Chapter

An Empirical Survey on Load Balancing: A Nature-Inspired Approach

Surya Teja Marella and Thummuru Gunasekhar

Abstract

Since the dawn of humanity man tried to mimic several animals and their behavior be it in the age of hunting of while designing the aero plane. Human brain holds a significant amount of power in observing the species around him and trying to incorporate their behavior in several walks of life. This mimicking has helped human to evolve into beings which we are now. Some typical examples include navigation systems, designing several gadgets like aero planes, boats, etc. These days these inspirations are several, and their inspiration is being utilized in several fields like operations, supply-chain management, machine learning and several other fields. The similar kind of approach has been discussed in this paper where we tried to analyze different phenomenon in nature and how different algorithms were designed from these and how these can ultimately be used to solve different issues in cloud balancing. Essential component of cloud computing is load balancer which holds a crucial role of task allocation in virtual machines and several kinds of algorithms were developed on different ways of task allocation procedures each holding its significance here we tried to find the optimal resource allocation in terms of task allocation and rather than approaching through traditional methods we tried to solve this issue by using soft computing techniques. Specifically, nature-inspired algorithms as it hold the key to unlocking massive potential regarding research and problem-solving approach. The central idea of this paper is to connect different optimization techniques to load balancer and how could we make a hybrid algorithm to serve the purpose. We also discussed several different types of algorithms each bearing its roots from different natural procedures. All the algorithms in this paper can be broadly tabulated into three different types SO (Swarm optimization techniques), GO (Genetic-based algorithms), PO (Physics-based algorithms).

Keywords: load balancing, cloud computing, nature-inspired algorithms, optimization techniques

1. Introduction

Load balancing implies guaranteeing the even distribution of workloads and to adjust the load among the accessible resources ideally. It helps in accomplishing a high client fulfillment and asset utilization proportion. Many scheduling algorithms have been proposed to maintain load balancing. The primary point is the ideal assets utilization resulting in improved throughput, migration times or smaller response,

ideal adaptability, and overall system production [1]. There are individual difficulties in cloud computing, and that needed to be routed to give the most reasonable and productive effective load balancing algorithms. These challenges are:

1. Geographical/spatial distributions of the nodes: to design an LBA that works for spatially or geographically distributed nodes is an arduous task. It is because as the distance increases the speed of the network links among the nodes is influenced which thusly influences the throughput [2].

2. The complexity of algorithm: complexity affects the overall performance of a system. Generally, LBA has a less complicated implementation. Complexity lead to delays which further causes more problems [3].

3. Point of failure: the load balancing algorithm ought to be planned in a way that they abstain from having a single point of failure.

4. Static load balancing algorithm: a static load balancing algorithm works on the earlier state/previous data, not on the ongoing state. It cannot adjust to the load changes at run-time [4].

The challenges as mentioned above remain unsolved. Researchers are going on to bring about changes in the existing algorithms and to create new algorithms to overcome these challenges.

Cloud load balancing is the activity of regulating the workload and computing assets in a cloud computing environment to accomplish high performance at potentially lower costs. This includes facilitating the dissemination of workload activity and requests that dwell over the Interweb [5]. As we know that, a load balancing technique is basically appropriating workloads among the servers and processing assets in a cloud domain in which the number of clients where more significant than the servers so that there can be the burden on the servers so we need to balance the load so we distribute the tasks among the servers equally so it cannot be the bash with any other server and in this way we can increase the performance of a server [6]. By allocating the resources among the various computer network or server, Load balancing allows companies or organizations to manage the applications or workload demands, so load balancing in cloud computing that incorporate facilitating the distribution of workload activity and request over the system so first level every one of the customers have been composed, In second level all the servers have been organized and in between these two a load balance rare used to balance the load among the server and is generally used by the company or organizations to manage their applications. Cloud computing is an advanced worldview to give benefits through the Interweb [7]. Load balancing is an essential part of cloud computing and avoids the situation in which a couple of nodes wind up finished weight while the others are sitting still or have little work to do. Load balancing can upgrade the Quality of Service (QoS) estimations, including response time, cost, throughput, execution and asset utilization. In computing, Load balancing [8] enhances the distribution of workloads over various figuring assets, for example, PCs, a PC gathering, central processing units, network links, or disk drives. Load balancing means to enhance asset usage, maximize throughput, confine response time, and maintain a strategic distance from over-weight of any single asset. Using different parts with load balancing rather than a single component may increase reliability and accessibility through excess. The load balancing in clouds may be among physical hosts or VMs. This balancing segment scatters the dynamic workload fairly among every one of the hubs (hosts or VMs).

The heap adjusting in the cloud is additionally alluded to as load adjusting as an administration (LBaaS) [9] (**Figure 1**).

Load adjusting is to move the workload to computational assets that are underutilized, with an outrageous objective of lessening the general execution time. A considerable measure of research has been added to the point, and this case continues with framework figuring and disseminated computation [10]. In the area of multi-core computing, a typical multi-core framework comprises of same cores that communicate using shared memory space. Similarly stays consistent with GPUs also [11]. Thus, a notwithstanding dividing of the load among accessible cores should do the trick to deliver a base execution time. Regardless, this dispute is effectively countered by the manner in which that we can have a gathering of workloads, each with various or obscure computation requirements. An even or approach apportioning would not be doable. If a distributed memory system is utilized as an execution stage, correspondence overheads can turn into a genuine performance concern [11].

As a rule, legitimately dividing the workload is basic to boosting execution. Toward this objective, we ought to think about a comparable number of the stage (e.g., computational speed) and issue attributes (e.g., cost of data transmission) as conceivable [12]. Dynamic load adjusting insinuates a wide assembling of computations that perform or change stack assignments on the Web, i.e., in the midst of

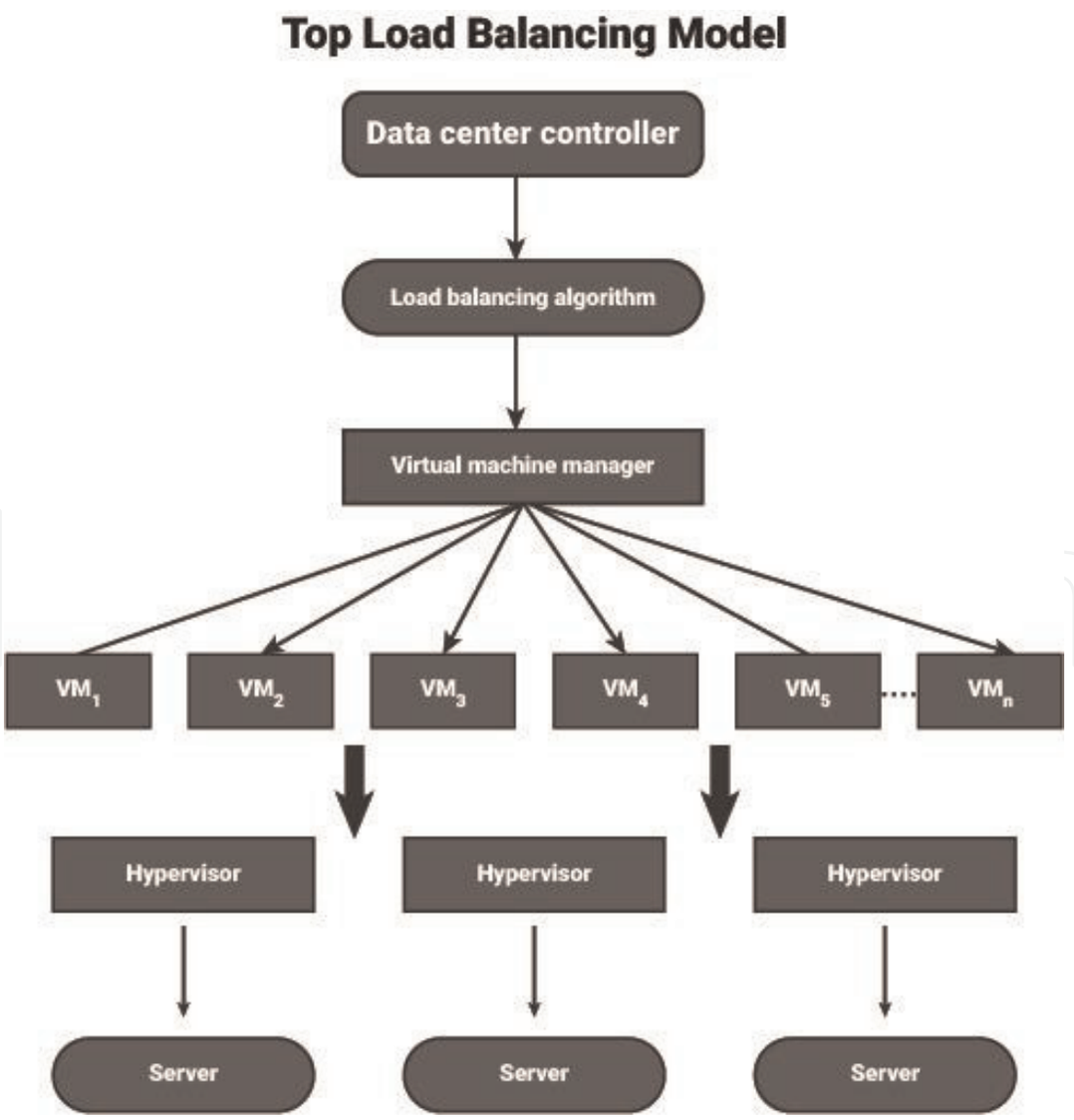

Figure 1.
Load balancing model.

the execution of a program [13]. Dynamic load balancing is depicted by the ability to adjust to changes to the execution organize (e.g., hubs going isolates, correspondence joins finding the opportunity to be congested, and whatnot.) yet to the burden of additional coordination overhead [14]. Static load modifying can give a close ideal answer for the load-partitioning issue, the exchange offs being the failure to adapt to run-time changes and the need to develop insinuate learning about the execution characteristics of the individual fragments making up the execution arrange [15].

To attain these goals, two types of load balancing algorithms are designed:

1.1 Static load balancing

In static load balancing, cloud requires knowledge of processing power, performance, nodes capacity and memory [5]. The cloud additionally requires learning of client necessities which cannot be changed on run-time. It is simpler to mimic the static condition, yet in the event that client necessities change static condition cannot adjust to it. Two well-known algorithms applied in static environments are [16].

1.1.1 Round Robin algorithm

In the Round Robin algorithm, assets are allocated to errand on First-come-First-Serve (FCFS) premise. It implies the errand which arrives first; assets are distributed to it first. In this algorithm, tasks are scheduled in time sharing manner [17].

1.1.2 Central load balancing decision model

It is an improved approach to Round Robin algorithm. It uses the basics of the Round Robin algorithm. This algorithm calculates total execution time spent by a task on the cloud. It uses this information to calculate time elapsed during client and server communication [18].

1.2 Dynamic load balancing

In a dynamic environment, various resources are installed. In a dynamic environment, cloud considers runtime statistics [19]. In a dynamic environment, the cloud allows changes in user requirements on runtime. Algorithms in a dynamic environment can quickly adapt to runtime changes. The dynamic environment is challenging to simulate [20]. Various load balancing algorithms implemented in a dynamic environment are weighted least connection (WLC) algorithm, load balancing min-min (LBMM) algorithm and opportunistic load balancing (OLB) algorithm [21].

Considering the scalability and free nature of the cloud, dynamic environments are preferred over static environments for cloud implementation as it also satisfies the particular goals of load balancing as given below,

Goals of load balancing are:

- To maintain the fault tolerance of the system.
- To maintain the stability of the system.
- To improve efficiency and performance of the system.
- To minimize job execution time.

- To minimize time spent waiting in the queue.

- To facilitate improved resource utilization ratio

1.3 Related works

Each technique holds significance and a different approach to solve the problem. Swarm optimization is generally used to find an optimal solution it might not be the best fit solution, Genetic algorithms generally try to find the best fit but it consumes much time, and physics algorithm generally used as hybrid or support algorithm to minimize other procedures in different algorithms. Swarm optimization is generally inspired by observing different flock behaviors be it in frogs, bats, fireflies and ants and each takes a specific approach to the problem of food gathering, communication, foraging, etc. [22]. A genetic algorithm is approximately enlivened by Charles Darwin theory of survival of fittest and algorithms like a genetic algorithm, mimetic algorithm come under this category. Physics algorithms are inspired from different physical phenomena like gravitation, mass-energy equivalence, simulated annealing, etc. These generally act as support factors for different algorithms. In every algorithm the process can be generalized as randomization, calculating fitness and arriving at a possible solution. The approach varies, but the process is more or less the same. The underlying algorithms used these days in different spheres are ant colony optimization; in the chapter, genetic algorithm and simulated annealing are also discussed. We also discussed concepts like task allocation based on a few traditional algorithms. Factors like green computing which affect the performance of the device considerably are also discussed. To conclude we tried to solve the optimal resource allocation in a natural way because nature itself looks for best fitting procedures for its procedures and mimicking it in computing could be advantageous [25].

To expand the general execution of the framework, load balancing is an intense instrument that aides in conveying bigger workloads into smaller processing workloads. To achieve proper resource utilization and excellent user satisfaction, it helps in the fair allocation of computing resources. It avoids bottlenecks and implements failover thus increasing the scalability. To transfer and receive data without any delay, load balancing divides the traffic between all the servers to get an optimum solution. The central vision of load balancing is to make sure that at any point in time, the processors in the system does the same amount of work. It is necessary for load balancing to utilize full parallel and distributed system's resources. Load balancing is classified as dynamic load balancing and static load balancing [23]. The processor's performance is decided at the beginning of execution in static load balancing. From that point onward, as per their execution, the workload is partitioned by the ace processor. It should be possible utilizing algorithms named "central manager algorithm," "threshold algorithm," and "round robin algorithm." The work is divided during the runtime in dynamic load balancing. With new information collected by the master, new processes it assigned to the slaves. Here, processes are allocated dynamically. It can be carried out using algorithms such as, "local queue algorithm" and "central queue algorithm" [24].

A load balancer can perform the following functions:

1. Distributes network load and requests of clients across many servers.

2. According to demand, it can add and subtract the servers.

3. Provides high scalability, reliability and availability to the online servers.

To scale with the increased demands, vendors of the cloud are more toward automatic load balancing services that allow the entities to increase the memory and count of CPU for their resources.

2. Models of nature-inspired algorithms

Nature-inspired algorithm is a capacity that aroused by activities that are perceived by nature. These registering approaches prompted the change of development named nature-inspired algorithms (NIA) [25]. This breakthrough is apt for computational agility. The objective of developing such algorithms is to optimize engineering problems [26] (**Figure 2**).

These algorithms use recombination and change overseers to streamline the perplexing issues, e.g., genetic algorithm and differential evolution et cetera. The basic objective of nature-inspired algorithms is to discover a universally response for a given issue. Two key factors normal in all nature-inspired algorithms are strengthening and expansion regularly named as Exploration and Exploitation [27].

Some of the algorithms are,

2.1 Hill climbing

It is a nature-inspired algorithm which takes its inspiration from a process of hill climbing. It is an iterative algorithm. This algorithm is thus helpful to find the minimal solution and can be used in the load balancer [28]. A tool called load balancer which is used to find the assets allocation in the cluster and several kinds of algorithms are used to find the resource allocation to the cluster and here to find the solution we can use the hill climbing algorithm.

Initially, the algorithm considers the cluster as a graph and mimics hill climbing behavior on that graph and generates a solution for optimal resource allocation [29].

To locate the ideal answer for the given issue this strategy can be utilized, this method can be utilized as a part of load adjusting to locate the ideal asset assignment for the given issue.

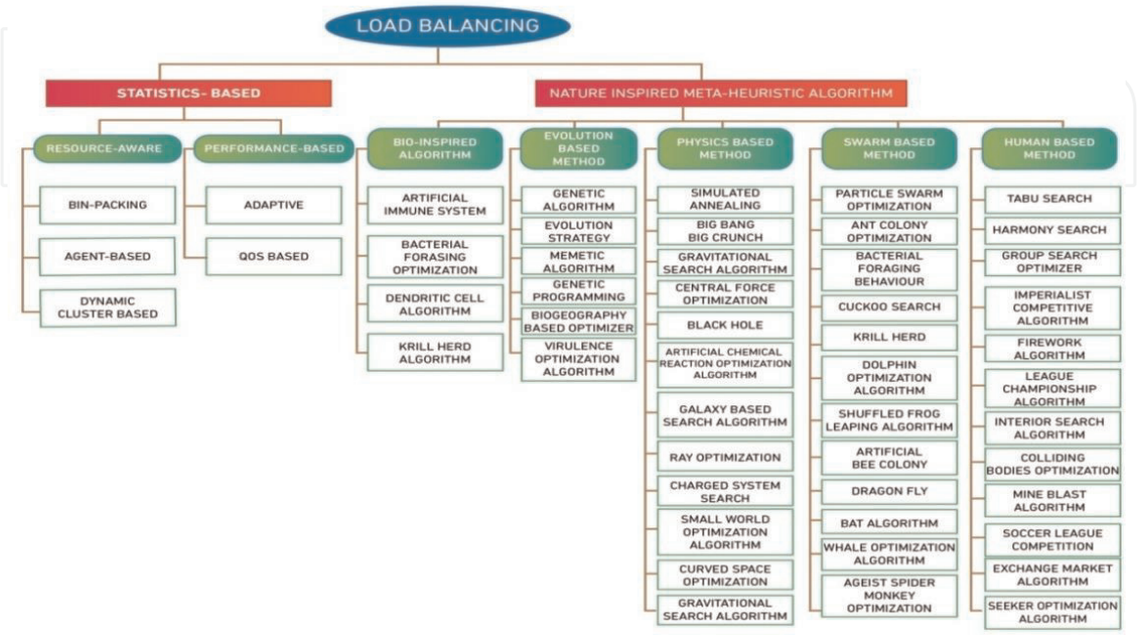

Figure 2.
Load balancing taxonomy.

This technique has three different variants,

1. **Coordinate descent**: this technique is optimally used to determine the optimal solution for the given problem but cannot be implemented as it has an exponential time worst case scenario.

2. **Stochastic hill climbing**: this technique does not guarantee an optimal solution, but it is better than stochastic hill climbing regarding the time taken to find the optimal solution.

3. **Traditional hill climbing approach**: this technique cannot be used in the load balancer as the solution we get out of this is not optimal.

Hence, here we describe a stochastic hill climbing approach as it is best regarding both time complexity and optimization (**Figure 3**).

```
Present node = initialNode;
 loop
   k=neighbours(presentNode);
nextEvalatuation = -knf
nextNode = NULL;
    for all x in L
       if (EVALuation(x) >nextEvalution)
nextNode = j;
nextEvaluation = evaLuation(x);
    if nextEvaluation<= evaluation(presentNode)

       return presentNode;
presentNode = nextNode;
```

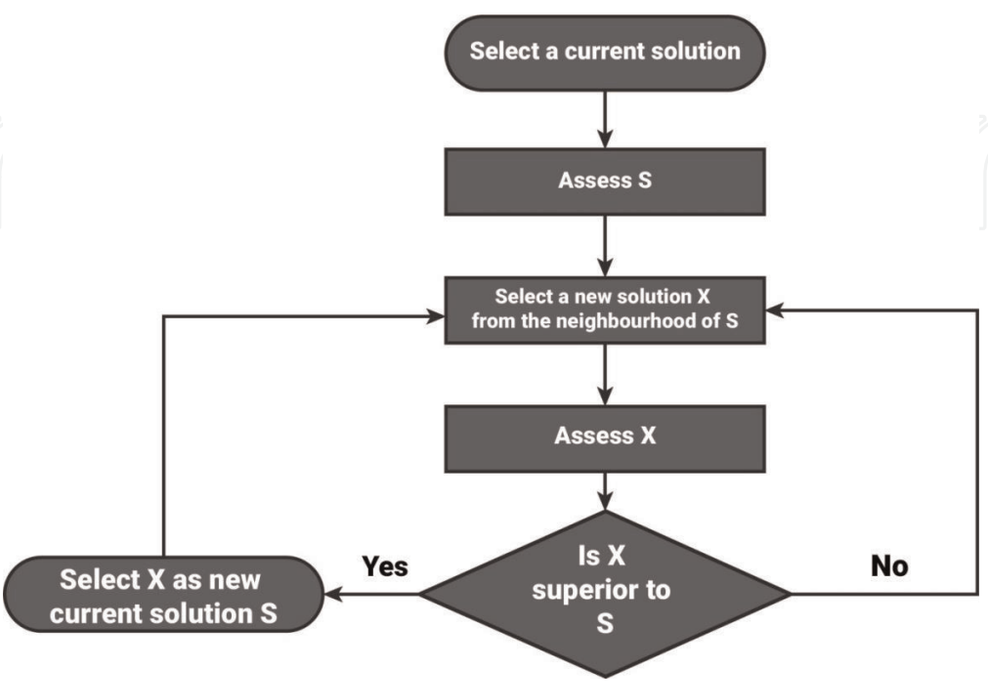

Figure 3.
Hill climbing.

2.2 Ant colony optimization

In nature-inspired algorithms the "ant colony optimization algorithm (ACO)" is a probabilistic intends to determine computational issues which can be diminished to finding the correct routes through graphs. Artificial Ants remain for multi-agent strategies motivated by the conduct of genuine ants. The pheromone-based communication of organic ants is frequently the dominating worldview utilized. Mixes of artificial ants and nearby local search algorithms have turned into a strategy for decision for various improvement tasks including some like vehicle routing and Internet routing [30].

Load balancing technique is essentially procedure which is used to find the optimal solution to given resource distribution problem in the given cloud computing scenario. Ant colony optimization considers the given problem as a graph and tries to find the optimal path and optimal resource allocation for the given problem can be found by using this specific procedure. The algorithm starts from randomness to optimization and alleviates the problem of the cluster. Initially, ants wander randomly and come back to the source by laying pheromone in their path. Then remaining ants follow the path set by the ants using pheromone [31]. There is a problem of pheromone getting evaporated this will allow for the shortest path determination. If it takes more time in this path, then the pheromone will evaporate there only shortest path pheromone survives to contribute to the shortest path. All the ants follow the same path to follow others eventually Thus after getting the shortest path whole ants to follow this path leading to the whole system following this path. This algorithm is more advantageous as it tackles the dynamic allocation problem easily. To solve the load balancing in a cloud environment the ant-based control system was designed. Each and every node was configured with capacity of being a destination, the probability of being destination, pheromone table. There are many variants for this technique namely Elitist depicts that global best solution gives pheromone update after every iteration, Max-Min ant system takes a min-max data structure and updates its value from minimum to maximum, Rank-based ant system takes all the possible solution and ranks according to sum of weights, continuous orthogonal ant colony takes additional angle parameter which makes for efficient searching, Recursive ant colony optimization takes solution and uses genetics to find out best solution [32].

The whole algorithm can be divided into different phases Edge selection and pheromone update. Initially randomly all ants are placed in the random order and after all the ants placed we updated pheromone level by using this formula Pxy<-(1-p)*pxy+ sum of all pheromone levels (**Figure 4**).

Here pxy is pheromone level which can be calculated by using this formula.

$$\text{Del}(\text{the k}) = \text{J here j is the perimeter of curve xy for straight line use } 0.$$

2.3 Artificial bee colony

The "artificial bee colony (ABO)" algorithm is a swarm based meta-heuristic algorithm. The algorithm is constructed unequivocally with respect to the model that is proposed for the scrounging conduct of honey bee settlements. The excellent comprises of three imperative parts: used and unemployed foraging honey bees, and sustenance sources. The underlying two sections, used and unemployed foraging honey bees, examine for rich sustenance sources, which is the third fragment, close to their hive. The model in like manner describes two driving techniques for lead which are basic for self-sorting out and total knowledge: enlistment of foragers to bounteous sustenance sources achieving positive information and surrender of poor

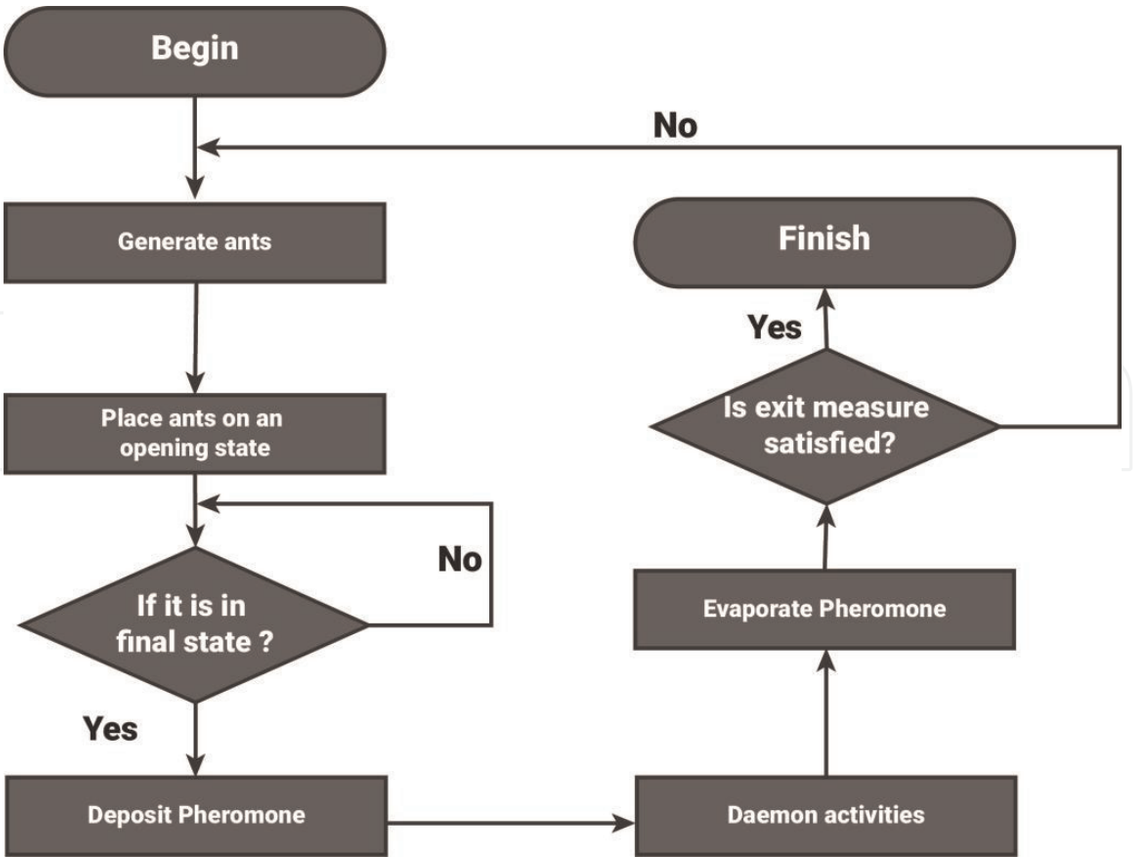

Figure 4.
Ant colony optimization.

sources by foragers causing negative feedback [33]. In ABC, a state of artificial forager honey bees check for rich phony sustenance sources. To apply ABC, the treated change issue is first changed to the issue of finding the best parameter vector which lessens a goal work. By then, artificial forager honey bees erratically discover a people of beginning arrangement vectors and a while later iteratively improve them by using the techniques: moving toward better arrangements utilizing a neighbor search mechanism while forsaking poor arrangements [34].

A technique called artificial bee colony optimization which is used to find the optimal solution to the given problem. It models the given problem as a graph and algorithm mimic the behavior of bees to approach a solution to the given Here problem refers to optimal resource allocation in given load balancer then resource allocation is done accordingly. ABC is an algorithm which takes inspiration from bees and tries to find the shortest path for the given graph cluster. Derviskaraboga developed it in 2005. The position of bees is modified to find out best position with the highest nectar since it is a population-based search procedure. Generally, bees perform a waggle dance to convey information regarding distances and directions. The whole graph system can be modeled into two significant components of food sources and foragers. Further classified of foragers can be done as unemployed foragers, employed foragers and experienced foragers. The algorithm can again be divided into four different phases [35].

(1) Initialization phase: the initial food sources are allocated randomly by using this formula.

$$\mathrm{Km} = \mathrm{lb} + \mathrm{random}(0,1) * (\mathrm{ub} - \mathrm{lb}) \tag{1}$$

Here lb, ub are **lower** and upper bound of solution space of objective function.

(2) Employed bee phase: the neighboring food source Nmi is determined by using the following formula.

$$\text{Nmi} = \text{Kmi} + \text{random}[-1, 1] * (\text{Kmi} - \text{Kki}) \tag{2}$$

Here i is any **randomly** selected parameter index.

Here fitness is **calculated** by using the following formula and a greedy algorithm is applied.

$$\text{Fitness}(\text{Km}) = 1/1 + \text{obj}(\text{Km}), \text{obj}(\text{Km}) > 0, \text{Fitness}(\text{Km}) = 1 + |\text{obj}(\text{Km})|, \text{obj}(\text{Km}) < 0$$

Here **obj**(Km) is objective function of Km.

(3) Onlooker bee phase: the quantity of food sources is the ratio of total fitness to the individual bee fitness.

$$\text{Profitm} = \text{individual fitness/sum of all fitness.}$$

Onlooker bee uses the formula (2) for neighboring food source.

(4) Scout phase: the new solutions are randomly searched by the scout bees.

The new **solution** Km is randomly searched by following formula.

$$\text{Km} = \text{lb} + \text{random}(0, 1) * (\text{ub} - \text{lb})$$

Here ub and **lb** are upper and lower bounds to the solution phase (**Figure 5**).

2.4 Genetic algorithm

The approval of GA is situated on four considerations: populace estimate, mutation rate, crossover rate and the number of generations. To control the first rate individual among a masses, qualities of comparing people are ordered against the objective function. The formative structure through which another successor is conveyed is crossover and mutation. In the crossover mechanism, an offspring is conveyed by joining the characteristics of consolidating the qualities of chose people among populace while transformation causes some irregular changes in qualities of an individual in this manner delivering new hereditary person. The transformative mechanism is finished to the moment that joining criteria are satisfied [36].

The genetic algorithm is used to find out the optimal solution to the given cluster. The approval of GA is situated on four contemplations: populace estimate, A genetic algorithm is a refinement algorithm natural selection (survival of the fittest) is the one inspired genetic algorithm, Essentially we have to show the given bunch and attempt to explain it by utilizing a hereditary calculation. The genetic algorithm performs optimization and searching using three different bio-inspired operators such as mutation, crossover and selection [37]. One genetic representation of all the possible solutions and a fitness function is required by typical GA to figure out the quality of the given solution. The given problem can be either modeled as a graph or tree, and a genetic algorithm can be applied to this to determine the optimal solution. The genetic algorithm borrows its properties from natural selection by Darwin or Darwinian evolution theory. Hence we need to

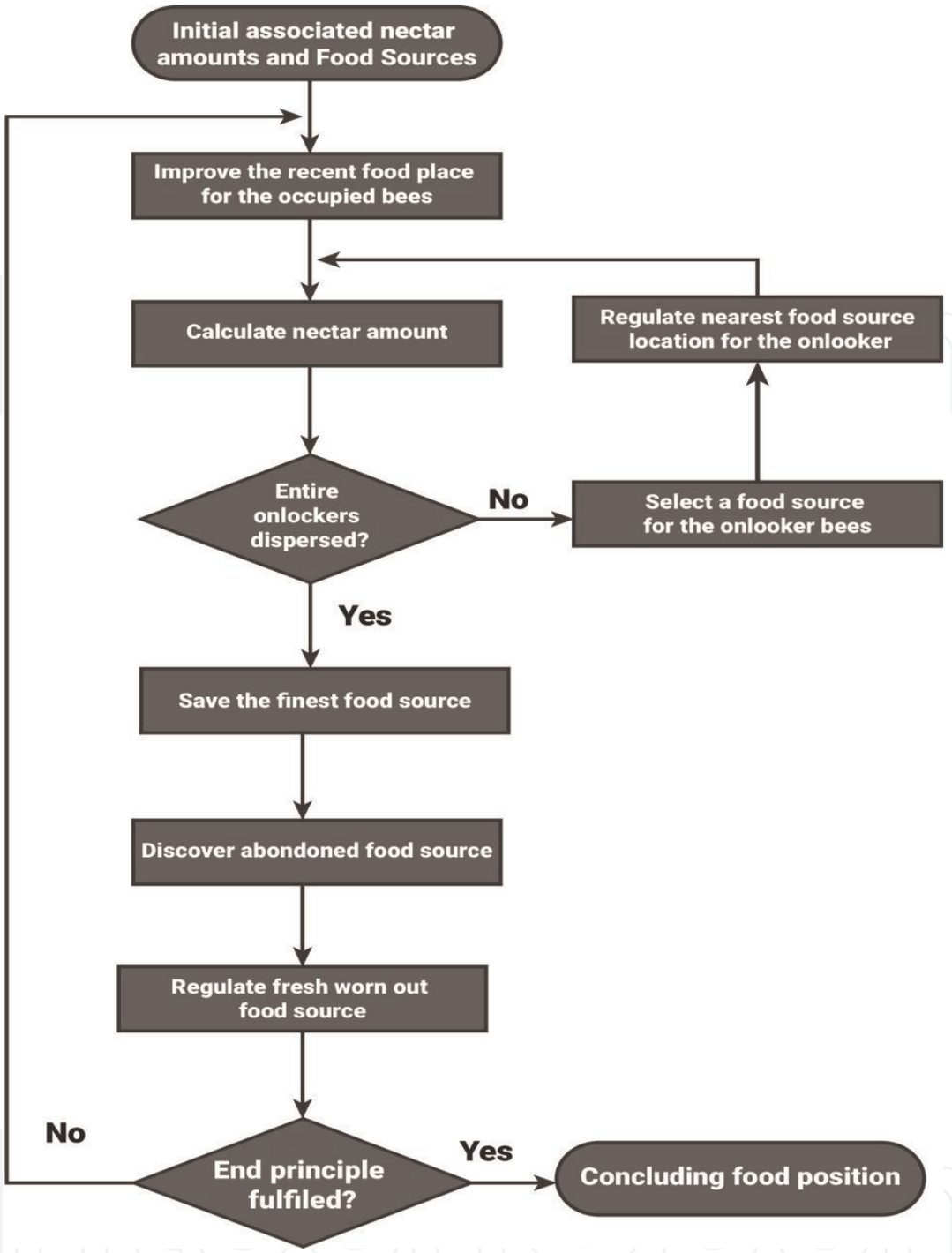

Figure 5.
Artificial bee colony (ABC).

implement these characteristics in our algorithm. The three significant steps in this algorithm are Initialization, selection and validation.

1. **Initialization step**: it is performed in randomness here we take all the possible solutions and list them out.

2. **Selection step**: with the help of probabilities possible solutions are listed out based on probabilities and parents are selected according to highest probabilities to form a solution.

3. **Termination step**: in the final step, we use different techniques like mutation, crossover, etc. to form a solution and this solution is evaluated by using fitness

function. This process is iterated until we form an optimal solution [38]. As it is an iterative technique this can perform for the infinite amount of time this should be terminated at some of the other instants to consider the optimal solution to the given problem. This can be terminated at either any of these cases:

a. If the latest solution persuades minimum criteria.

b. Fixed no of generations reached.

c. Allocated budget used up.

d. Manual inspection.

In its heart, genetic algorithm follows each of this basic mechanism:

a. Variation: this is implemented in the initialization part here we need to take variation as this would contribute to the formation of a solution.

b. Selection: this is implemented in the selection part here we take or consider the solution which has a higher probability to form a solution and remaining are eliminated.

c. Hereditary: this is implemented in the final stage whereby use of process like mutation allows the new solution to have two properties (**Figure 6**).

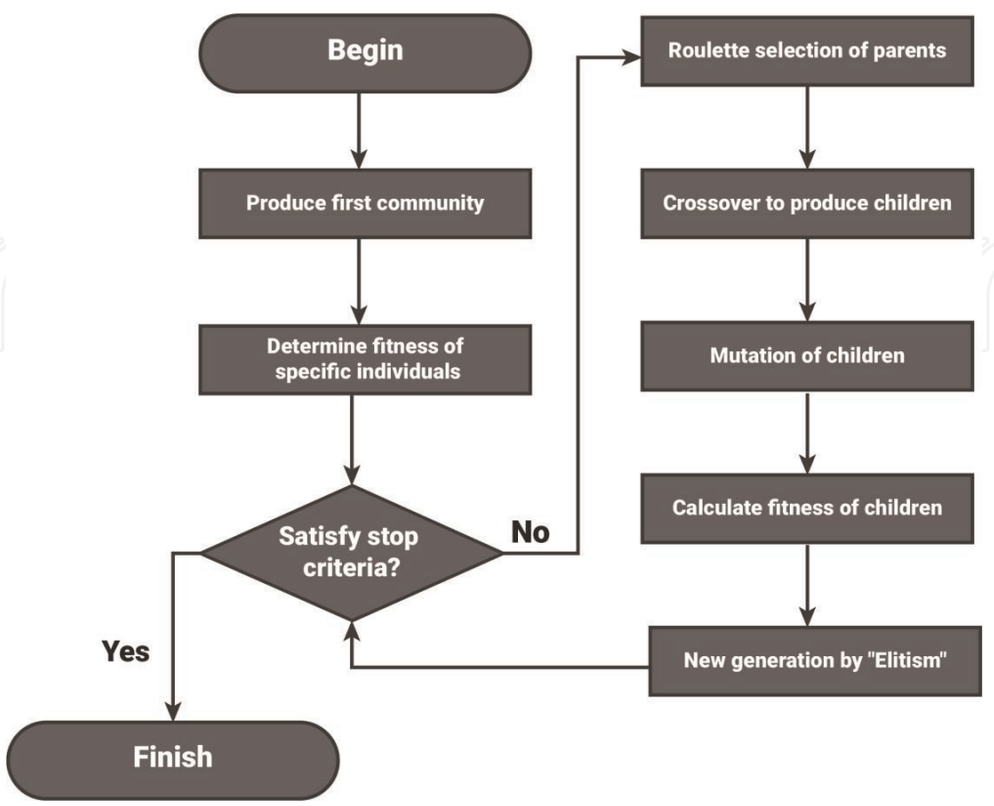

Figure 6.
Genetic algorithm.

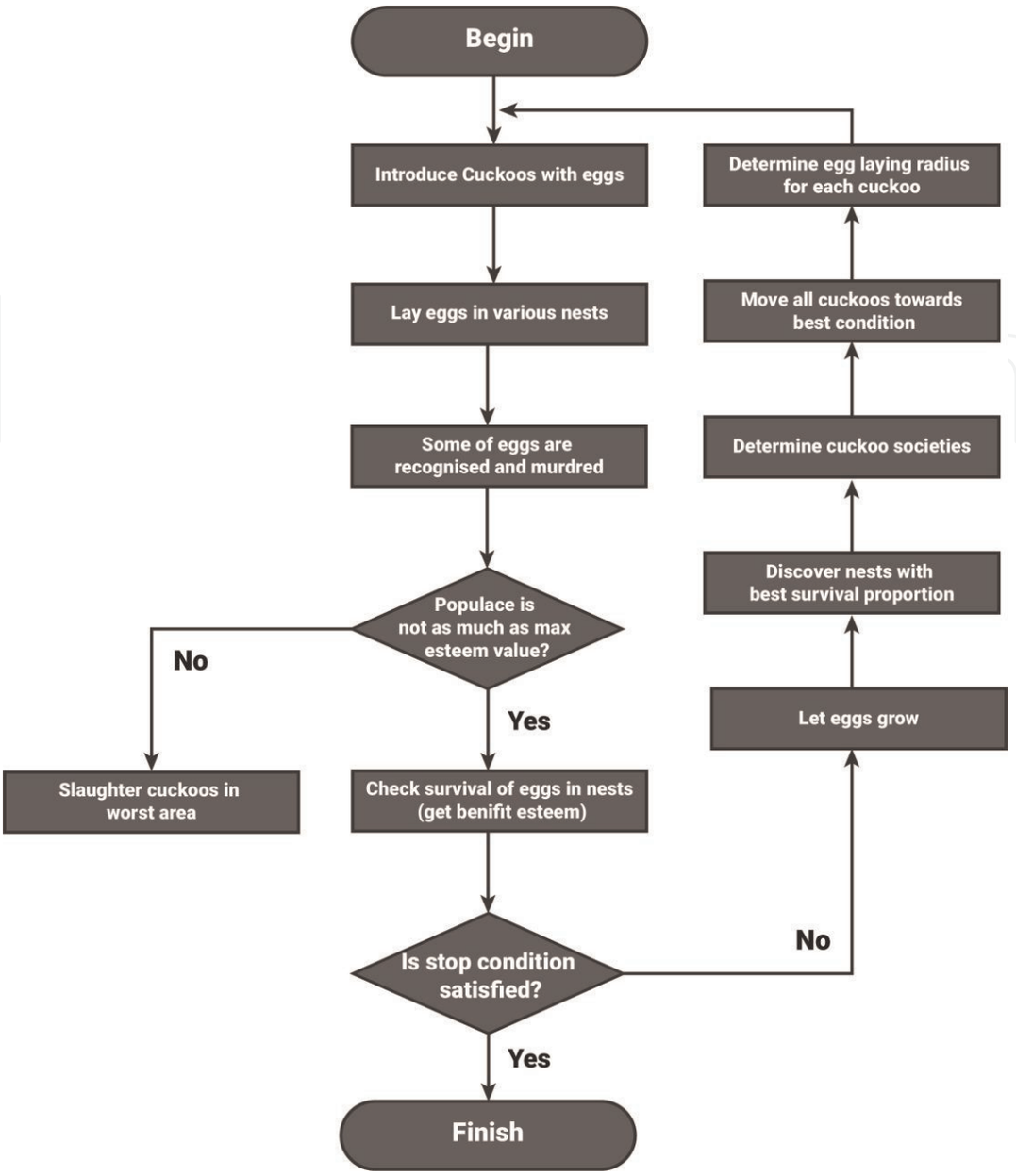

Figure 7.
Cuckoo search.

2.5 Cuckoo search

"Cuckoo search algorithm (CSA)" is another algorithm and is inspired by the raising behavior of the cuckoo bird they select their home by self-assertively accepting control over the home of some extraordinary winged animals for an age. They lay their eggs in the picked home of host cuckoo bird and drop the host winged animal's egg. The host bird either drop cuckoo fledgling's egg or surrender the whole home Some female cuckoo can copy their eggs like host fowl's egg and lay their eggs just before the laying of host feathered creature's egg. This grows the probability of their chick survival. Each egg in settle speaks to one arrangement, and the cuckoo flying creature's egg speaks to another arrangement. Wellness for every arrangement is handled, and settle with the high bore of eggs addresses the best game plan. The methodology is continued with aside from if a worldwide ideal arrangement is refined [39] (**Figure 7**).

To solve the problem of clusterin load balancer cuckoo search algorithm can be used. It does so by finding out the optimal resource allocation. Initially, we model

the given cluster into a graph and then find the optimal resource allocation to this graph. Global solution to the given problem can be found using this technique. It essentially mimics the behavior of cuckoos and their behavior in laying eggs [40].

a. The algorithm is roughly based on these ideas.

b. How cuckoos lay their eggs in host nests.

c. How the eggs are hatched by hosts, if not detected and destroyed.

d. How can we mimic this behavior to make an algorithm.

This algorithm can be divided into five different steps:
Generating initial population:
We are generating host nests for k nests.

$$(l1, b1), (l2, b2), \ - - - - - - - - - - (ln, bn)$$

These are optimal parameters

(1) Lay the cuckoo eggs (lk', bk') in the n nest.

n nest is randomly selected. Both have similar structure.

$$Lk' = lk + \text{Randomwalk}(\text{Levy flight})lk$$
$$bk' = bk + \text{Randomwalk}(\text{Levy flight})bk$$

(2) Compare the fitness of both eggs

The fitness can be found out by any statistical technique.

(3) Now replace the eggs according to the fitness value.

(4) If the host bird notices then leave that nest and search for other nest (to avoid local optimization).

(5) Repeat step 2–5 until we satisfy termination condition.

2.6 Firefly optimization

"Firefly algorithm (FA)" is the most heuristic algorithm for worldwide improvement, which is enlivened by the blazing behavior of firefly creepy crawlies. Xin-She Yang proposed this algorithm in 2008. The basic role of an (FA) is to go about as a flag framework to draw in different fireflies. Xin-She Yang defined this firefly algorithm by accepting that whole firefly are epicene with the goal that any single firefly will be pulled in to every other firefly. Engaging quality is relative to their shine, and for any two fireflies, the less brilliant one will be pulled in by the brighter one. In any case, the power diminishes as their aggregate length raises. If near are no fireflies luminous than an accustomed firefly, it will move randomly. The illumination should be related to the objective function [41] (**Figure 8**).

A load balancer is a device which is used to solve cluster problems in resource allocation. To solve the optimal resource allocation problem in the cluster we use

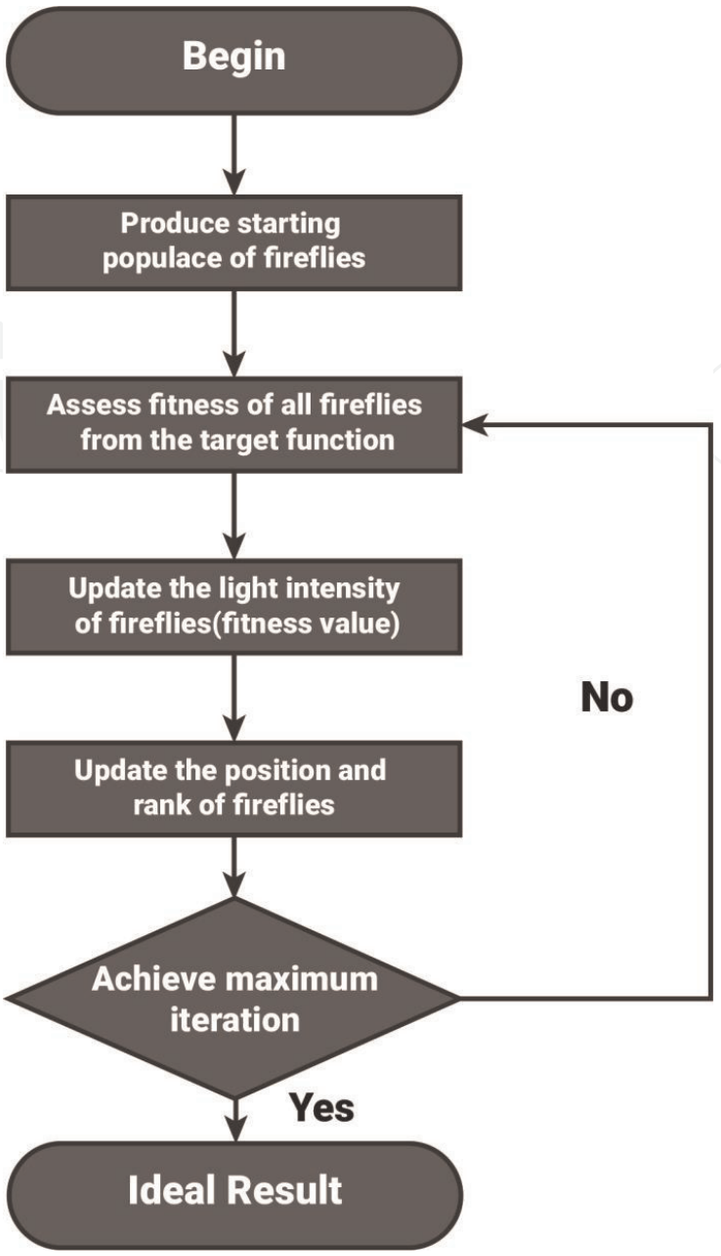

Figure 8.
Firefly optimization.

different kinds of algorithms. Here we are proposing this algorithm to solve the cluster of resource allocation in the load balancer. To solve the optimal resource allocation problem in the load balancer we use firefly optimization. Firefly optimization is unique in its approach as it tends to lead toward an optimal global solution. Essentially it mimics the behavior of fireflies, more explicitly flashing patterns of fireflies. Initially, we model the given problem into the graph, and we implement the firefly algorithm on this graph to find the optimal resource allocation [42].

1. Uses of flashing patterns like: communication, attracting prey, warning mechanism (female flies react toward the male's unique pattern of flashing in the same species).

2. Rules for the algorithm:
   Rule 1: fireflies are unisex.

Rule 2: attraction increases as the brightness of light increases and decreases as the distance increases.
Rule 3: the brightness of fireflies is generally determined by the objective function.

3. The principle of the algorithm:

This algorithm can be divided into six different steps,
(a) Initializing the objective function:

$$I = I\_0\, e^{(-kd^2)}$$

Here k is the absorption coefficient and d is the distance.
As we know,

$$I(d) = I(s)/d^2$$

(b) Generating initial firefly population: we use this equation to initialize the firefly population

$$Mt + 1 = Mt + Bo * \left(e^{-}k\left(d^2\right)\right) + a * e$$

second term attraction third term randomization.
(c) Determine the intensity of light: find brightness for every firefly using objective function equation.
(d) Calculate the attractiveness of Firefly:

$$B = B0 * e^{(-kd^2)}$$

(e) Move the less bright fireflies to bright ones.
(f) Rank the flies and find the current best, Update the intensities.

2.7 Simulated annealing

For approximating the worldwide ideal of an inclined function, "simulated annealing (SA)" is a valuable method. It is often worn when the pursuit space is detached. For issues where revelation a close global optimum is broader than completing up an obvious local optimum in a shot volume of age, simulated annealing may be attractive over decisions, for example, gradient descent [43].

Simulated annealing is a strategy for approximating the worldwide ideal of a given function. It is utilized for global enhancement in expansive search space. It is utilized when search space is discrete. Annealing technology in metallurgy rouses it. A procedure including both warming and simultaneously cooling to increase the size and to decrease abandons in the given material. This procedure is utilized to locate a right arrangement. A load balancer is an apparatus which is utilized to find the optimal asset allotment of a given cluster issue in the cloud, and we can utilize a few sorts of metaheuristics to locate an optimal solution here we utilize recreated toughening to locate the optimal solution for the given issue. This solution begins by modeling graph of the given cluster, and we have to apply this strategy on the chart which will correspond to the optimal resource allocation solution in the cluster [44].

Initial slow cooling can be explored as the probability to accept the worst solution when this solution is running iteratively we get the best possible solution. After generating each solution the algorithm checks for its fitness and compares with previous one and picks the best one [45] (**Figure 9**).

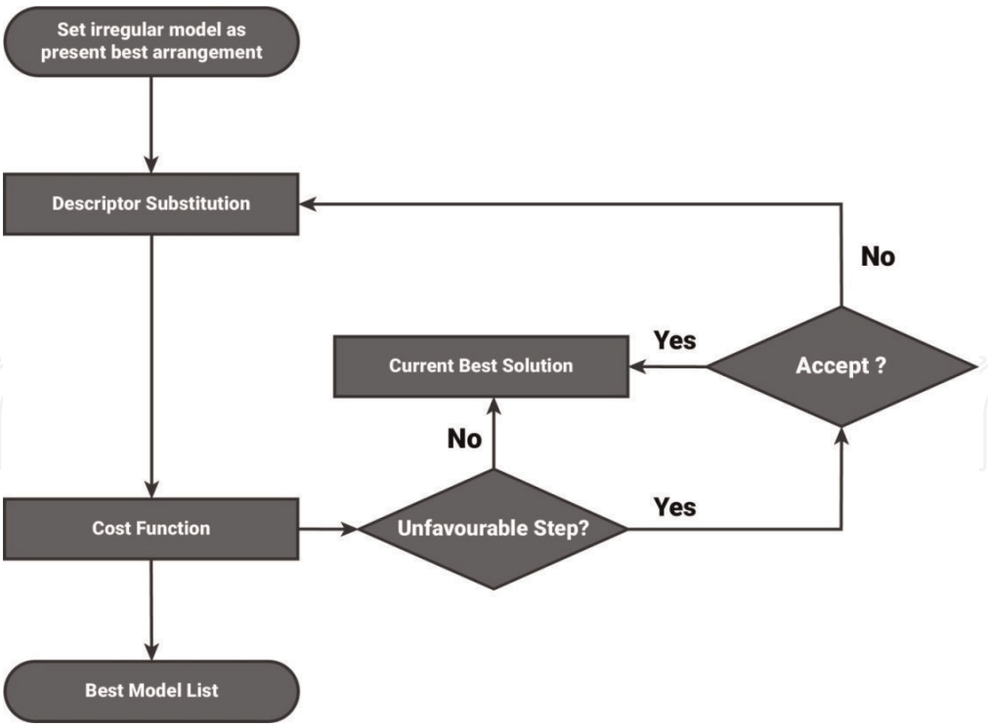

Figure 9.
Simulated annealing.

A process to find the best solution:

1. Selecting parameters: initially, we need to specify the following parameters in order to find an optimal solution:

    1. The space state.

    2. The energy goal function. E()

    3. The candidate generator neighbor()

    4. The acceptance probability p()

    5. The annealing scheduling temperature()

    6. Initial temperature()

2. Transition probability

3. Acceptance probability

4. Efficient candidate generation

5. Avoiding barrier

6. Cooling procedure

Each step can be mapped to the original simulated annealing process where we perform all these steps to get metal without defects here we use a similar procedure to find out the optimal solution.

Here we consider graph as metal and perform these tasks to find out the optimal solution.

2.8 Shuffled frog leaping algorithm

Load balancing using improved shuffled frog leaping algorithm. The "shuffled frog leaping algorithm (SFLA)" is a populace based algorithm excited by regular ridiculous. The virtual frogs go about as hosts or transporters of images where an image is a unit of social headway. The algorithm plays out an independent local search in each memeplex at the same time [46]. The local search is finished utilizing a particle swarm enhancement like technique adjusted for discrete issues, however, emphasizing a local search. To guarantee global investigation, the virtual frogs are intermittently rearranged and redesigned into new memeplexes in an approach indistinguishable from that well used in the rearranged complex evolution algorithm [47].

Load balancing technique essentially requires optimization technique to solve the given cluster problem of resource allocation. Any number of techniques can do this essentially we are using improved shuffled frog leaping algorithm which is also a part of swarm intelligence. To find the optimal solution to the given problem this algorithm imitates the behavior of frogs leaping and used this procedure. For load balancing with this technique, we require graph modeling of the cluster we get from

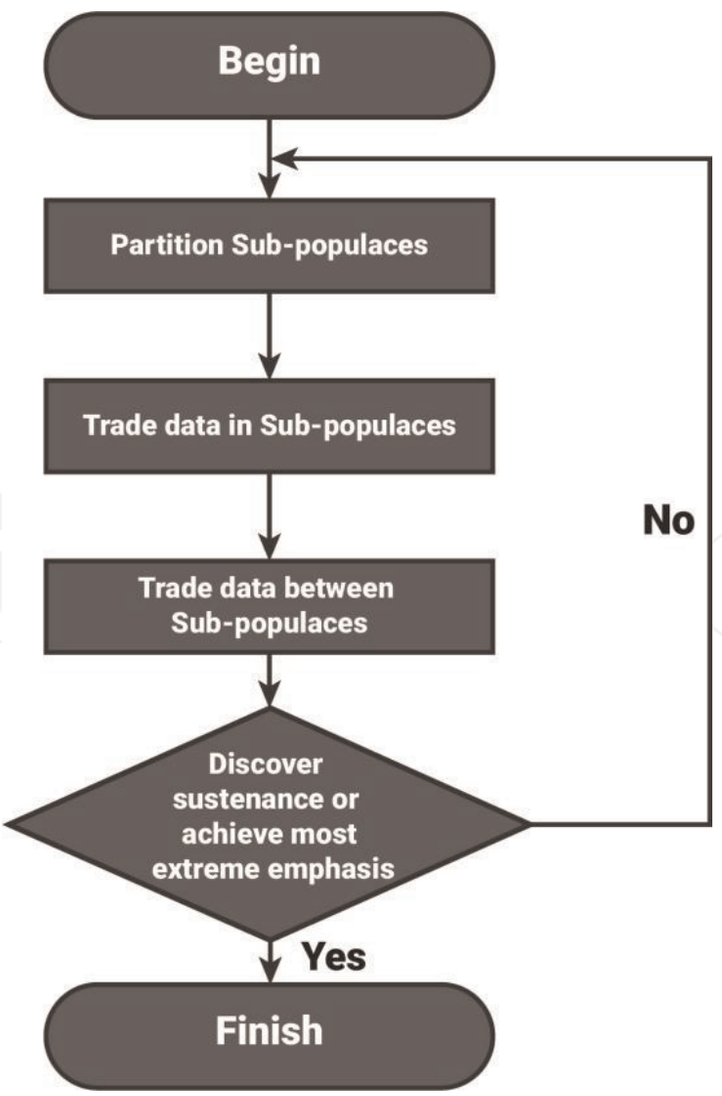

Figure 10.
Shuffled frog leaping algorithm.

the load balancer and then use this algorithm to find the optimal resource allocation. Every node in this algorithm consists of the capacity of VM, probability, etc. [48]. After graph modeling is done we continue with the stage of implementation using a frog leaping algorithm (**Figure 10**).

Essentially this algorithm can be classified into five phases:

1. Initialization phase: firstly, we need to declare the population size (k), no of subarrays (M), no of iterations in local exploration (u), no of algorithmic iterations (I), after that we need to generate no of frogs (k) randomly.

2. Fitness phase: firstly, we need to check the fitness of every frog using fitness functions.

3. Fitness functions: fitness = (1-(avg(load)/(avg(load)-least lode) + ((no of underloaded and overloaded nodes)/total no of nodes). Then after calculating the fitness of every frog, we need to sort them based on descending order of fitness values.

4. Formation of sub array and subarray: initially partitioning the sorted fitness array into subarrays as declared initially and now partitioning each subarray into another sub-array.

5. Local search phase: now perform a local search for every sub-array here is where this algorithm mimics frog leaping and perform search accordingly.

6. Convergence checking: now check for converging if convergence is satisfied this is optimal solution else repeat from frog fitness phase.

7. Off-spring generation phase: now perform this phase using fitness algorithm. Compare frogs and exchange information between sb, sw and continue this divide and conquer approach for u times, Now replace this final output by sb value.

The fitness for new product generated by the local search is calculated and this is updated solutions and this process continues until termination is given.

1. Termination phase:

    a. Number of generations.

    b. The fitness of the optimal solution.

    c. Time took.

2.9 Bat algorithm

The "bat algorithm (BA)" is the most eristic algorithm for global improvement. It was propelled by the allotment conduct of microbats, with fluctuating heartbeat rates of outflow and din. Xin-She Yang built up this algorithm in 2010. Each virtual bat flies heedlessly with a speed at a circumstance with a shifting recurrence or wavelength and tumult as it ventures and finds its prey, it changes recurrence, din and heartbeat surge rate. A local random walk escalates seek. Decision of the best continues to the point that specific stop criteria are met [49].

To solve the resource allocation problem in the given cluster a load balancer is an instrument which is utilized. We have to locate the ideal asset designation to take care of the issue of asset portion. The information for ideal asset portion is normally present in the cluster. We have to utilize any methods to locate the ideal answer for the given asset allotment issue. Here to locate the ideal answer for the given issue in the cluster we utilize the bat algorithm which impersonates the conduct of the bat. We need to use graph modeling of the given cluster to find the optimal solution initially. Essentially every node of cluster contains information about resource allocation, and when we use optimization on that graph, we find the optimal solution to the task which turns out to be optimal resource allocation for the given problem in the cluster. Here we are using Bat algorithm which is kind of swarm intelligence as it mimics the behavior of any flock of animals here we observe that we are mimicking the bat to find optimal solution hence bat algorithm [50].

Xin-She Yang proposed bat algorithm in the year 2010. It mimics the echolocation phenomenon seen in the bats. It uses sonar effect to navigate and communicate with other bats. Sonar is a form of ultrasonic sound only a few creatures like bats can understand and navigate through this. They calculate time delay to navigate, and the time delay is between emission and reflection. While reflecting bats use two notions zero means no emission one means maximum emission (**Figure 11**).

Ground Rules for this algorithm:

a. Bats can sense all kind of obstacles by this sound.

b. They fly randomly.

c. Loudness is always positive

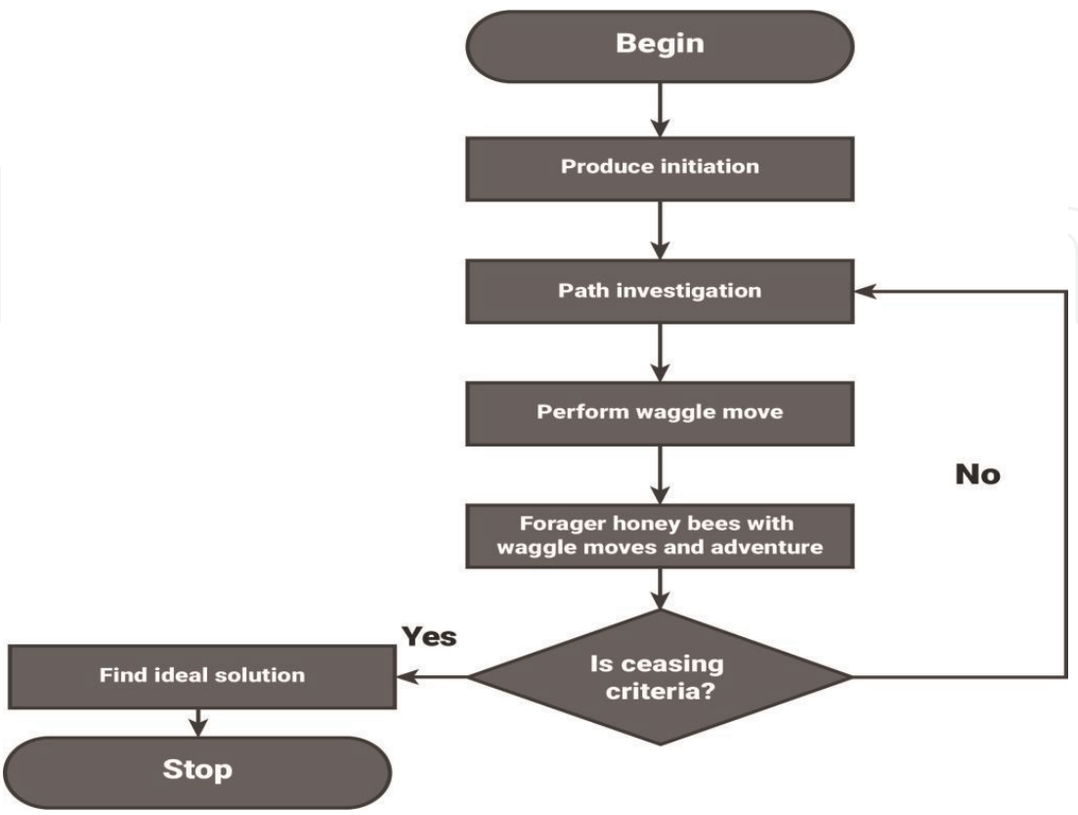

Figure 11.
Bat algorithm.

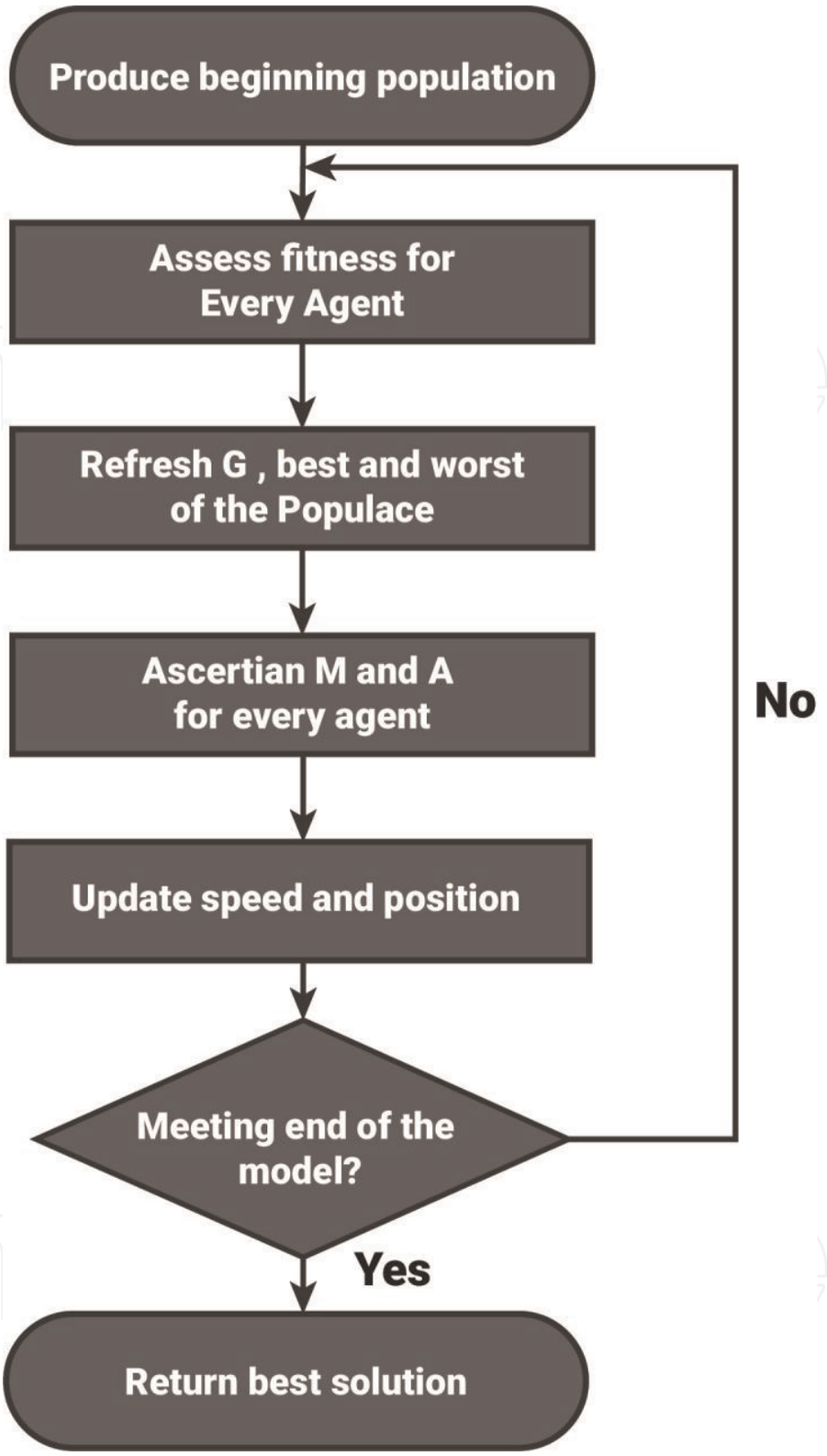

Figure 12.
Gravitational search algorithm.

Mathematical equations:

$$\mathrm{Fi} = \mathrm{f0} + (\mathrm{f1} - \mathrm{f0}) * \mathrm{K}$$

$$\mathrm{Vit} = \mathrm{vit} - 1 + (\mathrm{xit} - \mathrm{x}*)\mathrm{fi}$$

$$\mathrm{Xit} = \mathrm{xit} - 1 + \mathrm{vit}$$

K is random value between 0 and 1.
x^* is current best solution.

Algorithm	Advantages	Disadvantages	Utilization	Adaptive
1. Hill Climbing	1. Better optimization technique rather than FIFO, DFS, etc. 2. Can be used with less computational resources. 3. Takes lesser time. 4. Can be used in Dynamic allocation.	1. Co-ordinate descent cannot be used as it takes huge amount of time to determine optimal solution. Stochastic hill climbing cannot be used as it does not find the exact optimal solution. Is less efficient when compared to other iterative algorithms like ant colony based, Genetic algorithm.	1. Can be used to find out initial optimal solution and can be put in iterative optimal solutions to find best solution.	1. Yes, because there are two different variants of techniques with respect to the given problem.
1. Ant colony Optimization	1. Best optimal solution can be found out for both static and dynamic problems. Takes relatively less time than traditional algorithms. As it has many variants like recursive one it can be used based on context.	1. Best optimal solution can be found but is relatively less effective than genetic and such kind of algorithms.	1. This can be used to both find initial solution and better the solution by using iterative variant.	1. Yes, it is adaptable as it has many variants
1. Artificial bee colony:	1. Can solve any kind of optimization problem in this case can solve any kind of resource allocation. Simple, Flexible and robust. Ability to explore local solutions. Ease of implementation.	1. The solution we get is optimal but not perfect solution. Not adaptable because every problem cannot be modeled into a graph. Cannot tackle dynamic allocation. Takes up more amount of time. Uses up more computational resources.	1. Static environment, Initial solution	1. No, because not every problem can be modeled as graph.
1. Genetic Algorithm	1. Takes up less amount of time. More desirable than all the traditional algorithms in terms of both time taken and forming output.	1. Needs more computational assets to produce the ideal answer for the issue. The algorithm irrespective of its accuracy can find difficulty to find global maximum and sometimes struck at local maxima. The termination is sometimes unclear that is optimal solution is always comparative. Cannot handle dynamic allocation.	1. Generally used for static problem and can be used for iterative solution making.	1. Yes, this can be used on any kind of problem cluster.

Algorithm	Advantages	Disadvantages	Utilization	Adaptive
1. Cuckoo Search	1. Deals with multi criteria optimization problem. Easy to implement. Simple to understand. Aims to speed up convergence.	1. Cannot tackle dynamic resource allocation.	1. Similar to ABC it can be utilized if we compromise on the quality of solution.	1. It is not adaptive as it does not tackle dynamic resource allocation.
1. Firefly optimization:	1. It can deal with highly non-linear problems. 2. It does not use velocities. 3. Does not require good initial start for optimization.	1. Global searching. 2. Slow converging speed. 3. High possibility to get trapped in the local optimum.	1. It can be used to find the accurate solution.	1. Not adaptive as it has slow converging speed.
1. Simulated annealing:	1. It can deal with inconsistent and noisy data. 2. To approach global optimum. 3. It is versatile as it does not rely on restrictive property of model.	1. Lot of choices are required to make it into actual algorithm. 2. It takes lot of computation time	1. It can be used to find accurate solution as it approaches to find the global solution.	1. Not adaptable as it takes lot of computational time.
1. Shuffled Frog Leaping Algorithm	1. Need not to model the given cluster as a graph. 2. Studies show that this algorithm is directing toward global optimal solution. 3. Optimal solution is found out in iterative manner.	1. Takes up more computational resources. 2. Takes up more amount of time.	1. This can be used for both static allocation problem and dynamic allocation problem.	1. Yes, as it does not always require graph modeling.
1. Bat Algorithm	1. Automatic zooming Parameter control Frequency tuning	1. Limited accuracy Unable to predict best solution.	1. Can be used to find the immediate solution.	1. Not adaptive as it has limited accuracy.
1. Gravitational search algorithm:	1. Less execution time. 2. Less computational resource. 3. More optimal solution.	1. This algorithm cannot be used on its own but can be used as support algorithm in hybrid functionality or for calculating fitness in other algorithms.	1. Utilization: for static allocation	1. No, it is not adaptive as it requires graph modeling.

Table 1.
Comparison of various algorithms.

$$\text{Xnew} = \text{xold} + \text{H} * \text{at}$$

H is a random value between −1 and 1 (Loudness decreases as the bat have found prey).

2.10 Gravitational search algorithm (GSA)

"Gravitational search algorithm (GSA)" is a reality discovering algorithm which is picking up enthusiasm among scientific community these days. Gravitational Search Algorithm (GSA) is a populace search algorithm based on Newton's law of gravity and the law of movement. GSA is represented to be more instinctual. In GSA, the specialist has four parameters which are the position, inertial mass, unique gravitational mass, and uninvolved gravitational mass. The arrangements in the GSA masses are called specialists, and they speak with each other through the gravity compel. Its mass measures the execution of these agents. Each agent is considered an object and all object move toward different items with all the more extensive mass because of gravity compel [17] (**Figure 12**).

To solve the cluster problem of resource allocation Load balancing is the technique used, and several techniques can be used to solve this clustering problem. One such way is using gravitational search algorithm. The gravitational search algorithm uses evolutionary computing [51].

Evolutionary computing is more efficient than traditional algorithms because the solutions do not stagnate in local minima and are faster and robust when compared to other different algorithms.

It is memoryless takes up less computational time than other algorithms. The gravitational search algorithm is a likeness to the particle swarm algorithm. Gravitational search algorithm was proposed by Rashedi et al. in the year 2009. This metaheuristic comes in the category of computational intelligence. Initially, a cluster from the load balancer is taken as an input, and this is modeled into a graph. To find out the optimal resource allocation now Gravitational search algorithm is applied to the graph. This algorithm is inspired from Newtonian mechanics and specifically from Newton's second law and law of gravitation [52].

3. Advantages and disadvantages of various algorithms

Advantages, disadvantages, utilization and adaptivity of various nature-inspired algorithms are listed in **Table 1**.

4. Conclusion

In this paper we chew over different basic concepts of cloud computing, the primary focus has been kept to understand load balancer and how it functions to do task allocation and different traditional algorithms which help in task allocation have also been discussed, and nature-inspired algorithms were discussed in specific. These were classified into different types like swarm based, genetically inspired, physics-based and different algorithms from each category were explained, and each was explained with different advantages, limitations, etc., and all were compared with respect to their positive points and working methodology and with relevance to load balancer An attempt has been made to survey out the different algorithms present in nature and provide relevant solution for optimal resource allocation for load balancer in cloud. All these different algorithms are nature-inspired and are used for global

optimization. These algorithms are useful in finding optimized solutions for our lives by applying various laws and identifying the patterns of bats and flies like in Bat Algorithm, a search is identified by their local random walk and Firefly Algorithm.

5. Future scope

Different optimization techniques can be used as a hybrid to suit the appropriate usage to overcome the disadvantages of one over the other. New optimization techniques can be formulated from nature.

Author details

Surya Teja Marella* and Thummuru Gunasekhar
Department of Computer Science Engineering, Koneru Lakshmaiah Educational Foundation Vijayawada, Guntur, India

*Address all correspondence to: suryatejamarella@gmail.com

References

[1] Baran ME, Wu FF. Network reconfiguration in distribution systems for loss reduction and load balancing. IEEE Transactions on Power Delivery. 1989;**4**(2):1401-1407

[2] Randles M, Lamb D, Taleb-Bendiab A. A comparative study into distributed load balancing algorithms for cloud computing. In: 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA); IEEE; April 2010. pp. 551-556

[3] Al Nuaimi K, Mohamed N, Al Nuaimi M, Al-Jaroodi J. A survey of load balancing in cloud computing: Challenges and algorithms. In: 2012 Second Symposium on Network Cloud Computing and Applications (NCCA); IEEE; December 2012. pp. 137-142

[4] Velayos H, Aleo V, Karlsson G. Load balancing in overlapping wireless LAN cells. In: 2004 IEEE International Conference on Communications; IEEE; June 2004; Vol. 7. pp. 3833-3836

[5] Tantawi AN, Towsley D. Optimal static load balancing in distributed computer systems. Journal of the ACM (JACM). 1985;**32**(2):445-465

[6] Yousaf FZ, Taleb T. Fine-grained resource-aware virtual network function management for 5G carrier cloud. IEEE Network. 2016;**30**(2): 110-115

[7] Cybenko G. Dynamic load balancing for distributed memory multiprocessors. Journal of Parallel and Distributed Computing. 1989;**7**(2):279-301

[8] Performance Tradeoffs in Static and Dynamic Load Balancing Strategies; NASA; March 1986

[9] Shirazi BA, Kavi KM, Hurson AR. Scheduling and Load Balancing in Parallel and Distributed Systems. IEEE Computer Society Press; 1995

[10] Cardellini V, Colajanni M, Yu PS. Dynamic load balancing on web-server systems. IEEE Internet Computing. 1999;**3**(3):28-39

[11] Schoonderwoerd R, Holland OE, Bruten JL, Rothkrantz LJ. Ant-based load balancing in telecommunications networks. Adaptive Behavior. 1997;**5**(2): 169-207

[12] Brendel J, Kring CJ, Liu Z, Marino CC. Resonate Inc. World-wide-web server with delayed resource-binding for resource-based load balancing on a distributed resource multi-node network. U.S. Patent 5, 774, 660. 1998

[13] Willebeek-LeMair MH, Reeves AP. Strategies for dynamic load balancing on highly parallel computers. IEEE Transactions on Parallel and Distributed Systems. 1993;**4**(9):979-993

[14] Miyazaki T, Wada M, Kawahara H, Sato M, Baba H, Shimada S. Dynamic load at baseline can predict radiographic disease progression in medial compartment knee osteoarthritis. Annals of the Rheumatic Diseases. 2002; **61**(7):617-622

[15] Devine KD, Boman EG, Heaphy RT, Hendrickson BA, Teresco JD, Faik J, et al. New challenges in dynamic load balancing. Applied Numerical Mathematics. 2005;**52**(2-3):133-152

[16] Kim C, Kameda H. An algorithm for optimal static load balancing in distributed computer systems. IEEE Transactions on Computers. 1992;**41**(3):381-384

[17] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: A gravitational search algorithm. Information Sciences. 2009; **179**(13):2232-2248

[18] Zaki MJ, Li W, Parthasarathy S. Customized dynamic load balancing for a network of workstations. In: 1996, Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing; IEEE; August 1996. pp. 282-291

[19] Raz Y, Scherr AL, EMC Corp. Dynamic load balancing. U.S. Patent 5, 860, 137. 1999

[20] Trigui H, Cuthill R, Kusyk RG. Reverb Networks. Dynamic load balancing. U.S. Patent 8, 498, 207. 2013

[21] Raz Y, Vishlitzky N, Alterescu B, EMC Corp. Dynamic load balancing. U. S. Patent 6, 173, 306. 2001

[22] Hendrickson B, Devine K. Dynamic load balancing in computational mechanics. Computer Methods in Applied Mechanics and Engineering. 2000;**184**(2-4):485-500

[23] Sharma S, Singh S, Sharma M. Performance analysis of load balancing algorithms. World Academy of Science, Engineering and Technology. 2008; **38**(3):269-272

[24] Boyan JA, Littman ML. Packet routing in dynamically changing networks: A reinforcement learning approach. In: Advances in Neural Information Processing Systems; 1994. pp. 671-678

[25] Yang XS. Nature-Inspired Metaheuristic Algorithms. Luniver press; 2010

[26] Fister I Jr, Yang XS, Fister I, Brest J, Fister D. A brief review of nature-inspired algorithms for optimization. 2013. arXiv preprint arXiv:1307.4186

[27] Zang H, Zhang S, Hapeshi K. A review of nature-inspired algorithms. Journal of Bionic Engineering. 2010; **7**(4):S232-S237

[28] Mitchell M, Holland JH, Forrest S. When will a genetic algorithm outperform hill climbing. In: Advances in Neural Information Processing Systems; 1994. pp. 51-58

[29] Tsamardinos I, Brown LE, Aliferis CF. The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning. 2006;**65**(1):31-78

[30] Al Salami NM. Ant colony optimization algorithm. UbiCC Journal. 2009;**4**(3):823-826

[31] Dorigo M. Ant colony optimization. Scholarpedia. 2007;**2**(3):1461

[32] Yaseen SG, Al-Slamy NM. Ant colony optimization. IJCSNS. 2008;**8**(6):351

[33] Karaboga D. Artificial bee colony algorithm. scholarpedia. 2010;**5**(3):6915

[34] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation. 2009;**214**(1):108-132

[35] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization. 2007;**39**(3): 459-471

[36] Deb K, Pratap A, Agarwal S, Meyarivan TAMT. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation. 2002;**6**(2): 182-197

[37] Morris GM, Goodsell DS, Halliday RS, Huey R, Hart WE, Belew RK, et al. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. Journal of Computational Chemistry. 1998;**19**(14): 1639-1662

[38] Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist

non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: International Conference on Parallel Problem Solving From Nature; Berlin, Heidelberg: Springer; September 2000. pp. 849-858

[39] Gandomi AH, Yang XS, Alavi AH. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. Engineering with Computers. 2013; **29**(1):17-35

[40] Yildiz AR. Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. The International Journal of Advanced Manufacturing Technology. 2013;**64** (1-4):55-61

[41] Yang XS. Firefly algorithm, stochastic test functions and design optimisation. International Journal of Bio-Inspired Computation. 2010;**2**(2): 78-84

[42] Yang XS. Firefly algorithm, levy flights and global optimization. In: Research and Development in Intelligent Systems XXVI. London: Springer; 2010. pp. 209-218

[43] Corana A, Marchesi M, Martini C, Ridella S. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm corrigenda for this article is available here. ACM Transactions on Mathematical Software (TOMS). 1987; **13**(3):262-280

[44] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science. 1983;**220**(4598): 671-680

[45] Szu H, Hartley R. Fast simulated annealing. Physics Letters A. 1987;**122** (3-4):157-162

[46] Eusuff MM, Lansey KE. Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources Planning and Management. 2003;**129**(3):210-225

[47] Eusuff M, Lansey K, Pasha F. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. Engineering Optimization. 2006;**38**(2):129-154

[48] Rahimi-Vahed A, Mirzaei AH. A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. Computers & Industrial Engineering. 2007;**53**(4):642-666

[49] Yang XS. A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Berlin, Heidelberg: Springer; 2010. pp. 65-74

[50] Yang XS, Hossein Gandomi A. Bat algorithm: A novel approach for global engineering optimization. Engineering Computations. 2012;**29**(5):464-483

[51] Rashedi E, Nezamabadi-Pour H, Saryazdi S. BGSA: Binary gravitational search algorithm. Natural Computing. 2010;**9**(3):727-745

[52] Rashedi E, Nezamabadi-Pour H, Saryazdi S. Filter modeling using gravitational search algorithm. Engineering Applications of Artificial Intelligence. 2011;**24**(1):117-122