Transactions on Machine Intelligence 2(3) (2019) 191-204

ISSN Online: 2821-1693

Transactions on Machine Intelligence

Journal Homepage: https://tmachineintelligence.ir/index.php/TMI

Migration of Virtual Machines in Heterogeneous Mobile Cloud Computing Using Grey Wolf Optimization Algorithm

R. Zare Farkhadi[1,*], S. Vahabzadeh[2]

[1] Faculty Member, Department of Computer Engineering, Roshdieh Higher Education Institute, Tabriz, Iran
[2] Master's Student, Software Engineering, Roshdieh Higher Education Institute, Tabriz, Iran

ARTICLE INFO

Article History:
Received 12 June 2019
Received in revised form 16 August 2019
Accepted 22 September 2019
Available online 23 September 2019

Keywords:
Cloud Computing, Virtual Machine, Cloud Server, Grey Wolf Optimization Algorithm

ABSTRACT

Mobile cloud computing improves the performance of a mobile phone by executing an application on an efficient cloud server, which can minimize execution time compared to a resource-constrained mobile device. Virtual machine migration in mobile cloud computing brings cloud resources closer to the user, thereby minimizing response time to an application. Such resource transfers are highly effective for interactive and real-time applications. However, the main challenge is to find an optimal cloud server for migration that offers the maximum reduction in computation time. In this paper, a virtual machine migration model based on the Grey Wolf Optimization (GWO) algorithm is proposed for heterogeneous mobile cloud computing systems. The proposed approach optimizes the effectiveness of virtual machine transfer by considering user mobility and the load on cloud servers. The main objective of the proposed approach is to select an optimal cloud server for a mobile virtual machine and to minimize the total number of virtual migrations, which results in reduced job execution time. Additionally, a comprehensive numerical evaluation is presented to assess the effectiveness of the proposed model compared to advanced virtual machine migration policies. The proposed approach is simulated using MATLAB software, and the results are compared with those obtained using the Genetic Algorithm (GA). Experimental results indicate that the Grey Wolf Optimization algorithm has shown improvements in performance compared to the Genetic Algorithm.

1. INTRODUCTION

This paper presents a proposed approach using the Grey Wolf Optimization algorithm to solve the virtual machine migration problem. Initially, the virtual machine migration issue in mobile cloud computing is formally defined. Subsequently, various components of the proposed approach, such as encoding, prey encirclement, hunting, attacking the prey, and prey searching, will be described, followed by the presentation of results obtained from implementing the proposed approach compared to the Genetic Algorithm. All experiments were conducted on a Dell

* Corresponding Author: roya.farkhady@gmail.com
Faculty Member, Department of Computer Engineering, Roshdieh Higher Education Institute, Tabriz, Iran

http://dx.doi.org/10.47176/TMI.2019.191

computer with a Core i7, 2.4 GHz processor, and 6 GB of RAM. All simulations were carried out using MATLAB 2018b.

2. NETWORK MODEL

A mobile cloud computing environment [1,2] is considered where a set of M access points form the backbone network. Each access point is connected to one of L cloud slices, denoted by a set $S = \{S_1, S_2, S_3, \ldots, S_L\}$. It is assumed that multiple access points may share cloud slices. These cloud slices are connected via a backhaul network, with the bandwidth between cloud slice i and j represented as B(i,j). All cloud slices are remotely controlled by a main cloud. A cloud slice server has processing power Ci and memory Mi. Each cloud slice offers P virtual machines, denoted as $V = \{V_1, V_2, V_3, \ldots, V_P\}$user can offload their task to a dedicated virtual machine. It is also assumed that each user is mobile and runs a task on different virtual machines without mutual dependencies among them [3].

3. VIRTUAL MACHINE MIGRATION USING GREY WOLF OPTIMIZATION ALGORITHM

Virtual machine (VM) migration is a crucial aspect of cloud computing, enabling efficient resource management and load balancing. Several studies have proposed methods to optimize VM migration processes. Rabiatul Addawiyah Mat Razali et al. (2014) suggested migrating VMs based on CPU utilization thresholds to improve load balancing and reduce energy consumption[4]. Zhenzhong Zhang et al. (2013) focused on scheduling multiple VM migrations to minimize overhead and accelerate the migration process[5]. Changhyeon Kim et al. (2014) proposed a VM remapping scheme to reduce relocation time in cloud clusters[6]. Gurpreet Singh et al. (2022) developed an adaptive mechanism for VM migration to minimize migration frequency and reduce execution time[7]. These approaches aim to optimize resource utilization, improve load balancing, and enhance overall cloud performance. By implementing effective VM migration strategies, cloud providers can achieve better energy efficiency, reduced job execution times, and improved resource management in their data centers.

In migrating a virtual machine from one cloud slice, the virtual machines must be mapped to new cloud slices to reduce job execution time and interaction time. It has been proven that this problem belongs to the class of NP-hard problems [3,8,9], and thus, no deterministic algorithm can guarantee an optimal solution in polynomial time. Therefore, the proposed approach uses the Grey Wolf Optimization algorithm, which can provide near-optimal solutions within a limited time.

3.1. Encoding

A critical aspect of evolutionary algorithms is the method of encoding and representing a solution. Each solution to the problem in the proposed approach, termed a wolf, must represent the allocation of virtual machines to cloud slices. To this end, an array of integers of length N is used to represent each wolf, where N is the total number of tasks in the cloud slice. The index of each element in the array denotes the virtual machine, and the integer value stored in each array element represents the cloud slice number to which the virtual machine can migrate.

3.2. Fitness Function

All members of the population (wolves) in the Grey Wolf Optimization algorithm are evaluated using a fitness function. The proposed approach uses the following fitness function [3]:

$$F_n = \sum_{\forall k \in C} U_{n,k}^i \tag{1}$$

where F_n is the fitness function value for the n-th member of the population and $U_{n,k}^i$represents the utility of virtual machine k, defined by:

$$U_{n,k} = J_{n,k}^i + T_{n,k}^i + R_{n,l}^i \tag{2}$$

The normalized minimum completion time of a virtual machine k for the n-th member of the population, when migrating from cloud segment i to cloud segment j, is defined. Furthermore, the value is derived using Equation (3):

$$J_{n,k}^{i} = \min_{\forall j \in A_{k,i}} \left\{ \frac{J_{n,k}^{i,j}}{\max\limits_{\forall j \in A_{k,i},\, t \in G_p} J_{t,k}^{i,j}} \right\} \tag{3}$$

In addition to the above, the migration time of virtual machine k from cloud segment i to cloud segment j for the n -th population member is considered. When the destination cloud segment is the same as the source cloud segment, the migration time is considered zero, i.e., $T_{n,k}^{i,i} = 0$. The minimum normalized migration time of the virtual machine is also calculated using Equation (4):

$$T_{n,k}^{i} = \min_{\forall j \in A_{k,i}} \left\{ \frac{T_{n,k}^{i,j}}{\max\limits_{\forall j \in A_{k,i},\, t \in G_p} T_{t,k}^{i,j}} \right\} \tag{4}$$

In an interactive program, the data footprint increases with job execution time, allowing the cloud segment to determine the virtual machine migration. $R_{n,l}^{i,j}$ The degree of load correlation of virtual machine type l between cloud segments i and j aids the system in predicting the cloud segment load. The minimum normalized load correlation of the virtual machine is computed using Equation (5):

$$R_{n,l}^{j} = \min_{\forall j \in A_{k,i}} \left\{ \frac{R_{n,l}^{i,j}}{\max\limits_{\forall j \in A_{k,i},\, t \in G_p} R_{t,l}^{i,j}} \right\} \tag{5}$$

3.3. Gray Wolf Optimization Algorithm Phases

To mathematically model the social hierarchy of wolves when designing the Gray Wolf Optimization (GWO) algorithm, the best solution is denoted as Alpha (α). Consequently, the second and third-best solutions are named β and δ, respectively. It is assumed that the remaining candidate solutions are ω. In the GWO algorithm, hunting (optimization) is guided by α ،βand δ. Wolves follow these three leaders [10].

3.3.1. Prey Encirclement

As previously mentioned, gray wolves encircle their prey during hunting. To mathematically model the encirclement behavior, Equations (6) and (7) are employed [10]:

$$\vec{D} = \left| \vec{C}.\vec{X}_p(t) - \vec{X}(t) \right| \tag{6}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \tag{7}$$

where t represents the current iteration, and A and C are coefficient vectors. X_p denotes the position vector of the prey, and X signifies the position vector of a gray wolf. The vectors A and C are calculated using Equations (8) and (9):

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \tag{8}$$

$$\vec{C} = 2.\vec{r}_2 \tag{9}$$

where the components of vector $\vec{a}$ decrease linearly from 2 to 0 over the course of iterations, and r_1 and r_2 are random vectors in the range [0,1].

To observe the effects of Equations (6) and (7), a two-dimensional position vector and some potential neighbors are illustrated in Figure 1. As depicted, a gray wolf at position (X, Y) can update its position according to the prey's position (X*, Y*). By adjusting the values of vectors A and C, various positions around the best agent relative to the current position are achievable. The potential updated positions of a gray wolf in a three-dimensional space are also shown in Figure 2.

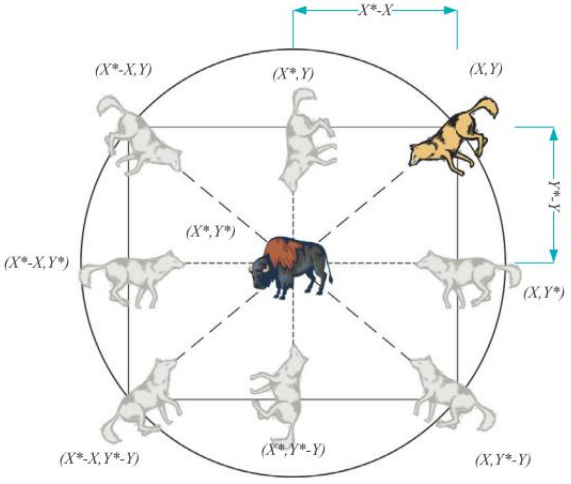

Fig.1. Two-Dimensional Position Vectors and Their Potential Subsequent Positions

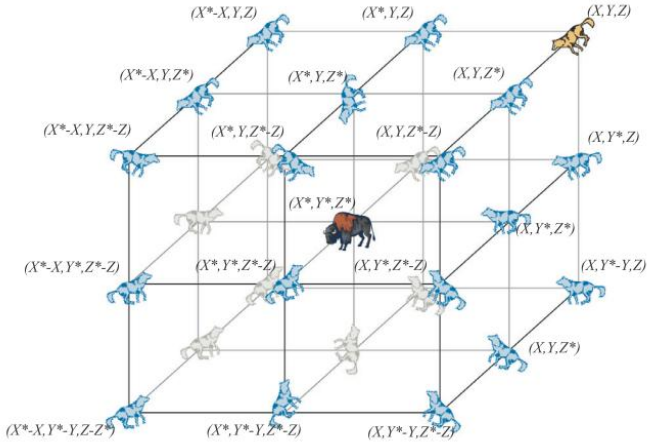

Fig.2. Three-dimensional Position Vectors and Their Potential Future Positions

A key point is that random vectors r_1 and r_2 enable wolves to reach any position between the points depicted in Figures 1 and 2. Thus, a gray wolf can change its position in the space surrounding the prey at any random location using Equations (6) and (7). This concept can be extended to the search space with n dimensions, where gray wolves move within a hypercube (or hypersphere) around the best solution obtained so far [10].

3.3.2. Hunting

Gray wolves have the capability to detect the location of prey and surround it. The hunt is usually led by the alpha wolf, with beta and delta wolves occasionally participating. However, in an abstract search space, there is no prior

knowledge of the optimal location (prey). To mathematically simulate the hunting behavior of gray wolves, it is better to assume that the alpha (best candidate solution) has superior knowledge of the potential location of the prey compared to the beta and delta wolves. Therefore, the top three solutions obtained so far are stored, and other search agents (including omega) are required to update their positions according to the positions of the best search representatives. Formulas (10) through (16) provide the details in this context [10]:

$$\vec{D}_{\alpha} = \left| \vec{C}_1 . \vec{X}_{\alpha} - \vec{X} \right| \tag{10}$$

$$\vec{D}_{\beta} = \left| \vec{C}_2 . \vec{X}_{\beta} - \vec{X} \right| \tag{11}$$

$$\vec{D}_{\delta} = \left| \vec{C}_3 . \vec{X}_{\delta} - \vec{X} \right| \tag{12}$$

$$\vec{X}_1 = \vec{X}_{\alpha} - \vec{A}_1 . (\vec{D}_{\alpha}) \tag{13}$$

$$\vec{X}_2 = \vec{X}_{\beta} - \vec{A}_2 . (\vec{D}_{\beta}) \tag{14}$$

$$\vec{X}_3 = \vec{X}_{\delta} - \vec{A}_3 . (\vec{D}_{\delta}) \tag{15}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{16}$$

Figure 3 illustrates how a search agent updates its position based on alpha, beta, and delta in a two-dimensional search space. It is observed that the final position lies randomly within a circle defined by the positions of alpha, beta, and delta in the search space. In other words, alpha, beta, and delta estimate the position of the prey, and other wolves update their positions randomly around the prey.

3.3.3. Attacking the Prey

Gray wolves conclude their hunt by attacking the prey when it stops. To mathematically model the approach to the prey, the parameter $\vec{a}$ is reduced. Note that the fluctuation range $\vec{A}$ is also decreased using $\vec{a}$. Specifically, $\vec{A}$ is a random value in the range [-2a, 2a], where a decrease from 2 to 0 over iterations. When random values A is within [-1, 1], the next position of a search agent can be anywhere between its current position and the position of the prey. Figure 4 shows that |A| < 1 compels wolves to attack the prey. With the operators proposed so far, the gray wolf optimization algorithm allows its search agents to update their positions based on alpha, beta, and delta and move towards the prey. However, the gray wolf optimization algorithm is susceptible to stagnation in local solutions with these operators. Although the presented encirclement mechanism provides some exploration, the gray wolf optimization algorithm requires more operators to emphasize exploration.

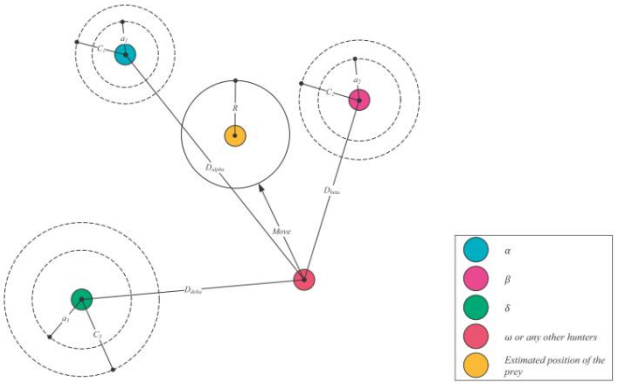

Fig.3. Position Updates in the Gray Wolf Optimization Algorithm

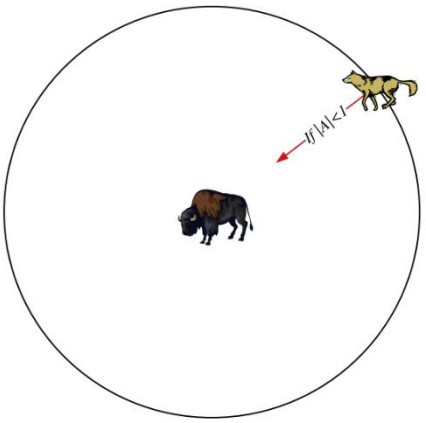

Fig.4. Attacking the Prey

3.3.4. Searching for the Prey

Gray wolves primarily search based on the positions of alpha, beta, and delta. They diverge to search for prey and converge to attack it. To mathematically model divergence, values of $\vec{A}$ greater than 1 or less than -1 are used to force the search agent to move away from the prey. This emphasizes further exploration and enables the gray wolf optimization algorithm to perform a global search. In fact, |A| > 1forces gray wolves to move away from the prey in the hope of finding a better prey. Another component of the gray wolf optimization algorithm that promotes exploration is the vector $\vec{C}$. As seen in Equation (9), the vector $\vec{C}$ contains random values in the range [0, 2]. This component provides random weights for the prey to make the hunting effect in Equation (6) appear more significant (C > 1) or less significant (C < 1). This helps the components of the gray wolf optimization algorithm exhibit more exploratory behavior during the optimization process, aiming to avoid local optima. It is noteworthy that, unlike A, C is not linearly decreased. In the gray wolf optimization algorithm, C is deliberately maintained to provide random values at all times, emphasizing exploration not only in the initial iterations but also in the final iterations. This component can also be viewed as representing obstacles in approaching the prey in nature. Typically, obstacles in nature appear in wolf hunting paths and, in reality, prevent quick and easy access to prey. This is precisely what vector C accomplishes. Depending on the wolf's position, it can randomly increase the weights of prey, making it harder and more distant for the wolf to reach them, or conversely, decrease the weights. In summary, the search process begins with creating a random population of gray wolves (candidate solutions). During iterative cycles, the alpha, beta, and delta wolves estimate the potential positions of the prey. Each candidate solution updates its distance from the prey. To emphasize both exploration and exploitation, the parameter a decrease from 2 to 0. When |A| > 1, candidate solutions move away from the prey, and when |A| < 1, they move closer to the prey. Finally, the gray wolf optimization algorithm terminates upon satisfying a termination criterion. The pseudocode for the stages of the gray wolf optimization algorithm is shown in Figure 5.

```
Initialize the grey wolf population Xi (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
Xα=the best search agent
Xβ=the second best search agent
Xδ=the third best search agent
while (t < Max number of iterations)
    for each search agent
            Update the position of the current search agent by equation (3.7)
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update Xα, Xβ, and Xδ
    t=t+1
end while
return Xα
```

Fig. 5. Pseudocode for the Stages of the Gray Wolf Optimization Algorithm

4. SIMULATION OF THE PROPOSED METHOD

To evaluate the performance of the gray wolf optimization algorithm, various experiments have been conducted, which will be presented subsequently. Additionally, the results of the proposed approach, referred to as GWO hereafter, are compared with those of the genetic algorithm (GA). Table 1 shows the parameter settings for the GWO and GA algorithms.

Table 1. Parameter Settings for Genetic and Proposed Algorithms

Algorithm	Parameter	Value
GA [13]	Initial Chromosome Count	80
	Crossover Operator	Two-Point
	Selection Operator	Roulette Wheel
	Crossover Rate	0.75
	Mutation Rate	0.06
GWO	Initial Wolf Count	80

For simulation purposes to assess the performance of the proposed approach in the first experiment, ten cloud instances were used, each with processors ranging from [1-3] GHz, main memory from [8-16] GB, and SATA hard disks from [250-350] GB. All cloud instances are connected to one or more access points and are interconnected via Ethernet links with speeds ranging from [2-20] megabits per second. Users access the resources of the cloud instances using mobile devices. Figure 6 illustrates the pseudocode for generating datasets for the experimental scenarios. The function `Randi` used in the pseudocode generates a random integer between two given parameters.

```
Function [Cloudlet , Links] = CreateDataSet(L)

   For i = 1 : L
      Cloudlet (i).CPU = randi([1,3]);
      Cloudlet (i).RAM = randi([8,16]);
      Cloudlet (i).HDD = randi([230,350]);
   End
   Links = zeros(L);
   For i = 1 : L
      For j = 1 : L
         Links(i , j) = randi([2,20]);
      End
   End

End
```

Fig.6. Pseudocode for Dataset Creation for Test Scenarios

5. TASK EXECUTION EVALUATION

In this experiment, the successful execution rate and average execution time of tasks were measured by varying the user's mobility from 4.25 km/h to 5.5 km/h in the pedestrian walking model. The results of this experiment are illustrated in Figures 7 and 8. Figure 7 demonstrates that as the user's mobility increases, the percentage of successful task executions decreases. Analysis of the results indicates that at lower speeds, the Genetic Algorithm (GA) performs better than the Grey Wolf Optimizer (GWO) algorithm, but with increasing speed, this trend reverses and the GWO algorithm outperforms the GA.

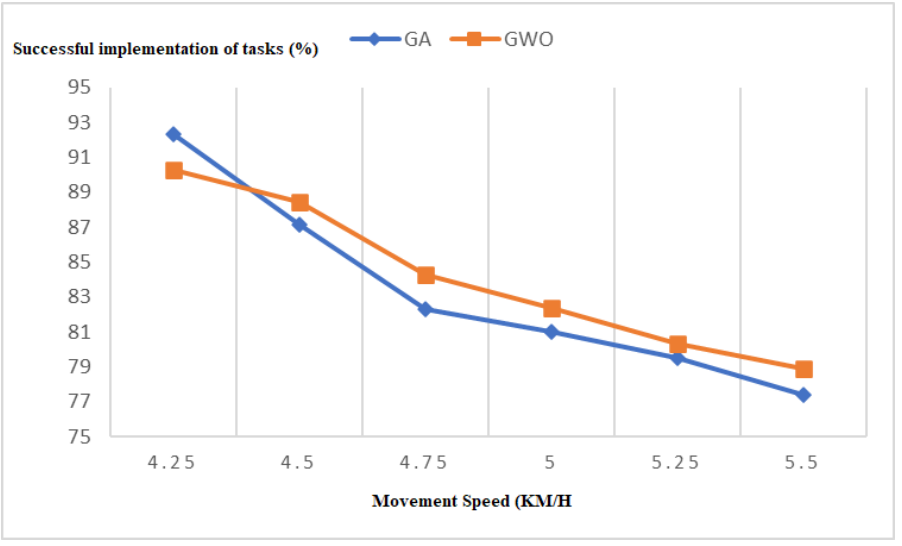

Fig.7. Impact of User Mobility on Task Execution Success Rate Compared to Genetic Algorithm

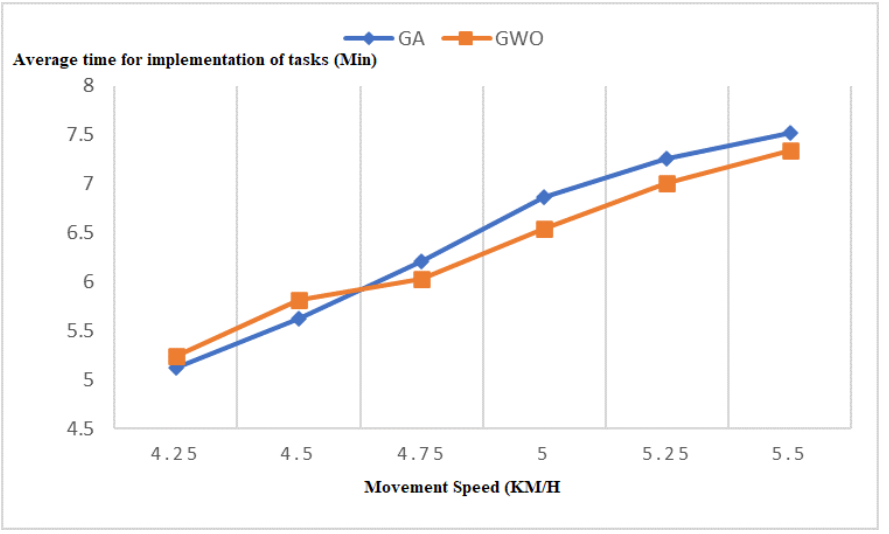

Fig.8. Impact of User Mobility on Average Task Execution Time Compared to Genetic Algorithm

6. EFFECT OF INITIAL POPULATION SIZE

To investigate the impact of initial population size on the quality of final solutions, the initial population size was varied from 10 to 100, and the proposed approach was executed on four different test scenarios. The results are shown in Figures 9 through 12. In these scenarios, the number of cloudlets was set to 5, 10, 15, and 20 respectively.

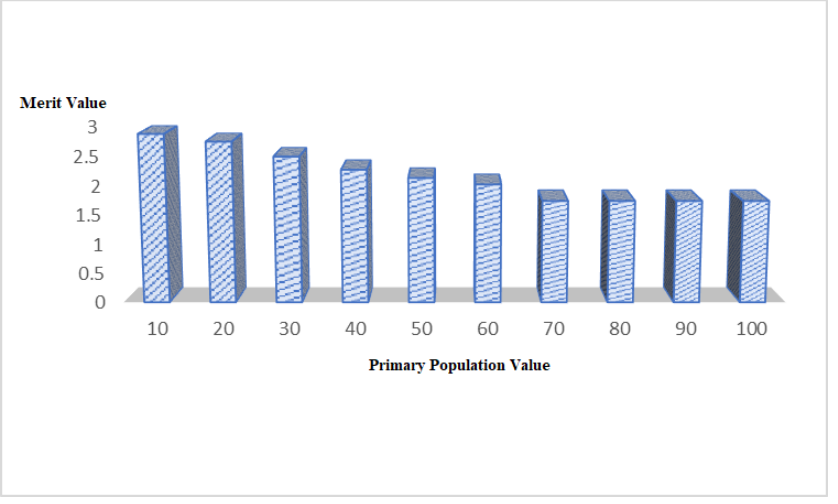

Fig.9. Impact of Initial Population Size on Fitness Value in Scenario 1

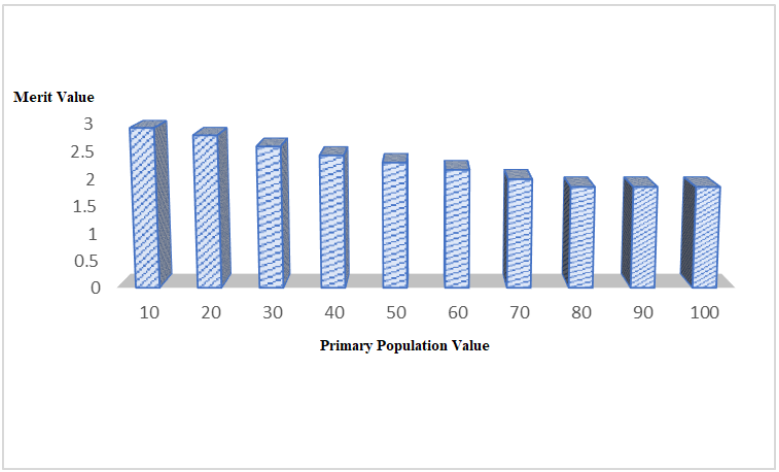

Fig.10. Impact of Initial Population Size on Fitness Value in Scenario 2

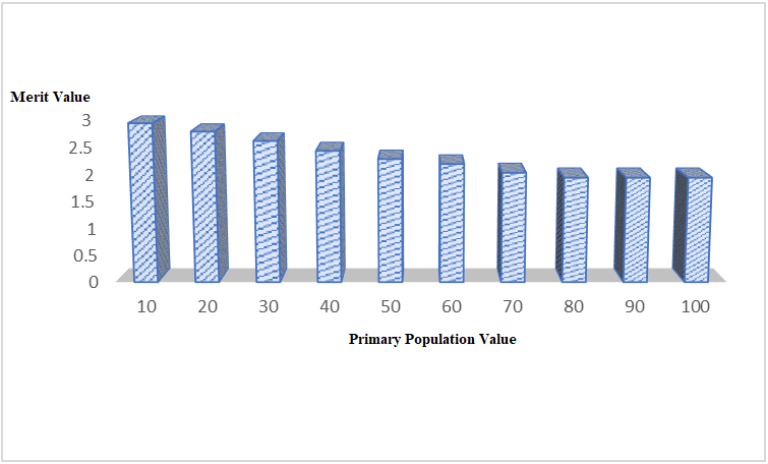

Fig.11. Impact of Initial Population Size on Fitness Value in Scenario 3

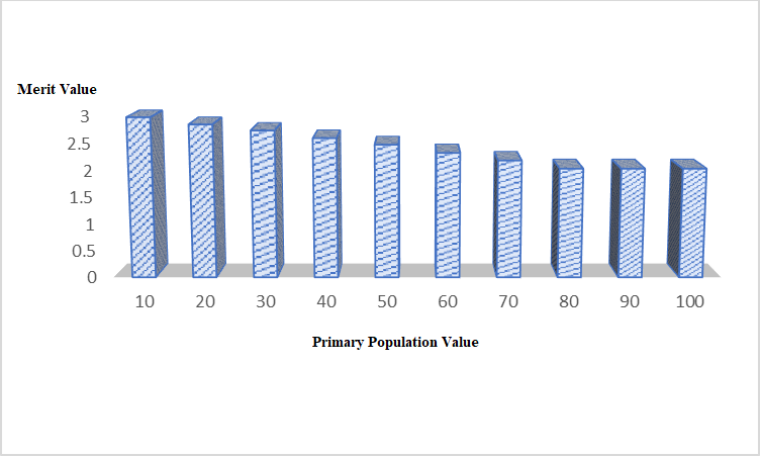

Fig.12. Impact of Initial Population Size on Fitness Value in Scenario 4

The analysis of the results shows that in test scenarios 2, 3, and 4, increasing the initial population size beyond 80 had no effect on the quality of the final solution and indeed on its fitness value. In contrast, for test scenario 1, increasing the initial population size beyond 70 had no impact, leading to the conclusion that an appropriate initial population size for all cases is 80.

7. CONVERGENCE TESTING

For convergence testing, the GWO and GA algorithms were executed for each of the four test scenarios mentioned in the previous experiment. Figures 13 through 16 respectively show how convergence to the final solution occurred in the four scenarios. In these figures, the horizontal axis represents the generation number and the vertical axis represents the fitness value.

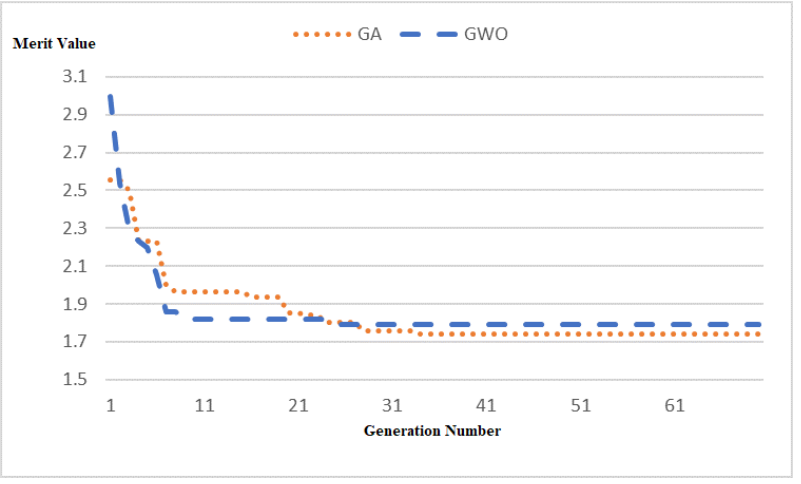

Fig.13. Convergence of the Proposed Approach Compared to GA (Scenario 1)

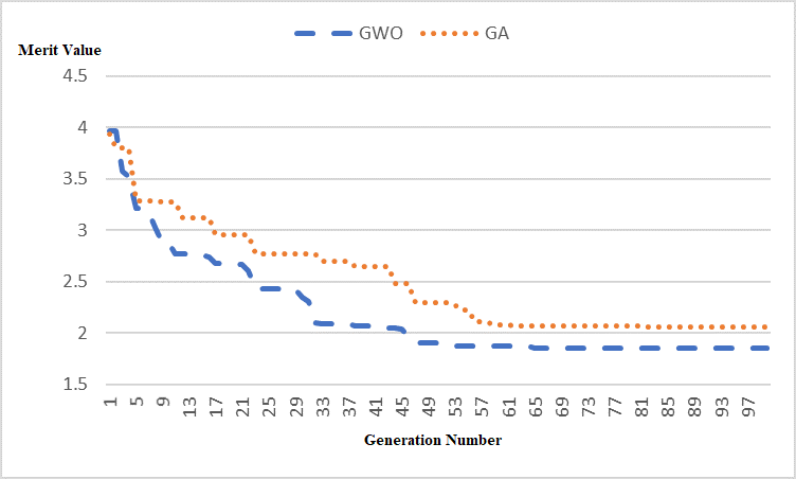

Fig.14. Convergence of the Proposed Approach Compared to GA (Scenario 2)

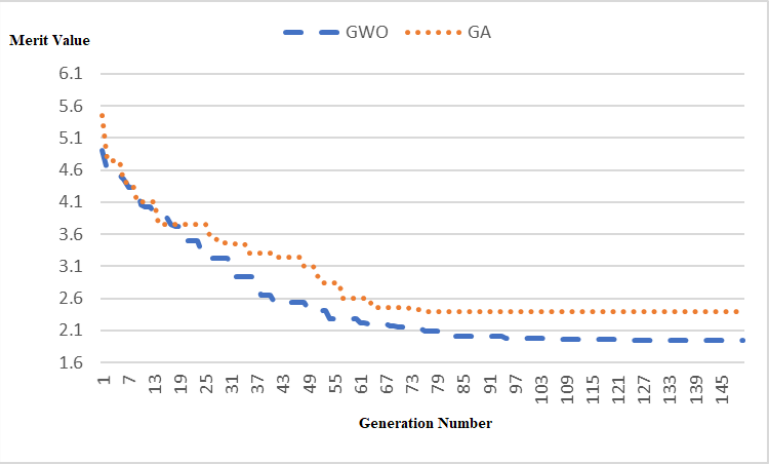

Fig.15. Convergence of the Proposed Approach Compared to GA (Scenario 3)

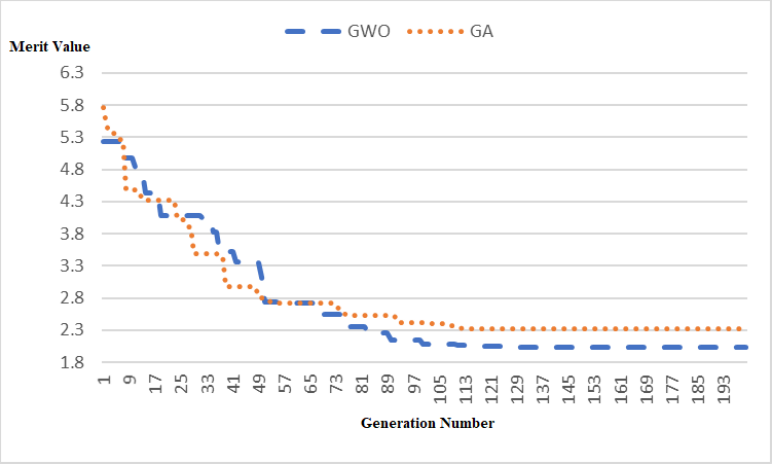

Fig.16. Convergence of the Proposed Approach Compared to GA (Scenario 4)

The results from the convergence testing indicate that the proposed approach demonstrates good convergence speed and reached the optimal solution in all test scenarios. However, it is worth noting that in Scenario 1, where the number of cloudlets is low, the GA algorithm performed better than GWO. Nonetheless, in other scenarios, the proposed approach showed a significant improvement over GA.

8. STABILITY TESTING

Metaheuristic algorithms have a probabilistic and random nature, so the stability of these algorithms needs to be examined. Stability refers to whether the algorithm produces consistent or close answers across different executions. For stability testing, scenarios 1 and 4 were selected, and the GWO algorithm was executed 10 times. The best fitness value obtained in each run is shown in Figures 17 and 18. The horizontal axis in the plots represents the run number, and the vertical axis represents the fitness value. The results from the stability testing indicate good stability of the proposed approach, with the GWO algorithm converging to the optimal solution in all runs.

9. CONCLUSION

Given that the virtual machine migration problem is classified as an NP-hard problem and, with the ongoing development of cloud data centers, the allocation and migration of virtual machines have become a significant

challenge in cloud computing. This paper employs the Grey Wolf Optimizer (GWO) algorithm to address this issue. Inspired by the hunting behavior of wolf packs, the GWO algorithm seeks to find the global optimum. Results from MATLAB experiments showed that the proposed approach exhibits good convergence speed and also maintains high stability. Additionally, the results from various test scenarios demonstrated that the proposed approach yields more suitable solutions compared to the Genetic Algorithm (GA).

Transparency Statement

The data supporting this study are available upon reasonable request to the corresponding author, subject to ethical and confidentiality considerations.

Acknowledgments

We would like to express our gratitude to all individuals who contributed to this project.

Declaration of Interest

The authors declare that they have no competing interests.

Funding

This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

REFERENCES

[1] Zheng, J., Cai, Y., Wu, Y., & Shen, X. (2019). Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach. IEEE Transactions on Mobile Computing, 18,(4), 771-786. http://doi.org/10.1109/TMC.2018.2847337

[2] Sundararaj, V. (2018). Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm. Wireless Personal Communications, 104,(1), 173-197. http://doi.org/10.1007/s11277-018-6014-9

[3] Islam, M., Razzaque, A., & Islam, J. (2016). A genetic algorithm for virtual machine migration in heterogeneous mobile cloud computing. In 2016 International Conference on Networking Systems and Security (NSysS) (pp. 1-6). IEEE. https://doi.org/10.1109/NSysS.2016.7400696

[4] Razali, R.A., Rahman, R.A., Zaini, N., & Samad, M. (2014). Virtual machine migration implementation in load balancing for Cloud computing. 2014 5th International Conference on Intelligent and Advanced Systems (ICIAS), 1-4. https://doi.org/10.1109/ICIAS.2014.6869540

[5] Zhang, Z., Xiao, L., Chen, X., & Peng, J. (2013). A Scheduling Method for Multiple Virtual Machines Migration in Cloud. IFIP International Conference on Network and Parallel Computing. https://doi.org/10.1007/978-3-642-40820-5_12

[6] Kim, C., Kim, J., & Jeon, C. (2014). A Virtual Machine Remapping Scheme for Reducing Relocation Time on a Cloud Cluster. Journal of the Korea Society of Computer and Information, 19, 1-7. https://doi.org/10.9708/jksci.2014.19.11.001

[7] Gilesh, M.P., Satheesh, S., Kumar, S.D., & Jacob, L. (2018). Selecting suitable virtual machine migrations for optimal provisioning of virtual data centers. ACM SIGAPP Applied Computing Review. https://doi.org/10.1145/3243064.3243066

[8] Venkata Krishna, J., Apparao Naidu, G., & Upadhayaya, N. (2018). A Lion-Whale optimization-based migration of virtual machines for data centers in cloud computing. International Journal of Communication

Systems, 31(8), e3539. https://doi.org/10.1002/dac.3539

[9] Suwarna, I. (2011). Virtual machine migration between servers. Google Patents.

[10] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in Engineering Software, 69, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007