Hindawi
Journal of Sensors
Volume 2023, Article ID 9828793, 1 page
https://doi.org/10.1155/2023/9828793

Retraction

Retracted: Cost-Aware Resource Optimization for Efficient Cloud Application in Smart Cities

Journal of Sensors

Received 19 December 2023; Accepted 19 December 2023; Published 20 December 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] P. Gupta, R. R. Kaikini, D. K. Saini, and S. Rahman, "Cost-Aware Resource Optimization for Efficient Cloud Application in Smart Cities," *Journal of Sensors*, vol. 2022, Article ID 4406809, 12 pages, 2022.

Hindawi
Journal of Sensors
Volume 2022, Article ID 4406809, 12 pages
https://doi.org/10.1155/2022/4406809

Research Article

Cost-Aware Resource Optimization for Efficient Cloud Application in Smart Cities

Punit Gupta,[1] **Ravindra R. Kaikini**,[2] **Dinesh Kumar Saini**,[1] **and Salma Rahman**[3]

[1]*Department of Computer and Communication Engineering, Manipal University Jaipur, Jaipur, India*
[2]*Department of Business Administration, Sahyadri College of Engineering and Management, Mangalore, Karnataka State, India*
[3]*North South University, Bashundhara, Dhaka 1229, Bangladesh*

Correspondence should be addressed to Punit Gupta; punitg07@gmail.com and Dinesh Kumar Saini; dkssohar@gmail.com

Received 21 October 2021; Revised 8 December 2021; Accepted 2 April 2022; Published 20 April 2022

Academic Editor: Pradeep Kumar Singh

In this generation of smart computing environment, every device in the system and multple system are interconnected to each other, which allows users to view, analyze data, and make smart decisions. Smart cities are one example of smart environments where every device is connected and computing is performed on the cloud. In such a situation, the system requires an efficient system to handle huge requests and deliver data. Cloud computing plays an essential role in solving this issue but suffers from resource optimization, cost optimization, and load balancing. This work is aimed at solving the issue of resource and cost optimization in cloud infrastructure to provide a high service rate and sustainable infrastructure to cloud applications in smart cities. The proposed model is inspired by artificial neural networks and nature-inspired algorithm to reduce execution cost, average start time, and finish time and to make the system power efficient at the same time to improve the utilization of the system. The result shows that the proposed model completes more tasks in the least time and execution cost as compared to the existing models. This showcases the smart cloud applications are cost efficient and can complete the tasks in less time.

1. Introduction

A smart city is an environment that comes with interconnected systems. A smart city is a set of interconnected systems which connect and communicate with each other for efficient solutions in a distributed environment. Cloud computing provides the best platform to such an application area, and it can provide a solution to all existing systems in smart city speaking from a traffic light, smart health care, smart transport system, and many more. Every system is connected to the cloud through cloud application. In such a scenario, cloud performance plays an important role in a widely distributed environment. The cloud suffers from various issues like resource optimization, cost optimization, load balancing, utilization, and power efficiency [1, 2]. Figure 1 shows the role of the cloud in smart cities with various types of systems interconnected to each other through cloud applications. In this figure, various connected networks in smart cities are showcased with multiple connected devices like mobile, smart camera, and surveillance devices using cloud services, where all the devices are connected to cloud and computation is done on cloud. This is a representation of huge tasks generated in smart infrastructure for cloud services.

In order to manage such a high-task load on cloud applications in smart cities, cloud optimization plays an essential role to serve the client with a high service rate and least cost. The cloud is basically a distributed environment with a scalable, pay-per-use, and reliable computing environment. Cloud applications are typically designed to provide real-time computing and require an efficient and cost-effective computing environment to provide a better, more cost-effective, and more reliable solution to the user [3–5]. The resource allocation in a cloud computing environment plays a vital role in improving the system's performance and serving users with high quality of service and the least cost. Services in smart city environments depend on remote computation and data analysis performed on the cloud. Cloud

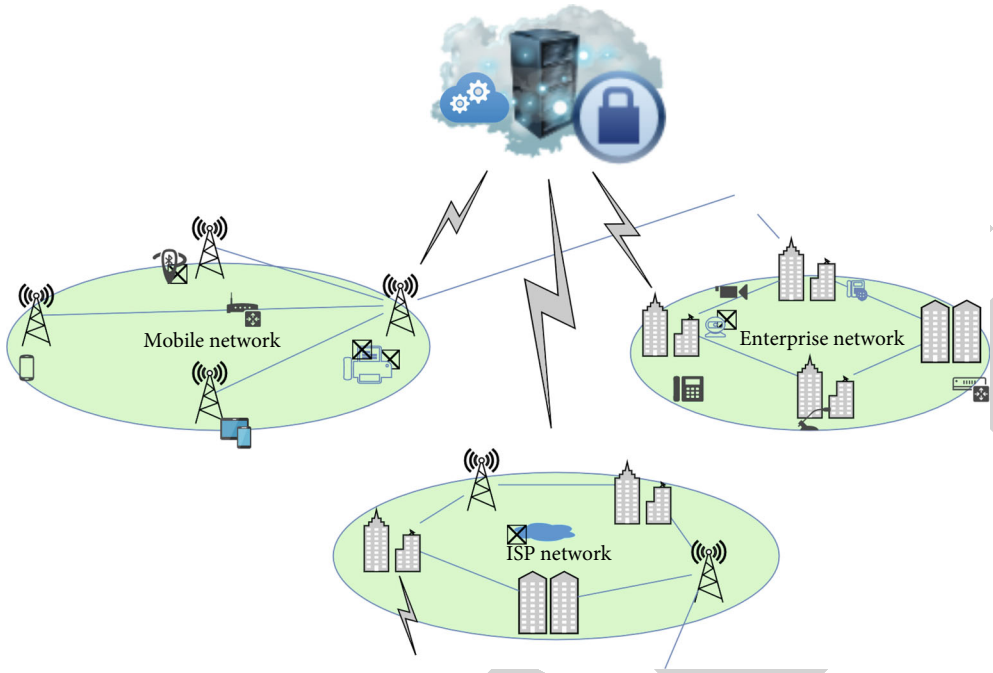

Figure 1: Cloud application service architecture for smart city applications.

optimization is critical for real-time services that most e-health, smart vehicles, and many other applications rely on.

The rest of this manuscript is organized as follows. Section 2 covers the related works. Section 3 presents the proposed cost-aware task scheduling approach and its implementation details. Section 4 incorporates the performance evaluation and analysis is included. Finally, Section 5 concludes the work.

2. Related Work

In this section, some of the related works from the field of task scheduling and resource optimization are discussed.

Cloud service has properties of pay per use which makes it a more advanced computational service [6]. The authors used group scheduling algorithm to schedule the tasks on the basis of their attributes with respect to the category been divided into four parts (user, type, size, and latency). After the tasks are divided into categories, scheduling algorithm takes place. Scheduling algorithm is devided into two parts. In the first part we will schedule the tasks on the basis of user type which will depend on category, which has high weightage that will schedule first. The second part the categories which are created in the first part will be subcategorize and scheduling of the tasks will be done accordingly. To do so, the authors used a matrix-based approach. The authors did not consider the workflow which is very important in the cloud environment [7]. The authors proposed a design for a workflow as a service to schedule the task in a more efficient manner to respond a continuous workflow execution of the task and schedule them in an efficient manner. To achieve the full utilization of cloud resource, the authors proposed algorithm that has been divided into two parts where in the first part, the authors used a workflow composer which uses a workflow service to check the workflow which interacted with the workflow scheduler to schedule task which forwards the tasks to the VM administrator for interaction with the clouds VM manager to schedule the task. The authors also proposed the WFaaS (workflow as a service) architecture to show how the tasks are being scheduled. With the WFaaS, the authors proposed static task scheduling, dynamic task scheduling, adaptive task scheduling, and greedy task scheduling algorithms to improve the scheduling in the cloud [8]. The authors proposed a credit-based scheduling algorithm which considers two factors: one is the task length and second is the user priority to schedule a task. The authors proposed an algorithm that works on credit which is given to the task length and priority; a task scheduler will consider their credit value while scheduling. During the scheduling, the task will be picked up from both sides: front and back sides. The task length and its credit value are being considered which are given on the basis of priority [9]. The authors proposed a multiobjective task scheduling algorithm to maximize the throughput and utilize the resource in the cloud system without violating SLA (service level agreement). By considering other aspects like VM availability and data center availability, other matrices also consider their proposed work [10]. The authors proposed a particle swarm optimization algorithm to schedule the task to maximize the utilization of resources. To maximize the resource utilization, the authors opt for fitness value to schedule the task [11]. The authors proposed a genetic ant colony optimization algorithm for an optimal solution to schedule the task over the cloud with a global search optimal solution [12]. The authors proposed a modified shortest job first algorithm which completes the last task

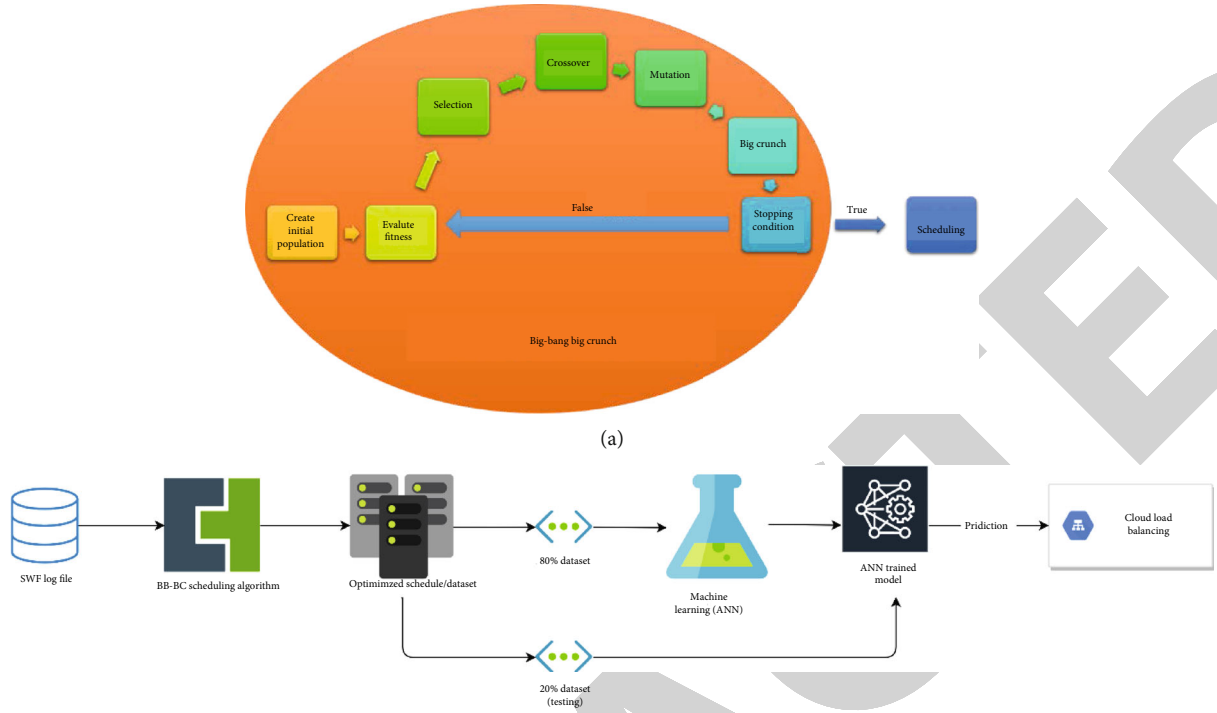

Figure 2: (a) The proposed BB-BC algorithm block diagram with various phases. (b) The proposed model architecture using the BB-BC and ANN model.

(makespan) and optimizes the response time with utilization of resource. Based on the average length, task scheduling takes place and maintains load balance over the cloud.

All the above existing works discussed show that the proposed task allocation algorithm makes optimization of time and resource utilization using nature-inspired and metaheuristic optimization models like ant colony, PSO, SHC, and GA. But the cloud is a pay-per-use model which requires cost optimization also at the same time, which is not fulfilled by either of the proposed model. On the other hand, some of the cost-aware algorithms are not able to find least execution time and cost-efficient solution at the same time. To overcome this issue, a BB-BC-based multiobjective time and cost-efficient solution is proposed to fulfill both the needs.

The previous work has focused only on one parameter/objective like execution time, power, utilization, and cost and could not be efficient when multiple parameters are considered. This work is aimed at overcoming the disadvantages of the existing approaches and proving a global best solution to improve the quality of service of the cloud.

3. Proposed Model

This section showcases the proposed algorithm for task allocation for cloud infrastructure. The novelty of the proposed Big Bang-Big Crunch (BB-BC) ANN cost-aware technique is to increase the execution time, cost, and convergence rate. It provides an optimal global solution with a fast convergence rate, providing a cost-efficient solution for the client.

The proposed algorithm is divided into two parts: trained model preparation and prediction. Figure 2(b) shows the proposed architecture of ANN-inspired cost-aware BB-BC algorithm. Figure 2(a) shows the flow of inputs from one module to another module and how the dataset is prepared for training of ANN and further testing. The trained ANN model is used to predict the cost-efficient solution. Figure 3 shows the working of the proposed model with all the phases on how the complete model is implemented and trained.

The proposed model is divided into 4 parts:

(i) Data preparation

(ii) Training

(iii) Testing

(iv) Prediction

(1) Data preparation

The BB-BC algorithm is a search-based heuristic optimization algorithm that selects the object on a fitness criterion for further processing. The algorithm is aimed at designing an optimized schedule for the tasks with a least cost.

The BB-BC consists of five phases:

3.1. Phase I (Initial Population). Initial population is the set of a random selection from the raw dataset. Every element

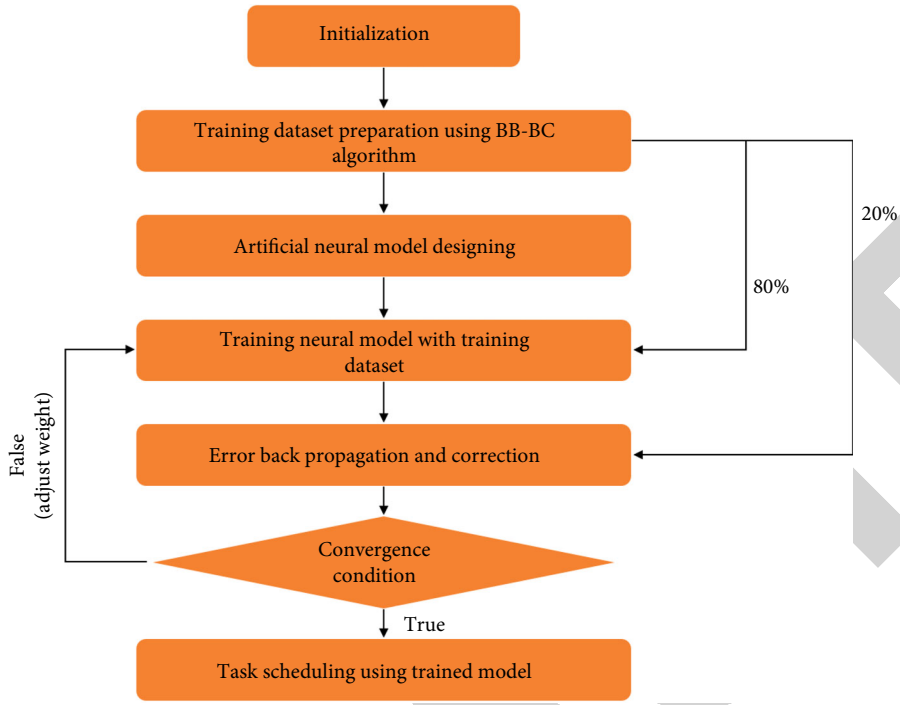

Figure 3: Flowchart of the proposed resource allocation technique.

of the set is characterized with some parameters. While deciding the initial population, the diversity and population size should be kept in mind. The population size is a random vector. The initialization can be done in two ways:

(1) Random initialization: the solution is given randomly without prior knowledge

(2) Heuristic initialization: the solutions are given on some prior known factors; this may cause the similarity and less diversity in the solution

After the initial population is determined, a predefined fitness function is applied to the initial population.

3.2. Phase 2 (Parent Selection). Selection is the most complex part of a genetic algorithm as if we select the fittest possible solution; it may not let any other solution to be selected, hence a premature convergence. Having a diversified set of solution is necessary for a successful BB-BC. There are various ways to select the solution.

(a) Fitness proportionate selection

In this method, every solution can become a parent with the probability proportional to its fitness. Hence, the fitter solutions will have higher chances of selection. Two possible models for this type of selection are as follows:

(1) Roulette wheel selection: a fixed point is defined on the circumference of the wheel. After the rotation whichever the region stops in front of the point is elected as a parent. The greater fitness solution will cover a large region; hence, it will have high chances of selection

(2) Stochastic universal sampling: we will have multiple fixed points; therefore, all the parents will be chosen at one spin

(b) Tournament selection

In an n-way tournament selection, n individual solutions are selected from the population. Process is repeated until the fittest parent is selected. This method can work with negative values too.

(c) Rank selection

When the fitness of the individuals is very close to each other, the fitness value is not taken into considerable as all the sets have the same fitness value. The higher ranked is preferred.

(d) Fitness function

For a parent selection, a predefined function is applied to every solution that is the fitness function. The selection of the individual is based on the fitness score. The fitness function should be fast enough to calculate the fitness of the given solution as we may have to repeat it multiple times until an optimal solution is found. Fitness may be exact or approximate which depends on the case.

The training set has a following features: task ID, expected cost, execution of all virtual machines, and finally

the actual output (i.e., the VM selected for allocation). The features of the training dataset help the ANN model to map the BB-BC algorithm's behavior. Equation (1) is used by BB-BC to evaluate the fitness of each population, where cost used in fitness function is evaluated for each task as given in Equation (2). Cost function requires cost of individual resource of RAM, MIPS (cost_mips), processor (cost_pe), and bandwidth (cost_bw). The total sum of resources used defines the total cost of the resources.

$$\text{fitness value}_i = \sum_{j=1}^{j=n} \alpha * \text{utilization}_j + \beta * \text{total\_execution\_time}_j + \gamma * \text{cost}_i, \tag{1}$$

$$\begin{aligned} \text{cost}_i &= \text{network\_delay}_i + (\text{task\_completion\_time}_i \times \text{cost\_per\_unit}), \\ \text{cost\_ram} &= \text{ram\_size} \times \text{costperram}, \\ \text{cost\_mips} &= \text{mips} \times \text{costpermips}, \\ \text{cost\_pe} &= \text{numberofpes} \times \text{costpersecond}, \\ \text{cost\_bw} &= \text{bw} \times \text{costperbw}, \\ \text{cost\_per\_unit} &= \text{cost\_ram} + \text{cost\_mips} + \text{cost\_pe} + \text{cost\_bw}, \end{aligned} \tag{2}$$

where $\alpha + \beta = 1$.

$$\text{Total\_execution\_time} = \sum_{i=1}^{n} \frac{\text{Task\_length}_i}{\text{MIPS}_j} + \text{Network\_delay}_i, \tag{3}$$

$$x^c = \frac{\sum_{i=1}^{i=N} (1/f_i) x_i}{\sum_{i=1}^{i=N} 1/f_i}. \tag{4}$$

Equation (1) defines the fitness function which is the summation of computational cost and the execution time in a proportion defined by α and β, where sum of α and β is 1. Equation (2) defines the function to evaluate the execution time used in Equation (1) for fitness evaluation, where the execution time is the sum of time required to complete the task and network delay.

3.3. Phase 3 (Crossover). In this, more than one parent is selected and one or more set is generated having the parent attributes. The crossover is applied with a high probability. There are various ways to perform a crossover.

(a) One-point crossover: the random point is selected from the parent(s) and generated a new solution with the swapping process

(b) Multipoint crossover: it is advanced to the one-point crossover; in this, we will have multiple random points to generate a new solution

(c) Uniform crossover: in this, there is no random point. We decide whether the property of parent will be selected or not by flipping a coin

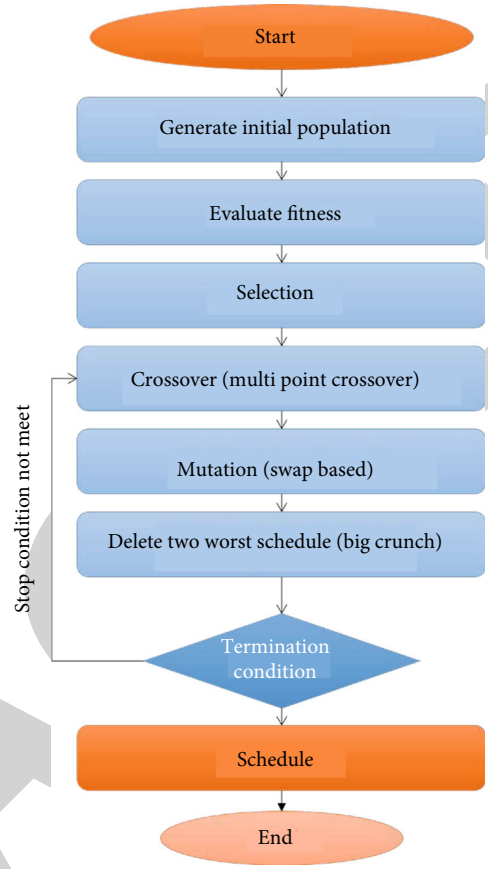

Figure 4: Flowchart of BB-BC algorithm.

Table 1: Configuration parameters of user tasks.

Parameters	Values
Task length	300/2000/3000/4000
Input file size	200 bytes
Output file size	400 bytes
PE	1-2

Table 2: Types of VMs.

	MIPS/RAM (mb)/PE/storage
VM1	1000/512/1/20000
VM2	2000/1024/2/2000
VM3	4000/2156/4/20000

(d) Whole arithmetic recombination: this takes weighted average of parents by the following formulae:

(1) Child1 = $\alpha \cdot x + (1 - \alpha) \cdot Y$

(2) Child1 = $\alpha \cdot x + (1 - \alpha) \cdot Y$

(e) Davi's order crossover (OX1): it is a permutation-based crossover that transmits relative information to the new solutions. It works as follows:

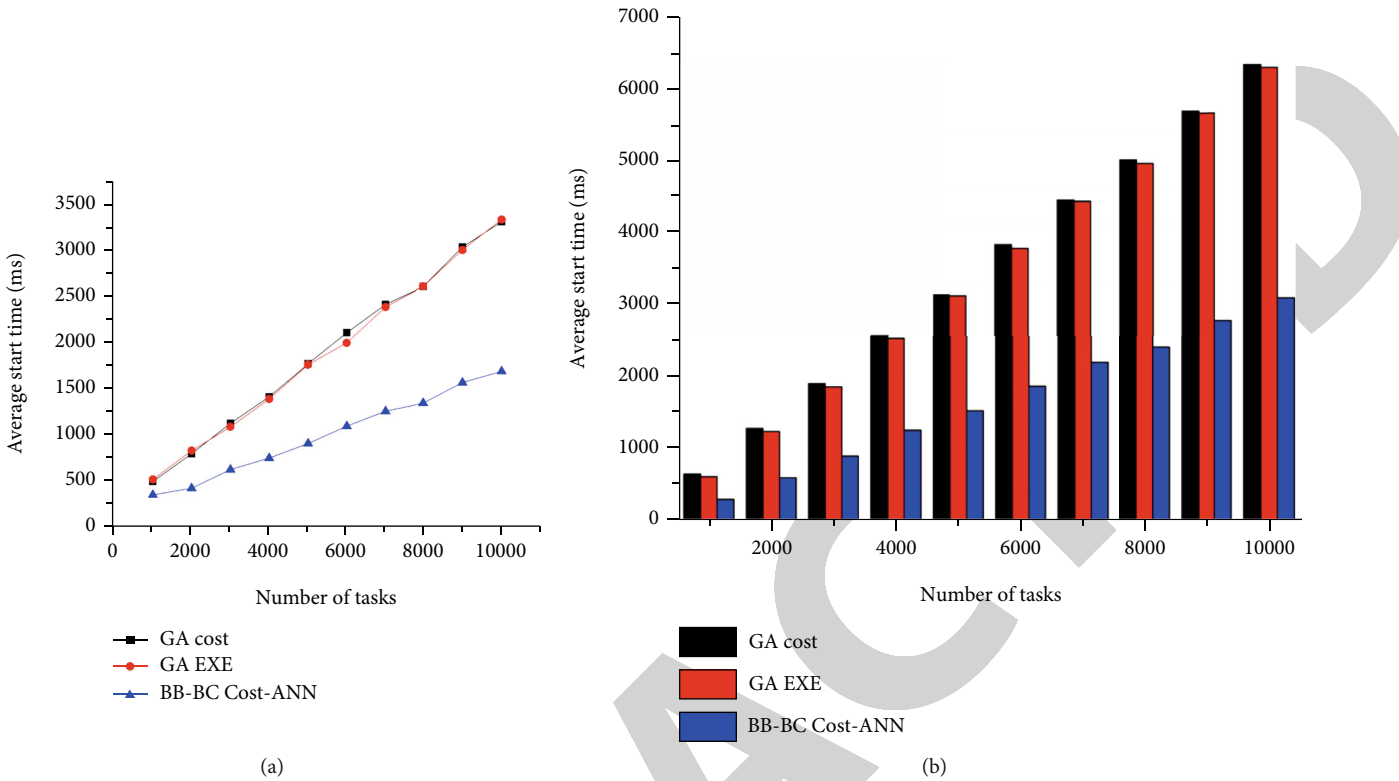

Figure 5: (a) Comparison of average start time for 5 VMs. (b) Comparison of average start time for 10 VMs.

(1) Define two random points in the parent and copy the segment between points and move it to the first new solution

(2) Now from the 2nd point in the second parent, copy the unused content and move to the first solution

(3) Repeat the second solution in a reverse parent manner

3.4. Phase 4 (Mutation). This process makes a random small change in the parent solution to generate a new solution. This occurs to maintain the diversity in the initial population set and works on low probability. The most common used mutations are as follows:

(a) Bit flip mutation: we select some random bits and flip them to get a new solution. Mostly used in binary encoded BB-BC

(b) Random setting: in this, a random integer value is assigned to a randomly selected solution

(c) Swap mutation: this combination-based encoding selects to random points from the parent and swaps them to generate a new solution

(d) Scramble mutation: in this, a subset of the solution is selected and scrambled/shuffled the subset to generate a new solution

(e) Inversion mutation: similar to the scramble selection mutation, we choose a subset, but instead of shuffle, we invert the list to generate a new solution

3.5. Phase 5 (Big Crunch and Termination Condition). In the Big Crunch phase, the worst solution with worst fitness value is rejected in each iteration. In each iteration, two such solutions are rejected. For termination condition, when the proposed algorithm provides a single set of solution, then the algorithm is terminated. In the beginning, BB-BC tends to be very fast, but at later stages, it comes to a saturation point where the improvements are very small. Termination is usually done when we are nearly close to an optimal solution or a solution is repetitive in nature.

Figure 4 shows the working of the proposed BB-BC algorithm taking into account the list of tasks and virtual machine list as input.

(2) Training

The schedule/data generated by BB-BC is used for training which has the following features: task length, task input, output size, processor required, start for each VM, and the VM ID of the selected VM. This data is divided into training and testing data in a proportion of 80:20 for verification of accuracy of the machine learning model. The ANN model used 2 hidden layers, 1 input layer, and 1 output layer. The hidden layer has 50 neurons in each layer and 1 neuron in the output layer. Learning rate is 0.2 with leaky ReLU as the

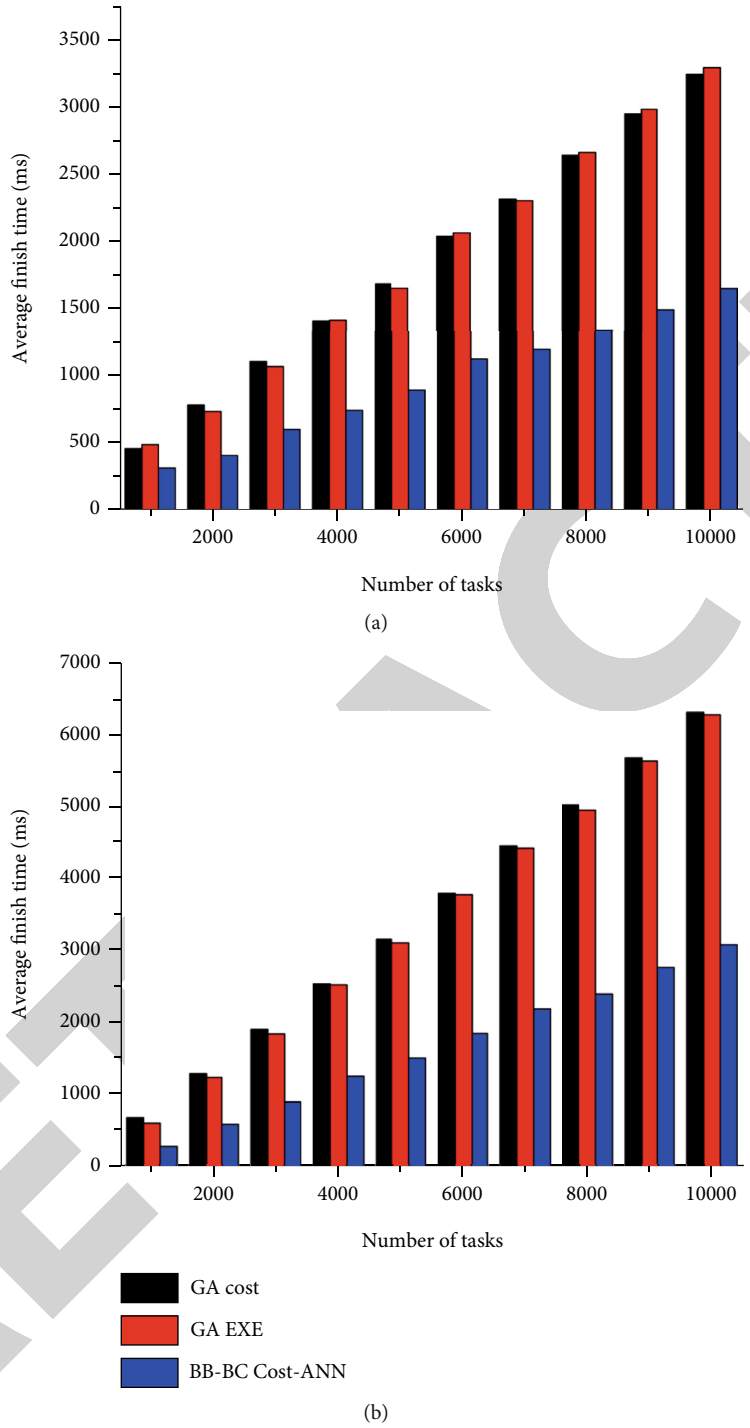

Figure 6: (a) Comparison of average finish time for 5 VMs. (b) Comparison of average finish time for 105 VMs.

activation function for the hidden layer and input layer and the sigmoidal function at the output layer. The training phase also includes the backpropagation and error correction which improves the accuracy of the model with the least mean error.

(3) Testing

In this phase, the trained model is tested with the 20% of the remaining data for testing the accuracy of the model if the model accuracy is found to be low; error correction is followed. This phase of neural network goes for various iterations which allow to improve the accuracy of the system.

Figures 2(a) and 2(b) show the architecture of the proposed BB-BC and BB-BC using ANN for task scheduling

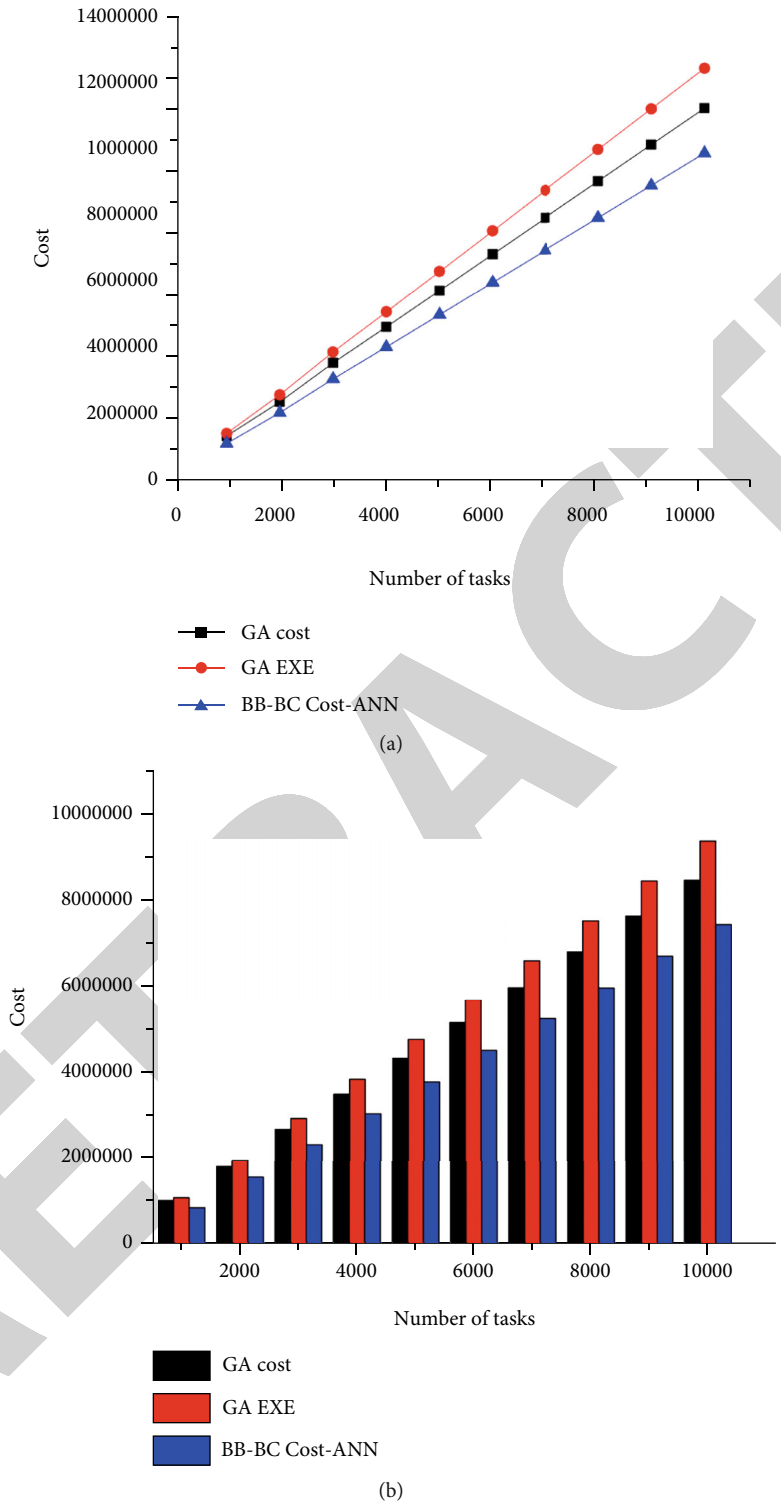

Figure 7: (a) Comparison of cost in $ for 5 VMs with increasing tasks. (b) Comparison of cost in $ for 10 VMs with increasing tasks.

in the cloud. Figure 2(a) shows the various phases in the proposed BB-BC algorithm to find the best optimal solution using the Big Bang and Big Crunch phases, which allow the model to find best fittest solution. The ANN model has its input and output layers with various activation functions used at the input layer, hidden layer, and output layer with error correction mechanism using backpropagation modules. For the model setup, mutation rate is 0.15; evolution and population are varied from 50 to 100.

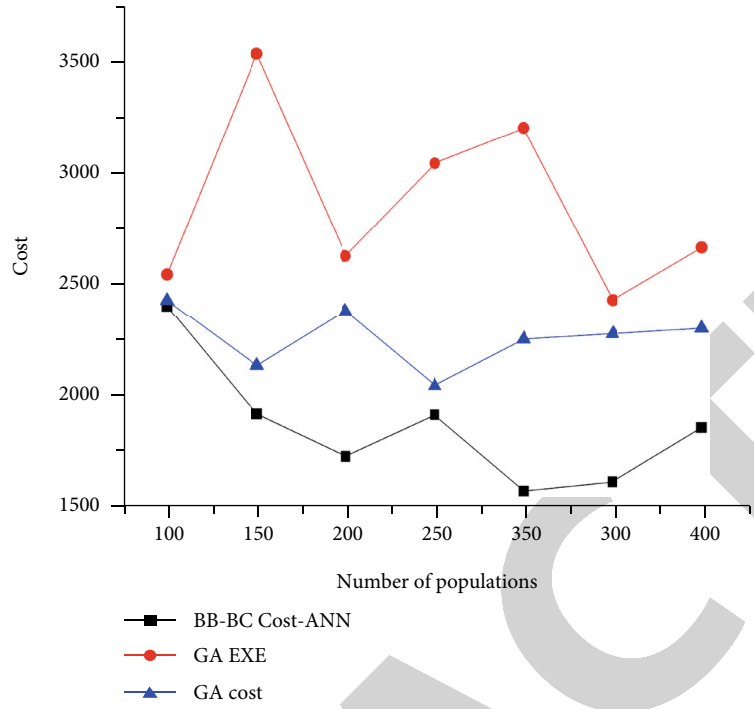

Figure 8: Comparison of cost in $ with increasing population size in the proposed algorithm.

4. Results and Discussions

In this work, experiment and results are showcased with comparative analysis with the existing works. For the simulation, cloudsim 4.0 is used. The experiment uses a real word SWF log file from the planned lab. The simulation is done taking the following performance matrix into mind:

(i) Cost ($)

(ii) Average start time (ms)

(iii) Average finish time (ms)

(iv) Execution time (ms)

(v) Network delay (ms)

Tables 1 and 2 show the configuration of various types of task and resource in terms of virtual machine taken into consideration. Where the tasks are of small large and medium type, similar resources are categorized into low-medium- and high-computing machines. For simulation, 5 data centers are taken into consideration with 2 hosts each. The cost for each of the resources is taken from AWS (Amazon Web Services) for a simulation purpose. The above discussed simulation parameters are used to formulate a cloud environment where the resources are spread geographically all over the globe. The tasks generated by smart city applications can vary from small, average, or large tasks, so in order to formulate this, the simulation uses the tasks varying from 200 instructions to 4000 instructions in a task.

Figures 5(a) and 5(b) show the comparative study of average start time with a scaling task load. The experiment is compared with change in resources to study the performance with increasing resources. Figures 6(a) and 6(b) show the comparative study of average finish time with a scaling task load. The experiment is compared with change in resources to study the performance with increasing resources. This shows most of the tasks get an early start without a long waiting time.

The proposed algorithm performs better than the existing algorithms in the case of the start time and finish time.

Figures 7(a) and 7(b) show the comparison of execution cost in dollars for using the resources with increasing tasks. The comparative study is also performed with scaling resources. In order to verify the performance of the proposed model, the performance with increase in population is also performed as shown in Figure 8.

Figures 9(a) and 9(b) show a study of network delay experienced by tasks before execution. This plays an important role in total execution time which is the sum of network delay and execution time. The comparative analysis shows that the proposed BB-BC-cost ANN performs better than GA with the least network delay with increasing task and scaling cloud resources. Similarly, Figures 10(a) and 10(b) show a comparative analysis of execution time of the tasks which shows that the tasks are complete on high-computing machines with the least

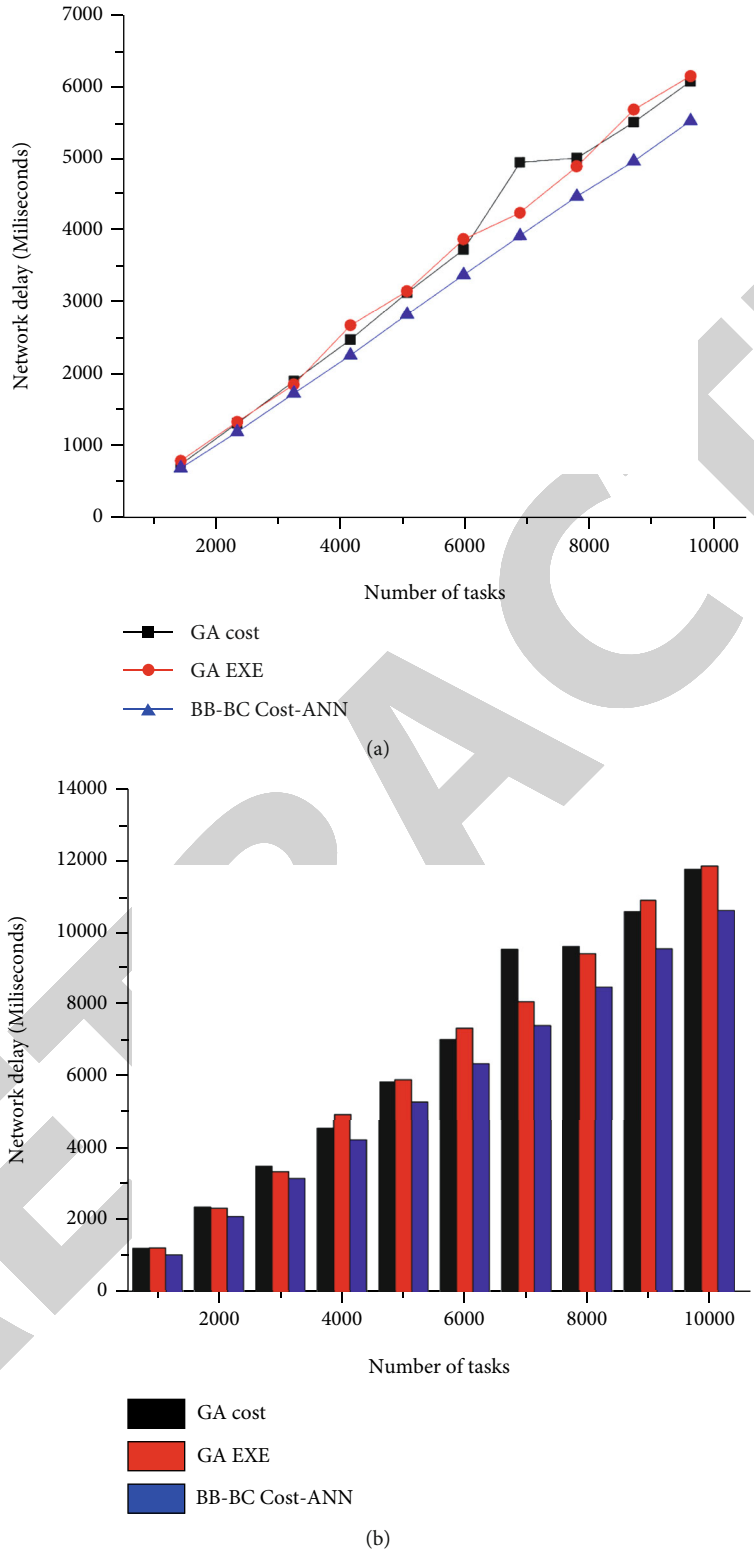

Figure 9: (a) Comparison of network delay for 5 VMs with increasing tasks. (b) Comparison of network delay for 10 VMs with increasing tasks.

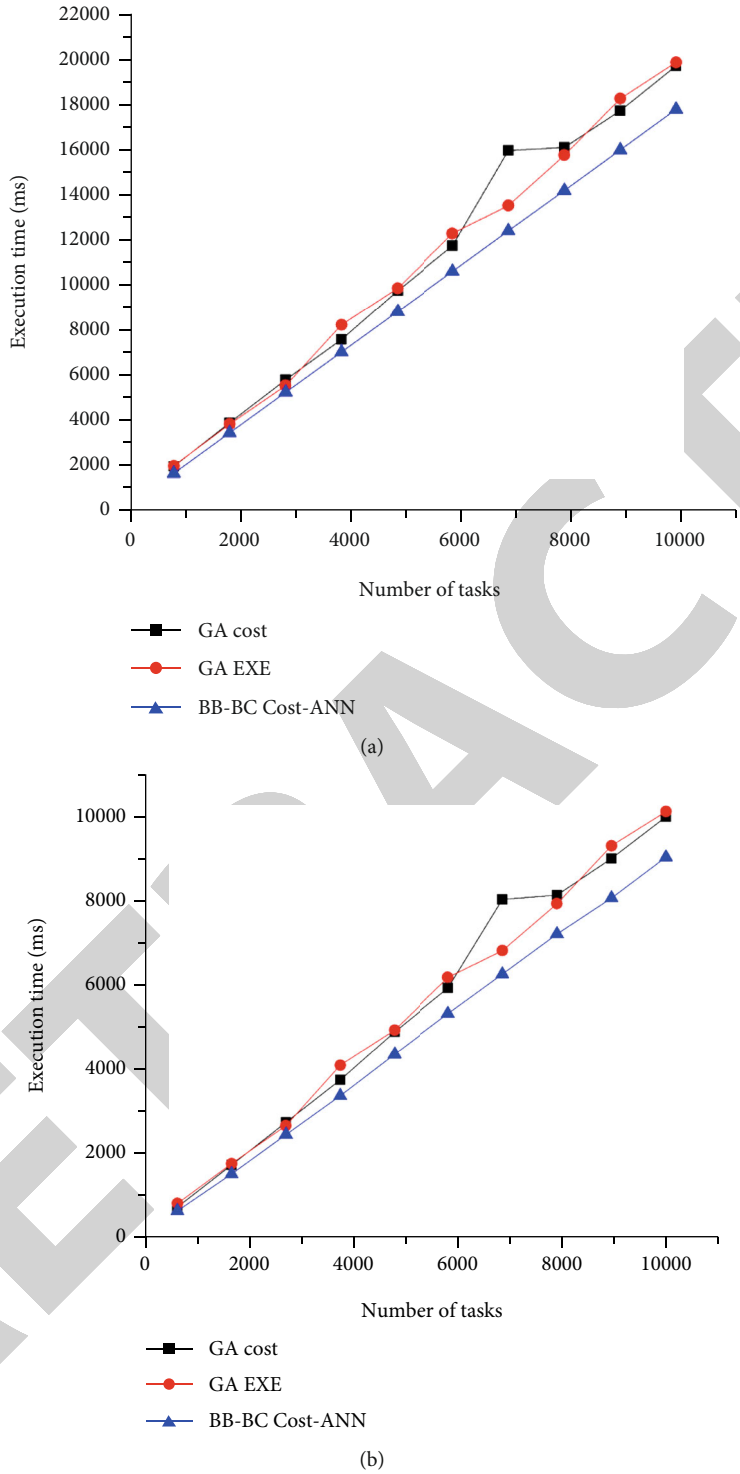

Figure 10: (a) Comparison of total execution time for 5 VMs with increasing tasks. (b) Comparison of total execution time for 10 VMs with increasing tasks.

execution time rather than some tasks with high delay and other with the least delay.

5. Conclusion and Future Works

In this work, an efficient resource allocation scheme is presented for cloud applications in smart cities. The proposed algorithm tries to improve the cost and network delay of the cloud environment and managing to improve the performance of the system at the same time using machine learning techniques. The proposed BB-BC-cost model outperforms the existing techniques. The results are compared using the finish time, start time, and power as the performance matrix. In this work, we focused on task scheduling

in the cloud using BB-BC and ANN algorithms to reduce the simulation time of finding the best solution and reducing cost at the same time. The work has proposed an ANN model trained with an optimized solution of BB-BC to predict the cost-efficient solution in a constant time. In the future, we plan to use the proposed model for resource scheduling on host and migration approaches for better utilization of resources to improve the running cost of cloud applications with the optimal resource. The proposed model has been evaluated under underloaded and overloaded conditions. The result shows that the proposed model proved to provide better results than the existing models.

Data Availability

The dataset for simulation is supported by parallel http://workload.com for real-time analysis.

Disclosure

We certify that the submission is the original work and is not under review at any other publication.

Conflicts of Interest

The authors have no conflicts of interest to declare, and there is no financial interest to report.

Authors' Contributions

Punit Gupta and Dinesh Kumar Saini have designed the model for the cloud with a mathematical proof. Ravindra R Kaikini and Salma Rahman have contributed in simulation and analysis of result. All co-authors have seen and agree with the contents of the manuscript.

References

[1] C. Zhao, Q. Liu, J. Xie, S. Zhang, and J. Hu, "Independent tasks scheduling based on the genetic algorithm in cloud computing," in *in Wireless Communications, Networking, and Mobile Computing, 2009. WiCom'09. 5th International Conference*, pp. 1–4, Beijing, China, 2009.

[2] Q. Wu, M. Zhou, Q. Zhu, Y. Xia, and J. Wen, "MOELS: multiobjective evolutionary list scheduling for cloud workflows," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 166–176, 2020.

[3] M. Rajasree and A. A. Beegom, "A particle swarm optimization based Pareto optimal task scheduling in cloud computing," in *in International Conference in Swarm Intelligence*, pp. 79–86, Springer, 2014.

[4] N. A. B. M. Shaari, L. Y. Por, T. F. Ang, and C. S. Liew, "Dynamic pricing scheme for resource allocation in multicloud environment," *Malaysian Journal of Computer Science*, vol. 30, no. 1, pp. 1–17, 2017.

[5] Z. Wang and A. Liu, "Grid task scheduling based on adaptive ant colony algorithm," in *in Management of e-Commerce and e-Government, 2008. ICMECG'08. International Conference*, pp. 415–418, Nanchang, China, 2008.

[6] H. G. E. D. H. Ali, I. A. Saroit, and A. M. Kotb, "Grouped tasks scheduling algorithm based on QoS in cloud computing network," *Egyptian Informatics Journal*, vol. 18, no. 1, pp. 11–19, 2017.

[7] J. Wang, P. Korambath, I. Altintas, J. Davis, and D. Crawl, "Workflow as a service in the cloud: architecture and scheduling algorithms," *Procedia Computer Science*, vol. 29, pp. 546–556, 2014.

[8] A. Thomas, G. Krishnalal, and V. J. Raj, "Credit based scheduling algorithm in cloud computing environment," *Procedia Computer Science*, vol. 46, pp. 913–920, 2015.

[9] A. V. Lakra and D. K. Yadav, "Multi-objective tasks scheduling algorithm for cloud computing throughput optimization," *Procedia Computer Science*, vol. 48, pp. 107–113, 2015.

[10] S. Zhan and H. Huo, "Improved PSO-based task scheduling algorithm in cloud computing," *Journal of Information & Computational Science*, vol. 9, no. 13, pp. 3821–3829, 2012.

[11] C. Y. Liu, C. M. Zou, and P. Wu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing," in *In 2014 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, pp. 68–72, Xi'an, China, 2014.

[12] M. A. Alworafi, A. Dhari, A. A. Al-Hashmi, and A. B. Darem, "An improved SJF scheduling algorithm in cloud computing environment," in *In 2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, pp. 208–212, Mysuru, India, 2016.