Received: April 8, 2023. Revised: June 14, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Cloud Computing for Task Scheduling Using Estimate of Distribution Algorithm – KrillHerd Method

Vasantha Muniyappa[1]* **Chandramouli Hattibelagal[1]**

[1]*Department of Computer Science and Engineering, East Point College of Engineering and Technology, Bengaluru-560049, Karnataka, India*
* Corresponding author's Email: vasantha.nhce@gmail.com

Abstract: Cloud computing is a popular technology that allows customers to use computing resources remotely on a pay-as-you-go basis. Task scheduling is one of the significant problems in cloud computing backgrounds since tasks must be scheduled properly to reduce implementation time and cost while optimizing resource efficiency. In this research, an estimate of the distribution algorithm for the Krill Herd (EDA-KrillHerd) method for multi-objective scheduling tasks in cloud computing is developed and compared with the particle swarm optimization (PSO) and Krill Herd algorithms. Most of the existing methods do not use EDA to the large potential in solving task scheduling problem resulting in high completion time of tasks. The main objective of this work focuses on effectively using EDA combined with KrillHerd algorithm to reduce the task completion time within task scheduling algorithms. The findings indicate that the proposed EDA-Krill Herd algorithm excels in terms of time efficiency and faster convergence when it comes to task scheduling, in comparison to the existing methods. Furthermore, in the presence of both small and large-scale activities, the EDA-Krill herd algorithm has achieved greater efficiency on Makespan of 1000.74s, throughput of 64.30%, and resources utilization of 99.90% respectively.

Keywords: Cloud computing, EDA-krill herd, Makespan, Multi-verse optimizer, Task scheduling, Virtual machines.

1. Introduction

The exponential growth of internet information processing resulted in the development of cloud computing systems. Cloud computing is essential for providing technological services through the internet. It delivers a resource to users, such as processing power, and data storage without requiring direct active control [1]. The concept of cloud computing was first proposed by Google. In later developments, Amazon, Microsoft and the Apache Foundation increased their research of cloud computing, and academia has further studied the theory of cloud computing [2]. Cloud computing is the most recent technological development, providing for the processing of large amounts of data. Cloud computing is an effective technique for addressing the requirements of large data applications. [3]. The cloud environment is a difficult system with numerous shared resources and unpredictability, and it is affected by unexpected external events [4].

Cloud computing consists of a variety of computing resources and data centers that receive a variety of tasks for execution every minute. The scheduling algorithm must select the optimal VM for the task depending on its specific requirements [5]. Cloud task scheduling is a non-deterministic exponential time-hard problem; finding an improved task scheduling solution in a multi-cloud context is challenging. [6]. In local scheduling tasks, a task scheduling ordering technique is employed initially to priority user tasks based on restrictions such as task flexibility and task life duration. [7]. Scheduling algorithms in cloud computing can have a direct impact on a system's resource utilization and operational costs. To increase the efficiency of cloud task executions, several metaheuristic algorithms and variants have been developed to optimize the scheduling [8]. A task scheduling method based on game theory is developed for large data in cloud

computing, and experiments demonstrate that the approach could enhance cloud computing energy management. [9]. However, scheduling a task is a critical issue in the cloud computing environment since it has a significant impact on system performance. It requires an upgraded, efficient algorithm to serve the work scheduling process [10]. The contribution of the work is as follows,

- To improve the efficiency of task executions in a cloud computing system, the EDA-Krill herd algorithm was proposed for multi-objective task scheduling in cloud computing to solve the problem.
- A new approach called EDA- Krill herd is proposed for efficient task scheduling by incorporating advanced optimization strategies to improve both convergence speed and accuracy.
- The design and implementation of EDA-Krill herd and compare it with some existing approaches including EMVO, MVO, PSO, and Krill herd. The experimental results demonstrate that EDA- Krill herd algorithm can achieve better performance on Makespan, Throughput, and Resource utilization for scheduling the tasks.

The rest of this paper is organized as follows. Section II described the related work. Section III described the proposed EDA-Krill herd approach and its implementation details in section IV. The conclusion of this research is given in section V.

2. Related works

Kusuma and Dinimaharawati [11] proposed a new metaheuristic algorithm known as a fixed-step average and subtraction-based optimizer (FS-ASBO) which is an improved version of the average and subtraction-based optimizer (ASBO). This was developed to replace the randomized step size in the guided movement with the fixed step size, to add an exploration mechanism after the guided movement in every iteration when the new candidate fails to find a better solution. However, for the purpose of solving a combinatorial problem, this proposed algorithm still has to be modified.

To address optimisation issues, Zeidabadi [12] created the mixed leader based optimizer (MLBO). The designed MLBO's goal was to combine the top population member with a random member to create a new member who would act as the algorithm's population's leader. The MLBO's optimisation results have demonstrated that the suggested algorithm is effective at handling a variety of optimisation issues. The future work of this research states that implementing MLBO on real time optimization issues can achieve in major contributions.

Shukri [13] designed an enhanced version of the multi-verse optimizer (EMVO) to minimize the cost and execution time of tasks. The EMVO was utilized to plan work, manage issues, and allocate resources. In terms of reducing makespan time and enhancing resource usage, EMVO significantly improves both PSO and MVO algorithms. However, one of the most difficult difficulties in EMVO was scheduling a task, which requires tasks to be planned to reduce operation cost and time while improving resource usage.

Rajakumari [14] proposed a dynamic weighted round- robin algorithm for improving task scheduling in cloud computing by solving the optimal task scheduling issues. Along with this, a hybrid particle swarm parallel ant colony optimization (HPSPACO) was also proposed to solve the task execution delay problem in deficit weighted round robin (DWRR) based task scheduling. In order to optimise work scheduling, the suggested fuzzy hybrid particle swarm parallel ant colony optimisation (FHPSPACO) on cloud computing reduced execution and waiting times, boosts system throughput, and maximises resource usage. However, the complexity and imprecise data handling are the limitations of this work.

Jamal and Muqeem [15] introduced a novel adaptive strategy that combines the best-worst multi-criteria decision-making (MCDM) with the compromise ranking method (VIKOR). This paper provided a method for mapping user requests to virtual machines (VMs), where many competing factors were taken into account in a cloud scheduling strategy. The limitations such as complex calculation of population, varying criteria will result in improper decision.

Doumari [16] implemented a novel ring toss game-based (RTGBO) for population-based optimisation. The purpose of RTGBO is to imitate player conduct and game rules in the construction of the implemented algorithm. However, for some optimisation situations, the RTGBO algorithm not be appropriate, hence, when solving issues with numerous variables or constraints, it not performs well.

Zeidabadi and Dehghani [17] presented puzzle optimization algorithm (POA) to solve the issues of various optimization and to represent the puzzle-solving process mathematically as an optimizer. The key benefit and characteristic of the POA is, it does not required parameter setting as POA has no

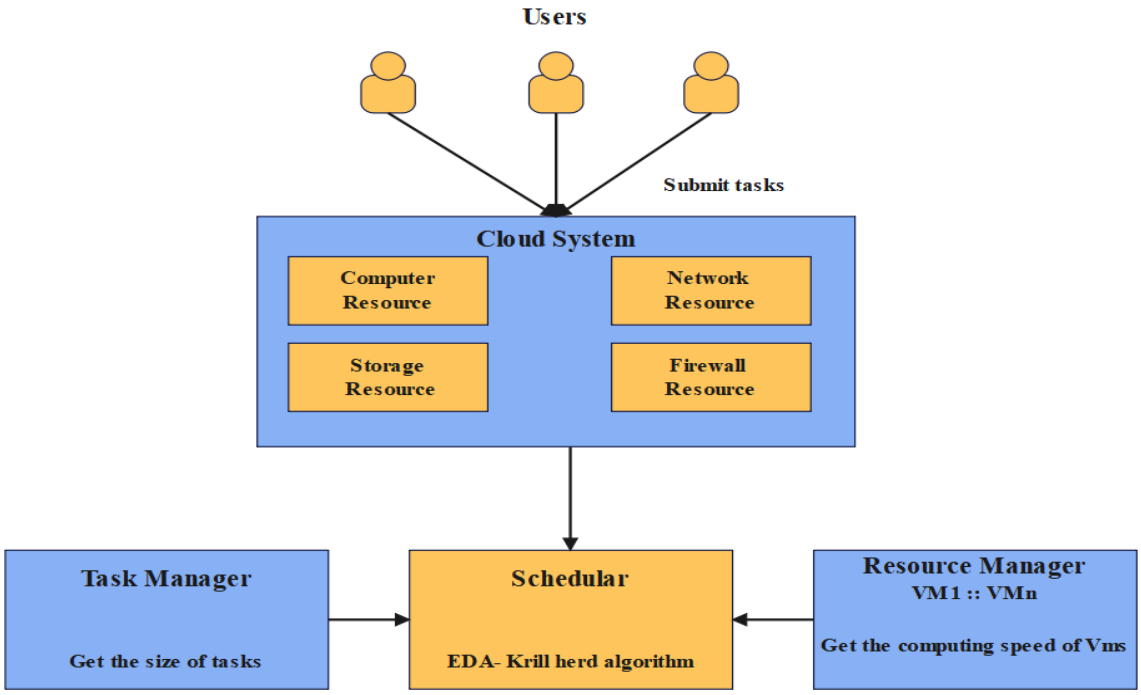

Figure. 1 The architecture of cloud computing for scheduling tasks

parameters of control. However, the POA has a loss of control, lack of flexibility, and misuse of authority.

Low precision, complexity, complex calculation of population, loss of control, lack of flexibility, and misuse of authority are the drawbacks of the existing methods. To overcome these methods an EDA-KrillHerd is proposed in this research.

3. Proposed method for Eda- Krill Herd Algorithm

To improve the performance of time-consuming issues in the existing methods, the EDA-krill herd algorithm is proposed. The EDA- krill herd algorithm was used to decrease the Makespan, throughput, and resources utilization time, which was discussed in the experimental results. Users submit tasks to the cloud system, and the cloud system includes three modules: task manager, resource manager and scheduler. Cloud computing delivers tasks to the taskbar and processed them to collect the data. It handles all virtual machines constantly and collects information for computing speed. The scheduler starts working after acquiring information on tasks provided by the task administrator for faster processing of virtual machines. According to this research, the EDA-krill herd plays a crucial role in the scheduling of virtual machine tasks. The structure of cloud computing for scheduling tasks was illustrated in Fig. 1.

3.1 Mathematical technique

The proposed approach for mathematically expressing the scheduling of n tasks across m virtual machines involves accounting for their differing computation rates. The multi-objective optimization problem addresses many objectives at the same time to identify load balancing and task completion time. The computational time for the suggested methodology was shown mathematically in the Eqs. (1-3).

$$\sum_{j=1}^{m}\sum_{r=1}^{n} x_{i,j,r} = 1, \qquad i = 1,2,\dots n \tag{1}$$

$$\sum_{i=1}^{n} x_{i,j,r} \le 1\ , \qquad j = 1,2,\dots m;\ \forall_r \tag{2}$$

$$\sum_{i_1=1}^{n} x_{i_1} j_{r+1} - \sum_{i_2=1}^{n} x_{i_2} j_r \ \le 0\ , \quad j = 1,2,\dots m;\ \forall_r \tag{3}$$

where Eq. (1) ensures the tasks that are scheduled on VMs and only once; Eq. (2) ensures that each VMs performs only one task at the same time; and Eq. (3) indicates the work on a certain VMs has been performed in a specific sequence.

3.2 Task model for completion time

From the input mathematical model to task model VMs and task size computation speed are identified, the ETC matrices can be considered at Eq. (4).

$$ETC\left(t_i, r_j\right) = \frac{TS_i}{VS_j} \quad (1 \le i \le n, 1 \le j \le m\ ) \tag{4}$$

Where, $\left(t_i, r_j\right)$ indicates i required time to

complete the tasks in virtual machine running time j.

The completion time of the virtual machine to aggregate the running times for all tasks given to it. Each virtual machine's completion time can be determined as Eq. (5).

$$time_j = \sum_{r=1}^{k} ETC\left(t_i, r_j\right) \quad (5)$$

where k represents the task numbers allocated to virtual machine j.

It describes that the entire runtime of total VMs in completion time was similar to cloud computing. The entire time for task completion was considered using Eq. (6).

$$Complete\ Time = \max\{time_1, time_2 \ldots . time_m\} \quad (6)$$

3.3 Model for load balancing

From the input task model to load balancing was to increase the degree of load balancing of the system is defined as Eq. (7).

$$DBL = \frac{\sum_{j=1}^{m} time_j}{M \times Complete\ Time} \quad (7)$$

DBL denotes the load balancing degree. When it comes to the execution of each VMs, has approximately equal total processing times, indicating the load for better balanced. As a result, the DBL was regulated the load with greater load balancing capacity.

3.3.1. Fitness function

From the load balancing to the fitness function, the EDA-Krill herd algorithm's population indicates a viable clarification to the challenge. The fitness function was utilized to assess solution quality, it is essential for avoiding an optimum and attaining the ideal solution. It may create various fitness functions based on the needs of the user. This paper considers load balancing degree and total task completion, the fitness function was determined in Eqs. (8) and (9).

$$GValue = \omega_1 * \frac{1}{Complete\ Time} + \omega_2 * DBL \quad (8)$$

$$\omega_1 + \omega_2 = 1\ (0 \leq \omega_1, \omega_2 \leq 1) \quad (9)$$

where ω_1 and ω_2 are weight coefficients, according to different user requirements weight coefficients can be set. For example, considering the task completion for factor time, set ω_1 and ω_2. The balancing load was considered, set 0 to 1, and two factors are considered consecutively of sets ω_1 and ω_2 correspondingly. The GValue was larger and better for the quality of the resolution [16].

3.4 EDA-Krill Herd hybrid algorithm

To schedule tasks in a computing cloud, the suggested EDA-Krill Herd hybrid algorithm is assigned the responsibility of load balancing. During the experimental setup used to obtain results on a specific scale, all options were set to a fixed value of 1/m for sampling purposes. Simultaneously, GValue recommends evaluating all options and selecting great ones. Second, employ the Krill herd to undertake mutation and crossover procedures on the selected great solutions, resulting in the generation of new resolutions. Finally, sort the outstanding resolutions from step 1 and resolutions from step 2 in descending order. The elite population is made up of the top p% of great answers. Finally, based on the updated probability and elite population model gives finite results to process the data. Run the algorithms in this manner while the halting condition is reached, and the output is the best result. The specific development of the EDA- Krill herd algorithm is mentioned as follows.

3.4.1. EDA operations

The data from the proposed method to EDA operations were classified into initialization, sampling method and fitness assessment described below.

3.4.1.1. Initialization

The distribution of solutions can be better understood and the characteristics of the problem can be more easily reflected through the use of a probability model. The possibility of techniques constructed as shown in Eq. (10).

$$P(g) = \begin{bmatrix} p_{11}(g) & p_{12}(g) & p_{1m}(g) \\ p_{21}(g) & p_{22}(g) & p_{2m}(g) \\ p_{n1}(g) & p_{n2}(g) & p_{nm}(g) \end{bmatrix} \quad (10)$$

In the gth iteration, $P(g)$ describes the mapping connection among m machines and n tasks. To assure the unpredictability of the initial population, all probability values are fixed to 1/m at startup.

3.4.1.2. Sampling method

This coding method encodes the virtual machines occupied by each task, and the length of each

individual is equal to the number of tasks. Each position in the individual represents the task number, and the value in this position represents the virtual machine number assigned to the task. This individual represents the first task assigned to the first virtual machine, the second task assigned to the fourth virtual machine, and the third task assigned to the second virtual machine.

3.4.1.3. Fitness assessment

The allocation of tasks on virtual computers can be achieved in the previous phase, based on the coding output of each task. The individual's fitness value is then calculated, and the tasks are ordered in descending order based on their fitness value.

3.5 Operations of Krill heard algorithm

The data from the proposed method to operations of krill herd the concentration appears to be quite enormous and is often seen at a distance of 10-100 m. The outcome of these arithmetic techniques cannot take into account the migration of krill that developed as Antarctic krill. The fitness characteristic for krill movement is represented based on multiple distance measures, that include the smallest space between each krill and the minimum gap between the largest herd solidity and each krill.

The decision of n-dimensional space is given by Eq. (11).

$$\frac{dX_i}{dt} = N_i + F_i + D_i \tag{11}$$

where N_i is a measure indicated by krill individuals, F_i foraging activity, D_i indiscriminate diffusion, i = 1 to nk number of krill individuals.

The movement of each krill is represented by Eq. (12).

$$N_i^{new} = \left[N^{max} \sum_{j=1}^{NN} \left[\frac{k_i - k_j}{k_{worst} - k_{best}}\right]\left[\frac{X_j - X_i}{X_j - X_{i+\varepsilon}}\right]\left\{2\left(rand + \frac{1}{I_{max}}\right)\frac{1}{k_{1,best}\, X_{1,best}}\right\}\right] + \omega_n N_i^{old} \tag{12}$$

where, k_i is the value of fitness, ith krill creature (i=1 to nk), k_j value of fitness for acquaintance ($j = 1\ to\ NN$), k_{worst} , k_{best} best case and the worst case of each krill, X exact location, ε positive minimum number, N^{max} large induced speed in ms^{-1} , I describe iteration calculation, I_{max} denotes utmost iteration calculation, $k_{1,best}$ rate of finest fitness of ith krill, $X_{1,best}$ position of $X_{1,best}$ of i th krill, ω_n indicates an inertial weight range of (0,1), N_i^{old} denotes final inertial weight from 0 to 1.

The equation for distance sense is described in Eq. (13).

$$d_{s,i} = \frac{1}{5N} \sum_{j=1}^{N} || X_i - X_j || \tag{13}$$

The calculation between the krill folks and formulated distance is obtained. If this gap is less, then they are assumed to be neighbors where N represents the movement induced by each krill, V_f foraging speed, ms^{-1}, ω_f weight inertia in the range (0, 1), f_i^{old} motion of last foraging.

3.6 Updating method

From the input krill herd algorithm to updating method, the possibility techniques were updated using the population-based incremental learning (PBIL) and elite population method was shown in Eqs. (14-15).

$$p_{ij}(g+1) = (1-\lambda)\, p_{ij}(g) + \lambda \frac{1}{E} \sum_{K=1}^{E} I_{ij}^{k}(g) \tag{14}$$

$$I_{ij}^{k}(g) = \begin{cases} 1, & if\ T_i\ on\ V_j\ in\ the\ k-th\ invidual \\ 0, & else \end{cases} \tag{15}$$

where $p_{ij}(g)$ describes the probability of task i allocated to virtual machine $j, \lambda \in (0,1)$ describes the rate learning, E describes the size of population elite ($E = PS \times p\%$), and $I_{ij}^{k}(g)$ describes the pointer function was similar to the k[th] elite population of individual tasks.

4. Experimental results and implementation

The suggested method is given to the experimental results, it contains experimental setup, performances, comparative analysis, and graphical representations and the outcomes of the suggested techniques were described and discussed below.

4.1 Experiment setup

The suggested techniques were implemented using Cloudsim version 3.0.3 and the outcomes were run on a PC with the features such as OS: Windows 10, RAM: 16 GB, Memory: 1TB, Software: NetBeans IDE 8.2 Java 1.8

Table 1 showed the characteristics of the algorithms. The main objective of the suggested algorithm is to discover the optimal sequence of tasks, where all tasks are allocated to VMs and completion time is reduced by utilizing the makespan time.

Table 1. Experimental environment parameters

Type	Parameters	Value
VM	Processor speed Memory Bandwidth Image size PesNumber VMM	1000 512 MB 1000 10000 1 Xen
Host	MIPS Storage VMM Monitor Memory Bandwidth	1000 1000000 Xen 25600 MB 50000
Data Center	Arch Operating system VMM Time_zone Cost Cost Per Memory Cost Per Storage Cost Per Bandwidth	X86 Linux Xen 10.0 3.0 0.05 0.1 0.1
KrillHerd	Size population	50
	Selection for mechanism	Roulette wheel
	Cross over for krillHerd	0.9
	Mutation for Krill Herd	0.2

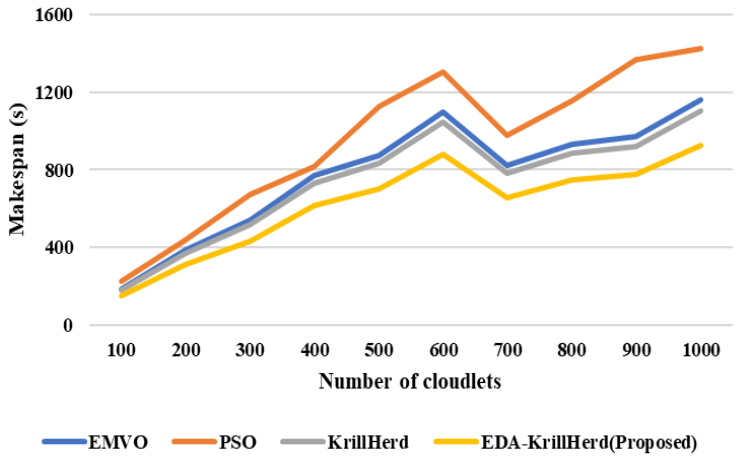

Figure. 2 Makespan time results for VMs=50

4.2 Experimental results

The experimental results showed that results are compared with existing method such as enhanced multi-verse optimizer (EMVO) algorithm as described below.

4.2.1. Quantitative analysis

The makespan time findings for the datasets are represented in Fig. 2 for the EMVO. As compared to previous results, the suggested EDA-Krill herd method achieves a superior make span of 927.02 seconds (s). As a result, the first assumption is that increasing the number of VMs to 50 can minimize the makespan time. In terms of efficiency, it can be shown that the suggested approach has achieved better performances when compared to existing techniques.

Fig. 3 shows the throughput results for this experiment, which reflect the efficiency of the algorithm in processing tasks proportional to the number of active VMs=50. As compared to previous findings, the suggested EDA-Krill herd method achieved a throughput of 68.80%. These findings show that the EDA-KrillHerd fared the best in both

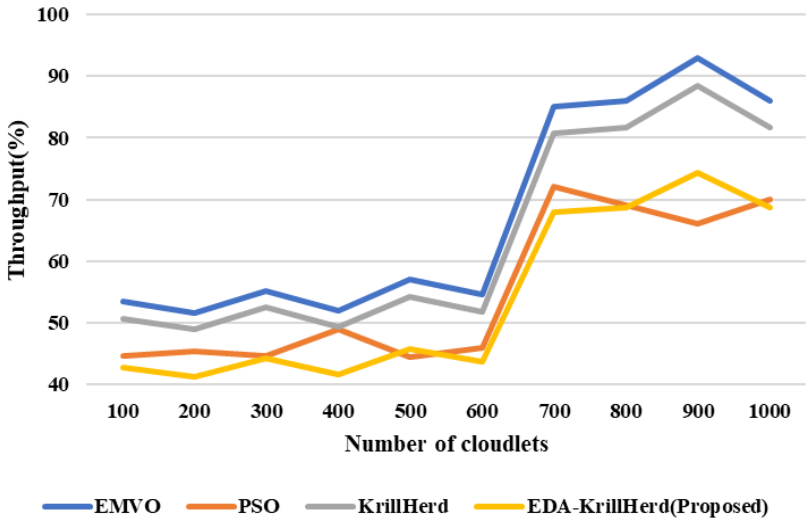

Figure. 3 Throughput results for VM=50

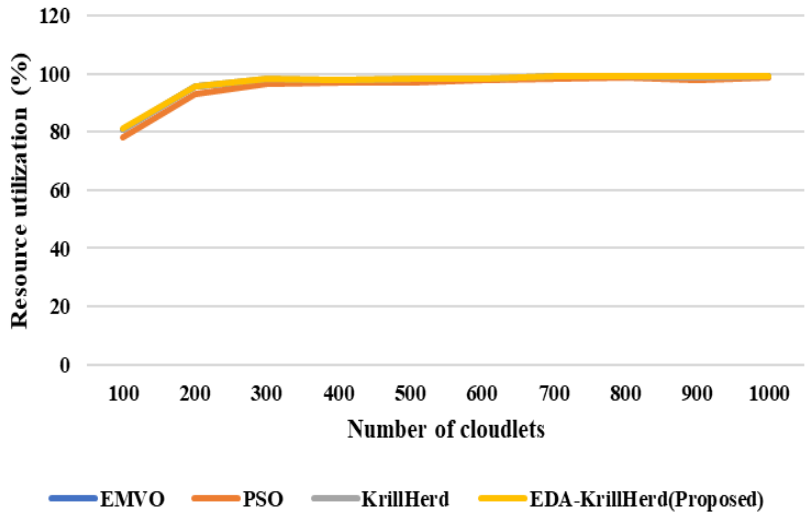

Figure. 4 Resources Utilization results for VMs=50

local and global searches for the optimal solution in terms of minimum makespan time and maximal throughput.

The number of utilization resources in the element is to be calculated in this experiment. The resources indicated by the VMs were utilized, which has VMs=50 for all datasets. MIPS ranged from 100 to 1000 for the VMs used. The average resource use for all datasets is shown in Fig. 4. As compared to previous findings, the suggested EDA-Krill herd algorithm performed well in terms of resource usage, with a range of 99.16.

This experiment was utilized to reduce the makespan time results, since this reduced execution and waiting time for a better scheduling technique. Fig. 5 shows the makespan time findings for the three techniques on regular-size datasets. By increasing the number of tasks, the makespan time was reduced to 1006.7 (s). It can also be observed that the EDA-KrillHerd method performed well when compared to EMVO methods in all datasets for a given number of VMs.

The throughput value is the second criterion for evaluation because more throughput equates to greater efficiency. The purpose of the tasks on better throughput VMs. Throughput time findings for regular-size datasets utilizing a varying number of VMs were shown in Fig. 6. The throughput time was reduced by increasing the number of tasks to 64.30 (%).

Fig. 7 shows that algorithms gave nearly identical results. This means that the number of tasks is equal to the three algorithms that were performed in equal values, with minor changes. By increasing the

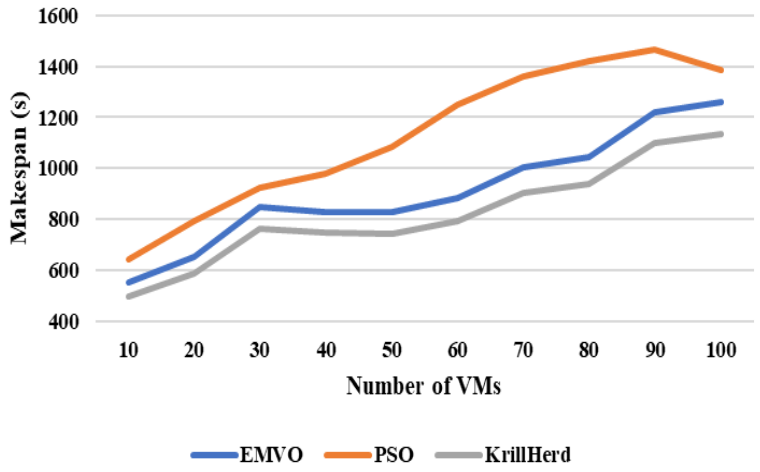

Figure. 5 Results of VMs for Makespan time

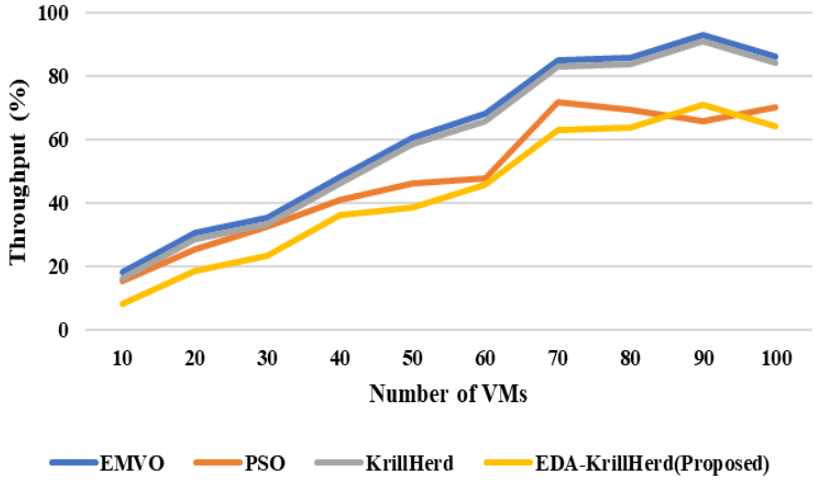

Figure. 6 Throughput time results of VMs

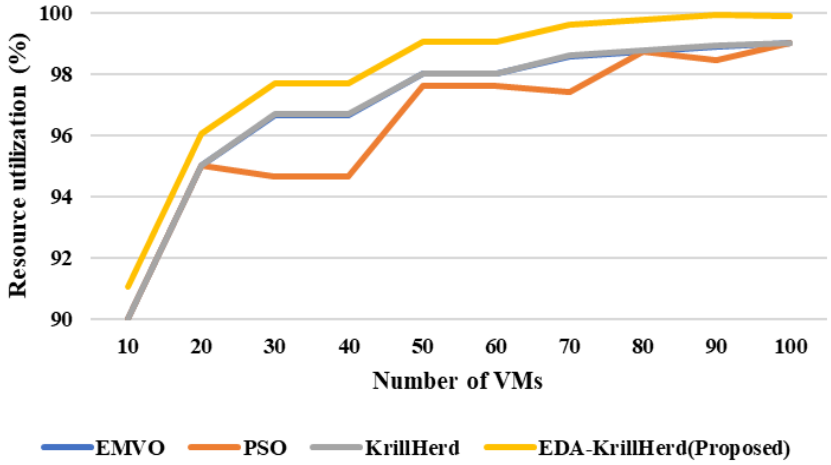

Figure. 7 Resources utilization time results of VMs

number of tasks, the resource time was reduced and 99.90 (%) was achieved.

4.2.2. Comparative analysis

In this section, the proposed method is compared

Table 2. Makespan outcomes of time for datasets using variable number of tasks

Number of tasks	Makespan (s)		
	FHPSPACO [14]	MCDM [15]	EDA-KrillHerd (Proposed)
10	20.00	50.00	18.45
20	22.45	54.00	20.34
30	25.00	59.00	22.46
40	27.56	80.00	25.36
50	30.00	100.00	27.52

Table 3. Throughput outcomes of time for datasets using variable number of tasks

Number of tasks	Throughput (%)	
	FHPSPACO [14]	EDA-KrillHerd (Proposed)
10	7.00	6.25
20	10.00	8.45
30	12.00	10.52
40	15.00	13.30
50	30.00	27.73

Table 4. Resource utilization outcomes of time for datasets using variable number of tasks

Number of tasks	Resource utilization (%)		
	FHPSPACO [14]	MCDM [15]	EDA-KrillHerd (Proposed)
10	93.00	72.00	96.00
20	90.00	76.00	94.00
30	87.00	66.00	91.00
40	84.00	68.00	88.00
50	80.00	88.00	85.00

to two existing techniques such as FHPSPACO [14] and MCDM [15] as given in Table 2. The obtain results shows that the proposed method archives better makespan by varying number of tasks from 10 to 50 compared to references [14, 15].

The FHPSPACO [14] method uses objective function related to resource availability and task parameters only, it doesn't consider objective cost and energy with respect to datacentre and also it uses a simple optimization algorithm of ACO which has lesser convergence over epochs in comparison to existing optimization techniques. The MCDM [15] method considers parameters defined manually which has conflicts over dynamic allocation of task scheduling strategy with respect to practical simulation scenarios to balance the load effectively. These limitations are overcome by the propose method by considering, cost and energy factors along with the appropriate task allocation in practical scenarios.

Table 5. Nomenclature

Terms	Representation
n	Number of tasks
m	Number of virtual machines
$x_{i,j,r}$	represents that task i is the r-th task processed on virtual machine j
$ETC_{n\times m}$	the matrix of size $n \times m$, represents the running time of all the tasks on each virtual machine.
TS_i	The size of task i
VS_j	The computing speed of virtual machine j
k	Total number of tasks assigned to virtual machine j

The throughput of the proposed method is compared to FHPSPACO [14] as shown in Table 3. The proposed method achieves better throughput compared to ref [14].

The resource utilization of the proposed method is compared to FHPSPACO [14] and MCDM [15] as shown in Table 4. The proposed method uses highest resources than the other task scheduling methods.

5. Conclusion

In this research, an EDA-Krill Herd method for multi-objective scheduling tasks in cloud computing was developed. The EDA-Krill Herd was then used to map some tasks into the required number of VMs to decrease makespan time, enhance throughput, and increase resource usage. The EDA-Krill Herd was utilized to schedule work, handle issues, and allocate resources and used to improve task scheduling in the cloud. The performance of the EDA-Krill Herd method was compared to the FHPSPACO and MCDM scheduling algorithms. The results demonstrate that the EDA-Krill Herd mapped workloads successfully with no extra overheads. As compared to existing techniques, EDA-Krill shows better performance and has a shorter computational time in Makespan of 1000.74s, throughput of 64.30%, and resource utilization of 99.90% respectively. In the future, task scheduling in cloud computing must concentrate on better scheduling techniques based on multi-objective functions and add some more parameters to improve the performance of the cloud scheduling system. The representation of terms which are used in the mathematical models are given in Table 5.

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] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing", *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 6A, pp. 2370-2382, 2022.

[2] B. Liang, X. Dong, Y. Wang, and X. Zhang, "A low-power task scheduling algorithm for heterogeneous cloud computing", *The Journal of Supercomputing*, Vol. 76, No. 9, pp. 7290-7314, 2020.

[3] X. Huang, C. Li, H. Chen, and D. An, "Task scheduling in cloud computing using Particle Swarm Optimization with time varying inertia weight strategies", *Cluster Computing*, Vol. 23, No. 2, pp. 1137-1147, 2020.

[4] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing", *Neural Computing and Applications*, Vol. 33, No. 19, pp. 13075-13088, 2021.

[5] K. R. P. Kumar and K. Kousalya, "Amelioration of task scheduling in cloud computing using crow search algorithm", *Neural Computing and Applications*, Vol. 32, No. 10, pp. 5901-5907, 2020.

[6] Q. H. Zhu, H. Tang, J. J. Huang, and Y. Hou, "Task Scheduling for Multi-Cloud Computing Subject to Security and Reliability Constraints", *IEEE/CAA Journal of Automatica Sinica*, Vol. 8, No. 4, pp. 848-865, 2021.

[7] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing", *Future Generation Computer Systems*, Vol. 108, pp. 361-371, 2020.

[8] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, and J. Murphy, "A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems", *IEEE Systems Journal*, Vol. 14, No. 3, pp. 3117-3128, 2020.

[9] J. Yang, B. Jiang, Z. Lv, and K. K. R. Choo, "A task scheduling algorithm considering game theory designed for energy management in cloud computing", *Future Generation Computer Systems*, Vol. 105, pp. 985-992, 2020.

[10] S. Velliangiri, P. Karthikeyan, V. M. A. Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing", *Ain Shams Engineering Journal*, Vol. 12, No. 1, pp. 631-639, 2021.

[11] P. D. Kusuma and A. Dinimaharawati, "Fixed Step Average and Subtraction Based Optimizer", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, pp. 339-351, 2022.

[12] F. A. Zeidabadi, S. A. Doumari, M. Dehghani, and O. P. Malik, "MLBO: mixed leader based optimizer for solving optimization problems", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 4, pp. 472-479, 2021.

[13] S. E. Shukri, R. A. Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments", *Expert Systems with Applications*, Vol. 168, p. 114230, 2021.

[14] K. Rajakumari, M. V. Kumar, G. Verma, S. Balu, D. K. Sharma, and S. Sengan, "Fuzzy Based Ant Colony Optimization Scheduling in Cloud Computing", *CSSE-Computer Systems Science and Engineering*, Vol. 40, No. 2, pp. 581-592, 2022.

[15] M. K. Jamal and M. Muqeem, "An MCDM optimization based dynamic workflow scheduling used to handle priority tasks for fault tolerance in IIOT", *Measurement: Sensors*, Vol. 27, p. 100742, 2023.

[16] S. A. Doumari, H. Givi, M. Dehghani, and O. P. Malik, "Ring Toss Game- Based Optimization Algorithm for Solving Various Optimization Problems", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 3, pp. 545-554, 2021.

[17] F. A. Zeidabadi and M. Dehghani, "POA: Puzzle Optimization Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 1, pp. 273-281, 2022.