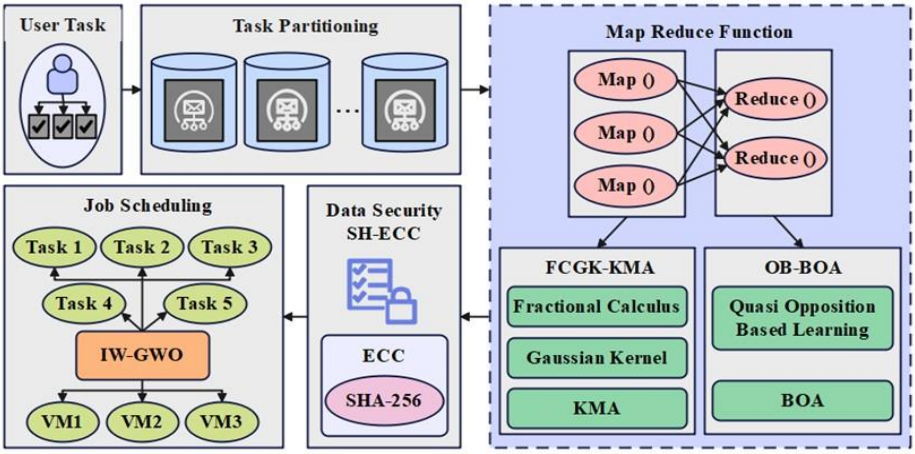

Fig. 1. Architecture of MapReduce Job Schedule Framework

A. Task Partitioning

Initially, for discovering cluster centroid to MR models, numerous amount of user tasks are separated into many smaller tasks. The n number of user tasks is represented as $(I_j = I_1, I_2, \dots, I_n)$ that is separated into m number of many tasks. The partitioned tasks are denoted as Eq. (1),

$$P_k = \langle P_1, P_2, \dots, P_m \rangle \quad (1)$$

Where, P_k is the partitioned tasks and P_m is the user task separated into m number of many tasks.

B. MapReduce Framework using BO-KMA

The MR is a programming technique for parallelly handling large sets of partitioning data which simplifies complex data processing tasks by breaking down into two basic operations, mapping and reducing. As an outcome, finding huge-scale data employing MR framework through allocating tasks on various clustering nodes. The Fractional Calculus integrates the Gaussian Kernel-centered K-Means clustering Algorithm (FCGK-KMA) and is employed as a mapper in developed BO-KMA method which obtains partitioning data as input and determines cluster centroids. Subsequently, an Opposition Based Butterfly Optimization Algorithm (OB-BOA) is utilized that selects the cluster centroid and performs as a reducer. The integration of OB-BOA in FCGK-KMA is labeled as BO-KMA. The MR framework is illuminated in below sections.

C. Mapping Function through FCGK-KMA

The FCGK-KMA is utilized to enhance the mapping reliability along with reducing the complications. The KMA is an enormously utilized clustering method that separates the task into numerous clusters. The Euclidean Distance (ED) is employed to measure the distance among every cluster. However, the time complexity is larger for measuring distance of large tasks. For minimizing time-complexity, ED is replaced through Gaussian Kernel (GK). Additionally, the KMA clustering performance was increased to enrich the quick convergence through fractional calculus.

Primarily, the partitioned task (P_k) are randomly clustered, then choose C cluster centers $(S_{k,i})$ as primary cluster center which is presented in Eq. (2),

$$S_{k,i} = (S_{k,1}, S_{k,2}, \dots, S_{k,C}) \quad (2)$$

Here, number of tasks is denoted through$k = 1,2, \dots, n$. The objective function is formulated in Eq. (3),

$$\zeta = \sum_{k=1}^{n} \sum_{i=1}^{C} \left\| P_k - S_{k,i} \right\|^2 \quad (3)$$

Where, GK Similarity (GKS) between portioned task input (P_k) and primary cluster centers (S_i) are formulated as$\|P_k - S_i\| = g(P_k, S_i)$. The GKS is calculated as Eqs. (4) and (5),

$$g(P_k, S_i) = exp\left[-\frac{|P_k - S_{k,i}|^2}{2\sigma^2} \right] \quad (4)$$

$$\sigma^2 = \frac{1}{n} \sum_{k=1}^{n} \left\| P_k - \left(\frac{1}{C} \sum_{i=1}^{C} S_{k,i} \right) \right\|^2 \quad (5)$$

Allocating every task for its adjacent cluster centroid positioned on GKS calculations. There are no changes in the cluster center, then it re-measured and assigns a task for each cluster $1 \le i \le C$. Therefore, the cluster centroids (L_i) are formulated as Eq. (6),

$$S_{k,i}(T+1) = \frac{1}{n_i} \sum_{\substack{k=1 \\ k \in i}}^{n_i} \left(\frac{g(P_k, S_{k,i})}{\sum_{i=1}^{C} g(P_k, S_{k,i})} \right) \quad (6)$$

Where, the number of tasks in ith mapper is denoted asn_i, the number of iterations is indicated as$T$. Then, fractional calculus is utilized to measure centroids which enriches quick convergence and solves fractional-order calculations through Laplace transform. At present iteration T is subtracted from $(T+1)$ iteration, then cluster centroid is measured which is denoted in Eq. (7),

$$S_{k,i}^{H}(T+1) = S_{k,i}(T+1) - S_{k,i}(T) \quad (7)$$

Based on the derivation of order ω, the Eq. (7) is indicated as Eq. (8),

$$d^{\omega}\langle S_{k,i}^{H}(T+1) = S_{k,i}^{H}(T+1)[\omega-1] + \frac{1}{2}\omega.S_{k,i}^{H}(T+1)\rangle + \frac{1}{6}\omega(1-\omega).S_{k,i}^{H}(T-2) + \frac{1}{24}\omega(1-\omega)(2-\omega).S_{k,i}^{H}(T-3) + S_{k,i}(T+1) \quad (8)$$

At iterations $(T-1), (T-2)$ and $(T-3)$, the cluster centroids are $S_{k,i}^{H}(T-1)$, $S_{k,i}^{H}(T-2)$ and $S_{k,i}^{H}(T-3)$ correspondingly. Hence, for calculating cluster centroid through mapper, the Eq. (8) is employed. As an output, each mapper result is dependent on the cluster center and input task. The n number of mapper output (M) is formulated as Eq. (9),

$$M = \{m_1, m_2, \dots, m_n\} \quad (9)$$

Where, M is the mapper output and the number of outputs is denoted by $m = 1, 2, \dots, n$.

D. Reducing Function through OB-BOA

To attain adequate clustering of centroids, the centroids produced through mapping functions are integrated. The data is clustered through a reducer in that the cluster number is user-specified. The reducing function is accomplished over OB-BOA. The BOA is a metaheuristic optimization algorithm that centered on food searching behaviors of butterflies [24]. In this sense, receptors were employed to find the food sources. The fragrances were generated by every butterfly which has a few intensities that are distributed and sensed through various butterflies in the area. If the butterfly changes their region, the fragrance intensity varies. If the butterfly changes to a stronger fragrance, it obtains a stronger fragrance produced through various butterflies. The OB-BOA procedure is presented as following:

Step 1: The primary population of butterflies is $\{r_j = r_1, r_2, \dots, r_N\}$ that is attained through clustering M, where some clusters in reducer are symbolized by r_N. The butterfly's fragrance magnitude (f^b) is expressed as sensor modality function (s), stimulus intensity (α), and power exponent (o) which is formulated as Eq. (10),

$$f^b = (s\alpha)^o \quad (10)$$

Where, (s) is the procedure of assessing butterfly energy, the physical stimulus magnitude is illuminated through (α) and the exponent to which butterfly intensity was high is defined by (o).

Step 2: The stimulus intensity (α) is related to butterfly fitness function and cluster centroids are discovered positioned on examined fitness. The fitness positioned on database index (τ) is formulated in Eq. (11),

$$\tau = \frac{1}{N}\sum_{j=1}^{N} X_j \quad (11)$$

Where, the function that selects the largest similarity score between r_j is indicated by x_j which is articulated as Eq. (12),

$$x_j = \max_{i \neq j} y_{j,i} \quad (12)$$

Where, the similarity measure among r_j centered on ED measure between the clusters is specified as $y_{j,i}$ which is indicated through Eq. (13),

$$y_{j,i} = \frac{p_j + q_i}{d_{j,i}} \quad (13)$$

The scattering measures between two butterflies is denoted as p_j andq_i. The highest clustering performance is denoted by the smallest distance among data points and the cluster centroid. The ED is measured as Eq. (14),

$$d_{j,i} = \left\| C_j - C_i \right\| \quad (14)$$

Where, cluster centroid is demonstrated through C_j, C_i. According to ED, p_j is measured cluster centered which is indicated as Eq. (15),

$$p_j = \frac{1}{N}\sum_{j=1}^{N}\left\| M - C_j \right\| \quad (15)$$

For food searching procedure, the global solution is designated on positioned fitness function.

Step 3: Butterflies travel towards fitness (b''') and global searching happens and the formula is indicated in Eq. (16),

$$r_j^{t+1} = r_j' + levy(\gamma) \times \left(b''' - r_j^{t}\right) \times f_j^b \quad (16)$$

Here, solution vector of tth iteration in jth butterfly is designated asr_j^t, the jth butterfly fragrance is illustrated through f_j^b and $levy(\gamma)$ is the levy flight distribution step size.

Step 4: The butterflies' local search is indicated as Eq. (17),

$$r_j^{t+1} = r_j' + levy(\gamma) \times \left(r_j^{t} - \hat{r}_j^{t}\right) \times f_j^b \quad (17)$$

Where, the butterflies are designated through Quasi opposition-based learning method that is denoted as $\tilde{r}_j^{t}$ which is formulated in Eq. (18),

$$\tilde{r}_j^{t} = \beta_j + \delta_j - r_j^{t} \quad (18)$$

Where, the corresponding butterfly interval is denoted as β_j, δ_j and the Quasi opposition number $\hat{r}_j^{t}$ is formulated as Eq. (19),

$$\hat{r}_j^{t} = rand\left(\frac{\beta_j + \delta_j}{2}, \tilde{r}_j^{t}\right) \quad (19)$$

Where, β_j, δ_j is the corresponding butterfly interval and $\tilde{r}_j^{t}$ is the quasi opposition-based learning method.

Step 5: Here, the distribution of Levy flight is calculated which is indicated in Eq. (20),

$$levy(\gamma) = \begin{cases} 1, & \gamma < 1 \\ \gamma^{-t}, & \gamma \geq 1 \end{cases} \quad (20)$$

To quicken butterflies in local search, the levy flight is employed by providing new solutions among the produced better solutions.

Step 6: By applying the above fitness function, the innovative solution was generated and estimated following the searching procedure. Then, lowest fitness is substituted through dominant solution and develops optimum solutions. Therefore, the optimum solution i.e., better centroids is observed in minimum task which is formulated in Eq. (21),

$$\phi_j = \langle \phi_1, \phi_{2,} \dots, \phi_M \rangle \quad (21)$$

Where, ϕ_j is the better centroid and M is the number of reduced task.

E. Proposed Job Scheduling

The Inertia Weight based Grey Wolf Optimization (IW-GWO) is utilized and M number of tasks ($\phi_j = \langle \phi_1, \phi_{2,} \dots, \phi_M \rangle$) are scheduled with minimum cost and execution time. The inertia weight strategy is utilized for controlling the exploration and population growth capability which effectively enhances the algorithm search capability and balance the connection between local development and global exploration. The GWO algorithm displays iterative optimization of grey wolves, present position individuals, and historical optimum position information (α, β, δ) of groups are taken and the search is shown through the position of α, β and δ. The position updating of GWO ignores the exploration capability and easily falls into the local optimum solution [25]. Because of this, the wolves are based on α, β and δ for justice the prey distance and the speed of convergence are low in the upcoming stage.

To improve the exploration capability and convergence speed, this paper proposed an IW-GWO algorithm. The evaluation of inertia weights according to the performance of real-time population in an iterative procedure. At every iteration, each gray wolf in the population, inertia weight scores are adjusted through the random factor, minimum, maximum, and average fitness score of the exponential function in the population. The inertia weight is mathematically presented in Eq. (22),

$$\varpi^t = \frac{f_{avg} - f_{min}}{f_\alpha - f_{avg}} \cdot e^{-k} \cdot r_1, \quad f_\alpha - f_{avg} \neq 0 \quad (22)$$

Where, ϖ^t is inertia weight at iteration$t$, f_α, f_{avg} and f_{min} is the maximum, average and minimum fitness scores of present populations correspondingly. k is the random number in the range of [0,1] and r_1 follows uniform distribution of [0,1]. This inertia weight strategy is presented for enhancing wolf pack position updating to exchange actual behavior which is formulated as Eq. (23),

$$X(t+1) = \varpi(t) \cdot \frac{(X_1(t) + X_2(t) + X_3(t))}{3} \quad (23)$$

Initially, the iterative procedure of fitness score in every individual gray wolf is estimated and arranged. According to the maximum, average and minimum fitness scores of individual populations, the inertia weight score was dynamically adjusted and the variation of population fitness was considered for improving the optimization accuracy. Next, the inertia weights of GWO were affected through the exponential function. Compared with GWO, the inertia minimizes quickly at following iterations which highly enhances the local development capability and obtains the best optimization. Lastly, the inertia weight of GWO was affected by random factors that assist in managing the diverse population to a certain extent.

3. Experimental Results

The IW-GWO is simulated by utilizing a Python with the system configuration: 16GB-RAM, Intel core i7-processor, and Windows 10-operating system. The IW-GWO performance is estimated based on the metrics of makespan, throughput, execution time and cost, presented in Eqs. (24), (25), (26), and (27).

$$Makespan = \sum_{i=1}^{n} \sum_{j=1}^{m} E_{ij} X_{ij} \quad (24)$$

$$Throughput = \sum_{t=0}^{m} \frac{Completed\ task}{T} \quad (25)$$

$$Execution\ time = \sum_{t=0}^{m} End\ execution\ time(t) - \frac{Start\ time\ (t)}{m} \quad (26)$$

$$Cost = \sum_{j=1}^{n} \sum_{i=1}^{m} Cost(t_{ji}) \quad (27)$$

Where, E_{ij} and X_{ij} is the execution time and boolean representation of the task. The n and m is the number of tasks running in the system and $Cost(t_{ji})$ is the predictable cost when t_j is handled at v_i.

A. Quantitative and Qualitative Analysis

The IW-GWO performance is evaluated by four various metrics namely, makespan, throughput, cost and execution time and varied number of nodes 100-500, taken for examining the IW-GWO. The Cat Swarm Optimization (CSO), Whale Optimization Algorithm (WOA), Ant Colony Optimization (ACO) and GWO are used for estimating performance.

Table I and Fig. 2 display the performance of IW-GWO with makespan, measured under varied number of tasks. The proposed model attains a superior makespan of 2.37s, 2.82s, 4.86s, 6.95s and 8.49s for 100, 200, 300, 400 and 500 nodes, respectively. The proposed IW-GWO model shows best performance than CSO, WOA, ACO and GWO for various nodes.

TABLE I. THE PERFORMANCE OF IW-GWO IN MAKESPAN AND THROUGHPUT

No. of nodes	Performance Metrics	CSO	WOA	ACO	GWO	Proposed IW-GWO
100	Makespan (s)	9.53	7.62	5.88	4.79	2.37

	Throughput (task/s)	28.92	32.45	35.72	40.57	45.91
200	Makespan (s)	13.68	10.24	8.93	5.70	2.82
	Throughput (task/s)	42.61	45.73	49.45	55.45	59.75
300	Makespan (s)	15.34	12.73	10.65	6.36	4.86
	Throughput (task/s)	47.92	52.51	56.59	60.43	64.53
400	Makespan (s)	18.29	15.82	12.51	9.77	6.95
	Throughput (task/s)	61.79	65.48	69.81	72.56	76.39
500	Makespan (s)	23.75	19.58	15.83	11.64	8.49
	Throughput (task/s)	69.39	73.67	77.66	82.49	85.42

Table I and Fig. 3 display the performance of the IW-GWO with throughput, measured under various number of tasks. The proposed model attains commendable throughput of 45.91 tasks/s, 59.75 tasks/s, 64.53 tasks/s, 76.39 tasks/s and 85.42 tasks/s respectively for 100, 200, 300, 400 and 500 nodes. The IW-GWO shows best performance than the CSO, WOA, ACO and GWO for various nodes.

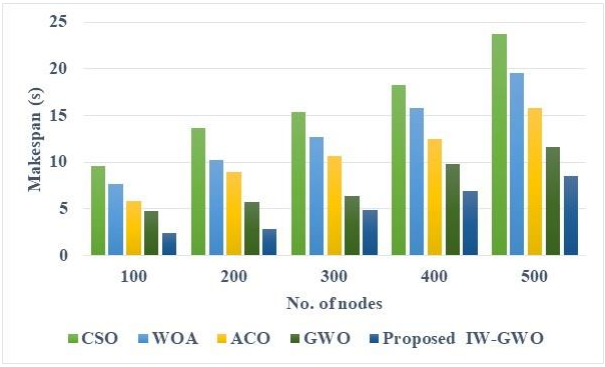

Fig. 3. The performance of IW-GWO in Makespan

Table II and Fig. 4 demonstrate the IW-GWO performance with metrics of cost, measured under various number of tasks. The proposed model achieves a preferable cost of 45$, 52$, 60$, 55$ and 70$ simultaneously for 100, 200, 300, 400 and 500 nodes. The IW-GWO model shows better performance than the CSO, WOA, ACO and GWO for various nodes.

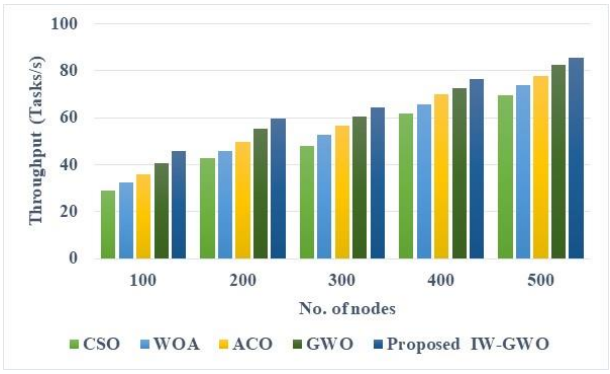

Fig. 2. The performance of IW-GWO in Throughput

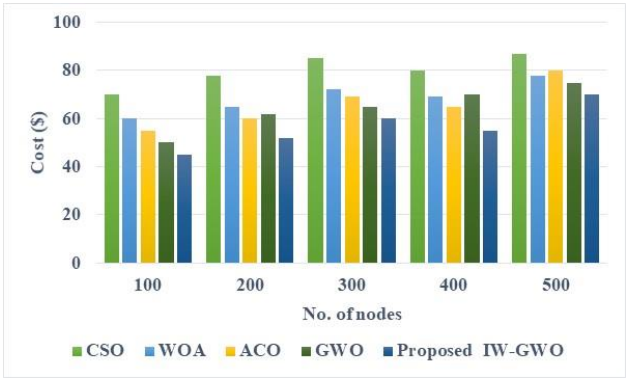

Fig. 4. The performance of the IW-GWO in Cost

TABLE II. THE PERFORMANCE OF IW-GWO IN COST AND EXECUTION TIME

No. of nodes	Performance Metrics	CSO	WOA	ACO	GWO	Proposed IW-GWO
100	Cost ($)	70	60	55	50	45
	Execution time (ms)	192.71	245.92	363.86	240.62	169.58
200	Cost ($)	78	65	60	62	52
	Execution time (ms)	520.55	559.73	458.42	395.51	381.62
300	Cost ($)	85	72	69	65	60
	Execution time (ms)	568.49	580.44	520.87	470.64	446.29

400	Cost ($)	80	69	65	70	55
	Execution time (ms)	720.92	680.39	650.93	665.69	639.46
500	Cost ($)	87	78	80	75	70
	Execution time (ms)	865.83	870.65	846.93	880.63	823.75

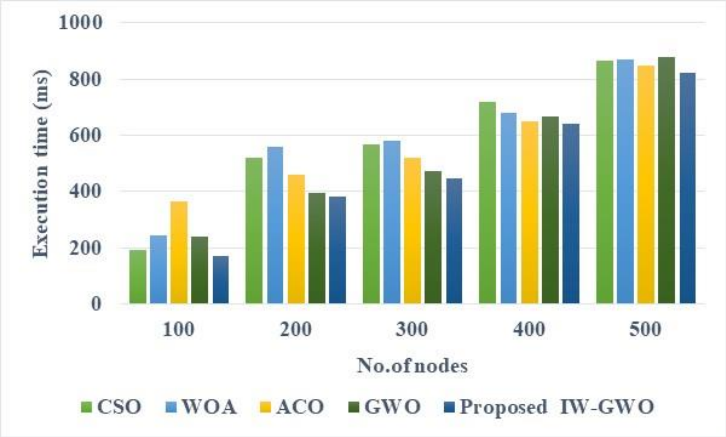

Fig. 5. The performance of IW-GWO in Execution time

Table II and Fig. 5 represent the IW-GWO performance with metrics of execution time, measured under various number of tasks. The proposed model achieves commendable execution time of 169.58ms, 381.62ms, 446.29ms, 639.46ms, and 823.75ms simultaneously for 100, 200, 300, 400 and 500 nodes. The proposed IW-GWO model demonstrates preferable performance than the CSO, WOA, ACO and GWO for various nodes.

B. Comparative Analysis

The comparative analysis of the proposed IW-GWO is illustrated in this section with evaluation metrics of makespan, throughput, cost and execution time as shown in Table III. The existing result such as MOABCQ_LJF [21], HWACO [22] and IMOMVO [23] are utilized to evaluate the ability of classifier for 200 nodes. The MOABCQ_LJF [21] method attains makespan of 3.78s, throughput of 55.28task/s and cost of 70$. The HWACO [22] method attains makespan of 25.9s, cost of 61$ and the IMOMVO [23] method attains execution time of 385.34ms. By adjusting inertia weight, the IW-GWO adapts its search behavior leading to faster convergence towards optimal solutions. The inertia weight strategy is utilized for controlling the exploration and population growth capability which effectively enhances the algorithm's search capability and balance the connection between local development and global exploration. The results obtained from Table III evidence that the proposed IW-GWO attains better performance like a makespan of 2.82s, throughput of 59.74tasks/s, cost of 52$ and execution time of 385.34ms for 200 nodes when compared with the existing methods.

TABLE III. COMPARATIVE ANALYSIS OF IW-GWO WITH EXISTING METHODS

Author	Method	No. of nodes	Makespan (s)	Throughput (tasks/s)	Cost ($)	Execution time (ms)
Kruekaew and Kimpan [21]	MOABCQ_LJF	200	3.78	55.28	70	N/A
Chandrashekar et al. [22]	HWACO		25.9	N/A	61	N/A
Otair et al. [23]	IMOMVO		N/A	N/A	N/A	385.34
Proposed method	IW-GWO		2.82	59.75	52	381.62

1) Discussion

Advantages of the IW-GWO and the drawback of previous techniques are discussed in this section. The previous technique has few drawbacks such as the following: the MOABCQ_LJF [21] model has slow convergence, particularly in complex problems and large solution space. The HWACO [22] model considers smaller number of tasks only, and the performance is affected when enhancing the

number of tasks. The IMOMVO [23] model has local optima issues and poor search scalability. The proposed IW-GWO model overcomes these existing model limitations. The inertia weight strategy is deployed for controlling the exploration and population growth capability, effectively enhancing the algorithm search capability and balance the connection between local development and global exploration. By adjusting inertia weight, the IW-GWO adapts its search behavior, leading to faster convergence towards optimal solutions.

4. Conclusion

Hadoop has emerged as a widely adopted approach for cost-effective data analysis and storage. As an open-source MapReduce framework, it has enabled industries to process an unprecedented volume of data on a daily basis. Numerous scheduling algorithms are created for Hadoop-MR model which varies in behavior and design, managing various issues like user share impartiality, data locality along resource consciousness. Therefore, as a secured and optimized scheduling technique, this paper proposes an innovative algorithm like Inertia Weight-based Grey Wolf Optimization (IW-GWO). The obtained results prove that the proposed IW-GWO attains superior outcomes in terms of makespan, throughput, cost, and execution time of 2.82s, 59.75 tasks/s, 52$ and 381.62ms for 200 nodes respectively, which is comparatively higher than the existing methods like MOABCQ_LJF, HWACO and IMOMVO. In future, the developed model can be deployed in a real-time environment for monitoring the IW-GWO performance.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

For this research work, all authors' have equally contributed in Conceptualization, methodology, validation, resources, writing—original draft preparation, writing—review and editing.

References

[1] T. Dreibholz and S. Mazumdar, "Towards a lightweight task scheduling framework for cloud and edge platform," *Internet Things*, vol. 21, p. 100651, Apr. 2023. https://doi.org/10.1016/j.iot.2022.100651

[2] R. Nath and A. Nagaraju, "Genetic algorithm based on-arrival task scheduling on distributed computing platform," *Int. J. Comput. Appl.*, vol. 44, number 9, pp. 887–896, 2022. https://doi.org/10.1080/1206212X.2021.1974751

[3] Z. Peng, P. Pirozmand, M. Motevalli, and A. Esmaeili, "Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework," *Math. Probl. Eng.*, vol. 2022, p. 4290382, Sep. 2022. https://doi.org/10.1155/2022/4290382

[4] A.G. Gad, E.H. Houssein, M. Zhou, P.N. Suganthan, and Y.M. Wazery, "Damping-Assisted Evolutionary Swarm Intelligence for Industrial IoT Task Scheduling in Cloud Computing," *IEEE Internet Things J.*, vol. 11, number 1, pp. 1698-1710, Jan. 2024. DOI: 10.1109/JIOT.2023.3291367

[5] M. Subramanian, M. Narayanan, B. Bhasker, S. Gnanavel, M.H. Rahman, and C.H.P. Reddy, "Hybrid electro search with ant colony optimization algorithm for task scheduling in a sensor cloud environment for agriculture irrigation control system," *Complexity*, vol. 2022, p. 4525220, Oct. 2022. https://doi.org/10.1155/2022/4525220

[6] R. NoorianTalouki, M.H. Shirvani, and H. Motameni, "A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, number 8A, pp. 4902–4913, Sep. 2022. https://doi.org/10.1016/j.jksuci.2021.05.011

[7] M. Manikandan, R. Subramanian, M.S. Kavitha, and S. Karthik, "Cost Effective Optimal Task Scheduling Model in Hybrid Cloud Environment," *Computer Systems Science & Engineering*, vol. 42, number 3, pp. 935–948, Feb. 2022. DOI: 10.32604/csse.2022.021816

[8] A.-N. Zhang, S.-C. Chu, P.-C. Song, H. Wang, and J.-S. Pan, "Task Scheduling in Cloud Computing Environment Using Advanced Phasmatodea Population Evolution Algorithms," *Electronics*, vol. 11, number 9, p. 1451, Apr. 2022. https://doi.org/10.3390/electronics11091451

[9] C. Li, C. Zhang, B. Ma, and Y. Luo, "Efficient multi-attribute precedence-based task scheduling for edge computing in geo-distributed cloud environment," *Knowl. Inf. Syst.*, vol. 64, number 1, pp. 175–205, Jan. 2022. https://doi.org/10.1007/s10115-021-01627-8

[10] S. Liu, X. Ma, Y. Jia, and Y. Liu, "An energy-saving task scheduling model via greedy strategy under cloud environment," *Wireless Commun. Mobile Comput.*, vol. 2022, p. 8769674, Apr. 2022. https://doi.org/10.1155/2022/8769674

[11] V. Seethalakshmi, V. Govindasamy, and V. Akila, "Real-coded multi-objective genetic algorithm with effective queuing model for efficient job scheduling in heterogeneous Hadoop environment," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, number 6B, pp. 3178–3190, Jun. 2022. https://doi.org/10.1016/j.jksuci.2020.08.003

[12] S. Gupta, S. Iyer, G. Agarwal, P. Manoharan, A.D. Algarni, G. Aldehim, and K. Raahemifar, "Efficient Prioritization and Processor Selection Schemes for

HEFT Algorithm: A Makespan Optimizer for Task Scheduling in Cloud Environment," *Electronics*, vol. 11, number 16, p. 2557, Aug. 2022. https://doi.org/10.3390/electronics11162557

[13] M. Yadav and A. Mishra, "An enhanced ordinal optimization with lower scheduling overhead based novel approach for task scheduling in cloud computing environment," *J. Cloud Comput.*, vol. 12, p. 8, Jan. 2023. https://doi.org/10.1186/s13677-023-00392-z

[14] Y. Lu, L. Liu, J. Gu, J. Panneerselvam, and B. Yuan, "EA-DFPSO: An intelligent energy-efficient scheduling algorithm for mobile edge networks," *Digital Commun. Networks*, vol. 8, number 3, pp. 237–246, Jun. 2022. https://doi.org/10.1016/j.dcan.2021.09.011

[15] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," *J. Cloud Comput.*, vol. 12, p. 24, Feb. 2023. https://doi.org/10.1186/s13677-023-00401-1

[16] X. Wang, C. Wang, M. Bai, Q. Ma, and G. Li, "HTD: heterogeneous throughput-driven task scheduling algorithm in MapReduce," *Distrib. Parallel Databases*, vol. 40, number 1, pp. 135–163, Mar. 2022. https://doi.org/10.1007/s10619-021-07375-6

[17] N. Sharma, Sonal, and P. Garg, "Ant colony based optimization model for QoS-Based task scheduling in cloud computing environment," *Meas.: Sens.*, vol. 24, p. 100531, Dec. 2022. https://doi.org/10.1016/j.measen.2022.100531

[18] R. Jeyaraj and A. Paul, "Optimizing MapReduce Task Scheduling on Virtualized Heterogeneous Environments Using Ant Colony Optimization," *IEEE Access*, vol. 10, pp. 55842–55855, May 2022. DOI: 10.1109/ACCESS.2022.3176729

[19] S. Mangalampalli, G.R. Karri, and M. Kumar, "Multi objective task scheduling algorithm in cloud computing using grey wolf optimization," *Cluster Comput.*, vol. 26, number 6, pp. 3803–3822, Dec. 2023. https://doi.org/10.1007/s10586-022-03786-x

[20] S.P. Praveen, H. Ghasempoor, N. Shahabi, and F. Izanloo, "A hybrid gravitational emulation local search-based algorithm for task scheduling in cloud computing," *Math. Probl. Eng.*, vol. 2023, p. 6516482, Feb. 2023. https://doi.org/10.1155/2023/6516482

[21] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803–17818, Feb. 2022. doi: 10.1109/ACCESS.2022.3149955

[22] C. Chandrashekar, P. Krishnadoss, V.K. Poornachary, B. Ananthakrishnan, and K. Rangasamy, "HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing," *Appl. Sci.*, vol. 13, number 6, p. 3433, Mar. 2023. https://doi.org/10.3390/app13063433

[23] 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 Comput.*, vol. 25, number 6, pp. 4221–4232, Dec. 2022. https://doi.org/10.1007/s10586-022-03650-y

[24] W. Long, T. Wu, M. Xu, M. Tang, and S. Cai, "Parameters identification of photovoltaic models by using an enhanced adaptive butterfly optimization algorithm," *Energy*, vol. 229, p. 120750, Aug. 2021. https://doi.org/10.1016/j.energy.2021.120750

[25] K. Li, S. Li, Z. Huang, M. Zhang, and Z. Xu, "Grey Wolf Optimization algorithm based on Cauchy-Gaussian mutation and improved search strategy," *Sci. Rep.*, vol. 12, p. 18961, Nov. 2022. https://doi.org/10.1038/s41598-022-23713-9