OPTIMIZATION OF LOAD BALANCING ALGORITHMS TO DEAL WITH DDOS ATTACKS USING WHALE OPTIMIZATION ALGORITHM

NASIBA MAHDI ABDULKAREEM[*] and SUBHI R. M. ZEEBAREE[**]
[*]Dept. of IT, Technical College of Informatics-Akre, Duhok Polytechnic University, Duhok, Kurdistan Region-Iraq
[**]Dept. of Energy Engineering, Technical College of Engineering, Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq

(Received: May 29, 2022; Accepted for Publication: August 16, 2022)

ABSTRACT

Load balancing algorithms are used to deal with DDoS attacks to identify the optimal server and are responsible for the optimal allocation of requests to the servers. The managing method of the distribution of the requests between servers directly impacts network performance. Denial of service (DDoS) attacks are malicious attempts to interrupt regular operation or network traffic in a targeted manner, making disturbances in internet traffic by disrupting the infrastructure of different servers and thus causing problems such as slow site performance. This paper addresses the optimization of load balancing algorithms to deal with DDoS attacks using Whale Optimization Algorithm (WOA) with other closest algorithms (Round-Robin (RR), Particle Swarm Optimization (PSO), and Genetic Algorithms (GA)). The results obtained from implementing these algorithms showed that the WOA performed better than other algorithms in term of speed up the response time to client requests. The whale optimization algorithm can prevent unexpected traffic and block the regular operation of Internet websites by providing a proper plan for distributing requests between servers and reducing the average response speed. So, the authors can prevent DDoS attacks by applying the whale optimization algorithm. It necessary be noted that the use of HAProxy to prevent DDoS is not enough, and depending on the type of attack, several layers of software and hardware security necessary be used.

KEYWORDS: Load Balancing Algorithms, DDoS Attacks, Round-Robin Algorithm, Whale Optimization Algorithm, Genetic Algorithm, Particle Swarm Algorithm

1. INTRODUCTION

DDoS attacks have increased significantly over the past year. In 2020, many people affected by the Corona pandemic worked online at home or used online services. According to a report from NETSCOUT, more than 10 million DDoS attacks were launched last year, targeting many of the essential remote services that people had to perform, such as healthcare, distance learning, e-commerce, and other services. It also disrupted business operations and, in some cases, led to extortion by attack agents. Attacks (DDoS) are attempts to disrupt the traffic of a targeted server, service or network, which in turn leads to disruption and increased traffic flood [1]. When requests for access to information about a server, site, or network become too high, a DDoS attack can effectively shut down or make that server vulnerable. It can also disrupt the normal operation of an organization's business [2]. A load balancer effectively increases server-side performance and reliability by splitting traffic loads across multiple servers and very powerful settings. Although load balancers were not originally designed to prevent DoS attacks, they become an effective security layer to counter DoS attacks with the correct settings. Instagram is one of the most important load balancer customers to prevent DDoS attacks [3]. Load balancing algorithms are used to identify the optimal server. Providing a proper scheduling method can lead to resource efficiency by reducing request response time and reducing costs. Load balance is simply the transfer of load from an overloaded server to a low load server, which results in a balanced load transfer. Still,this load balance must be such that the maximum use of server resources is

Nasiba.mahdi@dpu.edu.krd and subhi.rafeeq@dpu.edu.krd

considered at the same time as the task scheduling speed [4].

For example, after forwarding each request to the intended Backend, HAProxy uses an algorithm to identify the optimal server with less traffic load, such as Round Robin. The Round Robin algorithm is enabled by default in HAProxy settings. The Round Robin algorithm defines a loop as a queue and considers a fixed time quantum. Each request can only be executed with this quantum and, in turn. If a request is not completed in a quantum, it will return to the queue and wait for the next turn. The main advantage of this algorithm is that the request is executed on time and there is no need to complete previous requests [5]. Therefore, there is no lack of access to resources for other requests in this schedule. However, if the intended queue is full or the workload is very heavy, it will take a long time to complete all the requests. In addition, it is difficult to select a suitable time quantum for scheduling in this algorithm. One of the disadvantages of the Round Robin algorithm is the overload of a large number of switches between process execution and the relatively high average execution time in long processes, so how to manage the work schedule has a direct effect on the load balancer to distribute traffic load [6]. Providing an appropriate scheduling approach can reduce resource efficiency and reduce the effectiveness of DDoS attacks by decreasing response time and costs. Work scheduling optimization algorithm by selecting the appropriate strategy in allocating user requests leads to reducing processing time and waiting time for user requests. But, these algorithms have their own strengths and weaknesses. Therefore, in this study, load scheduling algorithms in load balancing are reviewed to deal with DDoS attacks.

The load balance problem is proposed using various methods to deal with DDoS attacks. Being single-purpose, not paying enough attention to reducing response time, too much computational time is one of the disadvantages of using these methods. Also, in these load balancing optimization algorithms, when the number of requests increases, there is a decrease in the variability of the algorithm and causes rapid convergence to the local optimization. Therefore, in this paper, the whale optimization method is used to achieve the appropriate response distribution of requests among servers to deal with DDoS attacks that can achieve the appropriate response at the desired time and under acceptable conditions. In the whale optimization method, with the cooperation of members of the population, the search space for the optimal global selection becomes wider and the algorithm has the ability to search in a wide range of problem solutions.

The rest sections of this paper have been organized as bellow: Section two is the related previous works. Section three addressed the Load balancer. Section four describes modelling the problem statement. Finally, section five illustrates the conclusion and recommendations.

2. Related Work

In order to load balancing in the cloud, various algorithms have been proposed, each of which has advantages and disadvantages depending on the state of the cloud computing environment. Load balancing algorithms are generally divided into dynamic and static categories.

The static method is used when the complete set of tasks and the resources required for them are estimated before execution. This strategy is implemented under two assumptions: first, that the tasks enter at the same time, and second, that the time of existing machines is updated after each task schedule. The most commonly used static algorithms are weighted round-robin (WRR) and round-robin (RR) algorithms. Still, static algorithms only work properly when there is little change in the load of virtual machines. Therefore, these algorithms are not suitable for cloud computing environments where workloads vary at various time points [6, 7].

Based on the attributes allocated to each machine, the dynamic load balancing method dynamically redistributes load across computers. Because they ignore important considerations like response time and overall system throughput, the most prevalent load balancing algorithms, Weighted Least Connection (WLC) or Least Connection (LC), will not be able to satisfy the demands of this dynamic environment [6].

In [8], the exploratory behaviour of bees has been used to balance the effective load between virtual machines. The decisions about which task necessary be assigned to which machine are made based on the load and availability of the virtual machine. This method improves overall throughput and reduces the time a task has to wait in the virtual machine queue, which led to a decrease in the response time and completion time of the tasks. The main purpose of this algorithm is to respond the requests quickly and manage virtual machines as well as possible. The search of virtual machines in this algorithm is exactly like the bees' search for food. The search for machines at the first stage of the algorithm is random, and there is a competition to find the best source, like searcher bees. This competition is between powerful virtual machines, and each powerful car will earn more. In this algorithm, in parallel with the allocation of resources from high-level resources to low-level resources, costs are also received relative to the level of resources requested by the applicant. This function is the same as the bees that collect the most nectar from the best source. In each work assignment to the virtual machine, their efficiency is also recorded. In this algorithm, tasks that require a quick response get the best resource by paying the cost. As a result, task prioritization is automatically implemented in this algorithm.

In [9], a new hybrid algorithm called QMPSO for dynamic load balancing between virtual machines is proposed using a combination of two modified particle swarm optimization (MPSO) and the Q-learning algorithms. The process of hybridization is performed to adjust the MPSO speed via gbest and pbest based on the best action generated by improved Q-learning. The purpose of hybridization is to increase device performance by balancing the load between VMs, maximizing VM performance, and keeping the balance between tasks priorities by optimizing task waiting times. The robustness of the algorithm is confirmed by comparing the QMPSO results obtained from the simulation process with the existing load balancing algorithm and programming. A comparison between the simulation result and the actual platform result shows that the QMPSO algorithm performs better than the MPSO and the Q-Learning algorithms.

In [10], a work scheduling algorithm in cloud computing based on the balance improvement of the ant colony algorithm (D-ACOELB) was proposed. The main part of this research is about balancing the load while reducing run-time in the whole system. A comparison between ACO, MACO, and DEACOLB algorithms shows that the run-time average and the degree of load imbalance have been reduced.

In [11], a new method for dynamic load balancing between virtual machinesusing a combination of TBSLB and particle swarm optimization (PSO) algorithms is proposed. This method is an algorithm for balancing the load of the whole system, which offers the best load distribution model among the load distribution models between virtual machines.In this method, compressed computations and compressed data are considered, which used bandwidth to transfer compressed data and used a number of high-performance CPUs on the virtual machine to transfer compressed computations. The TBSLB method contains a list of programs that are in the cloud computing programmer layer and are responsible for managing virtual machines.Virtual machine location information and cloud computing programmer layer applications are given as input to the TBSLB-POS algorithm, and the output of the algorithm is the allocation and transfer time information of virtual machines, based on which the cloud computing programmer layer tasks are updated.

In [12], a Distributed Flow-by-Flow Fair Routing (DFFR) algorithm was introduced with the aim of balancing the flow in cloud computing. This algorithm was considered as an adaptive and distributed method because it uses all network resources. Due to the re-routing of distributed algorithms, which inherently suffers from the complexities of redirection, therefore, in case of network position fluctuations, the authors need a large convergence mode to stabilize the position. The problem has been handled by DFFR and the process has been done with the support of load traffic information. In this study, better outputs are obtained while creating minimal inconsistencies for the use of cumulative bandwidth. The simulation results show that the DFFR algorithm performs better than the static routing determination protocol. The evaluation results show that DFFR is an effective load balancer for data center networks with random traffic patterns.

In [13], a new load-balancing algorithm is proposed by Adhikari et al. for the IaaS cloud. It is an efficient server configuration strategy based on the number of input tasks and their size which has been developed to identify suitable VMs for task allocation and to maximize the use of computational resources. The proposed algorithm is tested by performing simulations and comparing the simulation results with existing algorithms using different performance criteria. Finally, it is shown that the proposed algorithm performs better than the existing algorithms.

In [14], a hybrid approach based on ant colony optimization and bee colony optimization has been presented to address the problem of load balancing in the cloud. A set's features are taken into account in the suggested procedure. The revised bee colony algorithm's other parameters are coupled with this approach to form a new ACO technique. The Cloud Analyst tool is used to model the new hybrid approach. When compared to the ACO algorithm and the ABC algorithm, the hybrid algorithm performs better than the ACO algorithm and ABC algorithm.

In [15], a multi-objective genetic algorithm (GA) for a cloud data center is proposed to dynamically predict the amount of consumption of energy and usage of available resources. The issue of multi-objective resource allocation optimization was expressed by showing virtual and physical machines' memory and CPU usage. The resource requirement for subsequent time slots is predicted by the proposed genetic algorithm based on the data from the available time slots. Then, the GA prediction results were used using the VM replacement algorithm to allocate virtual machines in the next time slots. The prediction results were better than other available prediction methods. According to the data, this strategy minimizes energy used and maximizes CPU and memory consumption in the cloud data center.

Table (1): Comparison of load balancing algorithms

Reference	Algorithm	Advantages	Disadvantages
[6]	Round-Robin	Optimal use of resources in a short time	The method of allocating resources is sequential. Therefore, it reduces the load balance in virtual machines of cloud data centers.
[9]	Q-learning with two modified particle optimization algorithms (MPSO)	Hybridization Increase device performance by balancing loads between VMs Maximize VM performance Maintain a balance between work priorities by optimizing waiting times	Being a single Objective Insufficient attention to reduce response time Extremely computational time
[10]	Ant colony optimization	High-speed run-time	When the number of resources increases, a decrease in the algorithm occurs and causes rapid convergence.
[11]	Particle Swarm Optimization (PSO) and Task-Based System Load Balancing (TBSLB) (PSO)	Load balance optimization	Being a single Objective Insufficient attention to reduce response time Extremely computational time
[12]	Algorithm for Distributed Flow-by-Flow Routing	Increase system throughput Increase system efficiency Decentralization of load balance	Only the load balancing criterion has been proposed, and the time for completing the entire processing has not been addressed
[13]	Two-step load balancing	Run time Resource efficiency	It does not work on large scales
[14]	A bee colony algorithm and an ant colony optimization	Load balance optimization	When the number of resources increases, a reduction of variability occurs in the algorithm and causes rapid convergence.
[15]	Multi-objective genetic algorithm (GA)	Reduce run-time Increase balance Possibility of combining several algorithms	Low speed and local search

3.Load balancer

The method used in load balancing is that requests such as opening a web service or viewing a site from users are first received by a load balancer. Then each of these requests is sent to them according to the amount of load and the number of servers, which means that even in cases where a lot of requests are sent to a site or server, they are distributed evenly between the host servers. the authors will no longer see the interruption and slowness of the servers. When a server becomes inaccessible, the load balancer forwards all requests of the requesting server to another active server [16].

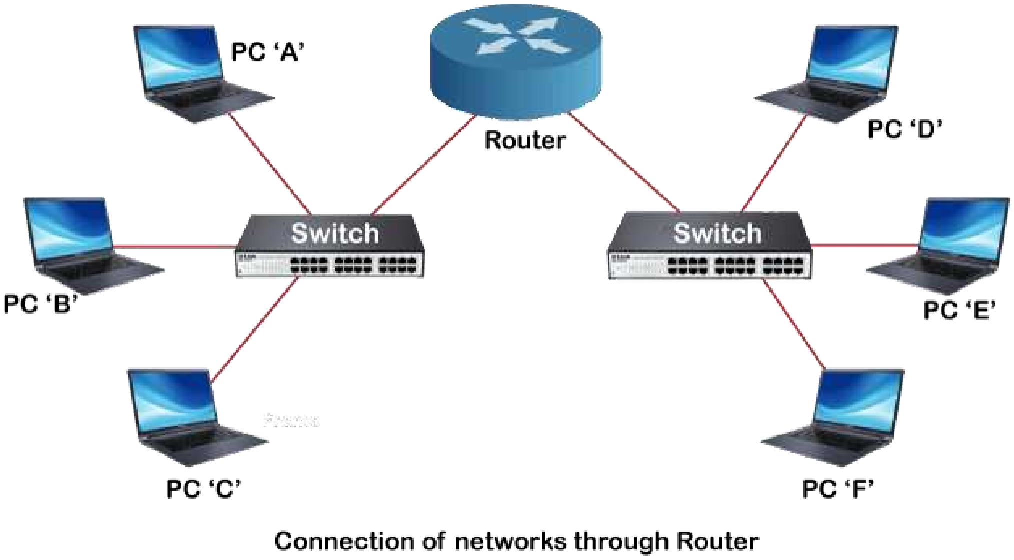


Fig.(1):- Existence of load balancer in the network [16]

-The requester tries to communicate with the server through the load balancer.

- Load balancer, accept the connection and decide which of the servers is more suitable for receiving traffic. Then, it changes the destination IP address (and, in some cases, the port) in the received packet header according to the selected server and service. It necessary be noted that the source IP does not change.

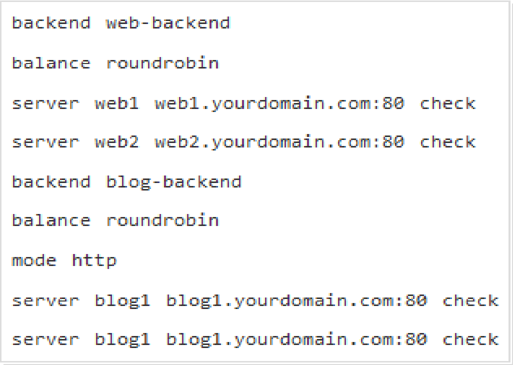
- The server accepts the received traffic and, after processing the information, sends the request to the source, the requester, through the default path, the load balancer.

- Load balancer separates the returning packet coming from the server from the traffic and now changes the source IP address (as well as the port) in the packet header, according to the IP address and port, and tries to send the packet again to the requester.

- The requester receives the package, and the processing operation continues [16].

3.1 HAProxyload balancer

HAProxy is an open-source load balancer that effectively increases server-side efficiency and reliability by splitting traffic load between backends. The backend includes one or a set of servers that receive forwarded requests from HAProxy and sends them to defined servers. Although HAProxy was not originally designed to prevent DOS attacks, with the right settings, it becomes an effective security layer to deal with simple DOS attacks (Slowloris…). GitHub, Imgur, Instagram, and Twitter can be mentioned as prominent customers of this software. In Fig.2 HAProxy load balancing structure is presented [17].



```
backend web-backend

balance roundrobin

server web1 web1.yourdomain.com:80 check

server web2 web2.yourdomain.com:80 check

backend blog-backend

balance roundrobin

mode http

server blog1 blog1.yourdomain.com:80 check

server blog1 blog1.yourdomain.com:80 check
```

Fig.(2): -HAProxy load balancing structure [17].

This structure defines two separate backends for the leading site (web-backend) and the blog (blog-backend). In each of the backends, two servers are limited, and the health of the servers is examined with the phrase check. A balance round-robin is an algorithm used to select the optimal server and load balancer. The question is whether it is possible to build a load balancer with non-round-robin algorithms that perform better. In the following, the authors will model this issue.

4. Modeling the problem

The network considered in this study includes a set of servers. One of these servers is selected as load balancing, which is responsible for scheduling user requests.

The difficulty in scheduling workflows is providing a mapping of requests to servers that reduce specific goals, including resource competition, cost, and energy consumption. This means that these goals conflict with each other. Assigning n requests to m nodes (servers) is considered an NP-completeness problem (Fig.1).

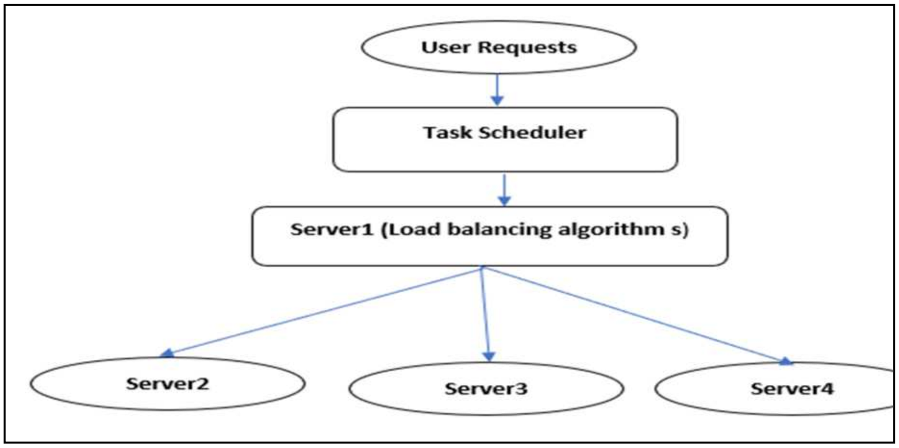


Fig.(3):- Structure of load balancing algorithms to deal with DDOS attacks

Based on the structure shown in Figure (3), requests are received from users in the first stage of load balancing. Then, the task scheduler receives the best request distribution model among the servers using the load balancing algorithm. In the following, the authors will examine load balancing algorithms.

The Round-Robin algorithm is enabled in the HAProxy software settings by default. In this algorithm, each server is used periodically.

The Least Connection algorithm forwards the request to the server with fewer working connections. It also acts as a Round-Robin if the server load is equal. This algorithm is suitable for activities that require long sessions [17].

The source algorithm connects each client to only a specific server. This means that all requests sent by each user are forwarded to a single server only (the user's IP is processed).

By applying the correct settings, the authors can prevent simple DDoS attacks in layer 4 or 7. It necessary be noted that the use of HAProxy to prevent DDOS is not enough, and for this purpose, depending on the type of attack, several layers of software and hardware security necessary be used.

4.1 Meta-heuristic algorithms as load balancing algorithms

There are a variety of situations that may benefit from the application of meta-heuristic algorithms, which have outflow strategies from local optimum locations. Some well-known meta-heuristic algorithms are population-based, including evolutionary algorithms optimization and bee colony algorithms (e.g. particle swarms or genetic algorithms). New meta-heuristic algorithms have emerged in recent years with respect to living beings in nature (nature-inspired), the most famous of which is the whale optimization algorithm. Meta-heuristic algorithms are designed to schedule the workflow of requests optimally. Each solution in this algorithm shows how to allocate *n* requests to *m* nodes (servers). This algorithm's objective function is selected based on the optimization of three objectives 1. Load balance 2. Planning speed 3. Productivity of design resources [18].

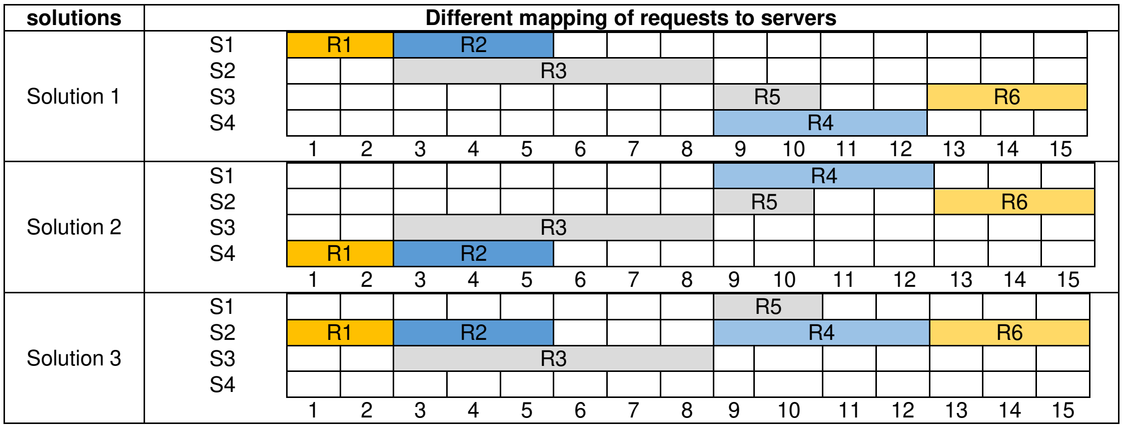


Fig.(3):- How to map n requests to m nodes (server)

In the fourth step, an evolutionary algorithm is proposed to reset the failed requests (requests that failed to receive resources) to achieve maximum service quality.

4.1.1 Particle Swarm Algorithm

The particle swarm algorithm (PSO) is one of the well-known population-based meta-heuristic algorithms. The person's interactions with their environment of living beings such as birds and fish groups, made them realize the effect of species cooperation to achieve their goals as a group. A large number of birds or fish gather together at the same time, suddenly changing direction, and it is based on personal and societal experience to say they disperse together and form a new group. This method uses the term "particle" to refer to each answer. Particles benefit from their prior and social experiences in a positive way. Particles not only know their optimal physical location (pbest), but they also know their optimal social position (gbest). The pbest, gbest, and the particle's current location are used to alter the paths and velocities of all the particles in motion (x_i^k) and velocity (v_i^k). Each iteration's dynamic matching is represented by pbest and gbest. Equations 1 and 2 are the PSO progression equations (2). Table 1 depicts the various factors and their corresponding ideas [19].

$$V_i^{k+1} = \omega v_i^k + C_1 rand_1 * (pbest_i - x_i^k) + c_2 rand_2 * (gbest - x_i^k) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^k \quad (2)$$

Table (1): -Parameters and the concept of parameters

Parameters	concept of parameters
v_i^k	the velocity of particle *i* in repetition *k*
v_i^{k+1}	the velocity of particle *i* in repetition *k+1*
x_i^k	position of particle *i* in repetition *k*
x_i^{k+1}	position of particle *i* in repetition *k+1*
ω	inertia weight
c_1, c_2	acceleration coefficients
$rand_{1,} rand_2$	a random number between 0 and 1
$pbest_i$	the best position of particle *i*
$gbest$	the best position of the whole particles in the population

Evolutionary algorithms, such as PSO, are analogous to the PSO algorithm. In the PSO, a potential solution to the underlying issue has n dimensions that are determined by the specific problem. Particles are fired at random locations and speeds. Particles have an adaptation value, which is assessed by the optimum adaptation function for each generation. Every atom has a preference for pbest or gbest. The particle's velocity and location in each generation will be updated by Equations (1) and (2) [19].

Algorithm 1: Using Particle Swarm Algorithm as Load Balancing Algorithm

Input: Server set (S), Request set (R)

Output: Provide the best distribution plan to the backends

1 Determine the initial population

1. Particles in the initial population represent the substitution matrix $X_{n\times m}$whose elements are defined as follows:

If the authors consider the number of requests to be 100, then the requests are forwarded to the backends (servers) based on a random distribution plan.

Table (2):- Requests based on random distribution plan (particles)

x_{pv}	$S1$	$S2$	$S3$
p_1	35	40	25
p_2	30	30	40
p_3	40	40	20
p_4	20	45	35
p_5	30	60	10
p_6	30	50	20

2. The objective function is defined based on the load balance criterion, which actually determines the amount of loads or tasks divided between the processors. The value of this criterion will be obtained based on Equation (3), and the lower the value, the better. So, the algorithm coulddistribute the load [20] better.

$$loadbalancing = \frac{Makespan}{Avg}$$

(3)

In the above relation, Avg is equal to the ratio of the total processing of each processor to the number of processors, and Makespan determines the completion time of the whole work in the programming. The lower the value of this criterion, the faster the algorithm is able to process and deliver tasks to users. The value of this criterion is obtained based on the following equation [21].

$$Makespan = MAX\ 1 < i \leq n\{T_i\}$$

(4)

For example, consider a network with four servers, the Gantt chart of its schedule is created in Figure (1).

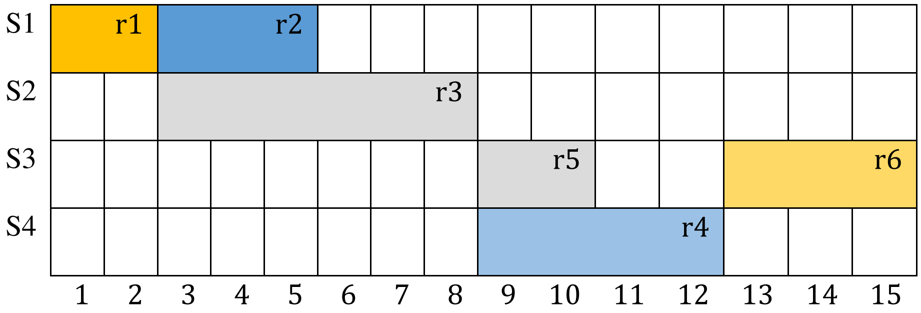


Fig.(4):- Gantt chart schedule of user requests

Makespan = 15

Time S1=5

Time S2=8

Time S3=15 $\Rightarrow Avg = \frac{5+8+15+12}{4} = 10$

Time S4=12

l

$oadbalancing = \frac{15}{10}$=1.5

3. Calculate the amount of best personal experience (pbest) for each particle and best group experience (gbest) for the whole population
4. Assignment of learning coefficients (*C1*, *C2*) and inertia (*W*)
5. Calculate the position and velocity of particles according to PSO relations
6. Calculate the fitness of all population particles
7. Update pbest and gbest values
8. Checking and determining the best weight coefficients
9. If the condition for termination of the number of repetitions *N* is met, go to 10, otherwise, go to 3
10. Determine the best distribution plan based on minimizing the load balance criterion

4.1.2 Genetic Algorithm

An algorithm's design variables are represented by strings of constant or variable length, which are known as chromosomes or individuals in biological systems. Response points in the search space are indicated by a string or chromosome. A chromosome's genotype and phenotype are two terms that refer to the same thing: the structure of strings and the parameters they exhibit. Generation and population are the terms used to refer to the sets of answers generated in each iteration (phase) in genetic algorithms. The primary search is carried out via genetic algorithms in the response space. "Seeding" is responsible for developing a collection of main search sites known as "initial populations," which are either picked carefully or randomly. As a result of genetic algorithms' use of statistical techniques to drive search operations toward the optimum point, the current population is picked in proportion to the fitness of its members for the next seeding. Genetic operators such as selection and crossover, mutation and other potential operators are then used to produce a new population. A new population then replaces it, and the cycle repeats. From seeding to seeding, the new population tends to be more fit (better suited to the task at hand). The search will be successful when the authors seeded the maximum feasible amount, achieved convergence, or satisfied the terminal criterion [22].

4.1.2.1 Using genetic algorithm as load balancing algorithm

Briefly stated, the following are the operators that make up the genetic algorithm:

- Encoding

At this point, algorithmic methods are likely to fail miserably at resolving the issue. Instead of focusing on the specifics of the issue, the genetic algorithm focuses on the encoded form of the parameters or variables. For example, using binary encoding, the response may be encoded as a series of binary integers (in the base of 2).

- Evaluation

The goal function or function to be optimized is converted into a fitness function using an appropriate conversion. Each string is given a numerical number that indicates its quality. If the response string is of high quality, it will be more likely to participate in the seeding of the following generation.

- Crossover

The crossover operator is critical in the genetic algorithm. To create a new generation of cells, older seeding chromosomes are combined with younger ones in the crossover process.

New members are born as a result of the pairings that were deemed parents earlier in the selection process. Good genes are able to discover each other via the crossover in the genetic algorithm, which reduces population dispersion or genetic variety.

- Mutation

Other alternative replies may be generated by using the mutation operator. After a new population is formed, each individual contains genes that have varying mutation probabilities. A previously absent gene may be inserted into a gene population, or a previously unknown gene might be deleted. When a gene is mutated, a change is made to that gene. Different mutation mechanisms are utilized based on the encoding of the gene. [22].

- Decoding

Decoding is the opposite of encoding. At this stage, after the algorithm has provided the best answer to the problem, it is necessary to apply the reverse of the encoding operation to the answers (decoding operation) so that the authors can clearly know the true version of the answer.

In general, when a genetic algorithm is applied, the following cycle takes place:

First, an initial population of individuals is randomly selected without considering any specific criteria. For all zero-seeding chromosomes (individuals), the fitness value is determined by the fitness function, which may be very simple or complex. Then, with different mechanisms defined for the operator, a subset of the initial population will be selected. Then, if it is necessary, the crossover and mutation operations will be applied to these selected individuals according to the problem. Now, these individuals to whom the mechanism of the genetic algorithm has been applied necessary be compared with the initial population (zero seeding) in terms of the amount of fitting. (the authors certainly expect first-seedingindividuals to be more competitive given the one-time application of genetic algorithms to them, but this will not necessarily be the case.) However, the people who have the most fitting value will remain. Such individuals will act as the initial population for the next stage of the algorithm [22]. Each iteration step (phase) of the algorithm creates a new seeding that will evolve according to the modifications made to it. It is worth noting that although genetic algorithms do not have a clear mathematical basis, they have proven to be effective as a reliable and executable model that is well implemented. The general outline of an algorithm is as follows:

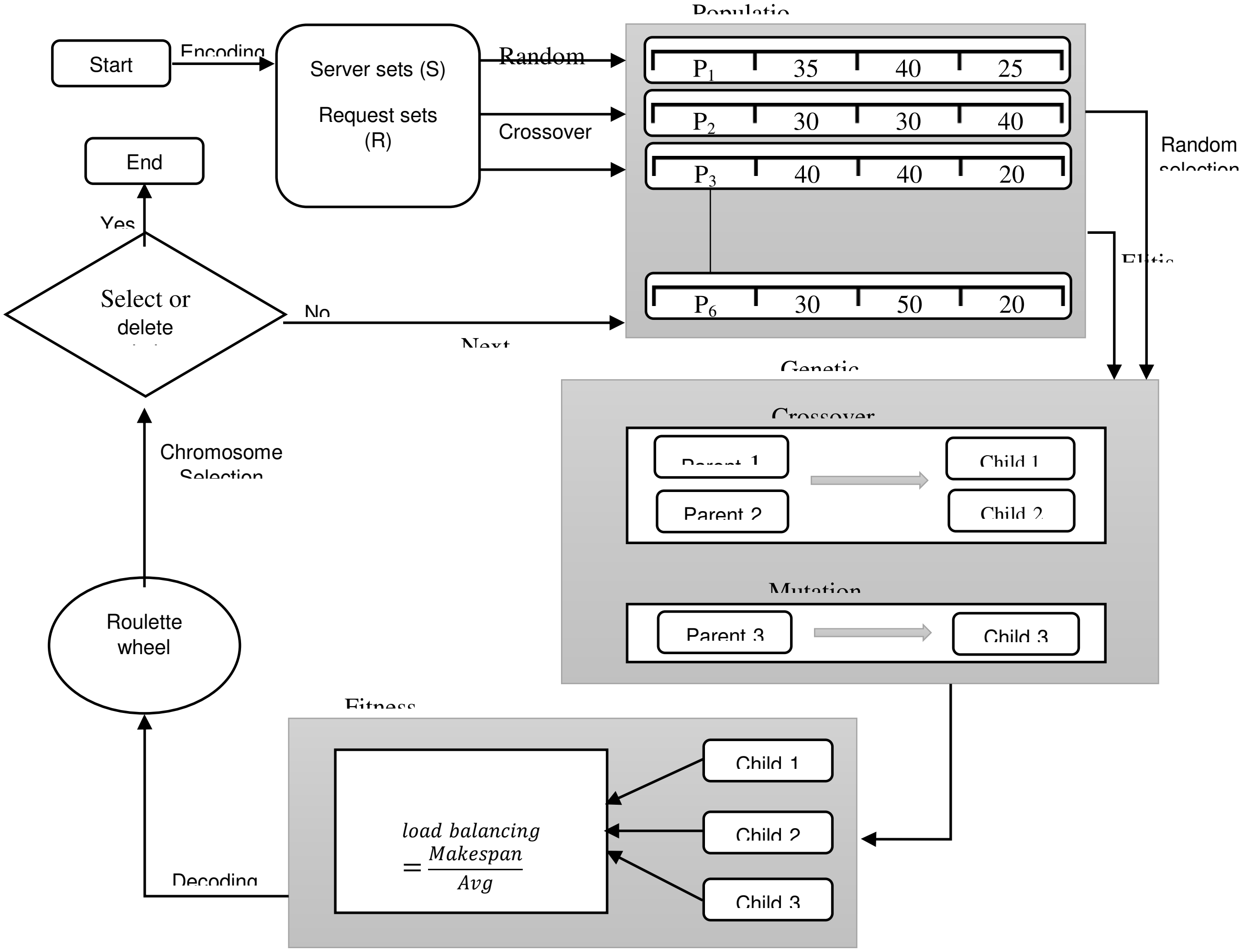


Fig. (5): -Flowchart of genetic algorithm as load balancing algorithm

Before a genetic algorithm can be implemented, a suitable encoding (or representation) must first be found for the problem. A fitness function must also be designed to give a value to each encoded solution [23.] Parents are chosen for reproduction and merged via crossovers and mutations in order to generate new offspring throughout the implementation process.Once the population is seeded, this procedure is repeated multiple times. As a result, if the requirements for convergence are fulfilled, the aforementioned procedure is completed.

4.1.3 Whale Optimization Algorithm

As a meta-heuristic method, the Whale Optimization Algorithm is an excellent choice thatcan avoid local optimization and achieve global optimization. This method is based on a bubble-net searching technique. The specific hunting humpback whale behaviour is described in the algorithm.The humpback whale goes down several meters in the water and then begins to produce spiral bubbles around the prey (small fishes) and then glides along with the bubbles upwards on the surface of the water [24].

4.1.3.1 Using the Whale algorithm as a load balancing algorithm

Humpback whales hunt for prey (small fishes) and then update their location relative to the ideal solution on the route of the increasing number of iterations.

$$\vec{D} = \left|C.\vec{X} \times (t) - X(t)\right| \qquad (5)$$

$$\vec{X}(t+1) = \vec{X} \times (t) - \vec{A}.\vec{D} \qquad (6)$$

here,$\vec{A}$and $\vec{D}$are coefficient vectors, t is the current iteration, $\vec{X} \times (t)$is the position vector of the optimal solution, and$X(t)$ is the position vector. The coefficient vectors$\vec{A}$and $\vec{D}$are calculated as follows:

$$\vec{A} = 2\vec{a} \times r - \vec{a} \qquad (7)$$

$$\vec{C} = 2 \times r \qquad (8)$$

here,a is the symbol of a variable that decreases linearly from 0 to 2 on the iteration path, and r is equivalent to a random number [0,1].

4.1.3.2 Modeling the bubble-net attack method

Two methods with the aim of the bubble-net behaviour of humpback whales are represented using the following model structure:

(A) The mechanism of shrinking encircling

This technique is applied by linearly reducing the value of $\vec{a}$from 2 to 0. The random value for the vector $\vec{A}$is in the range between [-1,1].

(B) Spiral updating position

The spiral motion between the humpback whale and its prey is shown in the following mathematical spiral equation:

$$\vec{X}(t+1) = \overrightarrow{D'} \times e^{bt} \times \cos(2\pi l) + \vec{X} \times (t) \qquad (9)$$

which in equation l is equal to a random number [-1,1], b is a constant, logarithmic form,$\overrightarrow{D'} = \left|\vec{X} \times (t) - X(t)\right|$is the distance between the i-thwhale and the best middle hunting solution. Note: the authors assume that there is a 50-50 percent probability that the whale follows a shrinking encirclingwitha logarithmic path during optimization. the authors have mathematically modelled it as follows:

$$\vec{X}(t+1) = \begin{cases} \vec{X} \times (t) - \vec{A}.\vec{D} & if \;\; p < 0.5 \\ \overrightarrow{D'} \times e^{bt} \times \cos(2\pi l) + \vec{X} \times (t) & if \;\; p \geq 0.5 \end{cases} \qquad (10)$$

which$p$ expresses a random number between [0,1].

(C) Search for prey

Vector $\vec{A}$ can be used for exploration in search of prey; also, the vector $\vec{A}$has values greater than 1 or less than -1. Exploration follows two situations: The exploration for the Whale Optimization Algorithm (WOA) is augmented by $\left|\vec{A}\right| > 1$to find the global optimal and avoid the local optimal.

$\left|\vec{A}\right| < 1$is selected to update the current search agent position / best solution.

The adaptive technique contains the best features; it contains less parameter dependence. It does not need todetermine the starting parameter and the size of the step relative to the optimal solution, which changes adaptively according to its functional proportionality relative to the iteration path. Therefore, meta-heuristic algorithms in integrated technique with adaptation lead to a less amount of calculation timeto achieve the optimal solution, local minimum acceleration and avoidance of pitfalls [24].

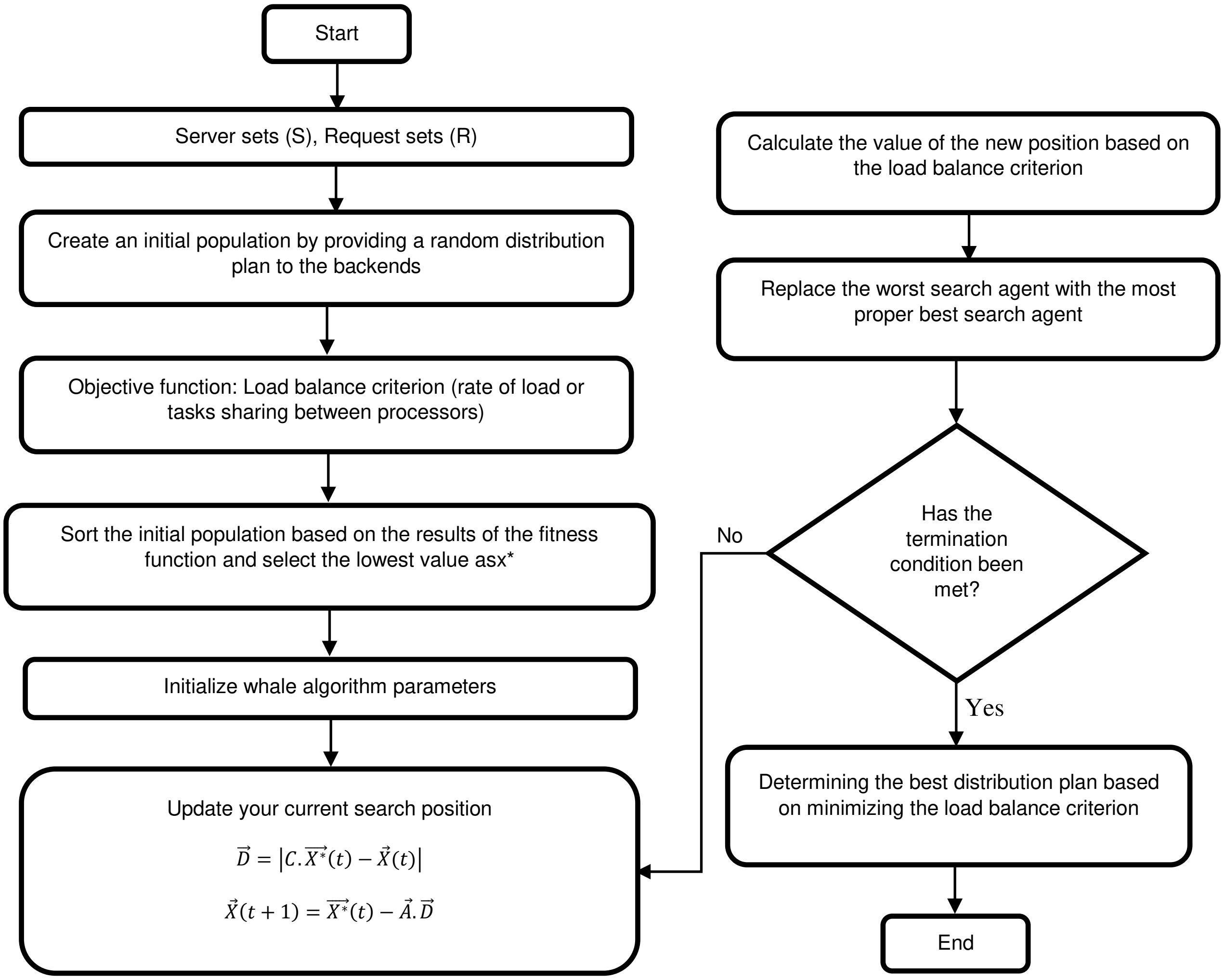

Start
Server sets (S), Request sets (R)
Create an initial population by providing a random distribution plan to the backends
Objective function: Load balance criterion (rate of load or tasks sharing between processors)
Sort the initial population based on the results of the fitness function and select the lowest value asx*
Initialize whale algorithm parameters
Update your current search position
$\vec{D} = |C.\vec{X^*}(t) - \vec{X}(t)|$
$\vec{X}(t+1) = \vec{X^*}(t) - \vec{A}.\vec{D}$
Calculate the value of the new position based on the load balance criterion
Replace the worst search agent with the most proper best search agent
Has the termination condition been met?
No
Yes
Determining the best distribution plan based on minimizing the load balance criterion
End

Fig.(6):- Flowchart of Whale optimization algorithm as load balancing algorithm

4.2 RESULTS

The proposed method was compared with the Particle Swarm, Greedy, Genetics, Differential Evolution, Bee Algorithm and Whale Optimization Algorithm methods. The parameters of these algorithms are presented in Table (1).

Table (3): -Parameter characteristics of the compared algorithms

Methods	Algorithm	Reference	Algorithm Parameter
APDPSO	Adaptive Pbest discrete PSO	[21] Miao, Yong Mei and Quanjun (2021)	Nparticle=50,C1,C2=2,Inersi=0.9
R-R	Round-Robin	[16]Pramono, L. H., Buwono, R. C., &Waskito, Y. G. (2018)	
GA	Genetic	[22]Vijarania, M., Agrawal, A., & Sharma, M. M. (2021)	Pop=100,CrossOver=0.9, Mutation=0.1
WOA	Whale Optimization	Proposed Method	Npop=50

The proposed method of simulation was performed by Python programming language using a system as a simulation platform and installation of five virtual machines to set up servers and clients. The characteristics of virtual machines and their operating systems are presented in Table 4.

Table(4): -Simulation environment, servers and client characteristics

N	Characteristics	Operating System	The number of Processors	The amount of memory	Storage Space
1	Simulation Platform	Windows 10	5	16 GB	1 TB
2	Server1 (Load Balancing Algorithms)	Ubuntu	4	3.7GB	20 GB
3	Server2	Ubuntu	2	2.6GB	20 GB
4	Server3	Ubuntu	2	3GB	20 GB
5	Server4	Ubuntu	2	3GB	20 GB
6	Client	Ubuntu	2	3GB	20 GB

According to the data in Table 4, in this simulation, four servers with 2-4 processors, memory 3-3.7 GB,and 20 GBof storage space have been done.

4.2.1Evaluation criteria

To compare the methods using alternative strategies, as shown in this study, the response speed (speed up) criterion has been used. In the response speed criterion, the speed-to-processing-time ratio and time for many tasks to be completed simultaneously in the processor is determined. This criterion is obtained from Equation (11). In this equation, $\sum_{i=0}^{n} T_i$ is equal to the set of time required to process requests.

$$Response\ time = \sum_{i=0}^{n} T_i$$
(11)

4.3 Simulation Results

The response speed to client requests is compared in round-robin, particle swarm, genetics and whale optimization algorithms.

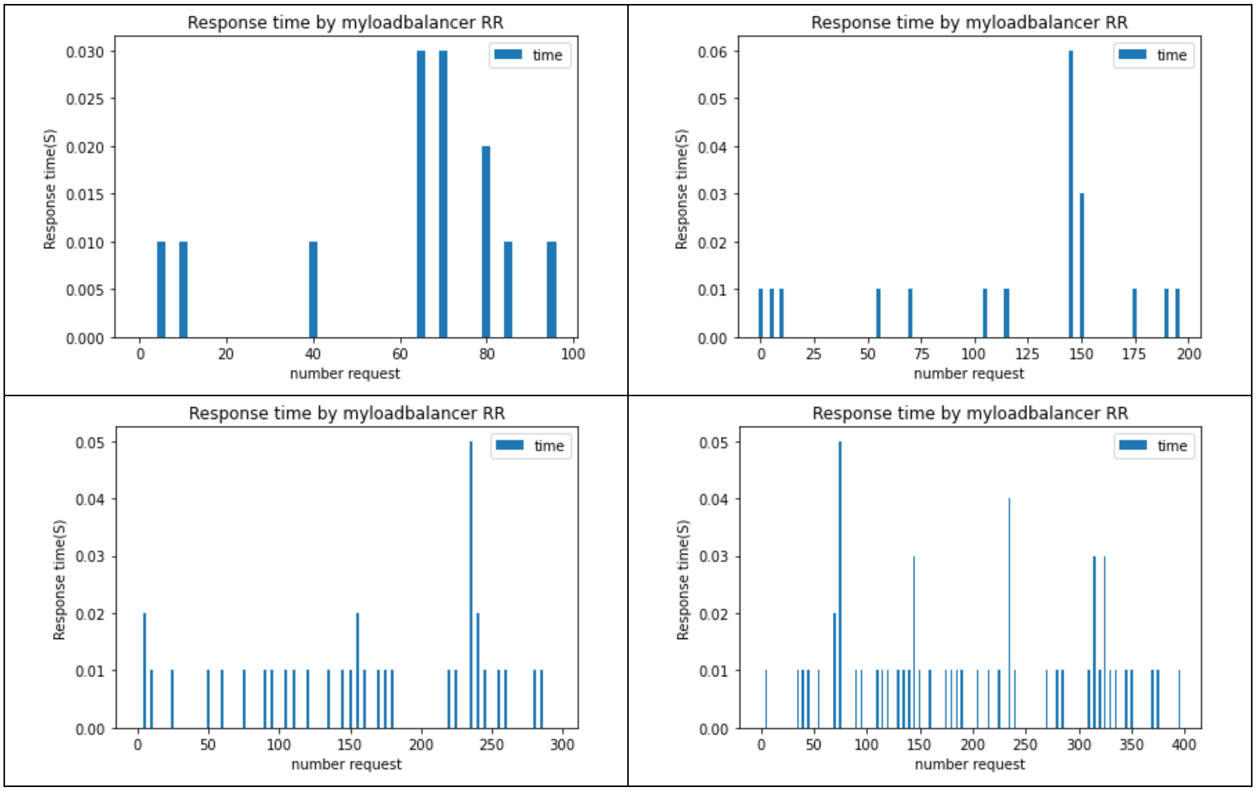


Fig.(7): -Response speed in the Round-Robin algorithm

Figure (7) shows the process of improving the speed of responding to client requests in the Round Robin algorithm. In Fig (7), the average response time in the round-robin algorithm has increased from 0.13 to 0.55, increasing the number of requests.

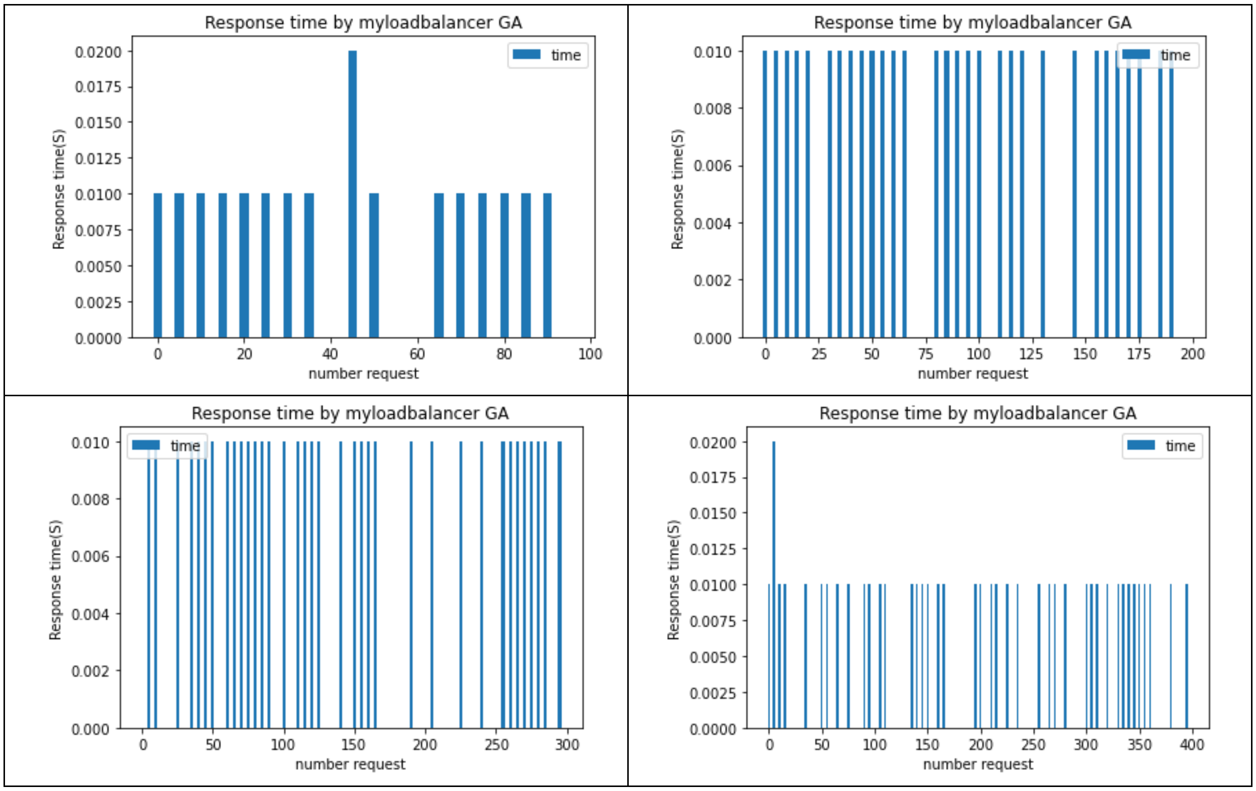


Fig.(8):- Response speed in GA algorithm

Figure (8) shows the process of improving the speed of responding to client requests in the GA algorithm. In Fig (8), the average response time in the genetic algorithm has increased from 0.17 to 0.43 with increasing the number of requests.

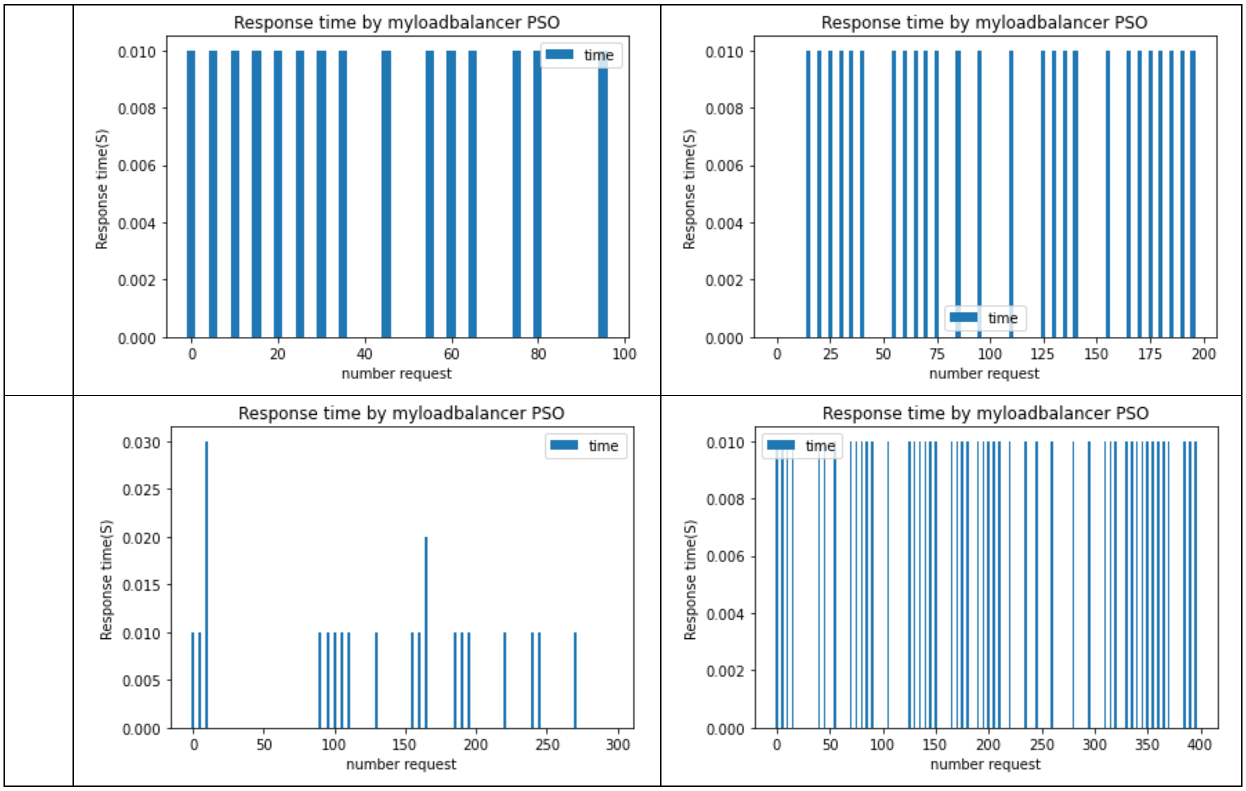


Fig. (9):- Response speed in PSO algorithm

Figure (9) shows the process of improving the speed of responding to client requests in the PSO algorithm. In Fig (9), the average response time in the PSO algorithm has increased from 0.15 to 0.49with increasing the number of requests.

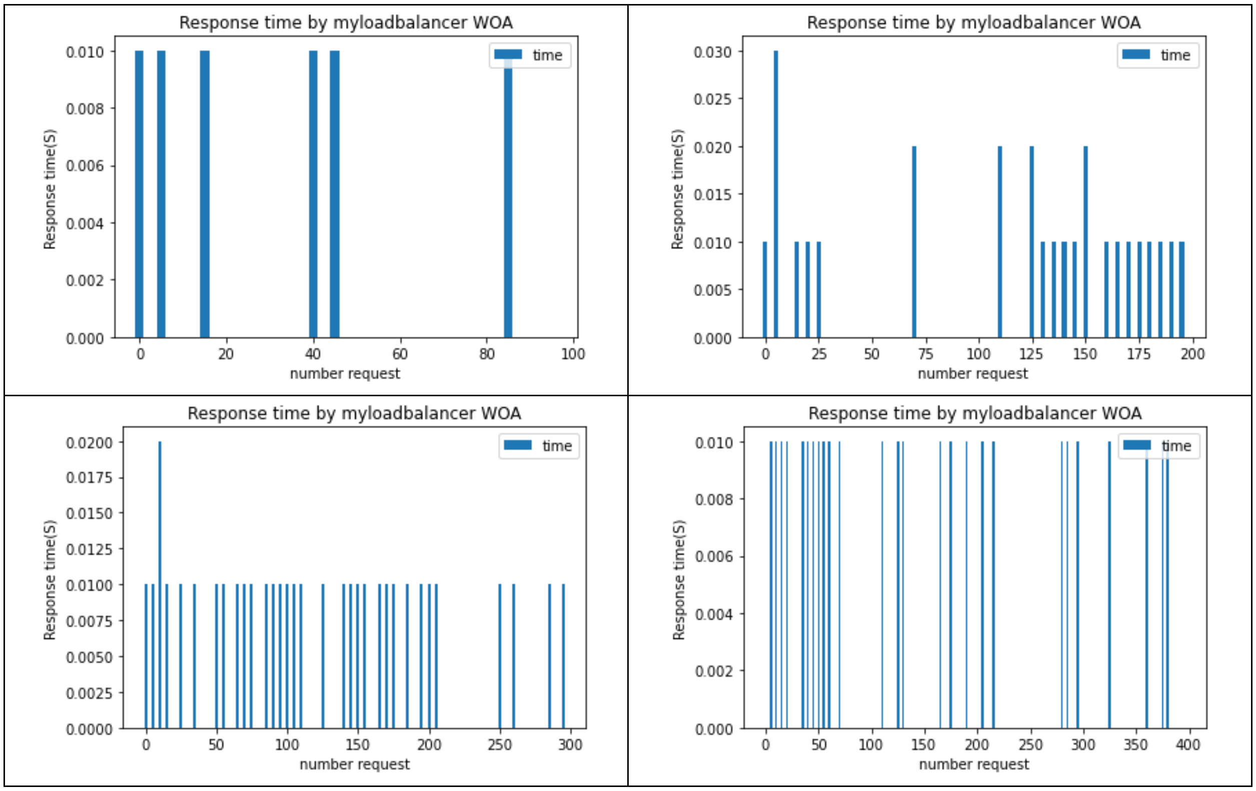


Fig. (10): -Response speed in WOA algorithm

Figure (10) shows the process of improving the speed of responding to client requests in the WOA algorithm. In Fig (10), the average response time in the WOA has increased from 0.06 to 0.34 with increasing the number of requests.

Table (5):- Comparison of the average response time to client requests in the evaluated algorithms

Number of Requests	Round Robin	PSO	GA	WOA
100	**0.13**	**0.15**	**0.17**	**0.06**
200	**0.19**	**0.26**	**0.3**	**0.25**
300	**0.35**	**0.22**	**0.36**	**0.26**
400	**0.55**	**0.49**	**0.43**	**0.34**
Average	**0.305**	**0.28**	**0.315**	**0.2275**

The results of Table (5) show that the response speed average to client requests in the whale optimization algorithm is 0.22 seconds, while in the round-robin algorithm is 0.30 seconds, in the particle swarm algorithm is 0.28 seconds, and in the genetic algorithm, is 0.31 seconds.

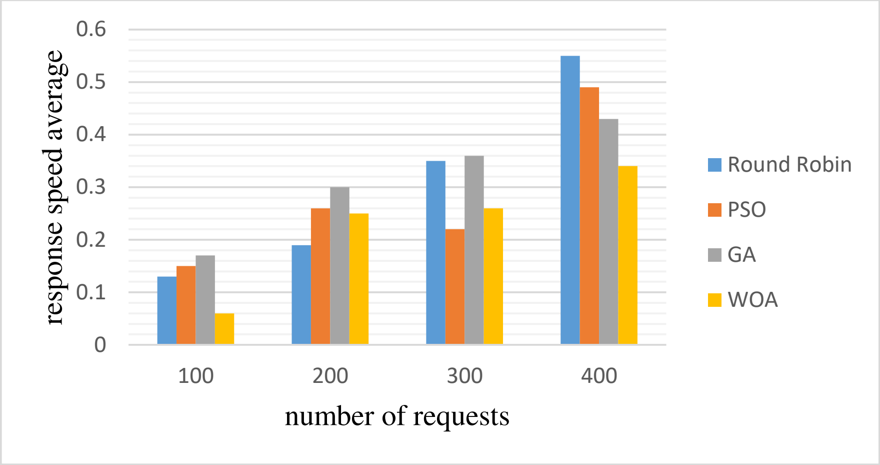


Fig.(11):- Comparison diagram of the average response speed in the evaluated algorithms

The comparison diagram of the average response speed in Figure (11) shows that the average response speed to client requests in the whale optimization algorithm was lower than in other evolutionary algorithms. So the authors can say that the whale optimization algorithm has a better choice for the distribution of client requests between backends. In the whale algorithm, more random selections are made to select the initial population, which leads to the coverage of the search space. In this algorithm, as soon as the ideal search engine has been identified based on the objective function (load balance criterion), other agents try to improve their position on the basis of where the top search agent currently has their employment. Finally, after updating the new position of all agents and ranking them, the best agent is selected as the problem solution (the best distribution of requests among the servers). Random definition of a spatial vector in a whale algorithm causes updating its position with relation to the current best solution for each search agent, thus providing the best possible location.

5. Conclusion and Recommendations

The search results showed that the whale optimization algorithm performed better than the round-robin, particle swarm optimization, and genetic algorithms in reducing the response speed to client requests. The whale optimization algorithm can prevent unexpected traffic and block the normal operation of internet websites by providing a proper plan for distributing requests between servers and reducing the average response speed. So, by applying the whale optimization algorithm, the authors can prevent DDOS attacks. It necessary be noted that the use of HAProxy to prevent DDOS is not enough alone. Therefore, several layers of software and hardware security necessary be used depending on the type of attack. The request distribution plan among the servers was considered a complete NP problem.

In this study, various optimization techniques Genetic algorithm, particle swarm optimization, and whale optimization algorithm, were among the first algorithms to be created. Although these are a good way to solve the optimization problem, the efficiency and quality of their solution depend on the proper adjustment of the control parameters of the optimization algorithm; because increasing the number of these parameters will lead to a lot of time spent. To overcome these problems, the authors will need to use a method that can find a suitable solution for load balancing in future times by learning the optimal distribution pattern of the requests between servers. To learn the pattern of task allocation, the authors can use the reinforcement learning method. It's a machine learning technique that allows a factor to learn via trial and error and feedback from their actions and experiences in an interactive environment. In reinforcement learning, the agent receives a reward when the agent chooses the appropriate task assignment pattern for the computational nodes in a particular case. In this type of machine learning, the goal of the agent will be to maximize the received reward in the long run. Therefore, given the extent of the solution space of the load balancing problem and the multi-objective nature of the problem, future researchers are recommended to use the reinforcement learning technique in load balancing to deal with DDOS attacks.

There is no related work that uses DDOS attacks, because we didn’t find any previous works addressed the Whale algorithm for the DDoS attacks. Adding to that, with the depended whale algorithm in this paper, the number of the requests can be increased without overload be occurrance. Also, the response time will Hence, t .decreasedhe results of Table (5) can be depended as findings from the comparison table: the response speed average to client requests in the whale optimization algorithm is 0.22 seconds, while in the round-robin algorithm is 0.30 seconds, in the particle swarm algorithm is 0.28 seconds, and in the genetic algorithm, is 0.31 seconds.

REFERENCES

Danielsen, V. (2021). Detecting Yo-Yo DoS attack in acontainer-based environment (Master's thesis, OsloMet-storbyuniversitetet).

Praseed, A., & Thilagam, P. S. (2018). DDoS attacks at the application layer: Challenges and research perspectives for safeguarding web applications. IEEE Communications Surveys & Tutorials, 21(1), 661-685.

Oricco, P. (2022). Analysis and implementation of load balancers in real-time bidding (Master's thesis, UniversitatPolitècnica de Catalunya).

Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., &Alzain, M. A. (2021). A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications. IEEE Access, 9, 41731-41744.

Oricco, P. (2022). Analysis and implementation of load balancers in real-time bidding (Master's thesis, UniversitatPolitècnica de Catalunya).

Hamid, L., Jadoon, A., &Asghar, H. (2022), Comparative analysis of task level heuristic scheduling algorithms in cloud computing. The Journal of Supercomputing, 1-19.

Sharma, V., & Sharma, H. C. A Review of Cloud Computing Scheduling Algorithms.

K.R. RemeshBabu, Philip Samuel (2015).Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud, Innovations in Bio-Inspired Computing and Applications, Advances in Intelligent Systems and Computing book series, volume 424, pages 67-78.

U.K. Jena, P.K. Das and M.R. Kabat (2020). Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment, Journal of King Saud University – Computer and Information Sciences, Available online.

K. JairamNaik(2020). A Dynamic ACO-Based Elastic Load Balancer for Cloud Computing(D-ACOELB)", Data Engineering and Communication Technology, volume 1079, pages 11-20.

F. Ramezani, J. Lu, & F.K. Hussain (2014). Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization, International Journal of Parallel Programming, volume 42, pages739–754.

C.-M. Cheung, K.-C. Leung (2018). DFFR: A flow-based approach for distributed load balancing in data center networks, Computer Communications, volume 116, pages 1-8.

M. Adhikari, T. Amgoth (2018). Heuristic-based load-balancing algorithm for IaaS cloud, Future Generation Computer Systems, volume 81, pages 156-165.

A.Tripathi, S. Shukla, D. Arora (2018). A hybrid optimization approach for load balancing in cloud computing", Advances in Computer and Computational Sciences, Springer, Singapore, pages 197–206.

F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, V.C.M. Leung (2018) Dynamic resource prediction and allocation for cloud data center using the multi objective genetic algorithm", IEEE Systems Journal Volume: 12, Issue: 2, pages 1688 –1699.

Pramono, L. H., Buwono, R. C., &Waskito, Y. G. (2018). Round-robin algorithm in HAProxy and Nginx load balancing performance evaluation: a review. In 2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) (pp. 367-372). IEEE.

Anicas, M. (2017). An introduction to HAProxy and load balancing concepts.

Fahim, Y., Rahhali, H., Hanine, M., Benlahmar, E. H., Labriji, E. H., Hanoune, M., &Eddaoui, A.

(2018). Load balancing in cloud computing using meta-heuristic algorithm. Journal of Information Processing Systems, 14(3), 569-589.

Jain, N. K., Nangia, U., & Jain, J. (2018). A review of particle swarm optimization. Journal of the Institution of Engineers (India): Series B, 99(4), 407-411

Mishra, S. K., &Manjula, R. (2020). A meta-heuristic based multi objective optimization for load distribution in cloud data center under varying workloads. Cluster Computing, 23(4), 3079-3093.

Miao, Z., Yong, P., Mei, Y., Quanjun, Y., & Xu, X. (2021). A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment. Future Generation Computer Systems, 115, 497-516.

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80(5), 8091-8126.

Vijarania, M., Agrawal, A., & Sharma, M. M. (2021). Task Scheduling and Load Balancing Techniques Using Genetic Algorithm in Cloud Computing. In Soft Computing: Theories and Applications (pp. 97-105). Springer, Singapore.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Advances in engineering software, 95, 51-67.