Chapter 5
Resource Management Framework Using Deep Neural Networks in Multi-Cloud Environment

S. Brilly Sangeetha, R. Sabitha, B. Dhiyanesh, G. Kiruthiga, N. Yuvaraj, and R. Arshath Raja

5.1 Introduction

Today's enormous growth in Information Technology paves the way for the development of various mechanisms for transmitting information. The multi-cloud model eliminates strategies to communicate directly with a server for all kinds of services as the cloud computing model. This model is preferable instead to the online mode of communication. The content of information stored on the websites is retrieved directly by means of relevant requests. The systems that can define the online data recovery procedures are integrated here, making it easier for the user to recover the required assets with the help of this model [1–5].

In this model, the main factor to take into account is asset management, which is considered the model's most important task. The next major factor is the load

S. B. Sangeetha (✉)
Department of Computer Science and Engineering, IES College of Engineering, Thrissur, India

R. Sabitha
Department of Electronics and Communication Engineering, Hindusthan College of Engineering and Technology, Coimbatore, India

B. Dhiyanesh
Department of Computer Science and Engineering, Hindusthan College of Engineering and Technology, Coimbatore, India

G. Kiruthiga
Department of Computer Science and Engineering, IES College of Engineering, Thrissur, India

N. Yuvaraj · R. A. Raja
Training and Research, ICT Academy, IIT Madras Research Park, Chennai, India

R. Nagarajan et al. (eds.), *Operationalizing Multi-Cloud Environments*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-74402-1_5

balancing task, which is ultimately intended to ensure that the load is distributed uniformly, allowing the entire group of users to use the resources efficiently [6, 7].

Ultimately, the maximum use of the resources would improve the throughput factor and thereby reduce the energy consumption rates. Another benefit of the load balance strategy is that it reduces the response time substantially. This work has proposed a load balancing strategy for improving resource delivery and the quality of service factors, based on the various strategies [8, 9].

Adequate task scheduling is important to ensure high cloud productivity. Scheduling for the disseminated frameworks is an entire NP problem, so that in such circumstances customary booking techniques do not yield significant productivity [10, 11]. Scheduling is a critical concern for suitable situations that have to address some errands in numerous assets while improving the use of assets and the make-up. Task booking system charts for the most part provide cloud-based assignments to accessible assets as shown by their compliance qualities [12–14].

In addition, problem strategies are sorted into heuristic and conjectured. Heuristic algorithms are suspected of the status of assets as workable, such as the load or the length of employment prior to the schedule of employment. Surveyed scheduling methods depend on similar information data and the formal computer model as ideal reservation strategies, but by reducing arranged space, they defeat the NP complement of ideal schedulers [15, 16].

The collection of all information clearly presents new practical problems, where the valuable heuristic systems are [17, 18]. Swarm approaches to knowledge in a dispersed schedule are extremely prevalent and impressive. The main reason is that progress problems can be illuminated without the need for excess data on the issue.

This article therefore uses the need for the properties of gray wolf optimization (GWO) on a genetic algorithm (GA) model, which can effectively compute the selection of the optimal path with iterative training instances. This model has an optimal routing principle, which takes on less compute complexity and more energy efficiency from the cloud-based dynamic sensor VMs.

The outline of the paper is given as follows: Sect. 5.2 provides the problem of the study. Section 5.3 discusses the proposed scheduler. Section 5.4 concludes the entire work.

5.2 Problem Formulation

The problem formulation [19] for the present study is considered under the following assumptions: (1) prior knowledge on computational time period of each task and (2) similar overheads prior scheduling of tasks.

Task constraint is modeled as:

$$\gamma_i = \frac{c_i}{p_i}, i = 1, 2, \ldots N \tag{5.1}$$

$$r_{ij}\left(\forall i\right)=\begin{cases}0 & j=1\\ d_{ij-1} & j=2,3,\ldots,n_i\end{cases} \tag{5.2}$$

$$d_{ij}=r_{ij}+p_{ij},\ i=1,2,\ldots,N,\ j=1,2,\ldots,n_i \tag{5.3}$$

$$s_{ij}<s_{i'j'}<f_{ij}, \tag{5.4}$$

$$\text{if priority}\left(\tau_{i'}\right)>\text{priority}\left(t_i\right)\text{and}r_{i'j'}\geq r_{ij}\forall i,j,i',j' \tag{5.5}$$

where

$i2$ – Task index
$j2$ – Index of j^{th} task executed
N – Total tasks
T – Scheduled time
n_i – Total task executed
r_{ij} – Release time (j) of task τ_i
s_{ij} – Start time (j) of task τ_i
f_{ij} – Finish time (j) of task τ_i
i – Time required for executing τ_i with τ_i

The following are the metrics utilized for estimating the resource allocation based on the available resources:

Makespan: The general requirement that shows the entire duration of the tasks. When the machining rate is at lowest, the method of task planning can be efficient for VM. The service quality and scheduling can be improved if the makespan is smaller.

$$W=\frac{\sum_{i=1}^{N}w_i}{N} \tag{5.6}$$

where

w_i – Total completion time of a task t_i

Average Waiting Time represents the performance of total throughput and overall processing capability of cloud.

$$M=\frac{\min\sum_{i=1}^{N}\tau_i}{N} \tag{5.7}$$

where

τ_i– Waiting time for a task t_i

CPU Utilization Rate of VMs represents the actual resource requirements required by VMs to consume and allocate the task in each VM at every instant of time.

$$U_{ij}\left(t_i\right) = \frac{\sum_{j=1}^{v_i} C_{ij}^{con}\left(t_i\right)}{\sum_{j=1}^{v_i} C_{ij}^{alc}\left(t_i\right)} \tag{5.8}$$

where

$C_{ij}^{con}\left(t_i\right)$ – Consumed tasks at time multi-val t_i
$C_{ij}^{alc}\left(t_i\right)$ – Allocated tasks at time multi-val t_i

Failure Tasks Scheduling Rate represents the cloud stability.

$$Q(\%) = \text{Time processing tasks / Total time} \tag{5.9}$$

5.3 Proposed Method

There are various VMs in the cloud computing system. Different requests from the users are received, and separate capacities for the processing of such received requests are identified and incorporated. The time of the task here depends on the processing power of the virtual machines (VMs).

Figure 5.1 shows the architecture of the IoT-multi-cloud proposed to sense, collect, and transmit the data from the source VM to the destination VM. Three different levels, including sensing, data, and control plane, are used in this architecture.

The detection aircraft is the collection of IoT inputs, which is a heterogeneous device that senses and gathers input data at a greater rate. In the sensor plane, the IoT VMs must be fixed and the proposed architecture should not be mobile. Greater input data collection creates explosive traffic so that the need for an adequate transmission path is critical across cloud.

The proposed architecture uses a control plane consisting of the GWO model integrated with GA algorithm to alleviate this challenge. The deep sense model is so responsible that the connection or path is not congested with path assignment, packet selection, and flow control. In some cases, bursty traffic is checked with optimum packet selection through routing paths to ensure better packet delivery by the cloud.

The data plane is then used to transfer massive IoT data into the destination VM or sink or the gateway in the form of packets. A GWO [11] model integrated with the GA model, which aligns the goal of the proposed mechanism to improve the scheduling in the cloud, controls the routing between cloud VMs.

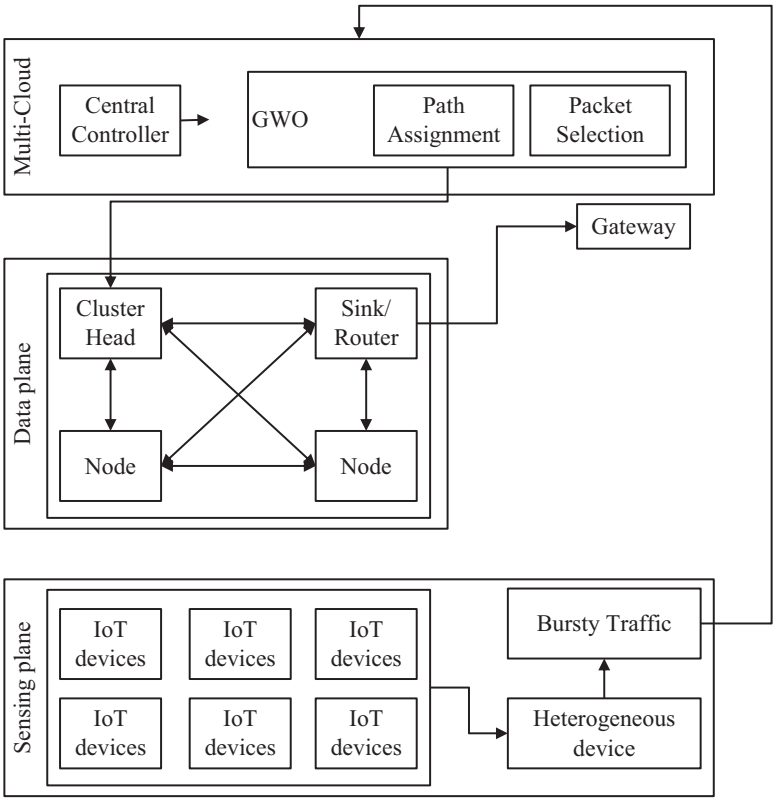

Fig. 5.1 Architecture for minimizing the time overhead using GWO

In the case of cloud data centers, the variations in computing power of the VM in question are seen, which means that there are significant differences in VMs during similar tasks. Tasks are generally different, and therefore, the performance efficiency of the various VMs makes the allotment of tasks to these VMs possible. Excess tasks would be assigned to VMs with better performance capacity. The above reason turns out to overload certain VMs, while the other machines remain idle. Ultimately, this issue would result in a waste of resources.

The efficiency of the cloud data centers is usually affected and reduced by the load imbalance of the VMs. This work thus focuses on the average load conditions applicable for the VMs, referring to the enhanced GWO to enhance the use of resources.

5.3.1 Scheduling Using GWO

The time multi-val between the submission of the IoT task in VM and its answer is viewed as defining the response time. The load balance factor is one of the factors influencing the process of reducing response time and increasing the reactivity of VMs. The easy procedure here is adopted to minimize the machining span, and the response time involves transferring the tasks from an overloaded VM to an overloaded VM, which is thus called the load balance procedure. In order to maintain the load conditions, it is important that the VM informs about its capacity to balance the load correctly.

The schedulers will be hosted based on the management nodes, from the computer nodes to the storage nodes. Based on the requests the scheduler selects relevant compute nodes for the VMs, it monitors requests that have been sent to a VM.

Task scheduling is a technique in cloud computing to ensure a dedicated resource is assigned work and that assigned work is completed. The resources include virtual computer elements or hardware. The programming activity in turn is performed by a programmer. The scheduler assigns multiple users to occupy a certain part of a resource for a QoS. The GWO scheduler allows the computer system to try multitasking for each CPU.

The scheduler prioritizes each task according to the user's needs and therefore the multitasks in parallel distributed applications set the schedule of work over idle VMs to complete the process soon. The main problem with task execution lies in the increase of parallelism, as it depends on another task to perform a task in cloud computing. Figure 5.2 shows the process of scheduling using GWO.

5.3.2 Load Balancing

This work classifies the load balance approach into two different phases. The first phase relates to the two-tier task planning strategy, which focuses on the dynamic needs of each user and the achievement of high resource utilization through the adopted load balance mechanism. In this case, load balances are used to map the IoT tasks to the entire VMs, followed initially by the VMs in the resource hosting tasks with the objective to improve the overall Cloud-IoT performance.

In the first phase itself, certain investigations, like identification of the CPU uses and the determination of memory requirements, must be completed with a determination of the number of available cycles, etc. The second stage involves determining the resources available and the amount of resources needed in the immediate future; on the basis of the resource requirements which instances would be either discarded or added, this would finally be seen as the user's prescribed status.

Load checking on VMs should be carried out regularly; this checking is wisely performed by the proposed algorithm; on the basis of observations, the following strategy is used to deduce load migration. It is the primary task of this algorithm to

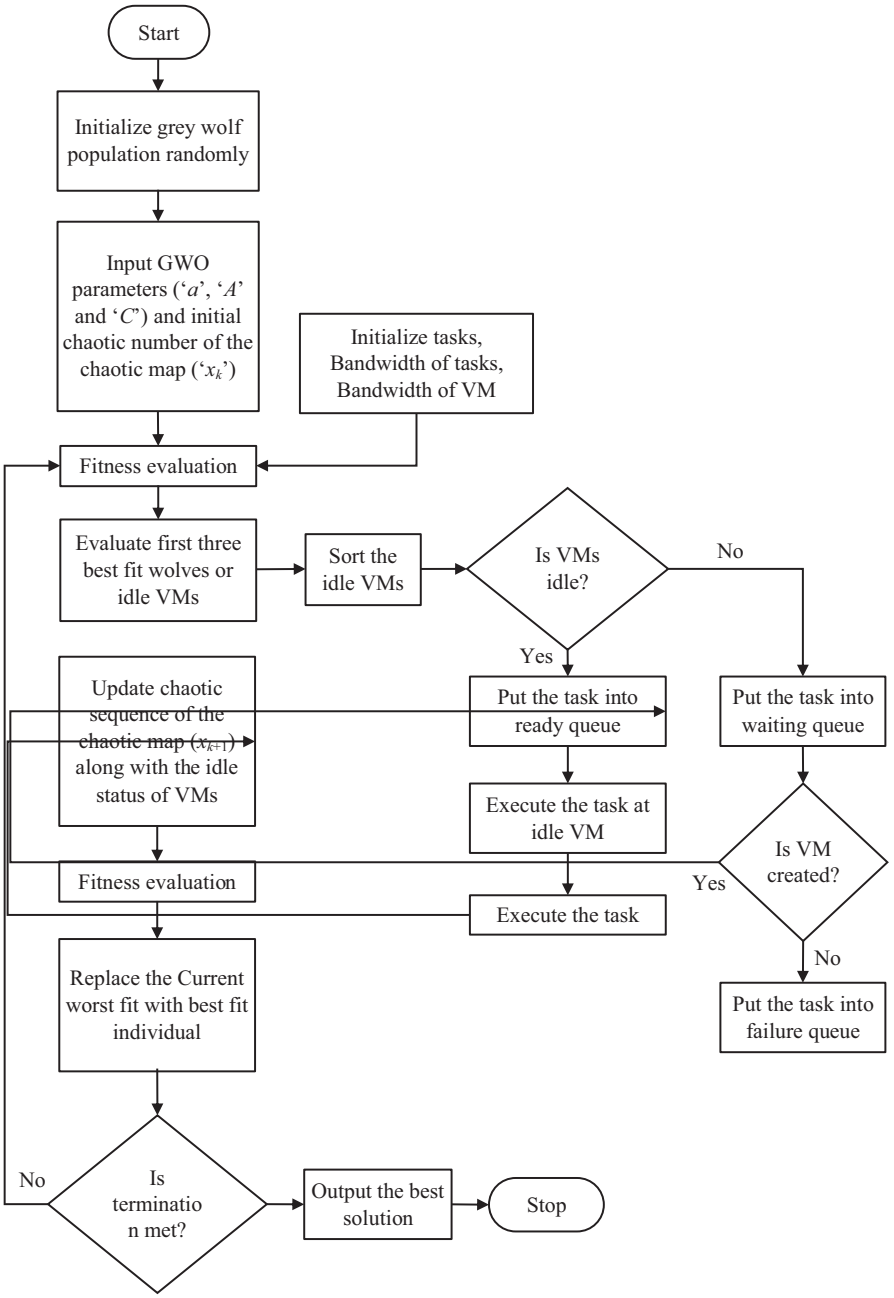

Fig. 5.2 Slot scheduling architecture

Pseudo Code of Cloud Scheduling

```
Initialize the wolfs
Initialize the VM in clouds
Initialize the IoT devices
Assign the value of threshold
While all VMs is found to be optimally balanced with loads
(apply GWO Algorithm – See pseudocode of GWO)
if (load < threshold)
if (load < threshold)
{
 do
 {
 Assign the scheduled task to the VM present in cloud
 Sort the scheduled task
 Estimate and update the distance of Wolfs
 } while (load < threshold)
}
else
Update the particles with updated solution
Search for similar neighbor VM
Finally check the load balanced VMs
Pseudocode of GWO
For i = 1 : n
 z_i ← i^th row vector elements of F;
 Generate initial search agents G_i (i=1, 2,….,n)
 Initialize the vector's a, A and C
 Estimate the fitness value of each hunt agent
Repeat
For i=1: G_s (pack size of grey wolf)
 Update the current hunt agent location
End for
Calculate the fitness value of the entire obtained hunt agents
Update the values of the vectors a, A and C
Update the value of first three best hunt agent
I=I+1
Check if I ε I_max (maximum iterations i.e. the Stopping criteria)
Output largest k eigenvectors G
End For
```

identify the loading conditions of VMs on the basis of the differentiation of machines from loaded VM and loaded VMs, followed by transferring loads from loaded VM to loaded VMs.

This process ensures that the loads are evenly distributed. This reduces the response time and improves the utilization of resources through a uniform distribution of load. This task for observation is like the bee movement, which periodically searches for loads. This search task continues to reach a threshold value. When the threshold value is reached, the optimization identifies the minimum loaded VM, and the task is set to minimum loaded VM.

5.4 Results and Discussions

This section discusses the experimental studies to concentrate on the efficacy of GWO. Optimization of GWO is verified using a range of performance measures including a package delivery ratio, average end-to-end delay, standardized payload routing and IoT, and cloud network life. To our best, it is the first IoT-multi-cloud GWO technique, and therefore a comparison is carried out between GWO routing with the Ant Colony Optimization (ACO) model and Bees Swarm Optimization (BSO). The following are the five performance metrics:

Packet delivery ratio (*PDR*): PDR is defined as the ratio from the total amount of packets generated and transmitted by IoT-source VMs at the Cloud-based base station.

End-to-end average delay (*EAD*): The delay is determined as the late time when packets from source IoT VMs are successfully transmitted to the cloud sink VMs and back to the source IoT VMs. This time delay includes queuing at cloud VMs, IoT cache latency, air propagation delay, MAC Cloud layer transmission delay, and GWO transformation time for tracking.

Normalized routing payload (NRP): For each data packet that is being delivered, NRP is the total number of control packets transmitted to the cloud sink VM. When the data is sent via one hop, the packets generated by the GWO occur.

Normalized MAC payload (NMP): NMP is defined as the total number of packets with address resolution, routing and control packets, and overhead packets that include overheads generated for each IoT data packet in the cloud MAC layer.

Network lifetime: IoT-multi-cloud network lifetime is defined as the total time taken to simulate the IoT cloud from the start to the last packet transmitted following the death of IoT VMs because VMs lie in a remote location and fail to receive continuous power.

The GWO routing efficiency with routing loads is considered important to the PDR and the EAD. MAC payload is an effective measure of wireless media for data streams in which the measurements are independent.

5.4.1 Simulation Results and Discussions

The data packet rate of the 10 IoT-source VM for session generation is between 45 and 54 Mbps. However, the optimal selection of control and data packet transmission via GWO makes the network congestion without affecting communication; this can lead to an increase in network congestion.

The simulation results of PDR in different sessions, which includes packet transmission at IoT source VMs and receipt of the cloud sink VMs, are shown in Fig. 5.3. The results show that the PDR is relatively less than the increase in pause times when pause times are reduced. However, the PDR is reduced by an increasing number of sessions from 10 to 40 than conventional ANN and enhancement.

The increase in performance in the GWO model is because of the efficient calculation of data transmission paths. The conventional systems fail to calculate the routes where the generated data rates on IoT devices do not coincide. GWO model performance is thus considered as stable and stabilizes the routing connection with increased route stability and minimal connection failure.

Figure 5.4 shows the EAD for various sessions when the EAD is reduced and with the increased break times it increases. The EAD is however much lower than ANN and reinforcement education with increasing sessions. The increased delay is due to the selection of longer routes while EAD is calculated, which causes severe cloud network congestion.

Thus, meta-heuristic algorithm learning does nothing to balance the loads. However, the GWO model, after certain iterations, increases the computing capability of the routes by minimizing the data transfer rate at that time. In contrast,

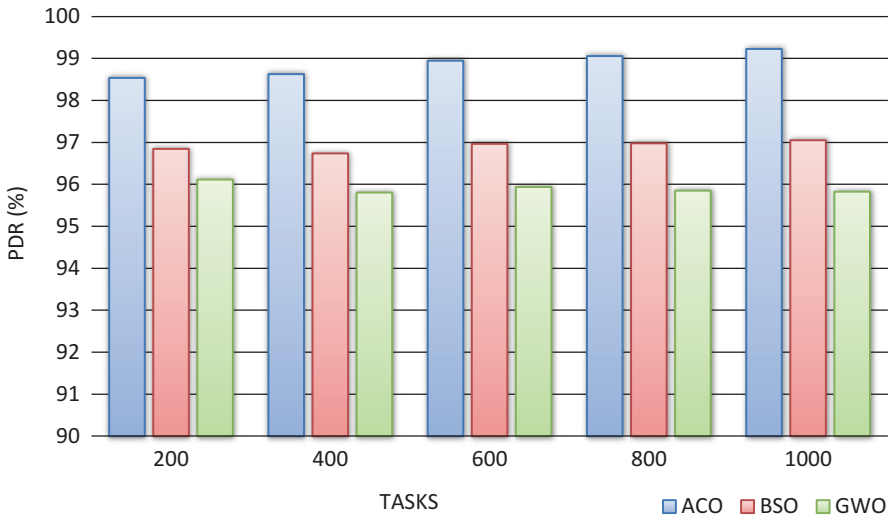

Fig. 5.3 Packet delivery rate

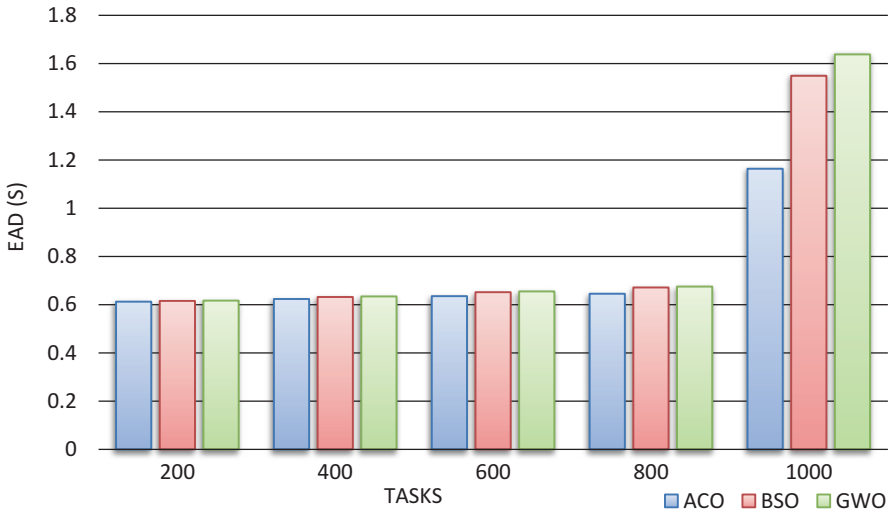

Fig. 5.4 Average end-to-end delay

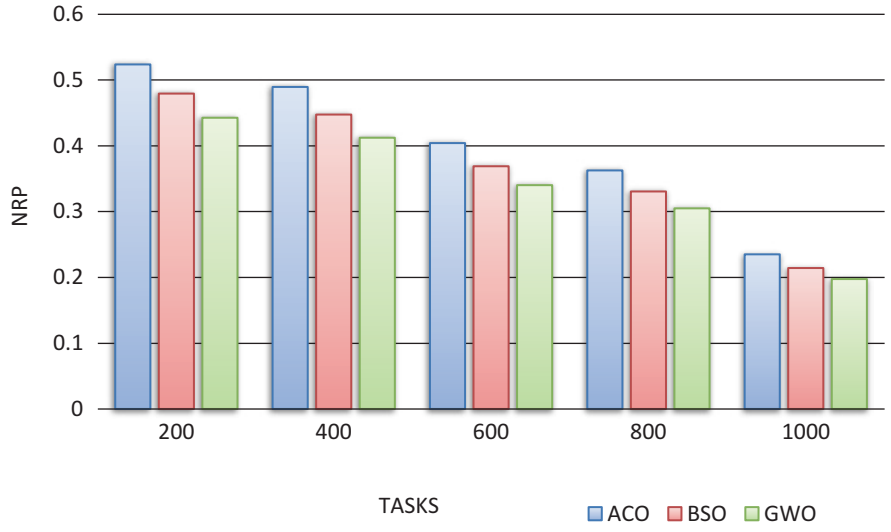

Fig. 5.5 Normalized routing payload

the fact that cloud VMs are delayed increases the EAD overall without affecting the kernels' stability.

The NRP and NMP for various sessions, which lower NRP and NMP for increased pauses, are presented in Figs. 5.5 and 5.6. The NRP and NMP

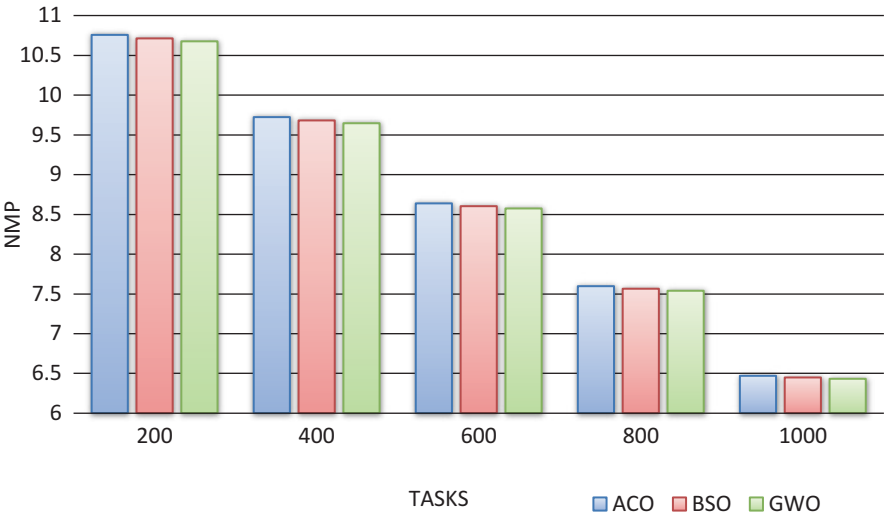

Fig. 5.6 Normalized MAC payload

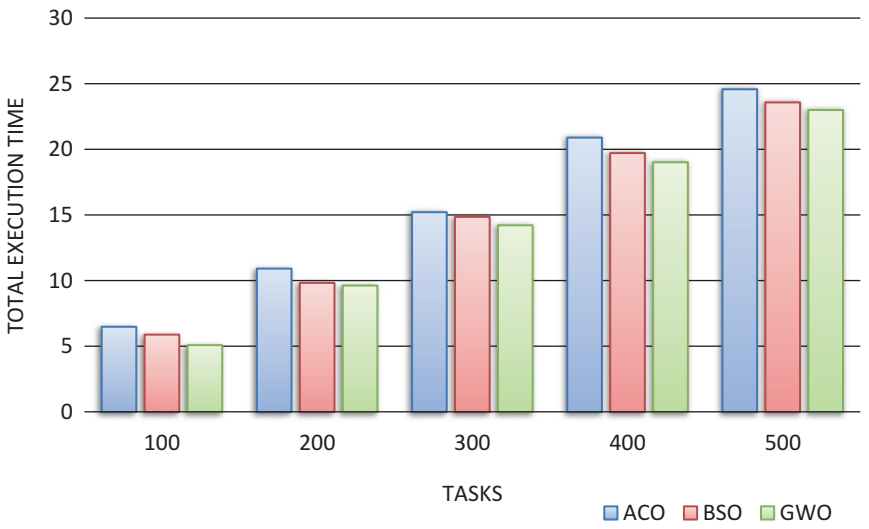

Fig. 5.7 Makespan for different number of tasks

significantly decrease with more and more sessions than ANN and enhancement learning. GWO handles address, routing, control packets, and overhead packets effectively with higher computation speeds.

Figure 5.7 shows the average overall makespan for different scheduling methods with different tasks. The GWO strategy consists of low make-up in comparison with the GWO strategy. The GWO does not waste energy with lower length tasks. The performance achieved by GWO has fallen by 13%. The main reason for this is that

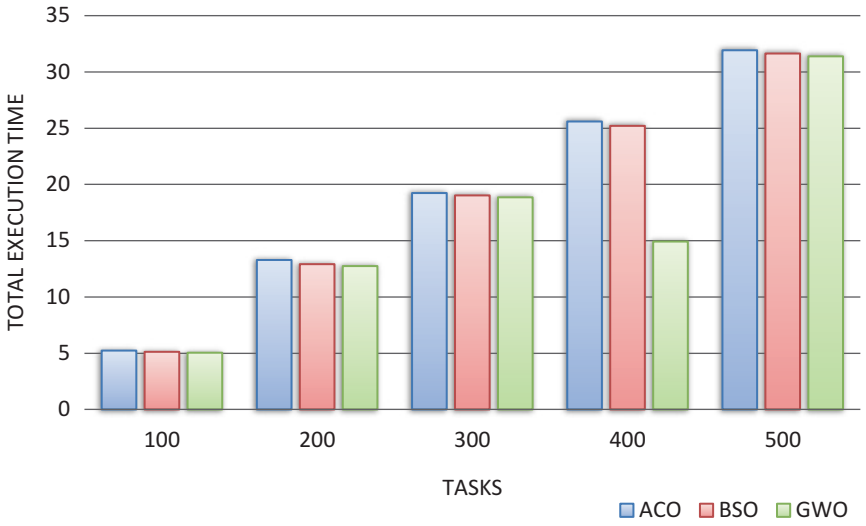

Fig. 5.8 Total execution time for different number of tasks

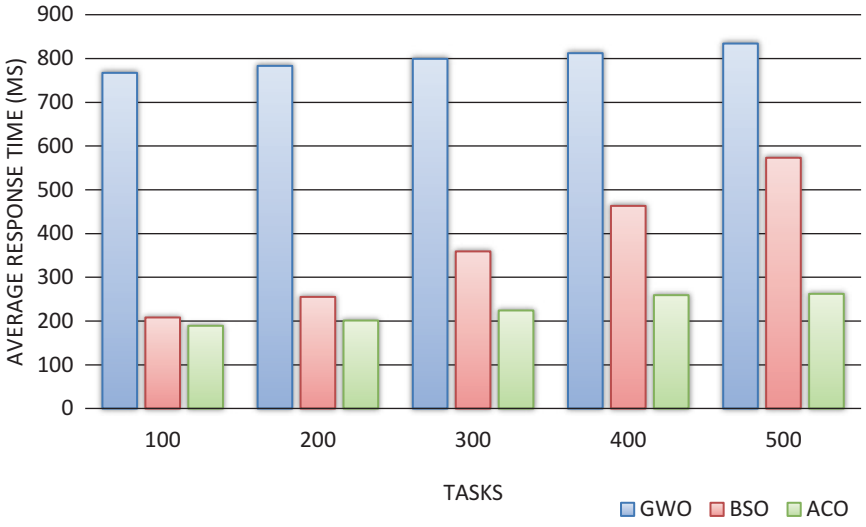

Fig. 5.9 Average response time in multi-cloud environment

the planned strategy takes on the characteristics of the tasks, total implementation time.

The resources are sufficient while there are small numbers of tasks. It takes a very long execution time (Fig. 5.8) to compare the GWO strategy. With discovering the whole area of search, wolf moves around the best position in the GWO algorithm. Compared to the entire method, GWO provides an average of 8–16% reduction in the total execution time. The metrics like increased average response time (Fig. 5.9),

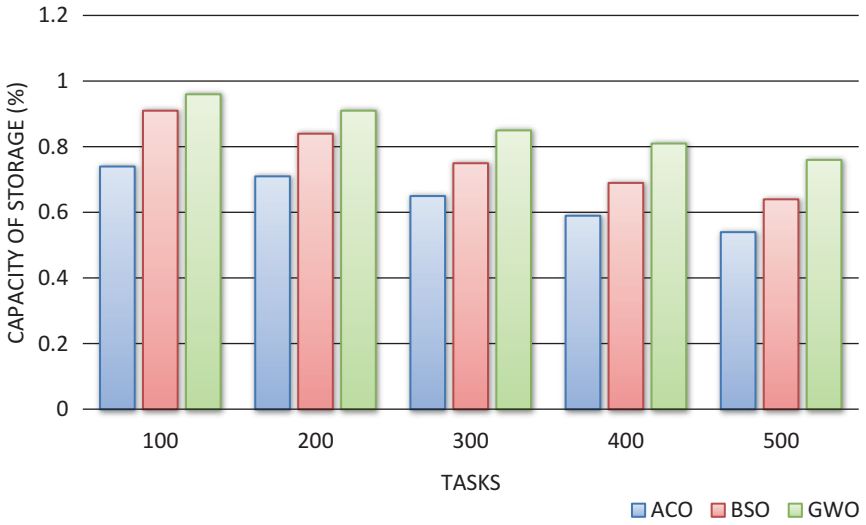

Fig. 5.10 Capacity of storage (%) in multi-cloud environment

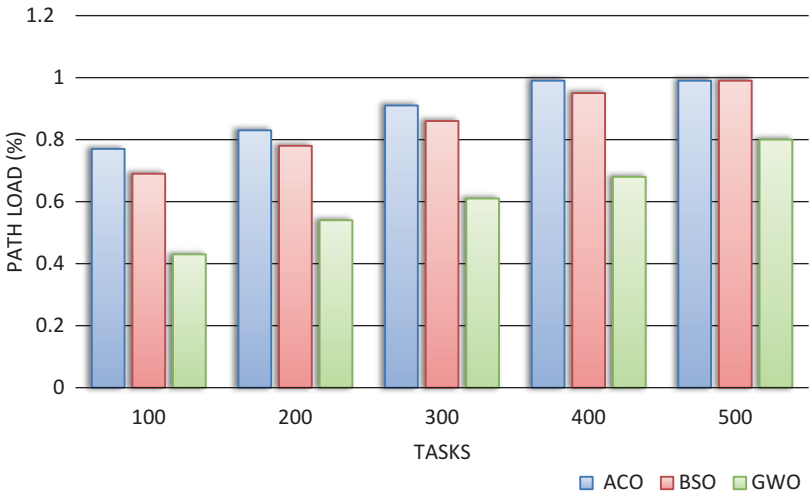

Fig. 5.11 Path load (%) in multi-cloud environment

increased storage capacity (Fig. 5.10), reduced path load (Fig. 5.11), and reduced cost (Fig. 5.12) shows improved performance by proposed GWO than existing methods.

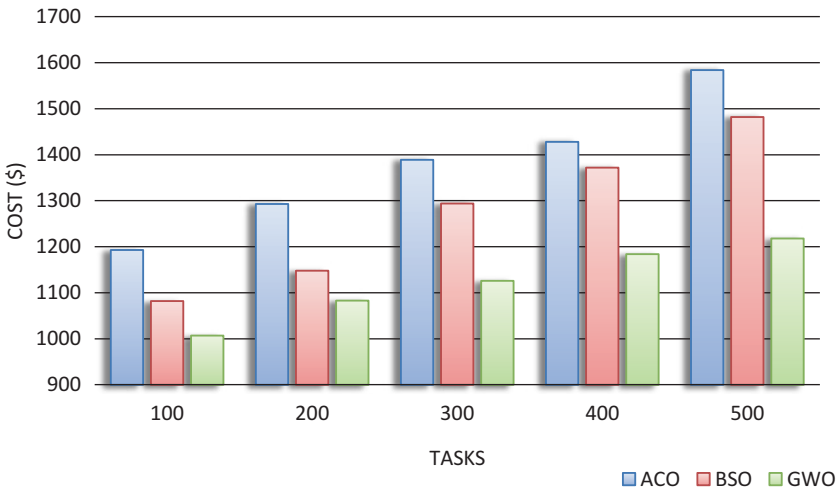

Fig. 5.12 Cost ($) in multi-cloud environment

5.5 Conclusions

In this paper, we framed a GWO model to enhance IoT-multi-cloud routing performance, thus increasing routing the performance of VMs. By adapting to the data acquisition speed by IoT VMs, GWO routes the packets effectively. IoT-multi-cloud multi-connection is effectively managed so that the data collection and transmission lack design defects. The results of the simulation show that the EAD is small and longer, with lower PDR, the scalability of the GWO is decreasing, and the risk of packet loss is high. But GWO reduces PDR on long routes which biases the samples and ensures a balanced scalability. The GWO routing model has a highly satisfactory scalability for shorter routes. The performance of the GWO model with high-speed packet routing provides improved packet transmission via Cloud, increasing IoT-multi-cloud longevity in IoT and Cloud sensor VMs as a result of increased residual energy.

References

1. Pham, X. Q., & Huh, E. N. (2016, October). Towards task scheduling in a cloud-fog computing system. In 2016 18th Asia-Pacific network operations and management symposium (APNOMS) (pp. 1–4). IEEE.
2. Basu, S., Karuppiah, M., Selvakumar, K., Li, K. C., Islam, S. H., Hassan, M. M., & Bhuiyan, M. Z. A. (2018). An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment. *Future Generation Computer Systems, 88*, 254–261.
3. Boveiri, H. R., Khayami, R., Elhoseny, M., & Gunasekaran, M. (2019). An efficient swarm-intelligence approach for task scheduling in cloud-based multi-net of things applications. *Journal of Ambient Intelligence and Humanized Computing, 10*(9), 3469–3479.

4. Ismail, L., & Materwala, H. (2018). Energy-aware VM placement and task scheduling in cloud-IoT computing: Classification and performance evaluation. *IEEE Multi-Net of Things Journal, 5*(6), 5166–5176.
5. Nguyen, B. M., Thi Thanh Binh, H., & Do Son, B. (2019). Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment. *Applied Sciences, 9*(9), 1730.
6. Al-Turjman, F., Hasan, M. Z., & Al-Rizzo, H. (2019). Task scheduling in cloud-based survivability applications using swarm optimization in IoT. *Transactions on Emerging Telecommunications Technologies, 30*(8), e3539.
7. Vashishth, V., Chhabra, A., & Sood, A. (2017, January). A predictive approach to task scheduling for big data in cloud environments using classification algorithms. In 2017 7th multi-national conference on cloud computing, data science & engineering-confluence (pp. 188–192). IEEE.
8. Hasan, M. Z., Al-Rizzo, H., Al-Turjman, F., Rodriguez, J., & Radwan, A. (2018, December). Multi-net of things task scheduling in cloud environment using hybrid wolf flame optimization. In 2018 IEEE global communications conference (GLOBECOM) (pp. 1–6). IEEE.
9. Wu, G., Bao, W., Zhu, X., & Zhang, X. (2018). A general cross-layer cloud scheduling framework for multiple iot computer tasks. *Sensors, 18*(6), 1671.
10. Yin, L., Luo, J., & Luo, H. (2018). Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. *IEEE Transactions on Industrial Informatics, 14*(10), 4712–4721.
11. Zhao, X., & Huang, C. (2020). Microservice based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network. *IEEE Access, 8*, 56680–56694.
12. Lin, W., Peng, G., Bian, X., Xu, S., Chang, V., & Li, Y. (2019). Scheduling algorithms for heterogeneous cloud environment: Main resource load balancing algorithm and time balancing algorithm. *Journal of Grid Computing, 17*(4), 699–726.
13. Xu, J., Hao, Z., Zhang, R., & Sun, X. (2019). A method based on the combination of laxity and ant colony system for cloud-fog task scheduling. *IEEE Access, 7*, 116218–116226.
14. Gawali, M. B., & Shinde, S. K. (2018). Task scheduling and resource allocation in cloud computing using a heuristic approach. *Journal of Cloud Computing, 7*(1), 4.
15. Cheng, N., Lyu, F., Quan, W., Zhou, C., He, H., Shi, W., & Shen, X. (2019). Space/aerial-assisted computing offloading for IoT applications: A learning-based approach. *IEEE Journal on Selected Areas in Communications, 37*(5), 1117–1129.
16. Li, W., Liao, K., He, Q., & Xia, Y. (2019). Performance-aware cost-effective resource provisioning for future grid IoT-multi-cloud system. *Journal of Energy Engineering, 145*(5), 04019016.
17. Kaur, K., Garg, S., Kaddoum, G., Ahmed, S. H., & Atiquzzaman, M. (2019). KEIDS: Kubernetes based energy and multi-ference driven scheduler for industrial IoT in edge-cloud ecosystem. *IEEE Multi-net of Things Journal*.
18. Deng, W., Yao, R., Zhao, H., Yang, X., & Li, G. (2019). A novel intelligent diagnosis method using optimal LS-SVM with improved GWO algorithm. *Soft Computing, 23*(7), 2445–2462.
19. Gen, M., Cheng, R., & Lin, L. (2008). Tasks Scheduling Models. Network Models and Optimization: Multiobjective Genetic Algorithm Approach (pp. 551–606).