The Scientific Temper (2025) Vol. 16 (5): 4220-4231
E-ISSN: 2231-6396, ISSN: 0976-8653
Doi: 10.58414/SCIENTIFICTEMPER.2025.16.5.07
https://scientifictemper.com/

RESEARCH ARTICLE

# Bradley Terry Brownboost and Lemke flower pollinated resource efficient task scheduling in cloud computing

**A. Jabeen[1*], AR Mohamed Shanavas[2]**

## Abstract

Cloud computing (CC) is extensively used across various domains, yet task and resource scheduling still demand significant improvement. In heterogeneous computing systems, effective task scheduling ensures optimal task-machine mapping, reducing makespan and enhancing resource utilization. One major challenge in cloud data centers is managing vast user requests while maintaining efficient scheduling. This work introduces the Bradley–Terry BrownBoost and Lemke flower pollinated resource optimization (BTB-LFPRO) method to enhance task scheduling and improve performance. The BTB-LFPRO approach includes two main steps: classification and optimization. First, the Bradley–Terry BrownBoost Classifier categorizes tasks into high- and low-priority based on pairwise comparisons. Then, the Lemke flower pollinated resource optimization algorithm selects the optimal virtual machine using swarm intelligence. This algorithm balances global exploration and local exploitation via Lévy flights to find the best scheduling path. Experimental results demonstrate that the BTB-LFPRO method significantly improves task scheduling efficiency by 24% and enhances throughput by 24%, outperforming existing techniques.

**Keywords**: Cloud computing, Bradley–Terry BrownBoost, Task scheduling, Lemke flower pollinated, Resource optimization

## Introduction

A ε-fuzzy dominance-based reliable green workflow scheduling (FDRGS) algorithm was designed (Rani, R. & Garg, R., 2022) to optimize application reliability and energy consumption. The fast Fourier transform (FFT) and Gaussian elimination (GE) task graphs reduced energy consumption and improved system reliability. Though the system reliability was improved, the computational complexity was not minimized by FDRGS. A fruit fly-based simulated annealing optimization scheme (FSAOS) was designed by Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z. (2022) to ensure efficient resource allocation in mobile edge-clouds. In the designed scheme, simulated annealing was integrated to ensure a balance between global and local search, thereby addressing premature convergence. A trade-off factor was employed with application owners to determine the optimal service quality that would minimize execution costs. However, the task scheduling efficiency was not improved by FSAOS.

A semi-dynamic real-time task scheduling algorithm was introduced in (Abohamama, A. S., El-Ghamry, A., & Hamouda, E., 2022) for bag-of-tasks applications in cloud–fog environment. The scheduling algorithm addressed the permutation-based optimization issues. The genetic algorithm was employed to provide different permutations for arrived tasks at each scheduling round. The tasks were allocated to a virtual machine with sufficient resources and minimum expected execution time. However, the task scheduling time was not reduced by a semi-dynamic real-time task scheduling algorithm. A new contract-based resource-sharing model was introduced by Xu, J., & Palanisamy, B. (2018) for federated geo-distributed clouds to perform efficient resource sharing contracts with individual data centers for specific time intervals. Individual CSPs used

[1]Research Scholar, PG and Research Department of Computer Science, Jamal Mohamed College (Autonomous), Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India.

[2]Department of Computer Science, Jamal Mohamed College (Autonomous), (Affiliated to Bharathidasan University), Thiruchirappalli, India

***Corresponding Author:** A. Jabeen, Research Scholar, PG and Research Department of Computer Science, Jamal Mohamed College (Autonomous), Affiliated to Bharathidasan University), Tiruchirappalli, Tamil Nadu, India., E-Mail: jabeen.ca@cauverycollege.ac.in

**How to cite this article:** Jabeen, A., Shanavas, A.R.M. (2025). Bradley Terry Brownboost and Lemke flower pollinated resource efficient task scheduling in cloud computing. The Scientific Temper, **16**(5):4220-4231.

Doi: 10.58414/SCIENTIFICTEMPER.2025.16.5.07

**Source of support:** Nil

**Conflict of interest:** None.



**Received:** 23/03/2025 **Accepted:** 25/04/2025 **Published**: 31/05/2025

contracts, cost, and duration-aware job scheduling and provisioning algorithms to complete and meet response time requirements. However, the computational cost was not reduced by the designed model.

Due to its coherence, effectiveness, and significance in global optimum identification, electric fish optimization (EFO) has received a great amount of attention as a metaheuristic model for addressing optimization issues. In Kumar, M. S., & Kumar, G. R. (2023), an electric fish optimization technique was proposed to address scheduling for cloud tasks with the objective of minimizing power utilization and turnaround time. By employing this optimization technique, energy consumption improved. Yet another comprehensive review of task scheduling algorithms was presented in Choppara and Mangalampalli (2024).

Despite numerous research works being performed on resource management in cloud computing environments, identifying materials and methods with the objective of enhancing job completion, minimizing costs, and maximizing resources has remained a top priority that has not been thoroughly explored. A novel hybrid method, with the potential to entirely change the game, employing Neural Network Task Classification (N2TC), was proposed by Mittal, P., Kumar, D. S., & Sharma, D. S. (2024). By integrating neural networks with genetic algorithms, both cost reduction and response time were minimized. In Thanka, M. R., Maheswari, P. U., & Edwin, E. B. (2019), a hybrid algorithm was proposed by combining artificial bee and particle swarm optimization. The hybrid algorithm optimized task scheduling by minimizing makespan, cost, and improving resource utilization in an efficient manner.

Due to the flexibility of the cloud, numerous business establishments are establishing additional data centers and transitioning their businesses to cloud technology. Optimal allocation of resources can be achieved through effective scheduling and load balancing to ensure quality of service (QoS) parameters. In Jain and Sharma (2023), a hybridized Firefly Salp Swarm Algorithm was proposed with the objective of improving not only throughput but also significantly minimizing waiting time. With the intention of exploiting cloud resources in an energy-efficient manner while providing extensive services to users, optimal cloud task scheduling is a prerequisite.

A new deadline-aware task scheduling method for cloud environments using the Firefly Optimization Algorithm (FOA) was designed by Ya-meng, B. A. I., Yang, W. A. N. G., and Shen-shen, W. U. (2023). With this design, a significant improvement was said to be achieved in both terms of makespan and power consumption. Yet another multiple-task scheduling algorithm employing regression and quartile range was presented in Singh, S. P., Nayyar, A., Kaur, H., & Singla, A. (2019) to focus on minimizing energy consumption during VM migration. A review of resource scheduling employing machine learning techniques was conducted by Raeisi-Varzaneh, M., Dakkak, O., Habbal, A., and Kim, B. S. (2023).

The overarching objective of this research is to enhance task scheduling performance while significantly improving throughput. A key objective is to predict the priority level of incoming user-requested tasks in a cloud computing environment as needed. To achieve this, we conduct a systematic analysis of the BrownBoost Classifier for user request task scheduling using the Bradley–Terry function in a cloud environment. Moreover, we analyze the requirements and consequences of utilizing quality of service in terms of task scheduling efficiency and makespan using the proposed Lemke-Flower-Pollinated Resource Optimization algorithm.

### *Related works*

The CC environment operates on a pay-for-usage model, where cloud users access services without possessing complete knowledge of distribution policies and hosting specifics. Due to the different types and levels of configurations, task scheduling becomes a predominant issue in the deployment of a significant cloud framework. Data localization for evaluating task scheduling was performed in Li, Y., & Hei, X. (2022) using the Hadoop platform. Despite extensive advantages, the CC environment poses demanding issues that circumvent the submission of significant workflow. In Hai, T., Zhou, J., Jawawi, D., Wang, D., Oduah, U., Biamba, C., & Jain, S. K. (2023), a Heterogeneous Earliest Finish Time (HEFT) algorithm was proposed, fine-tuned to generate enhanced results, according to two stages. In the first stage, ranks were calculated, followed by the selection of empty slots for task scheduling in the second stage. However, one of the significant metrics, energy consumption, was not focused on. To address this issue, a model based on eco-friendly aspects of reinforcement learning was presented in (Wang, Z., Chen, S., Bai, L., Gao, J., Tao, J., Bond, R. R., & Mulvenna, M. D., 2023). By using these mechanisms, in addition to minimizing waiting time for task scheduling, energy consumption was also considerably reduced. Yet another optimal resource allocation-based task scheduling approach, with a main focus on quality of service, was designed by Singh, H., Bhasin, A., and Kaveri, P. R. (2021).

The significance of cloud services is directly influenced by certain other paramount metrics, such as energy utilization and cost factors, that were not taken into consideration by several prevailing techniques. To address on this aspect, an efficient scheduling mechanism employing effective scheduling called the electric earthwork optimization algorithm was proposed in (Kumar, M. S., & Karri, G. R., 2023) that in turn not only scheduled the workload in a heterogeneous manner but also boosted the quality of service (QoS) in a significant manner.

A holistic survey of task scheduling mechanisms employing machine learning was conducted by

Hosseinzadeh, M., Azhir, E., Lansky, J., Mildeova, S., Ahmed, O. H., Malik, M. H., & Khan, F. (2023). Yet another descriptive literature review on efficient scheduling techniques in CC was presented by Sana, M. U., and Li, Z. (2021). However, the load-balancing aspects was not focused. In Muthusamy and Dhanaraj (2023), a dynamic load balancing approach based on Q-learning and reinforcement learning was presented to focus on the training time involved in task scheduling. Yet another work aimed at minimizing task runtime through efficient reinforcement learning was presented in (Jin, C., Han, Y., Deng, Z., Chen, Y., Liu, C., & Huang, J., 2023). Here, using a combinatorial optimization mechanism resulted in outstanding performance. In this context, a hybrid method combining cuckoo search and grey wolf optimization was presented in (Paulraj, D., Sethukarasi, T., Neelakandan, S., Prakash, M., & Baburaj, E., 2023). By using this hybrid mechanism, the success rate was said to be improved to a considerable extent.

The prevailing method, though, worked well; however, it suffered from issues such as resource utilization, increased boot time, and running costs. To address on these issues, a task scheduling model employing dynamic decision was proposed in (Ali, A., & Iqbal, M. M., 2022). Here, computational offloading ensures cost minimization, in addition to considerably reducing average boot time. Yet another profit maximization algorithm, employing simulated annealing and particle swarm optimization, was presented in Yuan, H., Bi, J., Tan, W., & Li, B. H. (2016). By using this hybrid optimization mechanism, throughput was increased considerably. A resource optimization algorithm with the minimum cost was proposed in Amer, A. A., Talkhan, I. E., Ahmed, R., & Ismail, T. (2022). A modified genetic algorithm employing integer coding was designed by Sun, F., Hou, F., Cheng, N., Wang, M., Zhou, H., Gui, L., and Shen, X. (2018) to ensure low latency and improve system stability. Despite improvements observed in terms of latency and stability, however, the energy efficiency aspects were not focused. To address this issue, a multi-access edge computing system was designed (Pliatsios, D., Sarigiannidis, P., Lagkas, T. D., Argyriou, V., Boulogeorgos, A. A., & Baziana, P., 2022). Here, resource offloading was introduced with the objective of minimizing total energy consumption.

A combination of local and global search strategies was developed in (Senthilkumar, G., Suvarnamukhi, B., Lekashri, S., & Mohammed Thaha, M., 2024) using an interactive school optimization algorithm. By using this optimization algorithm, the makespan was considerably reduced and the throughput was significantly improved. Yet another hybrid method, combining particle swarm optimization for allocating optimal resources and fuzzy inference to ensure task scheduling, was presented in Mansouri, N., Zade, B. M. H., & Javidi, M. M. (2019). However, energy efficiency was not ensured. An energy-efficient task scheduling algorithm using HEFT was proposed in Tang, Z., Qi, L., Cheng, Z., Li, K., Khan, S. U., & Li, K. (2016). Hazard Regressive Multipoint Elitist Spiral Search Optimization for Resource-Efficient Task Scheduling technique (HRMESSO) provides better scheduling efficiency, a lesser makespan, and reduced energy consumption (Jabeen, A., & Shanavas, A. R. M., 2024).

Except for the above works, most other research only concentrate on either lessening the training time or minimizing the makespan. The objectives of most existing task scheduling algorithms in cloud computing (CC) are to reduce the makespan without focusing on energy consumption. Unlike the aforementioned research, our optimal task-scheduling algorithm aims to minimize energy consumption. The elaborate description of the Bradley–Terry BrownBoost Lemke Flower Pollinated Resource Optimization (BTB-LFPRO) method is provided in the following sections.

## Methodology

The proposed Bradley–Terry BrownBoost Lemke flower pollinated resource optimization (BTB-LFPRO) method involves components such as cloud users that send access to certain tasks, a cloud service provider, and cloud servers. Figure 1 shows the proposed Bradley–Terry BrownBoost Lemke Flower Pollinated Resource Optimization (BTB-LFPRO) method. The figure shows the architecture of the BTB-LFPRO method.

As illustrated in the above figure, the system is designed such that the proposed BTB-LFPRO method is fed to the cloud service provider. Access requests for certain tasks are sent to the CC environment by cloud users. Here, numerous cloud users send the requests simultaneously. The cloud service provider evaluates these requests and the allocation of access requests to tasks is done based on the proposed Bradley–Terry BrownBoost algorithm. Access requests to tasks are processed based on a priority mechanism, and the selected requests are allocated to the respective cloud users and stored in the cloud server for further processing. Second

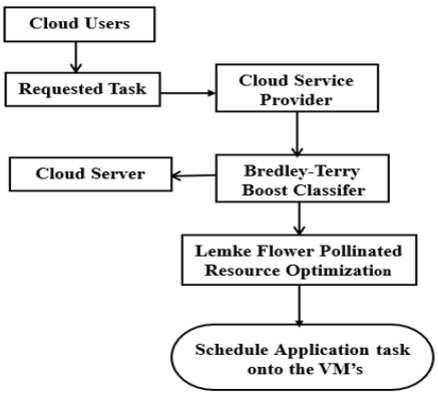


**Figure 1:** Block diagram of Bradley–Terry BrownBoost Lemke Flower Pollinated Resource Optimization (BTB-LFPRO) method

on the major issues in the CC environment is resource scheduling. Resource scheduling refers to the procedures of distributing scheduled tasks onto available virtual machines in CC environment. Due to the constraint on the number of VMs and their distinct potentialities, an effective resource scheduling model called Lemke-Flower-Pollinated Resource Scheduling is designed to meticulously schedule application tasks onto subsequent VMs.

### *Cloud application model*

According to the above explanation of the proposed Bradley–Terry BrownBoost Lemke Flower Pollinated Resource Optimization (BTB-LFPRO) method, designed according to priority constraints, the task flow with priority constraints in our work is represented by a Directed Acyclic Graph (DAG), defined as follows.

$$G = (T, E, C, W) \quad (1)$$

From the above equation (1), the DAG '$G$' is represented by means of cloud user requested tasks '$T = \{T_1, T_2, \ldots, T_n\}$', Edges between tasks '$E = \{E_{ij} = (T_i, T_j) | T_i, T_j \in T_{ij}\}$', cost '$C = \{C_{ij} | i, j\}$' involved in communicating between tasks '$T_i$' and '$T_j$', and computation cost '$W = \{W_{ij} | i, j\}$' between '$C = \{C_{ij} | i, j\}$' respectively.

### *Bradley–Terry BrownBoost Classifier-based Task Scheduling*

Task scheduling is used to schedule cloud user-requested tasks for optimal resource utilization by assigning specific tasks to definite resources at specific time instances. The issue of task scheduling encompasses two classes of users: cloud providers and consumers. On the one hand, cloud consumers run their tasks to address issues of various scales and complexity levels, and on the other hand, resources from cloud service providers are utilized in executing tasks requested by cloud users.

A priority assignment function is designed to utilize a new log likelihood data structure, known as the binary classification matrix, to assign priority to individual cloud user-requested tasks upon arrival. In addition to this, the binary classification queue implements a unique concept based on the principle of linear combination of weak hypotheses for extracting the cloud user's requested task with the highest priority. This work introduces a parallel algorithm for cloud user-requested task scheduling, in which the priority assignment to tasks and the building of linear combinations of weak hypotheses are executed in parallel, taking into account the non-preemptive and preemptive nature of cloud user-requested tasks.

In this work, the Bradley–Terry BrownBoost Classifier is used to categorize tasks into high-priority and low-priority tasks. The Bradley–Terry model is a probability model used to predict the outcome of priority levels in the BrownBoost Classifier. Based on priority level, the tasks are stored in the queue with minimal memory space consumption. Figure 2 shows the structure of Bradley–Terry BrownBoost Classifier model.

As illustrated in the above figure, the cloud user-requested tasks obtained from the dataset are provided as input to the classifier model to classify the tasks based on priority. First, the input cloud user's requested tasks are subjected to a log-likelihood function, followed by a pairwise comparison to obtain the text-categorized results. Next, the categorized text results are subjected to the BrownBoost classifier to generate strong classifier results, improving the prediction of priority levels and enhancing task scheduling efficiency and makespan.

Let '$Prob$' denote the probability that the task '$T_i$' is preferred in comparison with '$T_j$', then using the Bradley–Terry model, for paired comparisons between '$T_i$' and '$T_j$' the Bradley–Terry probabilistic function is formulated as given below.

$$Prob(T_i \text{ selected over } T_j) = \frac{T_i}{T_i + T_j} \quad (2)$$

From the above equation (2) '$T_i > 0$' denote the overall cloud user requested tasks placed in the cloud server and to be processed via cloud service provider. Let us further assume that the outcomes of all the comparison are independent and in addition let '$Res_{ij}$' denote the frequency of times '$T_i$' is selected over '$T_j$'. Then, the log likelihood of above probability measure takes the following form mathematically represented as given below.

$$\log(prob) \sum \left( Res_{ij} \log \left[ \frac{Prob_i}{Prob_i + Prob_j} \right] + Res_{ji} \log \left[ \frac{Prob_j}{Prob_i + Prob_j} \right] \right) \quad (3)$$

From the above equation (3), '$Res_{ij}$' denote the frequency of times '$T_i$' is selected over '$T_j$' with the overall probability rate of '$Prob_i$' and '$Prob_j$' respectively. Then, to obtain

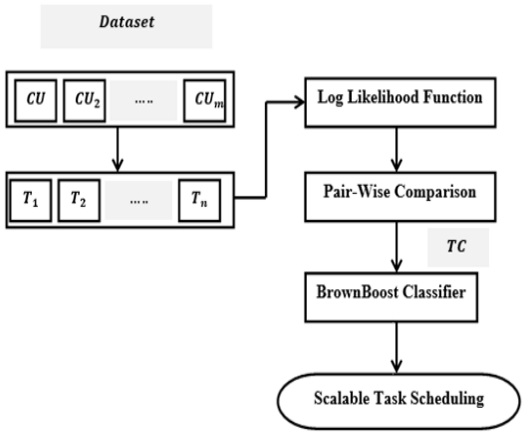


**Figure 2:** Structure of Bradley–Terry BrownBoost Classifier model

multi-class probability estimates by roughly calculating by pairwise comparisons is represented as given below.

$$Res'_{ij} = Prob\left(T_i \; in \; Class \; i | \; T_j \; in \; Class \; i \; or \; Class \; j\right) \quad (4)$$

From the above equation (4), results of multi-class probability estimates by pairwise comparisons '$Res'_{ij}$' is arrived at based on the probability of task '$T_i$' in either '$Class \; i$' (i.e., high priority) or '$T_j$' in either '$Class$' or '$Class \; j$' (i.e., low priority) respectively. Then, this work evaluates '$T_i = Prob(T \; in \; Class \; i)$' by minimizing the distance between '$Res'_{ij}$' and '$mean_{ij} = \frac{T_i}{T_i + T_j}$' and this is mathematically stated as given below.

$$TC = \min_{T} \sum T_{ij} \left( Res'_{ij} \log \frac{Res'_{ij}}{mean_{ij}} + Res'_{ji} \log \frac{Res'_{ji}}{mean_{ji}} \right) \quad (5)$$

From the above equation (5) '$T_{ij}$' represents the number of cloud user requested tasks in '$Class \; i$' (i.e., high priority) '$Class \; j$' (i.e., low priority) respectively. Also from the above equation results the task categorization '$TC$' results are obtained and accordingly '1' represents the high priority task whereas '0' denotes the low priority task. Followed by which, BrownBoost classifier is employed to predict the outcome of priority level.

Given '$n$' feature vectors (i.e., task) '$TC_1 = (TC_{11}, TC_{12}, \ldots, TC_{1p}), \ldots, TC_n = (TC_{n1}, TC_{n2}, \ldots, TC_{np})$' of size '$p$' and vector of class labels '$Res = (Res_1, Res_2, \ldots, Res_n)$' where '$Res_i \in K = \{-1, 1\}$' elucidating the class to which the task belongs to then the outcome of priority level is formulated as given below.

$$c = TC_i[erfinv^2(1 - \varepsilon)] \quad (6)$$

From the above equation (6), the only performance parameter of BrownBoost '$c$' is the time it runs that states that each hypothesis takes a discrete amount of time and hence priority variance in cloud user requested tasks are said to exist. Also a larger value of '$c$' means that BrownBoost will treat the cloud user requested task to be noisy and hence proceed with other set of cloud user requested tasks. On the other hand, a smaller value of '$c$' means that BrownBoost will treat the cloud user requested task as non-noisy and proceed with two-class classifier. Also, '$erfinv(TC)$' denotes the inverse error function with a target classification error '$\varepsilon$' defined as '$\varepsilon = \frac{1}{n} \sum_{i=1}^{n} (Prob[TC_i - Res_i])$' and '$Prob[TC_i]$' formulated as given below.

$$Prob[TC_i] = \text{erf}\left( \frac{\sum \alpha_i h_i(TC)}{\sqrt{c}} \right) \quad (7)$$

From the above equation (7) the probability results of the task categorization '$TC$' is arrived at '$Prob[TC_i]$' by taking into considerations the error function '$\text{erf}(TC)$', hypothesis '$h_i(TC)$' and weight '$\alpha_i$' respectively. In this manner, the core hypothesis of the BrownBoost classifier is that noisy task-categorized samples will be correctly labeled, whereas non-noisy task-categorized samples will contribute to the final classifier. Owing to this if the final classifier of task categorization is learned from the non-noisy task categorized samples, the generalization error of the final classifier can be much better than if learned from noisy task categorized samples. This, in turn, would not only improve task scheduling efficiency but also maximize the completion of the last subtask, thereby improving makespan in a significant manner. The pseudo code representation of Bradley–Terry BrownBoost Classifier-based Task Scheduling is given below.

As outlined in the above algorithm, with the objective of improving task scheduling efficiency and reducing the makespan, initially, cloud user-requested tasks are categorized into low-priority or high-priority tasks. For this, a probabilistic function and a log-likelihood function

**Input**: Dataset '$DS$', User requested tasks '$T = \{T_1, T_2, \ldots, T_n\}$'

**Output**: scalable task scheduling '$TS$'

Step 1: **Initialize** '$n$'
Step 2: **Begin**
Step 3: **For** each Dataset '$DS$' with User requested tasks '$T$'
Step 4: Formulate Directed Acyclic Graph (DAG) for task flow with priority constraints as given in equation (1)
Step 5: Formulate Bradley–Terry probabilistic function as given in equation (2)
Step 6: Evaluate the log likelihood of probability measure as given in equation (3)
Step 7: Evaluate multi-class probability estimates by pairwise comparisons as given in equation (4)
Step 8: Formulate task categorization by minimizing the distance as given in equation (5)
Step 9: **If** '$TC = 1$'
Step 10: **Then** cloud user requested task is high priority task
Step 11: Measure the outcome of priority level as given in equation (6)
Step 12: Obtain BrownBoost classified results as given in equation (7)
Step 13: **Return** task scheduling with high scalable cloud user request access tasks '$TS$'
Step 14: **End if**
Step 15: **If** '$TC = 0$'
Step 16: **Then** cloud user requested task is low priority task
Step 17: **End if**
Step 18: **End for**
Step 19: **End**

**Algorithm 1:** Bradley–Terry BrownBoost Classifier-based Task Scheduling

are employed to obtain multi-class probability estimates through pairwise comparisons. Finally, the BrownBoost classifier is applied to the task-categorized results to ensure scalable task scheduling. As a result the task scheduling efficiency is improved with minimal makespan.

### *Lemke Flower Pollinated Resource Optimization*

Next to identify optimal virtual machine tracking global optimum solutions remains a highly demanding task. Frequently, traditional optimization mechanisms do not perform satisfactorily in cases of optimal virtual machine identification in CC environments, as they may be trapped in local optima. Motivated by the pollination behavior of flowers, several research works have applied the flower pollination algorithm (FPA), a novel heuristic optimization technique. However, the global and local search processes of flower pollination algorithm are highly susceptible and are sensitive to the process of distributing tasks scheduled onto available virtual machines. Owing to the reason that the virtual machines are constrained in number an efficient scheduling model is required to schedule tasks on to the virtual machine. To solve this issue, an improved flower pollination algorithm based on the Gaussian Normal Distribution with Zero Mean and Variance, called Lemke Flower Pollinated Resource Optimization, is proposed. Figure 3 shows the block diagram of Lemke Flower Pollinated Resource Optimization model.

As shown in the above figure, the two element of flower pollination are cross-pollination and self-pollination. On the one hand, cross-pollination reveals that pollination occurs among different flowers due to biological factors, and on the other hand, self-pollination reveals that pollination occurs among different flowers due to spontaneous diffusion. Cross-pollination corresponds to a global search process and is formulated as follows.

$$VM_i^{t+1} = VM_i^t + LD(VM_i^t - CF[VM]) \quad (8)$$

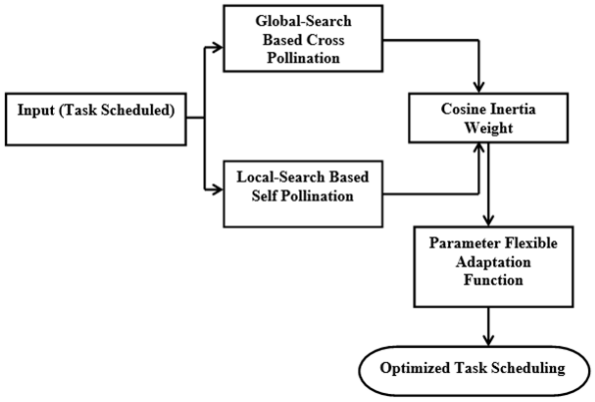


**Figure 3:** Structure of Lemke Flower Pollinated Resource Optimization model

From the above equation (8) the global search cross-pollination for the virtual machine '$VM_i^{t+1}$' with '$VM_i$' possessing specific configurations like main memory '$VM_{mem}$', storage '$VM_{st}$' and processing power '$VM_{pp}$' is formulated by taking into considerations the levy distribution '$LD$' of the current fittest virtual machine '$CF[VM]$' in action. On the other hand, self-pollination corresponds to local searching process and is mathematically represented as given below.

$$VM_i^{t+1} = VM_i^t + \varepsilon \left(VM_j^t - VM_k^t\right) \quad (9)$$

From the above equation (9), '$VM_j^t$' and '$VM_k^t$' two pollens (i.e., virtual machine) individual positions (different virtual machines with different specific configurations) in the population that are distinct from pollen (i.e., virtual machine) '$i$' and '$\varepsilon$' follows a uniform distribution in the range of '$[0, 1]$'. The inertia weight in the conventional flower pollination is said to be an arbitrary value. Despite larger inertia weight contributing to better global searching and smaller inertia weight contributing to better local searching, leads to exploitation. Hence, to ensure trade-off between exploitation and exploration a cosine inertia weight is provided for global search cross-pollination as given below.

$$w = weight_{af}\left[1 - \cos\left(\frac{\pi}{2}\frac{Curr_{iter}}{Max_{iter}}\right)\right] + inert_{af}BetaRND(i,j) \quad (10)$$

From the above equation (10), global search cross-pollination or local search self-pollinations, cosine inertia weight '$w$' is arrived based on the current iteration '$Curr_{iter}$', maximum number of iterations '$Max_{iter}$', weight adjustment factor '$weight_{af}$', inertia adjustment factor '$inert_{af}$' along with beta random numbers '$BetaRND$' with parameters '$i, j$' respectively. The deviation degree of '$w$' employed in our work manage the inertia weight value in turn therefore facilitating the global searching as well as the local searching potentiality.

Next, with the obtained global search cross-pollination or local search self-pollination, the cosine inertia weight crossover operation is performed using the parameter flexible adaptation function. For different stages of the virtual machine selection, process, it is mandatory to formulate different parameters and should meet the following requirements, select appropriate parameters (i.e., task scheduled) for a specific region (i.e., population) in the target scope (i.e., set of virtual machine) and eliminate inappropriate parameters (i.e., inappropriate resources) respectively. With this objective, initially, the parameter flexible adaptation function is formulated as given below.

$$ET\left(TS_i, VM_j\right) = \frac{TS_i}{Total\_MIPS(VM_j)} \quad (11)$$

From the above equation (11) the cloud service provider evaluates the execution time '$ET$' of each task scheduled

'$TS_i$' on each virtual machine '$VM_j$'. Next, the cloud service provider ensures that inappropriate parameters are eliminated and appropriate parameters are retained, with the purpose of maximizing resource utilization while assigning virtual machines to keep resources occupied as much as possible. Then, the average resource utilization is mathematically stated as given below.

$$AvgRU = \frac{\sum_{i=1}^{n} ET_i}{Obj} \quad (12)$$

From the above equation (12) with the objective of keeping as much resource as equipped by the cloud service provider the average resource utilization '$AvgRU$' is formulated based on the execution time '$ET_i$' and the modeled objective '$Obj$'. The objective function here remains in optimal task scheduled being allocating the tasks on virtual machine and is formulated as given below.

$$Obj = \min\left(\sum_{i=1}^{n} ET\left[TS_i, VM_j\right], Out_{ij}\right) \forall VM_j \quad (13)$$

$$Out_{ij} = \begin{cases} 1, if\ TS_i\ is allocated\ to\ VM_j \\ 0, Otherwise \end{cases} \quad (14)$$

From the above equations (13) and (14) '$Out_{ij}$' determines the binary decision output variable (i.e., '1') of either task scheduled '$TS_i$' being allocated to virtual machine '$VM_j$' or on contrary the decision output variable is (i.e., '0') with not allocation to virtual machine. In this manner, optimal task scheduling is ensured. The pseudo code representation of Lemke Flower Pollinated Resource Optimization is given below.

As outlined in the above algorithm, with the objective of reducing energy consumption and subsequently speeding up the process, the scheduled tasks of the corresponding cloud user should be allocated to a virtual machine. With this objective, the Lemke Flower Pollination algorithm, which introduces cosine inertia weight and flexible parameter adaptation, is designed. Here, by using the cosine inertia weight a trade-off between exploitation and exploration is said to be achieved. Next, by introducing parameter-flexible adaptation, both the cloud service providers' and cloud users' requirements are said to be achieved in a fine-tuned manner.

| |
|---|
| **Input**: Dataset '$DS$', Virtual Machine '$VM = \{VM_1, VM_2, \ldots, VM_l\}$' |
| **Output**: Energy and convergent-efficient task scheduling |
| Step 1: **Initialize** task scheduled '$TS$', '$l$', '$\varepsilon = [0,1]$'<br>Step 2: **Begin**<br>Step 3: **For** each Dataset '$DS$' with task scheduled '$TS$' and virtual machine '$VM$'<br>Step 4: Formulate global search cross-pollination as given in equation (8)<br>Step 5: Formulate local search self-pollination as given in equation (9)<br>Step 6: Formulate a cosine inertia weight to ensure trade-off between exploitation and exploration as given in equation (10)<br>Step 7: Formulate parameter flexible adaptation as given in equation (11)<br>Step 8: Evaluate average resource utilization to meet requirements of cloud service provider as given in equation (12)<br>Step 9: Evaluate modeled objective function to meet requirements of cloud user tasks to be accomplished in virtual machine as given in equations (13) and (14)<br>Step 10: **Return** virtual machine allocated<br>Step 11: **End for**<br>Step 12: **End** |

**Algorithm 2:** Lemke Flower Pollinated Resource Optimization

### *Experimental setup*

In this section, the experimental evaluations of the proposed Bradley–Terry BrownBoost and Lemke flower pollinated resource optimization (BTB-LFPRO) for optimized task scheduling and existing methods namely ε-fuzzy dominance based reliable green workflow scheduling (FDRGS) (Rani, R., & Garg, R., 2022) and fruit fly-based simulated annealing optimization scheme (FSAOS) (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) are implemented in Java with the CloudSim network simulator. To ensure fair comparison Personal Cloud Datasets is considered for validating and analyzing the results obtained from http://cloudspaces.eu/results/datasets. Detailed comparative analysis is performed for four different performance metrics, makespan, energy consumption, task scheduling efficiency and throughput respectively.

## Discussion

In this section detailed analysis of four different performance metrics, namely, makespan, energy consumption, task scheduling efficiency and throughput using the proposed Bradley–Terry BrownBoost and Lemke Flower Pollinated Resource Optimization (BTB-LFPRO) and two existing methods, ε-fuzzy dominance based reliable green workflow scheduling (FDRGS) (Rani, R., & Garg, R., 2022) and fruit fly-based simulated annealing optimization scheme (FSAOS) (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) respectively.

### *Performance analysis of makespan*

First, in this section, makespan or completion time representing the total time consumed in processing a set of cloud user requested tasks for its complete execution is measured. In cloud computing environment as far as optimal task scheduling is concerned, makespan denotes the time when the last subtask is accomplished. Minimization of makespan can be achieved by allocating the set of cloud user requested tasks to set of virtual machines. To be more specific, makespan in CC environment denotes the maximum completion of last subtask and is mathematically represented as given below.

$$MS(CT_{max}) = \max \sum_{i=1}^{n} T_{i,1}F_{i,1}, T_{i,2}F_{i,2}, \ldots, T_{i,l}F_{i,l} \quad (15)$$

$$F_{i,l} = \begin{cases} 1, if\ T_i \rightarrow VM_l \\ 0, Otherwise \end{cases} \quad (16)$$

From the above equations (15) and (16) the maximum completion of last subtask or makespan '$MS(CT_{max})$' is represented by means of all the tasks '$T_i$' and '$T_i \rightarrow VM_j$' denotes that task '$T_i$' is allocated to '$VM_j$' and '$T_{i,n}$' denoting the completion time of task '$T_i$' on virtual machine '$VM_j$' respectively. Table 1 shows the comparison of makespan. It is clear that the average makespan for existing methods, FDRGS (Rani, R., & Garg, R., 2022) and FSAOS (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) is 96.2ms and 109.1ms respectively. However, the proposed BTB-LFPRO archives minimum makespan of 83.7ms. This clearly shows the minimum convergence speed of the proposed method. This is due to the calculation of comparison of each cloud user request tasks in the proposed method.

**Table 1:** Makespan

| *Number of user tasks* | *Makespan (ms)* | | |
|---|---|---|---|
| | *BTB-LFPRO* | *FDRGS* | *FSAOS* |
| 1000 | 35 | 53 | 68 |
| 2000 | 38 | 58 | 73 |
| 3000 | 55 | 63 | 78 |
| 4000 | 68 | 79 | 85 |
| 5000 | 75 | 80 | 98 |
| 6000 | 90 | 98 | 110 |
| 7000 | 105 | 115 | 123 |
| 8000 | 113 | 125 | 138 |
| 9000 | 120 | 138 | 155 |
| 10000 | 138 | 153 | 163 |

Figure 4 shows the variation of overall makespan using the proposed BTB-LFPRO and existing two methods, FDRGS (Rani, R., & Garg, R., 2022) and FSAOS (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022). With x-axis representing the distinct numbers of cloud user requested tasks y-axis denotes the makespan measured in terms of milliseconds (ms) using the three methods, proposed BTB-LFPRO, (Rani, R., & Garg, R., 2022) and (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022). While performing simulations with 1000 cloud user requested tasks, the makespan using the proposed BTB-LFPRO was 35ms, and for (Rani, R., & Garg, R., 2022) is 53ms and for (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) is 68ms. However, the proposed BTB-LFPRO method achieves minimum makespan. This clearly shows the minimum amount of time consumed even for the accomplishment of last subtask. The reason was due to the application of Bradley–Terry BrownBoost Classification algorithm. By applying this algorithm, Bradley–Terry probabilistic function was applied in Directed Acyclic Graph (DAG) for measuring task flow with priority constraints. Also log likelihood of probability measure along with multi-class probability estimates by pairwise comparisons was performed for each cloud user requested tasks. With this even the time consumed in accomplishing last subtask was taken into consideration that in turn reduced the overall makespan of the proposed BTB-LFPRO method by 15% compared to (Rani, R., & Garg, R., 2022) and 26% compared to (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022).

### *Performance analysis of task scheduling efficiency*

Cloud user request task scheduling by the cloud service provider refers to the allocation of best suitable resources by taking into considerations different factors, like, cost, makespan, throughput, resource utilization and so on. In other words, task scheduling efficiency refers to the ratio of number of successfully scheduled tasks to the total number of cloud user requested tasks made by distinct cloud users

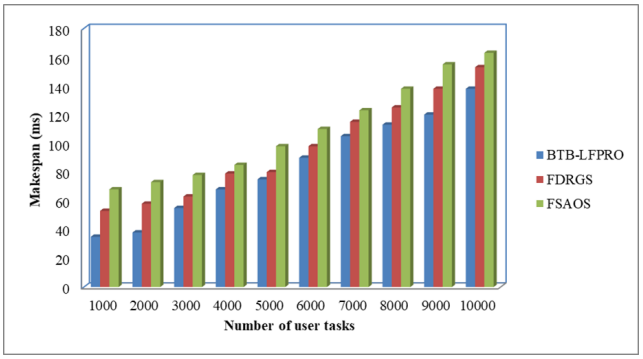


**Figure 4:** Variation of makespan

in cloud computing environment. This task scheduling efficiency is mathematically represented as given below.

$$Eff_{TS} = \sum_{i=1}^{n} \frac{Tasks\ scheduled}{T_i} * 100 \quad (17)$$

From the above equation (17), task scheduling efficiency '$Eff_{TS}$' is measured by taking into consideration the tasks scheduled '$Tasks\ scheduled$' and the cloud user requested tasks made by the cloud user '$T_i$'. It is measured in terms of percentage (%). Table 2 shows the comparison of task scheduling efficiency. The FDRGS (Rani, R., & Garg, R., 2022) method has average task scheduling efficiency of 85% and FSAOS (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) has an average task scheduling efficiency of 76%. However, the proposed BTB-LFPRO method has a high rate of 90%. This is due to the usage of efficient network resource utilization.

Figure 5 shows the variation of task scheduling efficiency. From the above figure x-axis referring to the distinct numbers of cloud user request tasks provided as input and y-axis denoting the task scheduling efficiency results by substituting the values in equation (17). With simulations performed for 1000 cloud user requested tasks 935 tasks were scheduled using the proposed BTB-LFPRO method, 905 tasks were scheduled using (Rani, R., & Garg, R., 2022) whereas 885 tasks were scheduled using (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022), therefore the overall task scheduling efficiency was observed to be 93.5%, 90.5% and 88.5% respectively. This clearly shows the high task scheduling efficiency results of the BTB-LFPRO method. This is due to the implementation of Bradley–Terry BrownBoost Classifier model. By applying this model, BrownBoost classifier was applied to the task categorized results for ensuring scalable task scheduling. Moreover, the BrownBoost function employed with discrete amount of time as hypothesis taken into consideration priority variance is employed and by classifying the resultant values, either cloud user requested task are considered to be noisy or non-noisy and proceed with two-class classifier. This in turn improves the overall task scheduling efficiency of proposed BTB-LFPRO method by 5% compared to (Rani, R., & Garg, R., 2022) and 18% compared to (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) respectively.

**Table 2:** Task scheduling efficiency

| *Number of user tasks* | *Task scheduling efficiency (%)* | | |
|---|---|---|---|
| | *BTB-LFPRO* | *FDRGS* | *FSAOS* |
| 1000 | 93.5 | 90.5 | 88.5 |
| 2000 | 91.35 | 86.15 | 76.13 |
| 3000 | 90 | 85.05 | 75.18 |
| 4000 | 88.25 | 83.45 | 73.25 |
| 5000 | 86.35 | 81.35 | 71.23 |
| 6000 | 89.35 | 84.25 | 74.28 |
| 7000 | 90.45 | 85.95 | 75.35 |
| 8000 | 91.25 | 86.35 | 76.31 |
| 9000 | 92 | 87.55 | 77.33 |
| 10000 | 93 | 88.45 | 78.38 |

### *Performance analysis of energy consumption*

Next, the energy consumption of CC environment chiefly includes, to name a few being memory, storage, CPU and network transmission. The energy consumption is mathematically represented as given below.

$$EC_{i.l} = Pow_l EC_{T_i^j} \quad (18)$$

$$TEC_{i.l} = \sum_{i=1}^{n} \sum_{j=1}^{m} Pow_l EC_{T_i^j} \quad (19)$$

From the above equations (18) and (19) '$EC_{T_i^j}$' represents the energy consumption of task '$T_i$' on virtual machine '$VM_l$' and '$EC_{i.l}$' represents the consumption of energy consumed by task '$T_i$' executing on '$VM_l$' respectively. Table 3 shows the comparison of energy consumption. It is clear that the average energy consumption of the FDRGS (Rani, R., & Garg, R., 2022) method is 75J and has the average energy consumption of the FSAOS (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) was 83J. However, the proposed BTB-LFPRO method had low energy consumption of 70J. This clearly shows the high scalability of the proposed method. This is due to the involvement of virtual machine for optimal scheduling.

Figure 6 shows the comparison of energy consumption efficiency. With x-axis denoting distinct numbers of cloud user requested tasks, y-axis represents the energy consumption measured by substituting the values in equations (18) and (19). With the simulations performed for 1000 cloud user requested tasks, 45J of energy was

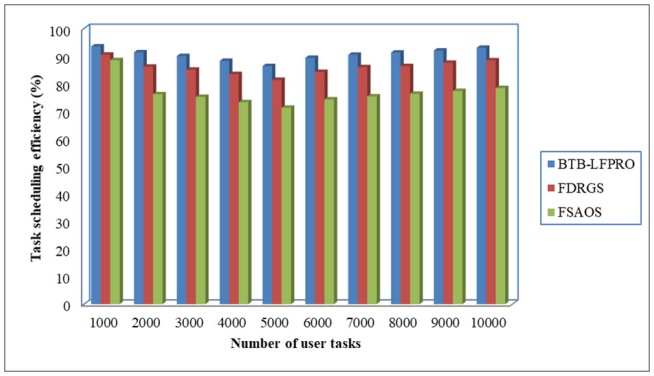


**Figure 5:** Variation of task scheduling efficiency

**Table 3:** Energy consumption efficiency

| *Number of user tasks* | *Energy consumption (J)* | | |
|---|---|---|---|
| | *BTB-LFPRO* | *FDRGS* | *FSAOS* |
| 1000 | 45 | 56 | 65 |
| 2000 | 49 | 54 | 62 |
| 3000 | 55 | 60 | 68 |
| 4000 | 58 | 63 | 71 |
| 5000 | 65 | 68 | 76 |
| 6000 | 73 | 78 | 84 |
| 7000 | 79 | 84 | 90 |
| 8000 | 85 | 89 | 95 |
| 9000 | 92 | 97 | 103 |
| 10000 | 105 | 110 | 116 |

**Table 4:** Throughput analysis

| *Number of user tasks* | *Throughput(tasks/sec)* | | |
|---|---|---|---|
| | *BTB-LFPRO* | *FDRGS* | *FSAOS* |
| **1000** | 195 | 143 | 112 |
| **2000** | 258 | 205 | 194 |
| **3000** | 340 | 278 | 247 |
| **4000** | 564 | 398 | 364 |
| **5000** | 622 | 512 | 495 |
| **6000** | 708 | 675 | 594 |
| **7000** | 837 | 745 | 718 |
| **8000** | 912 | 833 | 807 |
| **9000** | 1011 | 982 | 952 |
| **10000** | 1127 | 1013 | 1006 |

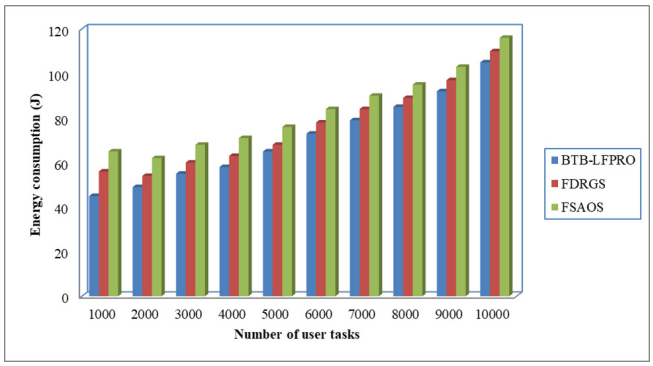


**Figure 6:** Variation of energy consumption efficiency

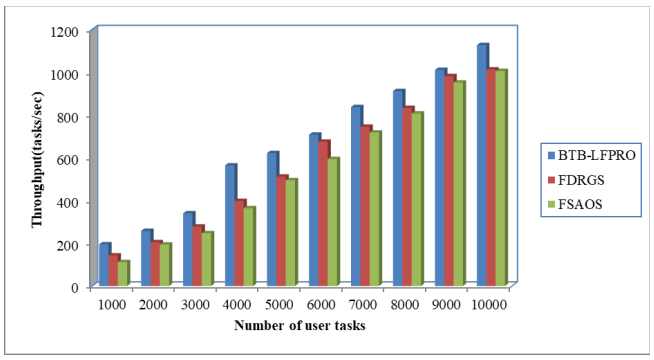


**Figure 7:** Variation of throughput

consumed in ensuring optimal task scheduling when applied with BTB-LFPRO method, 56J of energy consumed using (Rani, R., & Garg, R., 2022) whereas 65J of energy consumed using (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) respectively. This clearly shows the lower energy consumption of the proposed BTB-LFPRO method. This is due to the involvement of effective optimization model based on the network resource availability i.e., using Lemke Flower Pollinated Resource Optimization. By using this optimization model, to ensure trade-off between exploitation and exploration a cosine inertia weight was applied to both global search cross-pollination and local search self-pollination. This in turn reduced the energy being consumed in obtaining optimal task scheduled results. As a whole the overall energy consumption was found to be comparatively lesser using the proposed BTB-LFPRO method by 8% compared to (Rani, R., & Garg, R., 2022) and 16% compared to (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022).

### *Performance analysis of throughput*

Throughput is defined as the number of user-requested tasks completed by a particular time in the cloud. The mathematical representation for calculating throughput is given below,

$$TPT = \left[\frac{Number\ of\ tasks\ completed}{t\ (sec)}\right] \quad (20)$$

From the above equation (20) '$TPT$' indicates throughput, '$t$' represents time in second (sec), and it is computed in terms of tasks per second (tasks/sec).

Table 4 and Figure 7 portray a graphical representation of throughput with varying numbers of user tasks ranging from 1000 and 10000 considered for simulation. In Figure 7, the number of tasks is denoted in the horizontal axis, and throughput performance is represented in the vertical axis. The results illustrate that the throughput is comparatively increased than the existing methods. Let us consider that the number of tasks is 1000, and the throughput obtained for BTB-LFPRO, FDRGS (Rani, R., & Garg, R., 2022) and FSAOS (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022) is 196 tasks/sec, 143 tasks/sec, and 112 tasks/sec respectively. When the number of data points gets increased, the throughput gets increased correspondingly. From the graph, BTB-LFPRO improved than

the existing methods. This is because of the application of the Lemke Flower Pollinated Resource Optimization algorithm. By using this algorithm, the Parameter Flexible Adaptation function is employed for performing crossover operations. Appropriate parameters (i.e., task scheduled) were selected for a specific region (i.e., population), and discarded inappropriate parameters (i.e., inappropriate resources) with higher throughput. As a result, throughput is enhanced using BTB-LFPRO improved by 19% compared to (Rani, R., & Garg, R., 2022) and 29% compared to (Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z., 2022), respectively.

## Conclusion

Task scheduling is predominantly concentrated on identifying the optimal resources with the purpose of reducing the total execution time of virtual machines and hence the focus remains in improving the significant utilization of the shared resources. A plethora of meta-heuristic techniques have been deployed to solve the issue of optimal task scheduling in cloud computing environment. In this work, priority classification and resource optimization called, Bradley–Terry BrownBoost and Lemke Flower Pollinated Resource Optimization (BTB-LFPRO) is proposed. To begin, Bradley–Terry BrownBoost Classifier was applied to the cloud user request schedules to classify the tasks and prioritize according to the requirements. This eventually leads to a decrease in the makespan and improves task scheduling efficiency. The Lemke Flower Pollinated Resource Optimization algorithm is then used to ensure energy and convergent efficient task scheduling that in turn ensure trade-off between exploitation and exploration, therefore improving energy consumption and throughput subsequently. Results showed that the Bradley–Terry BrownBoost and Lemke Flower Pollinated Resource Optimization (BTB-LFPRO) method outperformed the counterpart. Moreover, the introduced BTB-LFPRO method minimizes the makespan, average energy consumption and throughput with improved task scheduling efficiency compared to the counterpart.

## Acknowledments

We sincerely acknowledge the Research convenor, Dr.A.R.Mohamed Shanavas, and Dr. D. I. George Amalarethinam, Principal of the institution, for providing the facility to complete this paper Successfully.

## Conflict of Interest

The authors have no conflict of interest regarding publication of this manuscript. This submission is original work has and not under review at any other journal/ conference etc

## References

Abohamama, A. S., El-Ghamry, A., & Hamouda, E. (2022). Real-time task scheduling algorithm for IoT-based applications in the cloud–fog environment. *Journal of Network and Systems Management*, *30*(4), 54. https://doi.org/10.1007/s10922-022-09664-6

Ali, A., & Iqbal, M. M. (2022). A cost and energy efficient task scheduling technique to offload microservices based applications in mobile cloud computing. IEEE Access, 10, 46633-46651. DOI: 10.1109/ACCESS.2022.3170918

Amer, A. A., Talkhan, I. E., Ahmed, R., & Ismail, T. (2022). An optimized collaborative scheduling algorithm for prioritized tasks with shared resources in mobile-edge and cloud computing systems. Mobile Networks and Applications, 27(4), 1444-1460. https://doi.org/10.1007/s11036-022-01974-y

Choppara, P., & Mangalampalli, S. (2024). An Effective analysis on various task scheduling algorithms in Fog computing. EAI Endorsed Transactions on Internet of Things, 10. DOI: https://doi.org/10.4108/eetiot.4589

Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., & Zakaria, Z. (2022). Dynamic scheduling of heterogeneous resources across mobile edge-cloud continuum using fruit fly-based simulated annealing optimization scheme. Neural Computing and Applications, 34(16), 14085-14105. https://doi.org/10.1007/s00521-022-07260-y

Hai, T., Zhou, J., Jawawi, D., Wang, D., Oduah, U., Biamba, C., & Jain, S. K. (2023). Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes. Journal of Cloud Computing, 12(1), 15. https://doi.org/10.1186/s13677-022-00374-7

Hosseinzadeh, M., Azhir, E., Lansky, J., Mildeova, S., Ahmed, O. H., Malik, M. H., & Khan, F. (2023). Task scheduling mechanisms for fog computing: A systematic survey. IEEE Access, 11, 50994-51017. DOI: 10.1109/ACCESS.2023.3277826

Jabeen, A., & Shanavas, A. R. M. (2024). Hazard regressive multipoint elitist spiral search optimization for resource efficient task scheduling in cloud computing. *The Scientific Temper*, *15*(02), 2143-2151.

Jain, P., & Sharma, S. K. (2023). A Load Balancing Aware Task Scheduling using Hybrid Firefly Salp Swarm Algorithm in Cloud Computing. International journal of computer networks and applications, 10(6), 914-914. DOI: 10.22247/ijcna/2023/223686

Jin, C., Han, Y., Deng, Z., Chen, Y., Liu, C., & Huang, J. (2023). Reinforcement Learning-Based Intelligent Task Scheduling for Large-Scale IoT Systems. Wireless Communications and Mobile Computing, 2023(1), 3660882. https://doi.org/10.1155/2023/3660882

Kumar, M. S., & Karri, G. R. (2023). Eeoa: cost and energy efficient task scheduling in a cloud-fog framework. Sensors, 23(5), 2445. https://doi.org/10.3390/s23052445

Kumar, M. S., & Kumar, G. R. (2023). EAEFA: an efficient energy-aware task scheduling in cloud environment. EAI Endorsed Transactions on Scalable Information Systems, 11(3). DOI: https://doi.org/10.4108/eetsis.3922

Li, Y., & Hei, X. (2022). Performance optimization of computing task scheduling based on the Hadoop big data platform. Neural Computing and Applications, 1-12. https://doi.org/10.1007/s00521-022-08114-3

Mansouri, N., Zade, B. M. H., & Javidi, M. M. (2019). Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory. Computers & Industrial Engineering, 130, 597-633. https://doi.org/10.1016/j.

cie.2019.03.006

Mittal, P., Kumar, D. S., & Sharma, D. S. (2024). Revolutionizing Cloud-Based Task Scheduling: A Novel Hybrid Algorithm for Optimal Resource Allocation and Efficiency in Contemporary Networked Systems. International Journal of Computing and Digital Systems, 15(1), 1551-1563. DOI:10.12785/ijcds/1501110

Muthusamy, A., & Dhanaraj, R. K. (2023). Dynamic Q-Learning-Based Optimized Load Balancing Technique in Cloud. Mobile Information Systems, 2023(1), 7250267. https://doi.org/10.1155/2023/7250267

Paulraj, D., Sethukarasi, T., Neelakandan, S., Prakash, M., & Baburaj, E. (2023). An efficient hybrid job scheduling optimization (EHJSO) approach to enhance resource search using Cuckoo and Grey Wolf Job Optimization for cloud environment. Plos one, 18(3), e0282600. https://doi.org/10.1371/journal.pone.0282600

Pliatsios, D., Sarigiannidis, P., Lagkas, T. D., Argyriou, V., Boulogeorgos, A. A. A., & Baziana, P. (2022). Joint wireless resource and computation offloading optimization for energy efficient internet of vehicles. IEEE Transactions on Green Communications and Networking, 6(3), 1468-1480. DOI: 10.1109/TGCN.2022.3189413

Raeisi-Varzaneh, M., Dakkak, O., Habbal, A., & Kim, B. S. (2023). Resource scheduling in edge computing: Architecture, taxonomy, open issues and future research directions. IEEE Access, 11, 25329-25350. DOI: 10.1109/ACCESS.2023.3256522

Rani, R., & Garg, R. (2022). Reliability aware green workflow scheduling using ε-fuzzy dominance in cloud. Complex & Intelligent Systems, 1-19. https://doi.org/10.1007/s40747-021-00609-1

Sana, M. U., & Li, Z. (2021). Efficiency aware scheduling techniques in cloud computing: a descriptive literature review. PeerJ Computer Science, 7, e509. DOI: 10.7717/peerj-cs.509

Senthilkumar, G., Suvarnamukhi, B., Lekashri, S., & Mohammed Thaha, M. (2024). Effective task scheduling based on interactive autodidactic school algorithm for cloud computing. Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije, 65(1), 159-166. https://doi.org/10.1080/00051144.2023.2288484

Singh, H., Bhasin, A., & Kaveri, P. R. (2021). QRAS: efficient resource allocation for task scheduling in cloud computing. SN Applied Sciences, 3, 1-7. https://doi.org/10.1007/s42452-021-04489-5

Singh, S. P., Nayyar, A., Kaur, H., & Singla, A. (2019). Dynamic task scheduling using balanced VM allocation policy for fog computing platforms. Scalable Computing: Practice and Experience, 20(2), 433-456. https://doi.org/10.12694/scpe.v20i2.1538

Sun, F., Hou, F., Cheng, N., Wang, M., Zhou, H., Gui, L., & Shen, X. (2018). Cooperative task scheduling for computation offloading in vehicular cloud. IEEE Transactions on Vehicular Technology, 67(11), 11049-11061. DOI: 10.1109/TVT.2018.2868013

Tang, Z., Qi, L., Cheng, Z., Li, K., Khan, S. U., & Li, K. (2016). An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. Journal of Grid Computing, 14, 55-74. https://doi.org/10.1007/s10723-015-9334-y

Thanka, M. R., Maheswari, P. U., & Edwin, E. B. (2019). A hybrid algorithm for efficient task scheduling in cloud computing environment. International Journal of Reasoning-based Intelligent Systems, 11(2), 134-140. DOI: 10.1504/IJRIS.2019.10021325

Wang, Z., Chen, S., Bai, L., Gao, J., Tao, J., Bond, R. R., & Mulvenna, M. D. (2023). Reinforcement learning based task scheduling for environmentally sustainable federated cloud computing. Journal of Cloud Computing, 12(1), 174.. https://doi.org/10.1186/s13677-023-00553-0

Xu, J., & Palanisamy, B., 2018) Xu, J., & Palanisamy, B. (2018). Optimized contract-based model for resource allocation in federated geo-distributed clouds. IEEE Transactions on Services Computing, 14(2), 530-543. DOI: 10.1109/TSC.2018.2797910

Ya-meng, B. A. I., Yang, W. A. N. G., & Shen-shen, W. U. (2023). Deadline-aware Task Scheduling for Cloud Computing using Firefly Optimization Algorithm. International Journal of Advanced Computer Science and Applications, 14(5). DOI: 10.14569/IJACSA.2023.0140553

Yuan, H., Bi, J., Tan, W., & Li, B. H. (2016). Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds. IEEE Transactions on Automation Science and Engineering, 14(1), 337-348. DOI: 10.1109/TASE.2016.2526781