electronics

MDPI

Review

A Survey of Swarm Intelligence Based Load Balancing Techniques in Cloud Computing Environment

M. A. Elmagzoub [1], Darakhshan Syed [2], Asadullah Shaikh [1,*], Noman Islam [2], Abdullah Alghamdi [1] and Syed Rizwan [2]

[1] College of Computer Science and Information Systems, Najran University, Najran 61441, Saudi Arabia; meabdullah@nu.edu.sa (M.A.E.); aalghamdi@nu.edu.sa (A.A.)
[2] Computer Science Department, Iqra University, Karachi 75500, Pakistan; darakhshan@iqra.edu.pk (D.S.); noman.islam@iuk.edu.pk (N.I.); syed.rizwan@iqra.edu.pk (S.R.)
* Correspondence: asshaikh@nu.edu.sa

Abstract: Cloud computing offers flexible, interactive, and observable access to shared resources on the Internet. It frees users from the requirements of managing computing on their hardware. It enables users to not only store their data and computing over the internet but also can access it whenever and wherever it is required. The frequent use of smart devices has helped cloud computing to realize the need for its rapid growth. As more users are adapting to the cloud environment, the focus has been placed on load balancing. Load balancing allocates tasks or resources to different devices. In cloud computing, and load balancing has played a major role in the efficient usage of resources for the highest performance. This requirement results in the development of algorithms that can optimally assign resources while managing load and improving quality of service (QoS). This paper provides a survey of load balancing algorithms inspired by swarm intelligence (SI). The algorithms considered in the discussion are Genetic Algorithm, BAT Algorithm, Ant Colony, Grey Wolf, Artificial Bee Colony, Particle Swarm, Whale, Social Spider, Dragonfly, and Raven roosting Optimization. An analysis of the main objectives, area of applications, and targeted issues of each algorithm (with advancements) is presented. In addition, performance analysis has been performed based on average response time, data center processing time, and other quality parameters.

Keywords: cloud computing; load balancing; swarm intelligence algorithms; comparative study

Citation: Elmagzoub, M.A.; Syed, D.; Shaikh, A.; Islam, N.; Alghamdi, A.; Rizwan, S. A Survey of Swarm Intelligence Based Load Balancing Techniques in Cloud Computing Environment. *Electronics* **2021**, *10*, 2718. https://doi.org/10.3390/electronics10212718

Academic Editor: Filipe Araujo

Received: 15 September 2021
Accepted: 2 November 2021
Published: 8 November 2021

1. Introduction

Cloud computing is a metaphor for the internet that provides computing as a utility to end-users. It is referred to as the systematic storage, computing, and access of data through the internet rather than one's hardware or office network. It is a computational paradigm providing various resources, software platforms, etc., hosted at vast data centers to users in order to enhance their productivity [1]. Providers of cloud computing deliver their "services" according to various models. Infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS) are the three common models [2]. These models provide growing abstraction; thus, they are mostly depicted as layers in a pyramid i.e., infrastructure, framework, and software-as-a-service. Several cloud commercial services providers have achieved users' trust despite the growing concern about relying on a third-party need for the computational purpose by end-users. The obvious reason is their user-friendliness, accessibility, and secure environment [3]. This includes Microsoft Azure, Google Cloud, and Amazon Web services, etc. In the past few years, some other academic and commercial service providers, viz., Jetstream, Helix Nebula, Open Science Data Cloud, etc., have achieved fame by introducing their services at lower rates with higher rates of proficiency [4]. As the creation and usage of big data continually increase, cloud computing has emerged in a significant role, grasping users' attention and promoting

several financial activities over it. It has been engaging many users who use its resources who pay a respective amount as per the required period [5].

Load balancing allows the facility to assign the workload such as the resources accessible. Its goal is to provide continuous service by providing and divesting the application instances along with proper use of resources in the event of failure of any service portion [6]. The goal of load balancing is to reduce the response time for operations and to maximize the usage of resources, which increases device efficiency at lower costs [7]. At the same time, it also targets providing scalability and versatility for applications in which its size will increase in the future and require more resources, as well as to prioritize tasks that need immediate implementation compared to others. Another main requirement in order to adapt load balancing [8] is that it decreases energy consumption, avoids bottlenecks, provides support, and meets QoS criteria with respect to improving load balancing.

Even though several algorithms are available for load balancing in cloud computing such as conventional algorithms, machine learning algorithms, and heuristics algorithms. This paper specifically targets swarm intelligence-based approaches to load balancing. As compared to heuristics algorithms, the benefits of swarm intelligence-based algorithms are discussed in the next paragraph.

Normally, when historical or traditional procedures are too expensive or fail to uncover accurate solutions, a heuristic is used to solve issues more rapidly. As a result, it is often referred to as approximation algorithms. The goal of heuristic techniques is to find a solution to a specific problem in an acceptable amount of time [9]. The results may not be the best, but they can come close to being the best. With a strategic estimate, this algorithm discovers the potential results. It can provide results on its own or in combination with other optimization strategies to improve efficiency. Heuristic approaches could not produce a near-optimal result; instead, they could only produce a small number of distinct alternatives. The major drawback of heuristic approaches is that they tend to halt at low-quality local optima when looking for a solution, which resulted in the invention of SI, an iterative optimization technique [10]. SI aims to combine relatively high approximate techniques to guide local optimization strategies in order to explore a solution space successfully and efficiently.

1.1. Objectives of the Research

The main objective of this research is to offer an organized and complete review of cloud computing research on load balancing by using swarm intelligence algorithms. This study examines state-of-the-art load balancing algorithms and various hybrid strategies from 2015 through 2021. Existing techniques for load balancing are reasonably grouped. The aim is to provide a clear and precise perspective of the underlying algorithms used by each strategy with this classification, i.e., to examine the work conducted on load balancing in a cloud computing environment. The objectives of this research can be summarized as follows:

(a) To present the systematic literature review of SI based load balancing algorithms;
(b) To present the algorithms along with the targeted issues and the areas of applications;
(c) To present a comparative as well as quantitative analysis of some swarm intelligence algorithms used for balancing the load in cloud computing.

1.2. Implications of the Research

This survey, in the long run, will be very advantageous for developing SI based load balancing algorithms, other amalgamative strategies, and the latest state-of-the-art techniques in order to facilitate the cloud with as much optimization as possible. The overview presented of algorithms in this paper furnishes a complete comparison that facilitates researchers in further improving quality parameters.

The benefits simply meet the requirements of cloud customers for efficiently optimized cloud services. Makespan, task rescheduling, scalability, efficiency, fast convergence,

performance, and high reliability, etc., are all issues for which their solutions can be taken into account.

1.3. Motivation

The motivation of this survey is to provide a comprehensive overview of existing and latest state-of-the-art load balancing algorithms inspired by swarm intelligence. By utilizing this survey, the targeted motivation is to inspire inexperienced researchers in this field to support the development of more optimal load balancing algorithms. This will enable interested scholars to perform future research in this area. For seamless operation, most firms and every organization in the future are expected to adopt efficient load balancing algorithms. As a result, focusing on this specific study topic in order to build better dynamic algorithms is very essential.

The method of fairly managing the workload on virtual machines for the proper allocation of resources is load balancing in the cloud [11]. In this research survey, different variations of load-balancing algorithms based on swarm intelligence are highlighted along with different performance parameters involved. SI based load balancing algorithms help to allocate resources to tasks equally at minimal expense for resource optimization and user comfort. It ultimately encourages us to identify and work on solving complex problems.

We end this section with a note on the comparison of virtualization and container technology. Virtualization emerged in the past as a crucial technique for cloud computing [12] in which virtual machines assist users in completing activities. All of the units are self-contained, and the user retains complete ownership and control over the software that is installed and used. By optimizing the resources, VM solves several difficulties. The developers are concerned that the code works great in the development environment but fails to work in the testing or production environments due to any variations in the environments. As a result, containerization was introduced to overcome these issues. Container-based virtualization, in particular, is a lightweight method for creating a virtual environment that runs at the software level on the host computer [13]. Due to minimal resource utilization and excellent portability, it has been rapidly increasing with rotating virtual machines (VMs) [14]. Furthermore, container-based micro-services are redefining application design methodologies [15]. In many non-container contexts, load balancing is a simple process, but when it comes to containers, it necessitates certain particular considerations. However, the fundamental issues remain the same, as the container is also a lightweight virtualization technique. Containers can be classified into two types: application container that has all the related things in one package for software application (e.g., docker, kubernetes, etc.) and a system container that is based on design goals and implementation (e.g., LXC). They provide diverse functions for container orchestration, such as load balancing, monitoring, scaling, and also file storage, deployment, pushing, and many more [16].

The rest of the paper is structured as follows. Section 2 describes the background study and design of a survey. Section 3 describes the literature review. Section 4 explores load balancing in cloud computing using SI. Section 5 presents the summarized view of the discussed SI algorithms along with their area of application and target issue(s). Section 6 comprises the comparative analysis among certain discussed SI algorithms based on performance and some quality parameters. Finally, Sections 7 and 8 discuss the future directions and conclusion of SI algorithms used for load balancing in cloud computing, respectively.

2. Background and Design of Survey

The next few paragraphs define the study design and address the collection of research papers, data sources, and exploration parameters discussed in the research.

2.1. Research Questions

The key exploration questions that are investigated and ultimately help to support our survey are mentioned below:

(a) What is the need for and the importance of load balancing in cloud computing?

(b) What is the main idea of SI?
(c) How the occurring issues in Cloud Computing can be reformulated by SI?
(d) How SI support load balancing in cloud computing?
(e) What are the current and future challenges associated with load balancing in cloud computing?

By providing efficient and precise Cloud Storage and load balancing data based on the review paper under the study direction, the above questions can be answered. The findings are portrayed below.

2.1.1. What Is the Need and Importance of Load Balancing in Cloud Computing?

Cloud computing's effective and beneficial features can be realized by properly managing cloud services. The virtual nature of these cloud services is one of the most fundamental properties of the cloud environment. The cloud service provider (CSP) is a company that rents out services to users. With the available virtualized computing resources, the job of the CSP in providing services to users is a very complicated one. As a result, academics have begun paying more attention to load balancing [17]. The response of the system is improved as a result of this load balancing. By load balancing, the CSP creates a trade-off between economic advantages and user satisfaction.

The idea of load balancing in a Cloud environment [18] can be described as the procedure of dividing workloads and various computing parameters associated with certain properties in order to manage the work or applications among computers, networks, or servers demanded by the enterprises [19]. Figure 1 shows the basic mechanism of load balancing.

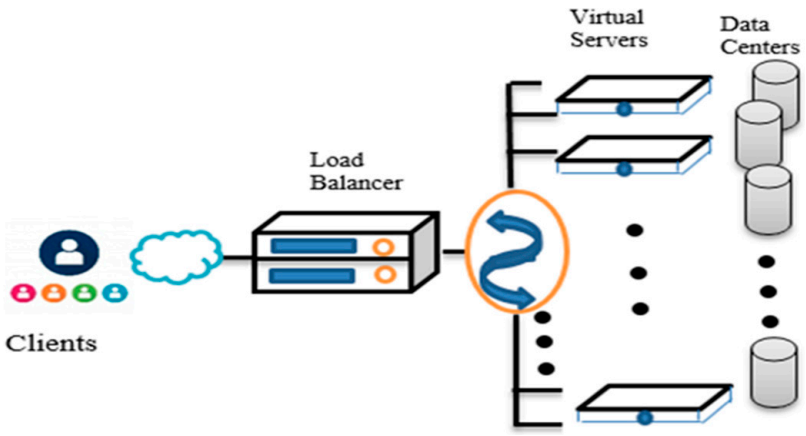

Figure 1. Mechanism of load balancing.

In other words, the cloud's load balancing includes possessing the circulation of traffic and demands that exists over the internet [20]. However, apart from managing the demand of the enterprises, the fact that the growth of traffic has been exponentially increasing cannot be ignored. Therefore, the workload on the servers is also increasing rapidly, which has been overloading the web servers unavoidably. Load balancing [19] is important for maximizing the use of cloud resources, such as processors, memory, and disks, and for achieving the purpose of good machine performance with proper resource utilization.

For those applications for which its size can increase in the future, load balancing often helps to provide scalability and versatility [6]. It also refers applications that need more time, and relative to other jobs, it gives preference to tasks that need immediate execution [21]. Other load balancing goals include reducing energy consumption, avoiding bottlenecks, providing resources, and meeting QoS criteria for improving load balancing [22]. Proper workload mapping and load balancing strategies that consider various metrics require consideration.

2.1.2. What Is the Main Idea of SI?

The metaphor of a swarm implies heterogeneity, probabilistic reasoning, irregularity, and unpredictability, but intelligence implies that the real concerned process is successful in some manner [23]. SI is the area that deals with mutual behavior and performance of decentralized and self-organized systems, whether natural or artificial. Specifically, SI systems center on the combined behaviors that result from the local interactions of the individuals with each other and with their respective environments [24]. Examples of such systems are colonies of ants and termites, schools of fish, flocks of birds, and herds of land animals [25].

Inspired from nature, especially the above-mentioned biological systems, simple agents have been designed. These agents follow simple rules without any centralized control [26]. Some computer programs that are designed to resolve issues of optimization and data processing have also been adapted by SI.

2.1.3. How Can the Occurring Issues in Cloud Computing Be Reformulated by SI?

The utilization of the concept of SI in cloud computing is an indispensable subject for researchers [20]. It is involved in the delivery of efficiently performed, optimized solutions to cloud-based services for reliable implementation, infrastructural stability, and security issues.

The vast knowledge of swarm intelligence starting from the bird's eye view to the state-of-the-art techniques that can be optimized for addressing various challenges of cloud computing. The occurring problems in the cloud environment that can be reformulated by SI advanced techniques include virtual machine allocation [27], Denial of Service assault, load balancing and optimization, deadline management, data leakage, power-aware profiling, fault tolerance, cost-effective architecture, and energy efficiency.

The use of evolutionary algorithms in the internet of things (IoT) information processing can make it considerably smoother and more efficient. Connected vehicles, IoT network architecture for forwarding large volumes of IoT data, and an Edge Computing system that optimizes IoT data analysis are all possible uses of the techniques. This would stimulate the growth of IoT networks while also improving the customer's experience of IoT solutions [28].

2.1.4. How Does SI Support Load Balancing in Cloud Computing?

As a consequence of inadequate scheduling, the challenges of services being underutilized and overutilized may develop, resulting in either cloud resource waste or service performance loss. As a result, the idea of introducing meta-heuristic techniques into job scheduling arose in order to optimally distribute complex and dynamic incoming workloads (cloudlets) across constrained resources in an acceptable amount of time [29].

SI finds its space in applications for routing and specialist task scheduling-related algorithms. These two applications of SI ultimately lead towards the most highlighted issue in cloud computing, which is "load balancing." SI renders load balancing conveniently possible by taking motivation from insects. SI follows the concept that social insects solve complicated issues together [30]. It proposed a very smart and decentralized solution in an appropriate manner, which is what is exactly required by Cloud computing in order to manage the load efficiently. Therefore, as a result, these collective, knowledgeable, and decentralized insect activities have become a model for solving the complex problem of load balancing in Cloud-based environments.

2.1.5. What Are the Current and Future Challenges Associated with Load Balancing in Cloud Computing?

Research suffers from various challenges with load balancing [31]. A list of a few problems with load balancing is addressed as shown in Figure 2.

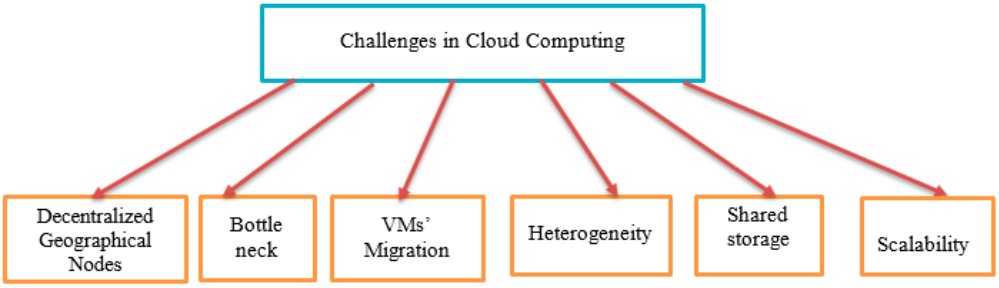

Figure 2. Challenges in cloud computing.

Decentralized Geographical Nodes: Some load balancing strategies are designed for a narrower area in which variables such as network delay, communication delay, the distance between distributed computing nodes, the distance between users and resources, etc., are not considered. As these algorithms are not sufficient for this setting, nodes located at very distant locations are a challenge [6].

Bottleneck: If the network server fails, then the overall computing system will be affected. Thus, it is a challenge to build certain distributed algorithms in which a single node does not dominate the entire platform [6].

VMs' Migration: Virtualization is another challenge associated with load balancing. The virtual machines' have various configurations and are independent in design. Some VMs need to migrate to a distant location by using a VM migration load-balancing method if a physical machine becomes overloaded [6]. It is the responsibility of the load balancer to appropriately balance load through migration so that none of the servers are overloaded.

Heterogeneity: As the software or platform changes, ultimately, the user may change their requirements. Due to the dynamically changing nature of the client, cloud computing includes executing them on heterogeneous nodes for efficient use of resources and reduced response time. It is, therefore, a challenge for researchers to devise successful load-balancing strategies for the heterogeneous world.

Shared storage: Partial replication can be adequate, but the availability of datasets can be a problem. At the same time, the sophistication of load-balancing approaches can be increased. Therefore, it is important to establish an effective load balancing strategy that considers the distribution of application and related data based on a partial replication method.

Scalability: Facilitating modifications effectively is another challenge of Cloud computing. A good load balancer should consider rapid changes in requirements. These changes can be in terms of computing capacity, storage, system topology, and so on. Dealing with all of them is an obvious challenge.

The future challenges incorporated in cloud computing include the following [32]: attacks targeting (shared-tenancy environment), the threat of VM-based malware, botnet hosting issues, service provider reliability, vendor lock-in, multilevel data privacy, etc.

To render this optimization target a reality, the cloud community must overcome various technological difficulties. Specific concerns revolve around the deployment of future infrastructure-as-a-service clouds, including the question of how to manage them reliably. It must incorporate clouds to supply flexible and dynamic provision of the services upon request, build cloud accumulation designs and technologies that allow cloud providers to interact and coexist, and improve the security, dependability, and renewable energy of cloud infrastructures.

The future of Cloud Computing is ultimately the Internet of things (IoT), and the obvious challenge is to stop it from becoming the Internet of Overwhelming things (IoOT). Dealing with complexity is another main challenge for load balancing in cloud computing as the network is tremendously increasing day by day. As the future money market is shifting towards digital currency, dealing with cryptocurrency is another highlighted future challenge associated with the Cloud environment. The findings of a real-world demand showed that by evenly splitting requests into VMs, SI techniques may boost resource utiliza-

tion and minimize response time and makespan by optimizing requests, their fluctuations, and the presence of different VMs. Future research will be conducted by generalizing SI approaches in geographical clouds with distributed datacenters. As a result, another issue that is of interest is to provide SI approaches such as autonomic multi-objective scheduling in order to optimize more competing objectives as soon as possible [33].

2.2. Benchmark of Search

In this work, a comprehensive review of balancing the load in a cloud environment focusing on SI based algorithms was carried out. Scan string terms were used as follows: cloud computing, load balancing, and swarm intelligence, SI algorithms, performance parameters, quality parameters, and challenges of load balancing in the cloud. Table 1 provides the detail of the search engines.

Table 1. Choice of search engine.

Finding Engine	Address of Mentioned Search Engine
IEEE Xplore	https://ieeexplore.ieee.org/ (accessed on 5 June 2021)
ACM	https://acm.org/ (accessed on 28 April 2021)
Academia	https://academia.edu/ (accessed on 15 April 2021)
Science Direct	https://sciencedirect.com (accessed on 10 May 2021)
Taylor and Francis	https://www.taylorandfrancis.com (accessed on 18 July 2021)
Springer	https://springer.com (accessed on 18 July 2021)

2.3. Origin of Data

A variety of different sources of data were considered for this survey. As data collection for the retrieval of associated research papers for conferences and journal research papers in websites, Google Scholar, journals, and books were primarily considered.

Figure 3 shows the percentage of different research papers studies from several sources from 2015 to 2021. These sources are IEEE Xplore, ACM digital library, Academia, Science Direct, Taylor and Francis, Springer, and different websites.

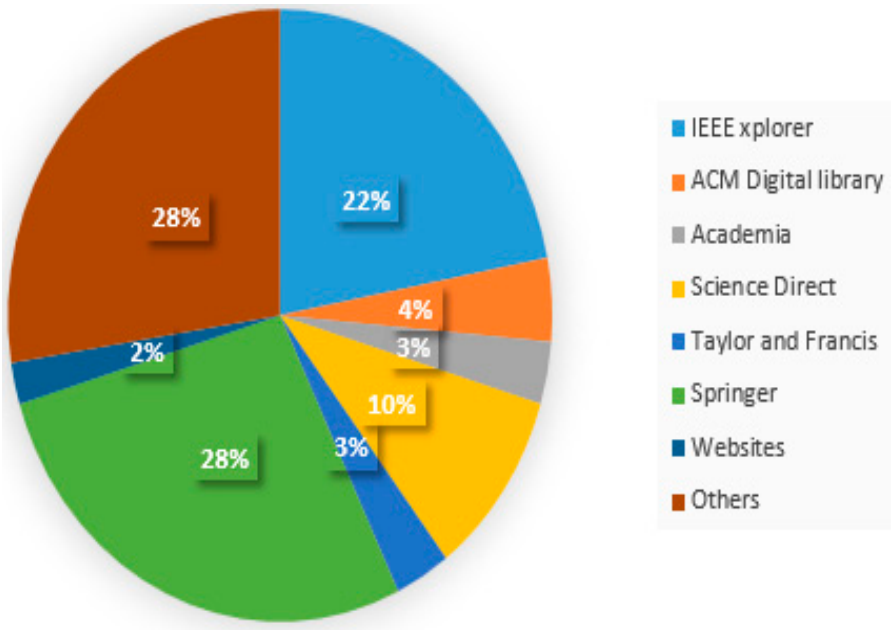

Figure 3. The percentage of a research paper read from multiple sources from 2015 to 2021.

2.4. Exploration Criteria

The study was conducted from 2015 to January 2021. The research papers that comply with quality assessment are scanned, and relevant materials are filtered accordingly. This

included research papers from Journals, Google Scholar, websites providing appropriate discussions, and books reviewed by peers. In Figure 3, the percentage of papers read between 2015 and 2021 is shown.

2.5. Quality Evaluation

The research background is one of the possible ways in which different types of basic quantitative research are conducted. A quality evaluation criteria for inclusion and exclusion of research papers are applied to the research papers searched. Any of the research papers were omitted after the initial abstract review. The main rules followed for inclusion and exclusion are shown in Table 2.

Table 2. Rules of the search inclusion and exclusion.

	Followed Rules of Research
Inclusion	• A research paper that is designed by professionals. • A complete research analysis was performed in the sense of cloud computing. • A complete research analysis was performed in the sense of load balancing. • A research study was conducted in the context of swarm intelligence algorithms that target load balancing issues. • A research paper is presented in the English language.
Exclusion	• A research study that did not focus on the load balancing issue in cloud computing. • A research paper that contains static algorithms.

3. Literature Review

In the field of cloud computing, a variety of studies have been conducted. This includes surveys on load balancing, resource scheduling, service broker regulations, resource allocation, and other general problems of cloud computing. The following section highlights the surveys conducted on load balancing in cloud computing.

Sughpal Singh and Inderveer Chana [34] surveyed several papers for studying resource allocation challenges in cloud computing. The main goal was to choose the best proficient and right algorithm from among the current resource scheduling techniques for a given workload. Their study centered on developing a wide methodological examination of the management and planning of cloud resources. They analyzed resource classification, resource planning evolution, percentages of different scheduling algorithms and related QoS parameters, an in-depth classification of resource scheduling algorithms, the distinction of resource scheduling algorithms as per distribution policies, and aspects of scalability. They suggested that before conducting rigorous resource planning studies, it is necessary to make progress in the same cloud search. They analyzed that by allocating resources based on the type of workload; the utilization of resources can be improved. Furthermore, they also suggested future directions.

MinxianXu et al. [27] provided a comprehensive and comparative overview of the available literature on load balancing techniques for virtual machines in cloud computing. The classifications throughout this research were expanded upon in previous studies by examining the various characteristics of VM load balancing elements in-depth, such as scheduling scenarios, management methodologies, resource type, VM type uniformity, and allocation dynamics. Then, the virtual machine LBA planning metrics were summarized, which may be utilized to assess load balancing effects and other scheduling goals.

AsrinVakili and NimaJafariNavimipour [35] proposed a concept for developing a new cloud service in cloud computing that incorporates existing services to save cost and time while increasing performance. Their contributions to the survey were as follows: (1) include a review of related issues in several problem domains related to the cloud service's composition; (2) outline the anatomy of certain key strategies in cloud service composition technologies, and (3) identifying significant topics for future research to improve the methods of service composition. Time, cost, scalability, optimization, and efficiency were

the five main factors that were addressed as well as for developing more effective service composition strategies; the challenges of current methods were highlighted.

ArunimaHota et al. [36] conducted a review of 29 research publications on various load balancing techniques. Based on the sort of method utilized, they divided load balancing algorithms into three categories: heuristic, metaheuristic, and hybrid. In their survey, they compared all three and discovered the following: (1) iIn comparison to metaheuristic algorithms, heuristic algorithms are simple to develop and find a satisfactory solution in a short period; (2) performance measurements for metaheuristic algorithms are determined by the nature of the issue, underlying configuration, and the method used for finding a solution; and (3) hybrid algorithms minimize both the calculation time and the cost. Furthermore, it outperforms other algorithms in terms of effectiveness.

Amrita Jyoti et al. [37] reviewed different papers related to cloud computing from 2015 to 2018. They compared and contrasted the various load balancing algorithms used in load balancers, as well as the brokering policies utilized for each service and its scheduling types. They classified and analyzed techniques based on key parameters. They discovered that LB algorithms are frequently employed in the development of resource utilization, energy conservation, and service quality. Furthermore, future directions have also been highlighted through a thorough examination of load balancing strategies and service brokers.

Based on the existing literature, it was found that very few of the studies analyzed load balancing solutions based on swarm intelligence techniques. The next section now analyzes swarm intelligence-based solutions.

4. Load Balancing in Cloud Computing Using SI

In this survey, SI techniques for load balancing are divided into two basic categories (as shown in Figure 4): One is a traditional algorithm, and the other is a Modern algorithm. The details of their variations are presented below.

4.1. Traditional SI Algorithms for Load Balancing

The idea to apply "collective intelligence" inspires the study of swarm intelligence. This intelligence is decentralized, coordinated, and distributed in an atmosphere [30]. Using swarm-based load balancing algorithms can efficiently solve the problem of load balancing in cloud computing environments [38].

Various selected types of algorithms based on Swarm Intelligence for load balancing are shown in Figure 4:

(a) Genetic Algorithms;
(b) Particle Swarm Optimization;
(c) Ant Colony Optimization;
(d) Artificial Bee Colony;
(e) Grey Wolf Optimization;
(f) BAT Algorithm.

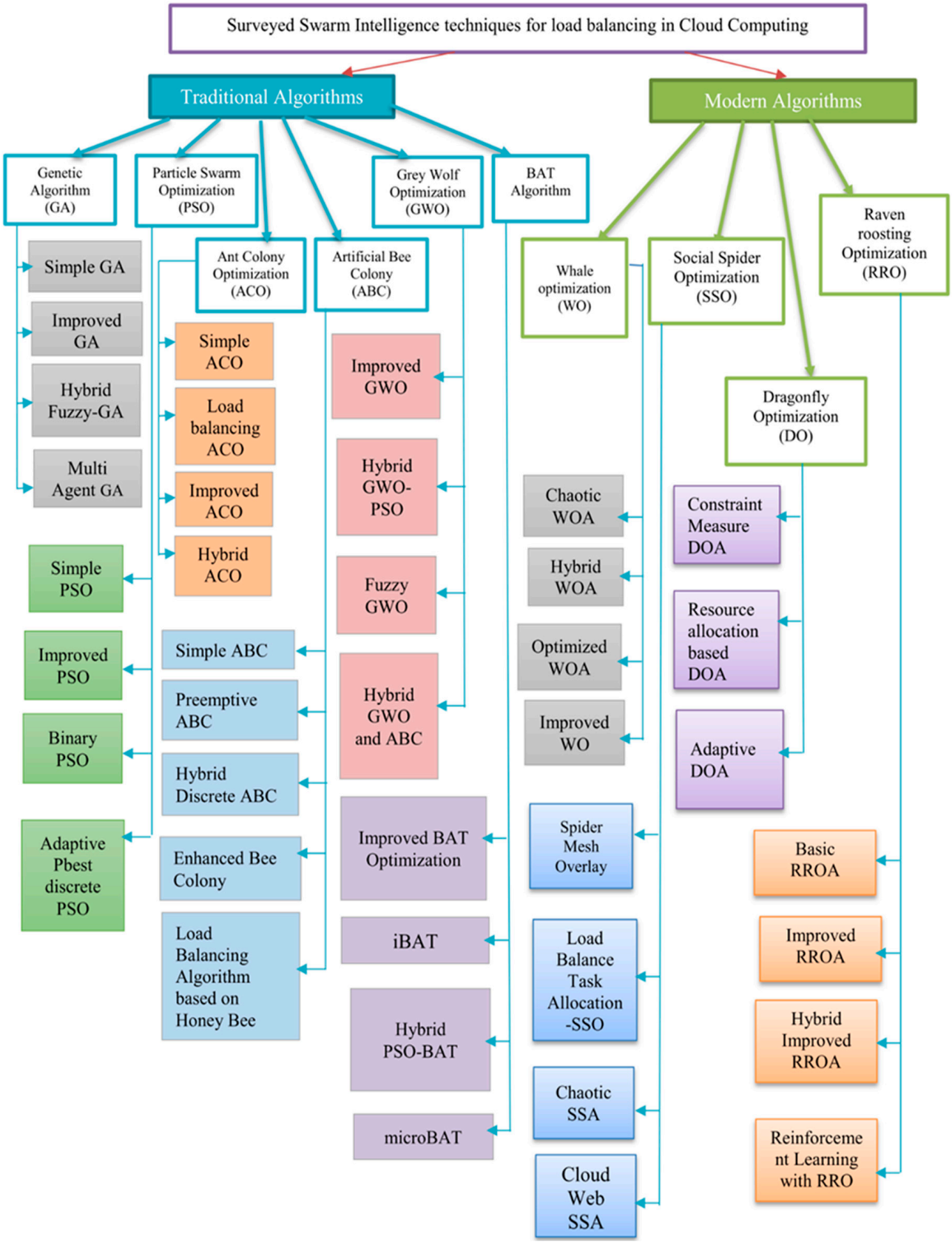

Figure 4. Swarm intelligence algorithms under discussion.

4.1.1. Genetic Algorithm (GA)

GAs are basic examples of artificial life strategies that takes biological systems as a model to produce computer programs. These computer programs then learn in the same manner as living systems [39]. Therefore, genetic algorithms are grouped with swarm intelligence in order to evolve the methods of optimization. They are concerned with simulating the biological progression of evolution [40]. Genetic algorithms reflect extension at the biological level. In the natural world, genetics convey an organism's inherited information. This biological information is carried by chromosomes in the form of genetic codes. Particularly, these codes can be used for identifying the characteristics of an organism. It has been concluded by Darwin that the survivability of any organism purely depends upon the characteristics that it has inherited from the parents. The organisms "most fit" to survive in an environment will be chosen to propagate and pass their genes. Three key parameters are involved in the GA [18]: selection, crossover, and mutation. In the first, the solutions with the best match are selected, crossover is selected for child generation, more than one parent is selected, and as far as the mutation is concerned, the gene meaning in the chromosome essentially changes. Figure 5 shows the main phases of GA. It comprises different operators such as natural selection, crosses over, and mutation. The fitness function is used in natural selection to select the next population.

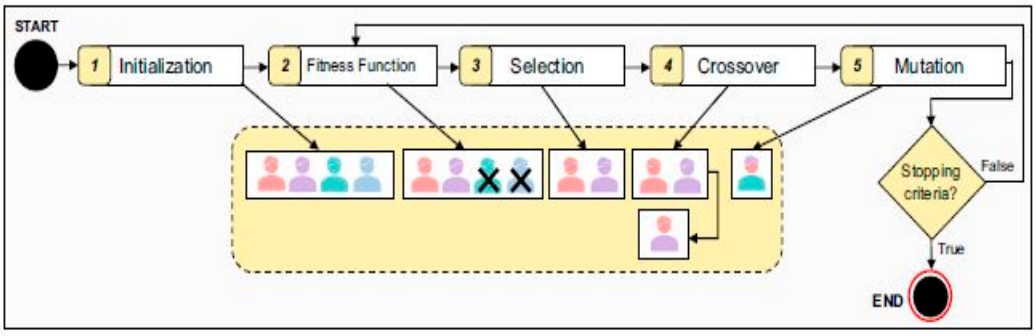

Figure 5. Genetic algorithm [41].

The steps involved are presented in the algorithm.

Pseudocode of GA [42]

1. Produce initial population "I" comprising chromosomes.
2. Utilize fitness function to measure the value of fitness of each genetic code.
3. Use the selection operator to pick the genetic codes that the next generation will make.
4. Utilize these chromosomes to perform the crossover procedure.
5. Now pick these chromosomes and execute the mutation process on them.
6. Produce the fitness value of offspring known as these newly produced genetic codes.
7. Improvise the population by changing irrelevant choices with genetic codes of better offspring.
8. Go to step 3 and execute until 7 until the situation of termination is reached.
The terminating scenario can be the last number of iterations possible or the fitness value of genetic code for all the possible runs is not able to update.
9. Return the best genetic code as a final result.

GA and Its Variations

Simple Genetic Algorithm: In 2016, Hussain A Makasarwala and Prasun Hazari performed load balancing by using GA [18]. The initialization of the population is dependent on the priority of the order and based on time (short time requests have more priority). After this, the selection of chromosomes is carried out based on fitness. Partial Mapping Crossover (PMX) is then used as a crossover operator for the ordered chromosomes. The block is interchanged between two parents in this process, and duplicates are sorted. Then, mutation is performed by swapping digits to enforce abrupt change in population. After

these steps, the chromosome is added to the new population and checks for termination conditions. Then, the algorithm ends.

In 2015, a combination of two methods has been proposed by Santanu Dam et al. [43]. GELS is particularly used for the initiation of GA populations. The initiation of the first population is based on the GELS determination of the velocity of chromosomes. Then, based on fitness, the two chromosomes are chosen, two-point crossover is applied, and mutation is performed accordingly. This entire process is repeated one more time and then added to the new population. Then, the algorithm ends. They used cloud analyst simulation of the proposed algorithm to obtain significant results.

Improved Genetic Algorithm: In 2017, an Improved Genetic Algorithm (IGA) was proposed [44]. It improves the resource utilization of simple GA. The improved GA keeps a continuous track of all the available virtual machines. If a VM is found in free state then it assigns a newly arrived task to it. If there is no free VM, then the improved GA techniques check for the VMs in which its task is about to finish. This is performed based on completion time. After checking, the newly arrived task is assigned to the VM that has less time left than compared to other VMs [45]. In this manner, this algorithm not only balances the load but also saves energy and cost as well.

Hybrid Fuzzy-Genetic Algorithm: A hybrid technique has been presented to intelligent person load balancing in cloud computing [46]. Ali Saadat and Ellips Masehian proposed two modules to achieve load balancing. The first module is uses a Genetic Algorithm for selecting optimum arrangements of tasks, and the second module is for fuzzy logic. It effectively creates the objective function of defining busy server states according to their unique task queues. The optimization of GA offers robustness, reliability, and generality where service availability is a fuzzy performance. In this study, computer tests are also performed, and the best solution obtained is demonstrated. This eventually contributes to greater user retention. The designed software is used to produce and distribute ten jobs to the user, which are passed by three machines in the workspace. Matlab had been used to program the model and to execute it. The results of using the generated model on multiple orders revealed that the best answer could be found in the first ten iterations of the projected 20 iterations. The optimum answer was found in half the time it was expected to take, resulting in increased satisfaction level of the client.

Multi-agent Genetic Algorithm: Anant Kumar Jayswal and Prem Chand Saxena [19] presented the idea for efficient load balancing by using multi-agent GA. The proposed algorithm takes the users' priority and the finishing time of the earliest job.

At the request of a specific user, the load is balanced between all VMs based on CPU memory availability. The fitness function is defined as the difference between each host's load and the average load of the system. The results are analyzed by using cloud analyst for simulating the experiments. It performs continuous execution of simulations for load balancing, and the output is displayed in graphical mode with a high level of flexibility.

The IGA [44] is further improved by Vishal Goar et al. [47], which is known as Enhanced Improved GA. The main approach adopted is to reduce the number of migrations. This can be accomplished by changing the mutation calculation points, which ultimately makes the execution faster and more reliable. The results of the proposed Enhanced IGA are tested on MATLAB and compared with IGA. After comparing the response time (RT), finish time (FT), energy consumption (E), cost (C), and number of migrations (N) of both algorithms, it was analyzed that EIGA is better than IGA. Table 3 represents the comparison between IGA and EIGA.

Table 3. Comparison between IGA and EIGA.

Algorithms	RT (ms)	FT (ms)	E (Joules)	C (Buffers)	N (in Numbers)
IGA [44]	2.8	2.7	0.53	0.53	10
EIGA [47]	2.5	2.25	0.5	0.44	7

In addition to the algorithms discussed, there are some other algorithms presented in the literature. This includes techniques based on Tabu search, multi-agent systems, PSO, multi-population GA, Job Spanning Time and Load Balancing (JLGA), and hybridization of good features of JLGA (HJLGA). The details of these algorithms can be observed in [9,36].

4.1.2. Particle Swarm Optimization (PSO)

The actual concept behind PSO is inspired by the flying swarm of birds. The swarm of flying birds finds a point to land, and the decision of which point to land is a complex problem to identify [48]. The landing depends on several factors. These factors include the availability of food and the risk of the existence of carnivores. Therefore, the birds fly synchronically unless or until they find the best place to land, and finally, they land at once [49].

The measurement of the performance of several other algorithms shows that particle swarm optimization combines the workload with a reduced reaction period for any assigned task. PSO has lower costs for computing, and it is easier to grasp and execute [50]. It also deals with the tradeoff between convergence and divergence by working in adaptive mode and ultimately finds the optimal solution to the complex issue of load balancing in a cloud-based environment. Figure 6 presents the flowchart of particle swarm optimization.

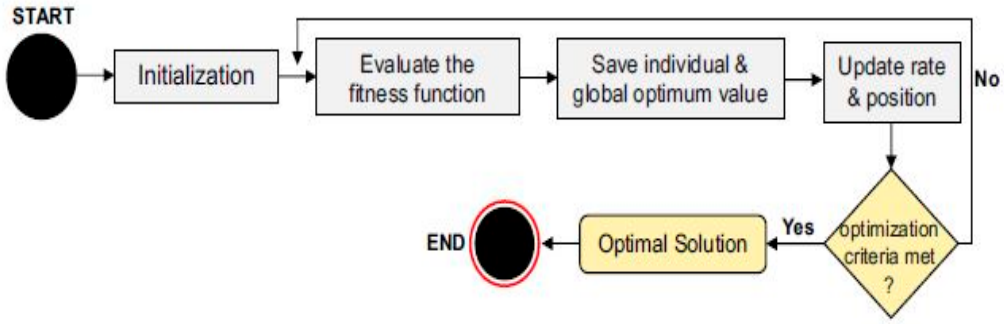

Figure 6. Particle swarm optimization flowchart [41].

The steps involved are presented in the algorithm. The next section presents various algorithms based on PSO. However, a range of other algorithms is also available, such as low time complexity and low cost BPSO, IPSO-TSA, and TBSLB-PSO [51–53].

Pseudocode of PSO [54]

1. Place vector and velocity vector initialization of each particle.
2. Convert the continuous vector of position to a discrete vector.
3. Calculate each particle's fitness value using a fitness function.
4. The "psobest" of each particle is given its best value of location until now. If the current fitness value of the particle is higher than the "psobest" of the particle, then substitute "psobest" with current position value.
5. From all particles, pick the particle with the best fitness value as pobest.
6. Using the following equations to change each particle's vectors:
Vj + 1 = qVj + i1n1 * (psobest − xj) + i2n2 * (pobest − xj), j is for iteration
Xj + 1 = Xj + Vj + 1
where,
q = interia
i1, i2 =acceleration coefficients
n1, n2 = Random numbers distributed arbitrary
pobest = In a population, the best place of entire particles
7. Go to step 2 and loop till 6 until the situation for termination is met. The last number of iterations or when the fitness value of particles for the next iterations stops changing is the terminating condition.
8. The best particle is created as the final solution.

PSO and Its Variations

Simple PSO Algorithm: Using the concept of PSO, Alguliyev et al. proposed this technique for load balancing in Cloud environments [55]. The presented method is to migrate the tasks requiring computing intensity relative to high-performing virtual machines. All scheduling processes are assigned with a corresponding weight to present the significance. These weights help to identify the optimal solution. In this research, the objective function has also been proposed, which is different than compared to the existing ones, and the scheduling of tasks is considered for heterogeneous virtual machines [56]. A task-based load balancing optimization method is used. The minimization of the task execution time and task transfer time as optimization objectives is considered by this optimization strategy. The software packages Jswarm and Cloudsim are used to assess the performance of the suggested solution. As compared to other optimization techniques, PSO has found more enhancing performance in dealing with discrete problems to be optimized [57].

Improved PSO algorithm: Another approach is presented by author Mahya Mohammadi Golchi et al. [58] to enable better average loads in cloud environments. In this research, they considered the disadvantage associated with PSO that it is highly sensitive to initial conditions. If there is any issue in choosing the initial population, then achieving a good result is not possible. To overcome this, a hybrid method has been proposed. This method promotes a hybrid approach of two concepts: One is based on firefly, and the other is on the improved version of PSO. First, the main target is to minimize the search range of optimal response by using the Firefly algorithm. The second target is to find the optimal response by using the improved PSO algorithm. In this manner, they are achieving tasks with high speed. It is concluded that the average load time of IPSO is 0.457 ms and Firefly is 0.47 ms, but the average load time of the hybrid IPSO-Firefly algorithm is 0.259 ms.

In another research, Kai Pan et al. [59] proposed a model to balance the load in cloud environments when the network is complex. The model utilized the concept of the PSO algorithm. The improved PSO algorithm proposed by the paper shows tremendous performance. This improved version performs some adjustments to define the particle's position and velocity. The rules followed for the continuous updates ultimately re-evaluate the fitness value to manage loads more intelligently.

Binary PSO algorithm: Jean Pepe Buanga Mapetu et al., [59] have proposed an approach based on a binary PSO algorithm. The target is to resolve the issue of optimal scheduling and balancing of diversified tasks onto different heterogeneous virtual machines without taking a long time. To achieve task scheduling and load balancing, a proper framework of the BPSO algorithm is proposed. This framework is based on three modules. The first module receives the user requests and splits them into multiple tasks. These tasks are received by the cloud manager. This cloud manager is the second module that ultimately outputs a task's local queue for each virtual machine. This module utilizes the IBPSO algorithm to balance tasks by assigning each local queue to the corresponding virtual machine after analyzing optimization constraints. At the same time, the pricing model determines the execution cost for all executed tasks. While virtual machine manager is the third module that relies on various hosts, its purpose is to manage the virtual machines used to execute the tasks of users. This idea provides scalability with proper load balancing and task scheduling.

Adaptive Pbest discrete PSO (APDPSO): Zhang Miao et al. [42] proposed an APDPSO algorithm to overcome the issues associated with the PSO algorithm. The major drawback experienced by PSO is randomness in the movement of particles which ultimately affects the discretization strategies. The proposed algorithm utilizes the stored good solutions to update the personal best positions of particles with a certain probability. A discretization method is also applied on PSO for continuous management of change in the velocity and position vectors of the particles.

4.1.3. Ant Colony Optimization (ACO)

ACO is a population-centered metaheuristic algorithm that can be used to estimate potential solutions to complex problems with efficient optimization [38]. This algorithm is inspired by ant colonies. It checks for good solutions to a given problem with a set of artificial ants (i.e., software agents). To find the best possible course, a weighted graph is designed. By moving on the graph, the software agents gradually construct solutions (the same as ants moving in a straight line). A presumptive solution construction mechanism is a collection of parameters linked with graph components (either nodes or arcs) for which its values are changed by the agents at runtime. Figure 7 depicts the activity of ants in the ACO. Some of the ant colony-based algorithms are presented in the next section. Interested readers are recommended [60,61] to other algorithms such as IMaxMin-ACO, CUDA-based ACO, IACO, and modified ACO.

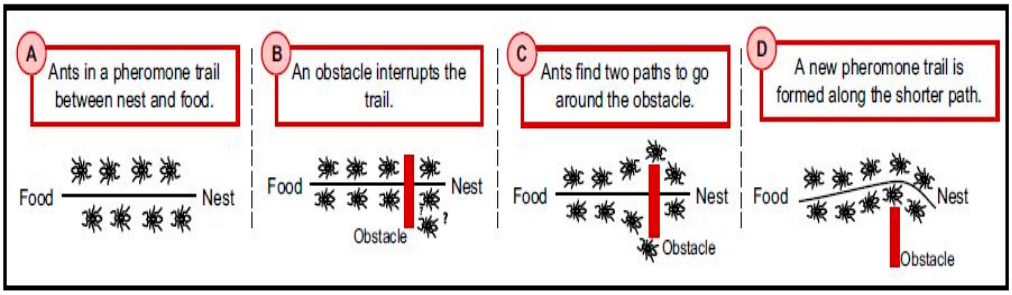

Figure 7. Environment of ant colony optimization [41].

The steps involved are presented in the algorithm.

Pseudocode of Ant Colony [62]

1. Initialize the Ant Colony, targeted limitations, and the memory required for the source of food.
2. Repeat steps 2–11 until stopping criteria not reached
3. Allocate the starting position to each ant
4. i = 1
5. For i<=ant perform the following operations
6. Use the state transfer rule to pick the next node.
7 Perform pheromone updations on each step
8. i++
9. End loop
10. Before each ant has established a solution.
11. Update the best one and perform updations on pheromone
12. Output: Print Optimal solution

ACO and Its Variations

Simple ACO algorithm: In an article about task scheduling termed SWIM (System Wide Information Management), Gang Li and Zhijun Wu focused [63] on the load imbalance problem. The users required better quality load balancing in terms of task scheduling. Project scheduling approaches are combined with the standard ACO algorithm to accomplish this large-scale network information system. Updating the pheromone uses the hardware output quality index and load standard deviation feature. Load balancing is implemented in this article by using the key principle to choose the best task scheduling order for the ant colony algorithm. The scheduler is seen as an ant in the specific scenario, and the scheduling phase is seen as the process of ant foraging. As indicated by the performance of hardware parameters, the pheromone is updated, keeping the total expected time at a minimum. To estimate the heuristic function of the ant colony algorithm, this projected length is then used. In the end, to deal with the assigned mission, a node that has high performance and low load witty minimum completion time is selected.

Load Balancing ACO algorithm: The LB_ACO algorithm [64] first performs pheromone initialization, followed by the task of selecting a VM based on that value. It maintains two matrices in which one is the completion time matrix, and the other is the matrix of pheromones. It computes the completion duration of all jobs using the beginning time of the corresponding job. After this, an arbitrary initialization is performed at the beginning time of the first task. It is possible to determine the beginning time of other tasks from the completion stamp of tasks previously assigned to the respective computers. At last, at the time when all activities are finished, the full completion time is then the magnitude of the plan. By the ACO algorithm, the best mapping of tasks to VMs is determined. By using the process followed by an ant, we obtain a more workable solution. The available resources are assigned by utilizing the ant concept. Assigning each task to any of the available resources without preemption is proposed. It first initializes the pheromone matrix. By calculating met heuristic probability, a subsequent task is assigned by using the transition rule. Depending upon the highest probability, the tasks are assigned to a particular VM (by keeping track of the completion time). For every ant, the fitness function is calculated to find the local solution. Later on, it ultimately changes the global solution. After performing non-dominated sorting on solutions, an optimal solution is achieved.

Improved ACO algorithm: Awatif Ragmani et al. [65] introduces an improved ant colony algorithm by introducing a fuzzy logic module with it. The fuzzy logic module calculates the pheromone value [66]. It also uses the Taguchi concept to optimize the ACO algorithm parameters. A CloudAnalyst is also used as a simulator, which confirms that the algorithm is appropriate for dealing with complex networks through experimental outcomes. The simulations proved that the technique adopted improves load balancing in the cloud environment while the response time is reduced to 82%.

Hybrid ACO algorithm: M Junaid et al. [21] has suggested a hybrid approach. This approach comprises two phases. In the first level, the vector machine support is updated to establish precise classifications across various file formats. Best classification over established validity metrics has been demonstrated by the initially classified results. Values for these metrics differ between 0 and 1. The performance of the File Type Formatting algorithm using the Help Vector Machine is the data that ACOFTF will obtain for scheduling purposes. The ACO algorithm performs QoS metrics for multi-object scheduling (i.e., reduced violations, minimum migration time, high optimization, reduced makespan, and high response time).

4.1.4. Artificial Bee Colony (ABC)

ABC is produced by inspiration from the colony of bees that consists of three bee groups: bees working, scouts, and onlookers. For each food source, one artificial bee is predicted. We can, thus, deduce that the number of working bees in the colony is the same as the number of food sources around the hive [67], and after coming back from the food source to their hive, they dance in this area. It becomes a scout after this bee whose food supply is refused, and it looks for new food. The artificial bee colony algorithm's operators, control parameters, and application domains are depicted in Figure 8. Some of the algorithms based on ABC are presented next. Some other variations of ABC algorithm such as ABC with EA, IABC, GB guided ABC, modified ABC, cross over and mutated ABC, Hybrid ABC, memetic ABC, etc., are also presented in various research studies [68].

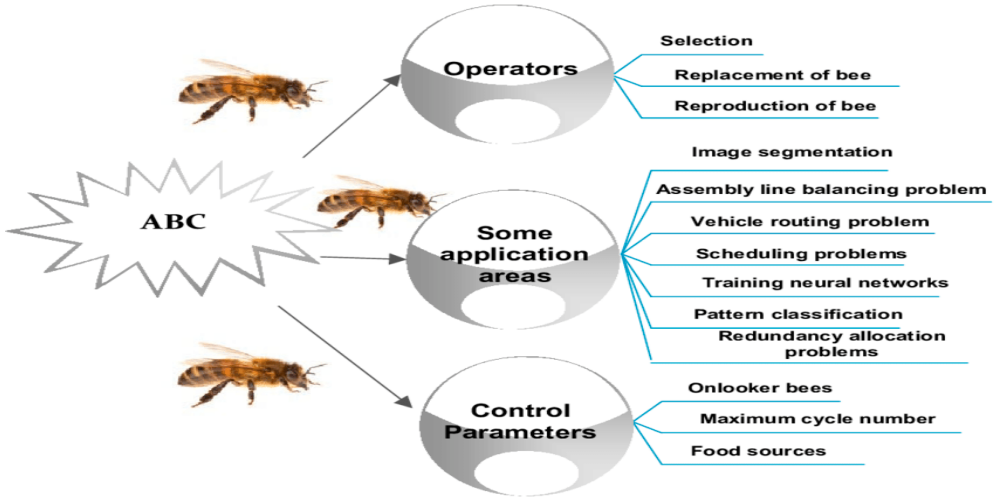

Figure 8. Artificial bee colony [69].

The steps involved are presented in the algorithm.

Pseudocode of ABC [70]

1. Initialize the bat population and evaluate the fitness.
2. Calculate the initial fitness value, f_init.
3. Set the best solution, s_best.
4. Set maximum no. of iterations, Noi.
5. Set the population size, N.
6. i = 1
7. Start loop end till iteration < Noi
8. Generate a random solution for all no. of employed bees.
9. Generate solution for all onlooker bees
10. Apply a random structure on the selected bees, s_Changed.
11. IF (s_best< s_Changed)
12. Then s_Changed=s_best
13. END IF
14. Scout bees identify the abandoned source of food and ultimately replace it with new
15. i ++
16. End loop
17. Output: the best solution

ABC and Its Variations

Simple ABC algorithm: ABC algorithm is a method of optimization that is very good at inspection but bad for manipulation [20]. The bee algorithm is suggested for performing optimization techniques by utilizing the intelligent foraging behavior of honey bees. The gathering of honey bees is called a swarm, which can adequately conclude the full mission by cooperation. Different researchers used the ABC algorithm for improvements in load balancing [49].

Preemptive ABC algorithm: To define an optimal schedule of tasks for virtual machines, task scheduling using artificial bee foraging (TSABF) optimization is proposed by Geetha Muthsamy [71]. Using the principle of ABC, the activities are preemptively prepared. In the suggested methodology, the purpose of task preemption is performed to turn down the execution time of the tasks about different priorities. The researchers compared the results of the experiments with another load balancing algorithm. It has been concluded from the experimental results that load balancing can be intelligently achieved by using TSABF with improved QoS than compared to existing honey bee behavior-inspired load balancing algorithms.

Hybrid discrete ABC algorithm: In a cloud environment, Junqing Li et al. [72] considers and solves the flexible task scheduling issue. To solve the problem, the hybrid discrete ABC algorithm is being used. It is suggested to first model the question as a hybrid flow shop scheduling problem. They considered both single and multiple goals. The proposed work considers three types of bees (i.e., the working bee, the onlooker bee, and the scout bee). As a primary contributor to coping with manipulation skills, various forms of permutation systems are investigated. An efficient selection approach is also used for the enhancement of the exploitation process. The convergence ability is also improved to increase by designing the abandoned solution. The performance of the algorithm is confirmed by performing proof of work (i.e., tested on authenticated benchmarks).

Enhanced Bee Colony Algorithm: The proposed method [73] utilizes the foraging behavior of bees to improve load balancing in cloud environments. Inspired from the ACO algorithm, rescheduling is performed after identifying the underloaded machines. For every task (honey bee), calculate the load on VM (honey bee foraging a food source) and make a decision either to shift load or not. After this, the VMs are grouped based on load. Then, by using the groups, sorting is performed to make sets of overloaded (honey bees starve for a food source) and under-loaded machines. The procedure of sorting is performed again to prioritize the overloaded tasks. After assigning priorities, the capacity of underloaded VMs is checked in order to identify a suitable VM for load allocation. After assigning load, the overloaded and underloaded machines are updated in order to be considered for future load balancing.

Load Balancing Algorithm Based on Honey Bee (LBA_HB): Walaa Hashem et al. [74] proposed the LBA_HB algorithm. It mainly focuses on distributing the load to the network links in the most balanced way. For proper management of workload, whenever there is a task that asks to be processed by the VM, it checks for two conditions:

1. No. of tasks being processed by the VM < No. of tasks processing by other VMs;
2. The deviation of the VM processing time from the average processing time of all VMs is less than the threshold. In this manner, the overall response times and data center processing times are successfully minimized (proved by simulations). As the algorithm intelligently focuses on distributing load to avoid under and overutilization of VMs, compared to other swarm algorithms i.e., ACO and ABC, the balancing performed by LBA_HB is more efficient.

4.1.5. Grey Wolf Optimization (GWO)

GWO algorithm [75] is developed for allocating tasks in order to balance load efficiently. This algorithm duplicates the hunting style of wolves. Similarly as the wolves that live in a pack of four levels, this algorithm breaks the strategy of load balancing into four levels. These four levels are categorized as omega, delta, beta, and alpha. On the first level is the one who leads the pack (independent from the gender) in terms of making decisions. On level 2, there are supporters who help the leader to decide and maintain control in the pack. Level 3 is for scouts to keep an eye on the limitations for protection. Level 4 includes the substitutions in the pack; they eat last. Figure 9 represents the behavior of the grey wolf optimization algorithm. In addition to the algorithms discussed in the next section, some other variations of the GWO algorithm such as DGWO, TLBO-GW, Modified GWO, hybridized versions GWO, PGWO [76–78], etc., are also presented in other research studies.

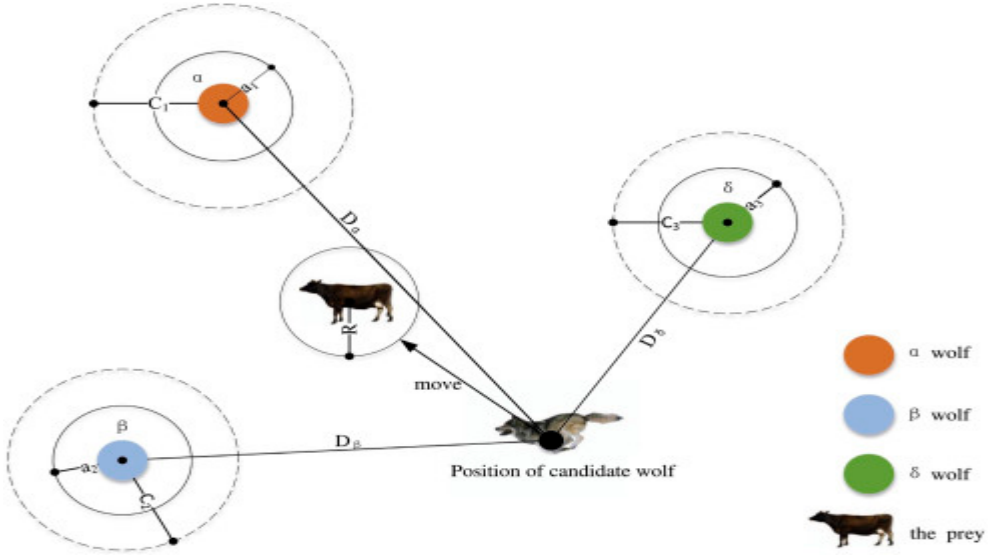

Figure 9. Gray wolf optimization algorithm [79].

The steps involved are presented in the algorithm.

Pseudocode of GWO [54]

1. Select the max no. of iterations L.
2. Initialization of the population X_i = (i = 1, 2, 3, 4, . . . , n)
3. Initialization of r, N, D, & T = 1.
3. Produce fitness of all the wolves (searching machines)
4. Let, P = searching machine (best)
5. Let, Q = searching machine (second best)
6. Let, R = searching machine (third best)
7. **Start while** (i < L) do
8.Repeat step 9 for every searching machine
9. Change the position's value of the recently available searching machine.
10. Change the values of r, N, and D
11. Find the fitness of all searching machines
12. Update P, Q, and R
13. i ++
14. end while
15.OUTPUT the best searching machine

GWO and Its Variations

Improved GWO Algorithm: Optimal scheduling of tasks on VMs is the goal of the Improved GWO Algorithm [80]. Firstly, it collects all the task and resource information and checks whether the allocation requirements are satisfied or not. After viewing the best possible option, a resource is assigned to the scheduler. Based on the information about CPU, storage, and bandwidth, cloud resource management updates the resource nodes accordingly. It target is to reduce the make span, total cost, and the maximum no. of task completion within the specified time. Improved GWO is used to find the optimal solution. The potential of the proposed technique has been proved by implementing it on CloudSim.

Hybrid GWO-PSO Algorithm: A hybrid system [81] is proposed using GWO and PSO algorithms for load balancing. In this approach, GWO is suggested to be used first in order to produce the best position as alpha, i.e., $X\alpha$. After this, instead of calculating the personal best position, PSO is executed by utilizing alpha. In this manner, the main contribution achieved using this approach is to contrive an objective function that efficiently manages the load in a cloud environment.

Fuzzy GWO algorithm: Fuzzy GWO algorithm [82] is adapted to implement a modified load balancing algorithm in cloud computing and IoT. The objective is to achieve better

response times with a balanced load. The algorithm saves all the information along with the assigned demands to the VMs as wolves first find the overloaded system. Whenever there is an allocation request, the least loaded system for assigning the assignment is found, just as the second move for the wolves is to identify the status. The researchers then modified the procedure by introducing fuzzy logic with GWO for making the system stable. This technique considers CPU speed and load as inputs for better load adjustment. They assign rules by using fuzzy logic, i.e., LOW when speed is ZERO (0), MEDIUM when speed is 0.4, HIGH when speed is 0.6–0.7, and VERY HIGH when speed is ONE (1).

A Hybrid GWO and ABC Algorithm: A hybrid solution is proposed by Soukaina Ouhame, Youssef Hadi, and Arifullah [83]. The resource allocation method in VM often fails because of a lack of proper load balancing. It is either overloaded or underloaded. To resolve this discrepancy, a hybrid of the GWO and ACO algorithms is proposed. The grey wolf performs the first improvement at local search; it is then utilized by the ABC algorithm for further improvements in fitness function. In this manner, this hybrid technique improves overall throughput, stability, execution time, and energy consumption. It efficiently improves the resource allocation system in VM to 1.25% than compared to simple ACO and GWO algorithms in cloud environments.

4.1.6. BAT Algorithm

BAT is an optimization algorithm that is inspired by echolocation behavior. BATs use their sounds to locate their target. When bats generate sound, it spreads to different prey available in frequency form. Based on this frequency, they calculate the distance after collecting all the signals [84]. The same idea can be adapted for load balancing as it runs individually and focuses the node to make the coordination localized.

Arif Ullah and Nazri Mohd Nawi [85] applied the BAT algorithm in VM to verify that this idea improves load balancing in cloud environments. The idea is that whenever there is a need to entertain a task, an optimal VM among all the available VMs is first searched for. In that way, this algorithm assigns the task to the optimally available VM. Figure 10 shows the flowchart of the BAT algorithm. Several variations of this algorithm are presented in the next section. Some other variations of the BAT Algorithm such as chaotic BAT Algorithm, directional BAT Algorithm, θ-Modified BAT, BAT mutation crossover, etc., can be seen in [86,87].

The steps involved are presented in the algorithm.

Pseudocode of BAT [88]

1. Input: Population of searching machine Z, dimension of solutions n, upper and lower bounds of solutions [$y_1, \ldots, y_n, L_1, \ldots, L_n$], Maximum iterations
Output: The best searching machine P
1. Initialize the grey wolf population $X_i = (x_1, x_2, \ldots, x_n)$
where ($i \in [N]$) and $x_j \in [y_j, Lj] \mid \forall j \in [n]$
2. Initialize a, A, C, & i = 1.
3. Find the fitness of each searching machine $f(X_i)$, where ($i \in [Z]$)
4. Let, P = the best searching machine.
5. Let, Q = the second-best searching machine
6. Let, R = the third-best searching machine.
7. while (i < Max iterations) do
8. for each search agent do
9. Change the position of the current searching machine accordingly
10. end for
11. Update a, A, and C
12. Calculate the fitness of all searching machine
13. Update P, Q, and R
14. i++
15. end while
16. Output P as the best solution

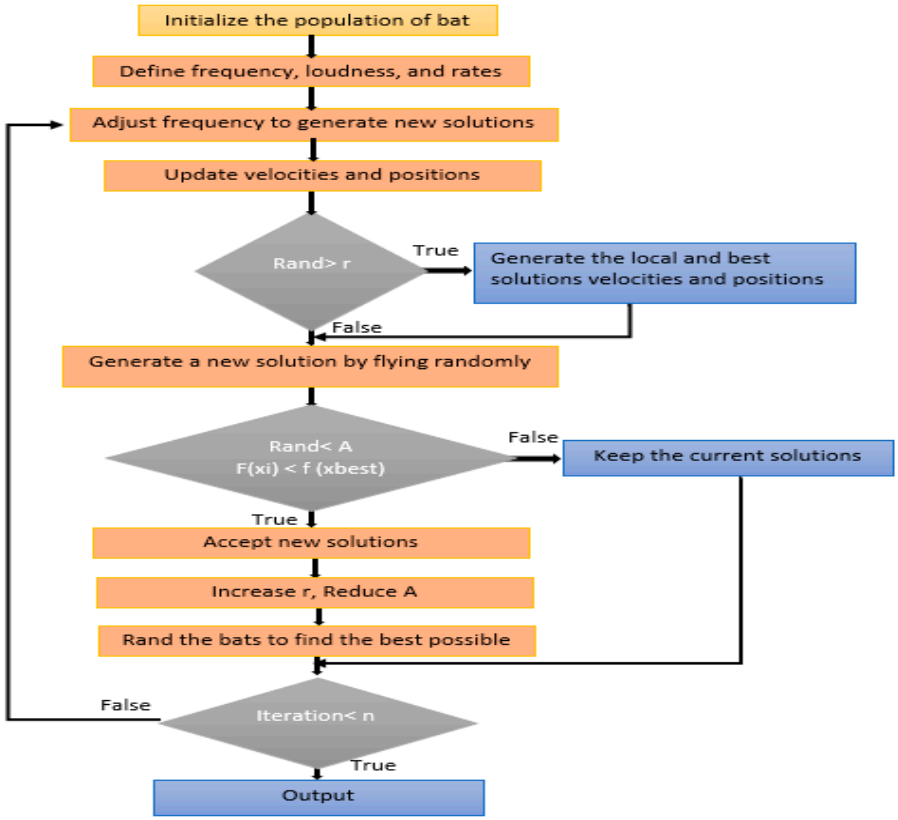

Figure 10. BAT algorithm flowchart.

BAT and Its Variations

Improved BAT Optimization (IBO) Algorithm: Improved BAT algorithm [89] is proposed to obtain more optimum and the finest results. To make this possible, the algorithm needs to be executed iteratively. Whenever there is a task for processing, the BAT algorithm finds the optimal server among the available. At the same time, the load scheduler also identifies the job type and resource required and selects the optimal VM for task execution. If the available server is meeting the requirements efficiently, then the load is assigned; otherwise, if the load is higher than it is distributed to more than one server. In that manner, this algorithm maintains load balancing by keeping all the servers busy and maintains them as neither under loaded nor overloaded. The suggested technique not only minimizes the response time but also performs load balancing without any delay.

iBAT Algorithm: A system is operated using an Improvised BAT algorithm for load balancing [62]. The iBAT algorithm utilizes the Minimum-Min and Max-Minimum algorithms, which aim to produce a population of more optimized virtual bats. The minimum-minimum and max-minimum algorithms are used for listing the least and most observed ending time of the tasks, respectively. In this manner, it schedules resources with the least execution time. In this manner, a more desirable population is generated to achieve the best results.

MicroBAT Algorithm: A system model [90] utilizes the concept of micro-bats as they recognize the shortest iteration to the prey. The model proposed improves its computations by adapting the echolocation method of bats. Youssef Fahim et al. separated their modeling in two sections. The first one is for the classification performed before allocation. This classification is based on the meta-heuristics bat algorithm. It pre-classifies the spots as per the available resources in the cloud environment. The second one permits the task to multiple VMs after checking the load (status and allocation). In this manner, this method allocates jobs to VMs with equal load distribution. It also guarantees all the

possible allocations in series or parallel mode. This technique also proposes shifting tasks from one VM to another in order to manage the load more efficiently (i.e., to avoid over and underutilization).

Hybrid PSO-BAT Algorithm: A hybrid approach of PSO and BAT algorithm is presented by Valarmathi, R et al. The idea is to swap and update the population by using both algorithms until the best population target is achieved. This technique mainly focuses on the main drawback of the PSO algorithm. Keeping in mind the fact that the convergence of PSO becomes slow after several iterations, the BAT algorithm increases this diversity. The fitness function monitors the poor population of both algorithms and transforms them into stronger ones. These good individuals are merged at the end to achieve enhanced optimization. The simulator proves the better performance achieved after performing task scheduling conducted in the cloud for load balancing. It reduces the cost and makespan. At the same time, it also improves the utilization of resources. Table 4 presents the pros and cons of the discussed traditional SI algorithms.

Table 4. Pros and cons of traditional SI algorithms.

Algorithm	Pros	Cons
GA	It makes greater use of resources and offers a better load balancing solution.	The numerous steps of computation add to computational complexity. When the search space is expanded, the efficiency suffers. It does not offer the same level of priority time.
PSO	It has a higher utilization rate. The load is redistributed from an overburdened virtual machine to a physical machine.	The performance of the algorithm depends on the problem.
ACO	Optimal resource usage through efficient load distribution among nodes.	Convergence takes a longer period time.
ABC	Enhances the maximum throughput limit.	When utilized in a sequential process, the lack of supporting material causes the process to slow down, and the solution raises the computing cost.
GWO	It is easier to use and converges more quickly due to less randomness and varied numbers of individuals assigned in global and local searching procedures.	The equal importance of the grey wolves' positions, which is not consistent strictly with their social hierarchy.
BAT	It boasts a high level of precision and efficiency. In terms of processing costs, it outperforms.	There is no mathematical analysis to link the parameters with convergence rates at an early stage; therefore, convergence occurs quickly.

4.2. Modern SI Algorithm for Load Balancing

As we have discussed in the last sections, adequate work is performed on traditional swarm intelligence algorithms to cope with the issues of load balancing in cloud-based task allocation [91]. There are certain issues associated with these algorithms that ultimately limit the pertinence of utilizing the best of them. Therefore, due to slow convergence, intricacy in implementation, and complexity to ensure scalability, some state-of-the-art techniques are highly required. In this section, the most increasingly competitive swarm intelligence-based load balancing strategies in the cloud include the whale, spider, dragonfly, and raven; roosting optimization algorithms are discussed.

4.2.1. Whale Optimization Algorithm (WOA)

In 2016, Australian researchers Mirjalili and Lewis [92] suggested the Whale Optimization Algorithm as a new swarm intelligence optimization algorithm. The model captures the humpback whale population's shrinking, encircling, loop updating location, and arbitrary hunting strategies, which were influenced by natural humpback whale hunting conduct. Figure 11 depicts the bubble hunting technique of the whale optimization technique. The main steps of WOA involve the following [93]:

1. Initialization phase: During this phase, the population is randomly created.
2. Fitness calculation phase: In this phase, a fitness function is calculated. This is used to calculate the fitness function. Depending on the outcomes of the test, the best whale (agent) is selected.
3. Encircling prey phase: The location of the prey is believed to be fixed during this process. Assuming that the current approach is the safest, the prey is covered by humpback whales. Other whales change their positions in accordance with the existing best agent. The exploration phase uses the following mathematical model (Equations (1) and (2)):

$$SD = |\, C.\, W^*(i) - W(i)| \tag{1}$$

$$W\,(i+1) = Wr(i) - A.SD \tag{2}$$

where i is the recent number of iterations;

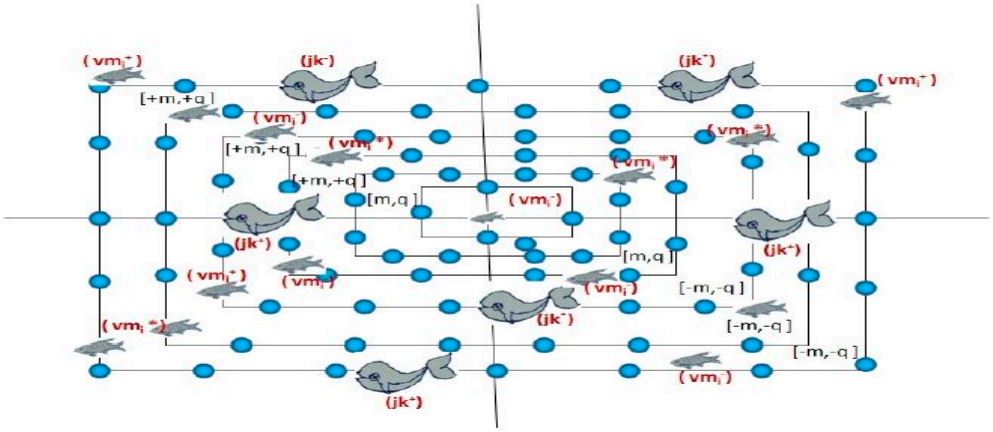

Figure 11. Bubble net hunting strategy of whale optimization load balancing technique [91].

W is the position vector;
W^* is the position vector of the best solution obtained;
C is the coefficient vector.
SD is the search distance;
Wr is the random position vector;
A is the coefficient vector.

The spiral equation for the humpback whale's spiral adjusting location strategy to prey follows Equation (3).

$$(i+1) = SD'.e^{bl}.\cos(2\pi l)W^*(i) \tag{3}$$

The steps involved are presented in the algorithm.

Pseudocode of Standard WO

1. Initialize the population of whales (agents) "W"
2. Calculate the value of the fitness function.
3. Randomly pick the search agent W*
4. While i = 1 and t < MAX_LIMIT
5. Repeat for each search agent
6. if (probability < 0.5)
7. if (| A | <1)
8. Update the position of *i*th search agent by Eqution (1)
9. else if (probability < 0.5 AND | A | $\leq$ 1)
10. Update the position of ith search agent by Eqution (2)
11. end if
12. else if (probability $\geq$ 0.5)
13. Update the position of ith search agent by Eqution (3)
14. end if
15. End inner loop
16. Calculate the fitness of W(t + 1) and update W*
17. End Outer loop

WOA and Its Variations

Chaotic Whale Optimization Algorithm (CWOA): Kaur et al. [94] targeted the slow convergence issue of standard whale optimization. To overcome this problem, the solution to improve the overall convergence speed is proposed. The study highlights the chaos technique to integrate with the standard WOA so that better performance will be achieved. The proposed algorithm begins by performing a probabilistic initialization of the whale group (VMs). It is then used to map a particular chaotic map, along with the setup of its first chaotic number and a variable. After that, the suggested variables are used to regulate the exploration and exploitation process and are as much the same as in WOA, which is ultimately configured. In addition, the chaotic number of the chaotic map is set. The fitness of all the whales (VMs) is initialized in the objective function. This fitness is then assessed by using various statistical benchmark functions in the next phase. The whale (VM) with the highest fitness is thought to be the strongest quest agent at the moment. The recommended procedure is used to balance workload and resource distribution in the cloud in order to reduce energy consumption.

Hybrid Whale Optimization Algorithm (HWOA): Strumberger et al. [93] proposed a modification in the original whale optimization technique. The research study targets the tradeoff between exploitation and exploration in simple WO algorithms. The Hybrid WOA performed the exploration as bee colony metaheuristics. This approach then uses additional hybrid controlling parameters to monitor new exploration criteria. Finally, this hybrid proposal integrates the firefly algorithm in order to further update the search equation. Comparative analysis showed that the Hybrid WOA surpassed Simple WOA by about 19% in the benchmark of 100 tasks.

Optimized Whale Optimization Algorithm (OWO): Farinaz et al. [95] highlighted the early convergence issue of simple WOA. To address this issue, a new concept for grouping whales is presented. The chosen strategy divides the filtered population into a specific number of groups, then randomly selects a member of each group for use in the whale optimization algorithm's encircling prey segment. The mean best fitness was then increased to aid with exploitation, discovery, and convergence rate. At high workloads, the Optimized Whale algorithm is used in a cloud computing environment to minimize average execution time, response time, and throughput in the cloud computing setting. Experiments prove its response time far better than the others. The progress in percentage when compared to standard WOA is about 37.60 percent, 81.21% when contrasted to bat, 79.60% when compared to Chaotic WO algorithm, and 79.90% when compared to particle swarm algorithm.

Improved Whale Optimization Algorithm (IWO): To boost the optimal solution exploration capability of the WOA-based system, Xuan Chen et al. [96] suggest an effective algorithm called Improved WOA for cloud task scheduling. The researchers are particularly interested in optimizing the makespan, load, and price expense of a cloud computing environment for specific tasks, and these considerations will be critical in ensuring that the VMs' overall functionality is as efficient as possible. In the very beginning, the proposed task scheduling scheme is used to map the foraging whale model. This model ultimately allows obtaining an estimated optimal solution. On this foundation, the study recommends Improved WOA for cloud task scheduling for load balancing, an advanced method aimed at enhancing WOA's optimal solution exploration ability. IWC starts by setting network input tasks and all the available specifications. It also identifies the underlying virtual machines, tasks, and resources, allowing them to be mapped according to a particular strategy. The task scheduler at the scheduler layer will then create an optimized task execution plan based on modeling to meet the assigned specifications. Ultimately, the configured control plan is sent to the task control center for operation, with the output results being sent to the clients. IWC outperforms the competition around the board, showing that it can efficiently minimize device costs while optimizing the load in VMs for cloud computing activities.

4.2.2. Social Spider Algorithm (SSA)

Spiders include a large order of nearly 50,000 species identified by scientists all over the world. Researchers have classified them into three groups [70]. These types include colonial, subsocial, and social spiders. Social spiders are unique among these three groups. These spiders congregate in groups and communicate with one another. Yu et al. [97] suggested SSA as one of the most prevalent methodologies for optimization techniques, which was influenced by the foraging behavior of social spiders, which can be observed in Figure 12.

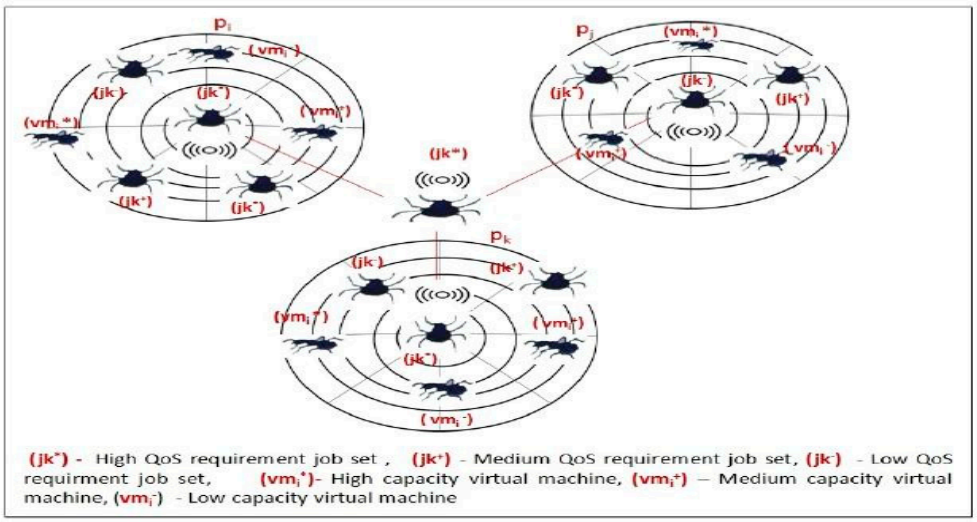

Figure 12. Behavior of social spider technique [91].

The steps involved are presented in the algorithm.

Pseudocode of SSA [98]

1. Inputs:
Total count of spiders "T"
Total No. of female spiders "T_f"
Total No. of male spiders "T_m"
2. Initialize the positions of male and female spiders
3. Ti = total no. of iterations
4. Output: The fitness value of the optimization problem and the optimal location of social spiders.
5. Process:
while i $\leq$ Ti do
6. Determine the mating radius for female and male spiders.
7. Determine the weight of the spiders.
8. Based on female and male collective operators, determine the response of female and male spiders.
9. Accomplish a mating procedure with a dominant male and a dominant female.
If the spider progenies are larger, then update the strategies.

SSA and Its Variations

Spider Mesh Overlay (SMO): Usurelu et al. [99] showed the various characteristics of a novel, naturally influenced spider mesh overlay that was used for load balancing non-pre-emptive jobs, each of which consumes a substantial amount of energy. Each overlay unit can provide the highest capacity for a specific resource while providing lower capacities for other devices. The routing policy removes the chances of redundant communications and ultimately improves performance by assigning nodes efficiently. The network's central point serves as a task manager. This task manager manages the distribution of jobs depending on the required resource. A node is matched by a task based on two methods. It first ensures that the node has the resources required to complete the task and then ensures that the task does not surpass the node's usable resource capacities. Both methods highlight that the server load and the overlay topology are stable. The suggested algorithm achieves an average integrated load of 50 to 85%.

Load Balance Task Allocation (LBTA-SSO): Mahato et al. [100] also targeted the issue of load balancing in cloud environments. The idea is to enhance the performance of cloud-based environments with improved efficiency. In order to achieve balance load task allocation, a modified algorithm of SSA is presented. The proposed technique focuses on providing the resources on a real-time basis. In this manner, there will be no superfluous rise in the traffic of requests. This scheme eventually results in no degradation in performance and reliability in any transaction.

The algorithm first generates the total available nodes. After that, it initializes the vibration of the target node for every node available. The algorithm maintains a transaction queue. It continuously monitors the targeted node's value for every transaction and node, respectively, until the transaction queue becomes empty. If the vibration of the targeted node is greater than the maximum possible value of vibration, then that transaction is assigned to the targeted node. Otherwise, the algorithm picks the best vibration among the available nodes' vibrations. This selected vibration is then compared with the targeted vibration. If it is greater than the targeted one, then the transaction is then allocated to the node that contains the best vibration. Finally, all nodes will update the fitness value accordingly.

Chaotic Social Spider Algorithm (CSSA): In a research paper [101], a chaotic social spider algorithm was proposed. The idea of this technique is driven by the social spider in order to solve the job planning problems in interconnected virtual machines. Each searching machine in Chaotic SSA has a memory that holds the position of the practical alternative as well as the fitness value of the VM in the form of a Broadcast Message. Furthermore, each search machine is set up to be dynamic, with the ability to communicate and transfer to another VM location at any time. CSSA considered the spider web as a cloud computing environment, where each resource, i.e., VM, is represented as a feasible

solution (food source in case of spiders) in the cloud. Each searching device in the cloud environment moves freely in the solution space to find the best VM for the user tasks. When the number of tasks is 200, the CSSA decreases the total cost by an average of 33.66% relative to Genetic, PSO, and ABC Algorithms. Furthermore, Genetic, PSO, and ABC algorithms show efficiency up to 79%, 83%, and 86%, respectively, whereas CSSA generates 97% efficiency. As far as other performance parameters are considered, CSSA proves effective performance improvements.

Social Spider Cloud Web Algorithm (SSCWA): Abrol et al. [102] proposed that the tasks will behave as spiders, and their QoS characteristics will be defined as the spider's fitness based on the Social Spiral algorithm. The tools function as prey, and their ability corresponds to the health of the prey. In contrast to other tasks, tasks with a higher QoS, i.e., a high resource consumption criterion, are given higher priority. The proposed procedure first generates the population of tasks and resources along with the quality constraints. The fitness value is obtained from the given tasks and every available resource. After this, depending upon the available resource utilizations, a vibration is produced. The task's vibration frequency is related to the resource's vibration intensity. If the vibration frequency of the resource is greater than the vibration intensity of the task, then the task will be assigned to the resource. Otherwise, another resource with a higher vibration rate than the previous one is found. The task's vibration frequency concerning service vibration strength is calculated. To find the best solution, a population of tasks and resources with QoS requirements is created. In terms of execution cost, throughput, execution, and response time, experimental results indicated that the QoS aware SSCWA surpasses ABC, PSO, and ACO.

4.2.3. Dragonfly Optimization Algorithm (DOA)

DA is imitating a dragonfly's swarming action [103]. Relocation and hunting are the two main causes of their swarming (dynamic swarm or static swarm, respectively). Small groups of dragonflies pass around a limited area to hunt other species in a static group. Social motions and sudden changes are characteristic of this form of swarming. Dynamic swarming, on the other hand, involves a large number of dragonflies forming a single community and moving in a specific direction for a long period.

Dragonflies should adjust their weights to respond to the transformation from intensifying to diversifying during the optimization phase, ensuring the convergence of dragonfly individuals. Figure 13 depicts the dragonfly swarming behavior of jobs within partitions, which involves assembling a swarm of jobs of varied sizes to travel towards the best-fit virtual machines.

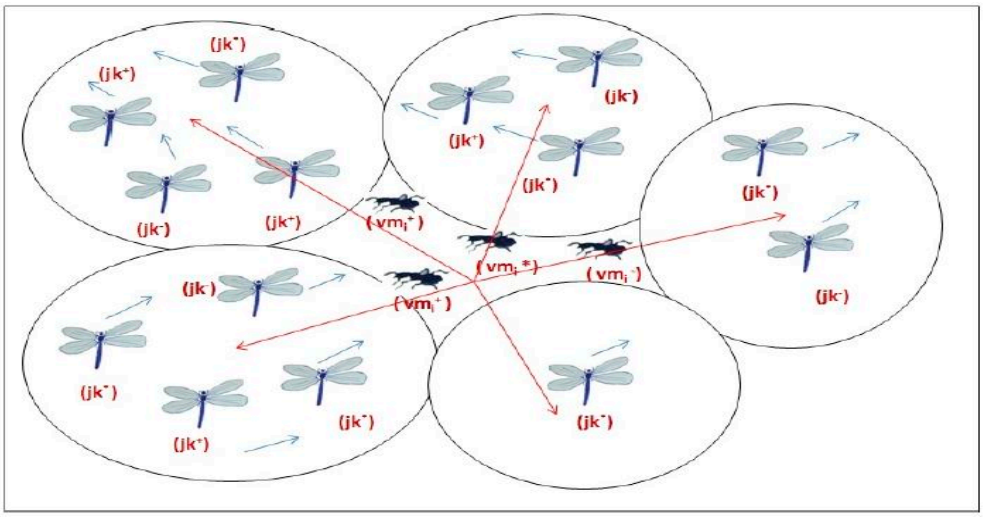

Figure 13. Behavior of dragonfly optimization algorithm [91].

The steps involved are presented in the algorithm.

Pseudocode of Standard DAO [104]

1. Initialization of data centers, host machines, and virtual machines
2. Demands for resource distribution are entered.
3. The demands are divided into several tasks.
4. Attempt to determine the number of VMs and the number of initial tasks to be executed
5. if (it is the end of the process)
6. Then (to balance the load) determine the execution requirements and resource time.
7. else
8. divide the activity between processors while maintaining load
9. Apply DFO for the scheduling of job while maintaining load
10. If (all VMs are active)
11. Then Go to step 4
12. else Go to Step 6

DOA and Its Variations

Constraint Measure Dragonfly optimization (CMDO): Polepally et al. [105] proposed a constraint measure algorithm for load balancing in cloud computing. According to the scheme, the requests from different users are entertained by the VMs present at the data center. These data centers hold physical machines. Each machine holds several VMs. While entertaining the jobs of any user, the VMs follow the constraint measure algorithm. The algorithm first decides among the available VMs and maintains a decision list. After this, a selection list is obtained for all the jobs requested by users. At this point, the dragonfly algorithm is used to produce the optimal maximum values. According to DAO, if the VM's load is higher than the controlled threshold level and the virtual machine has a higher deciding factor, the VM is removed from the decision list and the job with the highest selection factor is allocated to it. The delegated job has been excluded from the list of choices. Similarly, all of the functions are distributed evenly among these VMs.

Resource allocation-based dragonfly optimization algorithm (RDOA): Amini et al. [104] proposed the dragonfly optimization technique to manage resources so that load balancing becomes possible. This technique targets the task scheduling issue in the datacenters of the cloud. Therefore, initially, the datacenter starts the VMs to run the user's request. After that, the cloud environment finds the requests. Each of these requests will be divided into a sequence of jobs to be executed by VMs. After this step, VMs and several tasks will be picked for building or executing. Then, it will be checked to see if the VM's task evaluation has been completed by using the dragonfly optimization technique. If it is completed, then the simulation will stop; otherwise, the DOA will be used to schedule tasks. The proposed algorithm is also compared with other swarm algorithms such as ACO, PSO, and Hybrid ACO_PSO. However, the evaluated experimental results show that the dragonfly algorithm makes more substantial performance increases in scheduling tasks, balancing load, and allocating resources.

Adaptive dragonfly algorithm (ADA): Neelima [106] elaborates the ADA algorithm by highlighting the capability of minimization of execution time and cost. What underlies both of the main focuses is allocating the request to VM by using ADA so that the issue of under or overutilization can be avoided. To exclude this issue, this research proposes the idea of achieving load balancing in cloud computing. The algorithm at the very first instance considers the data center. The data center consists of multiple physical systems. To accomplish any job requested by the user, each physical system used multiple VMs: the load balancer. This load balancer works on the hybrid dragonfly and firefly algorithms. This load balancer firsts evaluate the fitness function by using dragonfly then firefly. After calculations, the load balancer compares the best fitness value between them. The greater fitness value is selected at the end of the iteration. In this manner, the load balancer shifts load to that particular VM by utilizing processing time.

4.2.4. Raven Roosting Optimization Algorithm (RROA)

Ravens usually reach roosts sometime before sunset and depart in highly contemporized groups the next day at sunrise [107]. These ravens initially decide on a space for roosting, which they remained fixed at for the rest of the time. Each raven is then assigned a randomly decided initial position to search for food. This ultimately results in evaluating the fitness value of all the ravens. Among them, the one who is at the best solution at the end of the entire evaluation is declared as a leader. After this decision, a specific number of ravens are selected. These selected ravens leave the roost along with the leader to find the best location present far away. The companion ravens first evaluate the radius of the hemisphere in the last best solution found. After evaluating, they pick any arbitrary point from it.

The jobs to identify suitable virtual machines resemble the social roosting behavior of ravens with respect to following or unfollowing the leader in order to find a huge number of food supply, as shown in Figure 14.

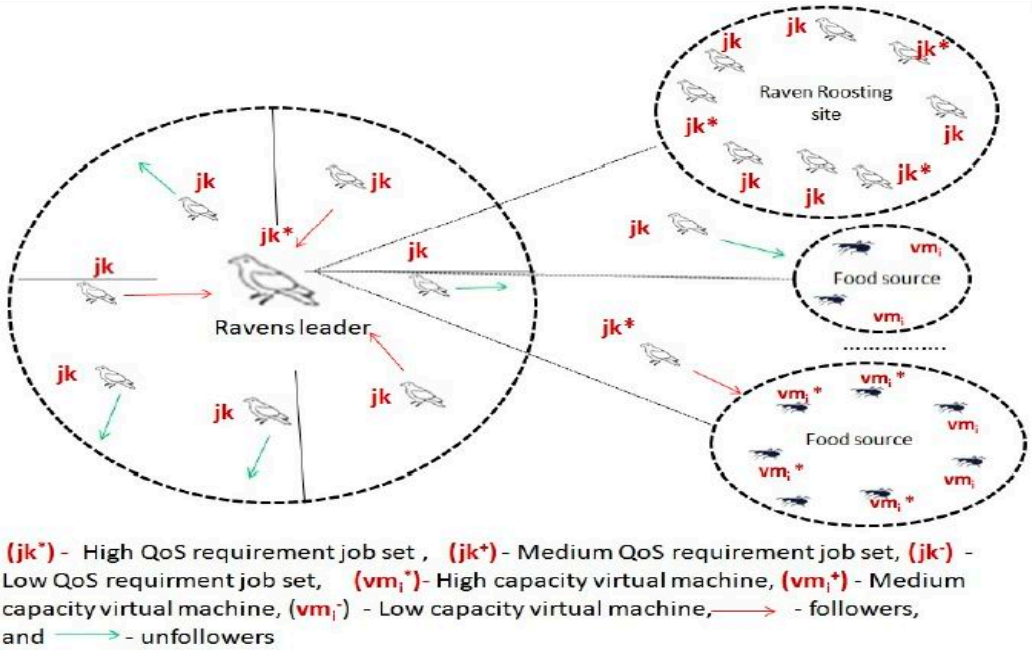

Figure 14. Social roosting behavior of raven roosting load balancing technique [91].

The steps involved are presented in the algorithm.

Pseudocode of Standard RRA [108]
1. Initialization of solutions and load of data centers
2. A proportion of the load is shifted to leave an overloaded VM and find the leading machine.
3. The arrived load searches for a VM for allocation as per the leading machine
4. if (i < the total number of VMs)
5. then
6. Search for the VM among the perception radius
7. Evaluate and reassign the ID
8. else
9. Evaluate and reassign the id if mandatory
10. if (terminating condition not meet)
11. then go to step 1
12. else
13. End the search process

RROA and Its Variations

Basic Raven Roosting Optimization Algorithm (RROA): Rani et al. [109] highlighted the procedure of simple RRA to manage task scheduling and ultimately balances the load in a cloud environment. According to research, birds and insects enroll in roosting amid their diversity. As far as raven roosting is concerned, the targeted problem is solved by the roosts (i.e., main servers). This technique targets finding the capacity of VMs. Based on the capacity and no. tasks assigned to a specific machine, the load is eventually transferred from one VM to another. This allocation of the load is random and will drastically shift the makespan. This algorithm improves the makespan, average response, and average waiting time as well. Above all, the issue with basic RRO is its premature convergence.

Improved Raven Roosting Optimization algorithm (IRROA): Torabi et al. [108] presented IRROA to resolve the premature convergence issue present in the basic RRO algorithm. The proposed algorithm first creates the initial solutions, sets the current amount of food to one, and checks the determination of the leader. After selecting these parameters, the weak and greedy raven will follow the leaders. The leaders continue the search process by using the perception radius in the leader's vicinity. The flight will continue the search process in a certain range possessing the perception radius and evaluate and update the location until or unless they reach the maximum no. of flight steps. If the maximum number is reached, then the last location will evaluate and update if required. The current amount of food is also updated. If the termination is observed, then the algorithm will stop; otherwise, it will continue with finding the current initial position and a leader who repeats the steps. This algorithm improves performance and better exploration.

The algorithm in this manner always assigns load to any VM after considering the recent load available.

Hybrid Raven Roosting Optimization algorithm (Hybrid IRRO): S. Torabi et al. [110] proposed a hybrid technique in which the IRROA is combined with chicken swarm optimization (CSO) to combine the good features of both of them. The CSO algorithm is proposed because of its efficiency in creating a balance between local and global search, while the IRRO algorithm is selected because it solves the issue of premature convergence and performs better in larger complex environments. The involvement of IRROA is emphasized at the start of this hybrid approach. IRROA tests the optimization requirements first, as well as all available resource details. IRROA measures the percentage of approved tasks based on this analysis. This mapping will finally be sent to the virtual machine. The population size and the best solution found thus far are then sent as the input parameter to the chicken swarm algorithm. Then, the CSO algorithm schedules based on the data gathered from the buffers as they are processed in real-time. If there are any tasks in the buffer, CSO will schedule them. In this manner, the VMs that will be sent to the resource manager will have the best possible mapping. In reality, the recommended mapping of jobs to VMs that is sent to the resource manager for ultimate allocation is the global optimal position in the CSO algorithm.

Reinforcement Learning with Raven Roosting Optimization Policy (RROP): Bhargavi et al. [111] considered all the challenges and QoS parameters of balancing load in a cloud environment, and after that, they suggest an RROP algorithm. Via reinforcement learning and the implementation of raven foraging behavior, the potential load balancer will proactively adjust to the complex data center. The effective task completion rate is reported to be significant, while the response time and blocking likelihood of devices are found to be minimal. The model presented here is for a cloud environment with a variety of tasks and a specific number of physical and virtual machines. All tasks are performed by the task scheduler, which divides them into many subtasks and performs the same with the available physical machines, as per the algorithm. Every physical system has Reinforcement Learning with RROA Policy Dependent Load Balancing Agent that manages and allocates the distribution of an incoming subset of tasks among the optimal VMs. Table 5 presents the pros and cons of the discussed modern SI algorithms.

Table 5. Pros and cons of modern algorithms.

Algorithm	Pros	Cons
WOA	Increases the rate at which tasks are completed successfully.	It frequently fails during the initial iterative cycle, resulting in delayed convergence.
SSA	Improves various QoS parameters by adopting the global best match.	As the number of comparable types of VMs expands, the rate of lucrative job execution falls.
DOA	Provides fast convergence to the global optimum solution with high resource utilization.	In the lack of a nearby solution, overall response rates are shown to be ordinary rather than quick and fast.
RROA	Efficiently prevents overloading and underloading.	Beginning iterations have a reduced rate of work completion.

5. Summary of the Reviewed SI Algorithms

Table 6 presents the summary (along with proper references) of the discussed SI based load balancing algorithms along with the main objective, area of application, and targeted issue(s). It summarizes the discussion about GA, PSO, ACO, Grey wolf, ABC, BAT, Whale optimization, SSO, DO, and RR optimization algorithm along with their variations. It presents the evolution of swarm-based algorithms with the main objective to assign load, better response time, distribution of resources, achievement of low time and low-cost complexities, optimize task scheduling, balancing heterogeneous resources, non-preemptive job allocation, etc.; all of these objectives ultimately lead towards balancing the load in cloud-based environments. All this demonstrates that SI based algorithms neither ensure the cloud to be neither one nor decentralized by maintaining migrations in the cloud. These algorithms exploit the capabilities of heterogeneous resources; consider the priority to obtain real-world virtualization by minimizing makespan; increasing throughput; reducing the response, waiting, and processing time; and improving the convergence rate with increased performance. This efficient utilization of resources makes the fitness function improve premature converge, and this fast convergence ultimately satisfies the QoS metrics.

Analysis in terms of time complexity and quality of solution

We observed the following through an extensive study:

- GA has average time complexity and average solution quality, and ABC has the best solution quality despite having the worst time complexity.
- In addition, PSO looks to have the best time complexity, but the quality of its solution is not comparable to that of ABC.
- ACO has a good solution quality and a lower time complexity.
- BAT has the poorest solution quality but surpasses ABC in terms of convergence speed, whereas GWO's convergence speed is more efficient in terms of ABC's solution quality.
- In terms of recent SI algorithms, dragonfly and raven have significantly better performance based on time complexity and solution quality; however, the whale's solution quality is below average with reduced time complexity.
- When compared to WWO, SSOs have an average time complexity and improved solution quality.

Table 6. Summary of the SI based load balancing algorithms for cloud computing.

Algorithm	Authors	Main Objective	Area of Application	Targeted Issue(s)
Genetic algorithm (GA)				
Genetic algorithm	Hussain A Makasarwala; Prasun Hazari, 2017	Load balancing in cloud environments to achieve better response time.	Considers the priority to obtain real-world visualization.	To increase the range of request IDs through permutation encoding. Better response time is achieved.
Genetic Algorithm and Gravitational Emulation	Santanu, Gopa Mandal, Dasgupta, and Paramartha Dutta, 2015	Distribution of load among VMs.	Minimizes the makespan as well as reduces the number of VMs that are going to miss their deadlines	The combination of both strategies reduces the response time of VMs and guaranteed specifications for QoS.
Fuzzy Logic-based Genetic Algorithm	Saadat, A., and Masehian, E, 2020	Load Balancing in Cloud Computing.	Make output scheduling decisions correctly and based on efficient methods.	The objective function enables scheduling the order with the least possible delay.
Multi-Agent Genetic Algorithm	Anant Kumar Jayswal, Prem Chand Saxena, 2020	Efficient Load Balancing in Cloud Computing.	Minimizes makespan time and significantly improves the average throughput performance and resource load balancing on VMs.	Utilizes the resources efficiently and ensures QoS.
Particle Swarm Optimization Algorithm (PSO)				
Particle Swarm Optimization Algorithm	R.M.Al Guliyev, Y. N. Imamverdiyev, F. J. Abdullayeva, 2019	Optimize the load.	Minimizes task execution and transfer time using target functions.	Time optimization for running and using energy.
Hybrid Firefly iPSO Algorithm	Golchi, 2019	Reach the better average load in a cloud environment.	Minimizes the makespan as well as increases the throughput.	Rapid convergence made it more effective and efficient. A high-speed response is achieved.
Binary Particle Swarm Optimization Algorithm	Jean Buanga Mapetu, Zhen and Lingfu Kong, 2019	Achievement of low-time complexity and low cost in load balancing.	Minimizes make-up, waiting for time, and degree of imbalance while minimizing scheduling time, cost of execution, and maximizing the use of resources.	It targets load balancing, cloud resource management task scheduling, and ensures device scalability.
Particle Swarm Optimization Algorithm	Vidya, S. H., and Prakash, R. M. (2020).	Maintaining the load of infrastructure in Cloud.	Reduces the response time and balancing of load.	Exploit the capabilities of heterogeneous resources.
Ant colony optimization Algorithm (ACO)				
Load balancing_ACO Algorithm	Gupta, A., and Garg, R., 2017	Maintain load by task scheduling.	Meet users' demands in terms of execution efficiency and increase resource utilization	Makespan optimization.
Ant colony Optimization Algorithm	Gang Li, Zhijun Wu, 2019	Optimize the load by task scheduling.	Maximize the load handling requirements of the system when efficiently completing the scheduling mission.	Better performance is achieved.

Table 6. *Cont.*

Algorithm	Authors	Main Objective	Area of Application	Targeted Issue(s)
Fuzzy-Ant Colony Algorithm	A. Ragmani, N. Abghour, and M. Rida, 2019	Ensures neither the load is under or over-allocated.	Reduces the response and processing time up to 82% and 90%, respectively.	For handling complex networks.
Ant colony Algorithm with Support Vector Machine	Junaid, M., Sohail, A., Ahmed, A., Baz, A., Khan, I. A., and Alhakami, H, 2020.	Maintaining accuracy in load balancing.	Reduce violations, minimum migration time, high optimization, reduced makespan, and high response time.	Targets scalability and robustness in cloud environments.
Artificial Bee Colony (ABC)				
LBA_Honey Bee	W. Hashem, H. Nashaat, R. Rizk, 2017	Maintaining load in a cloud-based environment.	Propose to avoid under and overutilization.	Reduces the execution time, reaction time, makespan, and load standard deviation.
Artificial Bee Colony	Arif Ullah, Nazri, Jamal Uddin, Samad Baseer, Ansam Hadi Rashed, 2019	Cloud Load Balancing with VM Strategy.	Convergence rates and global search efficiency are validated.	Fast convergence with high flexibility.
Artificial bee foraging optimization	Geetha Muthsamy, Suganthe Ravi Chandran, 2020	Managing the load of distributed systems.	Task preemption to reduce the tasks' response and execution time.	Improves QoS metrics.
Hybrid artificial bee colony Algorithm with multi-objective	Jun-qing, Yun-qi, 2020.	Flexible task Scheduling in cloud computing.	Improves the rate of convergence with increased performance.	Enhances the exploitation process.
Grey Wolf Optimization (GWO)				
Simple GWO	Patel, D., Patra, M. K., and Sahoo, B.,2020	Maintaining load in a cloud-based environment.	Propose resource allocation and load balancing.	Reduces the makespan.
Hybrid GWO_PSO Algorithm	Gohil, B. N., and Patel, D. R., 2018	Cloud Load Balancing with heterogeneous resources.	Convergence near the optimal solution.	Satisfy quality of service constraints.
Fuzzy GWO	Xingjun, L., Zhiwei, S., Hongping, C., and Mohammed, 2020	Managing the load on cloud-based IoT.	Ensures the utilization of resources efficiently.	Reduces response time and improves the degree of imbalancing.
Hybrid GWO_ABC	Soukaina Ouhame, Youssef Hadi, 2020	The resource allocation system is utilizing load balancing.	Improves 1.25% precision and reliability for cloud computing resource allocation systems in VM.	Improves efficiency, energy consumption, and the average time for network execution.
Improved GWO	Gobalakrishnan Natesan, Arun Chokkalingam, 2020	Optimize the Load by task scheduling and resource optimization.	In terms of make-up, cost, and the overall amount of work completed during the deadline, obtain the closest optimum schedule of tasks.	Improves the overall QoS.

Table 6. *Cont.*

Algorithm	Authors	Main Objective	Area of Application	Targeted Issue(s)
		BAT Algorithm		
iBAT Algorithm	Raj, Ranjan, Rizvi, Pranav, and Paul, 2017	Managing the load on cloud-based IoT.	To achieve a desirable population (i.e., better utilization).	Optimized makespan is achieved.
Micro BAT	Youssef, Hamza, Rahhali, Hanine, Benlahmar, Houssine, Mostafa, and Ahmed Eddaoui, 2018	To avoid under and overloading the VMs.	Improve the resource allocation in series or parallel mode.	Load shifting to avoid overloaded or idle VMs.
BAT Algorithm	Arif Ulah, Nazri Mohd, Mubashir Khan, 2020	To improve load in a cloud-based environment.	Select VM optimally.	Satisfy QoS.
IBO Algorithm	Vinothini, Balasubramanie, K. Arvind, 2020	Provide load balancing in a multi-server environment.	Convergence near the optimal solution.	Better performance is achieved.
Hybrid PSO and BAT Algorithm	Valarmathi, R., and Sheela, T., 2017	Optimize task scheduling in the cloud to achieve load balancing.	Fast convergence.	Reduces makespan and improves resource utilization.
		Whale Optimization Algorithm (WOA)		
Chaotic WOA	Kaur G and Arora S, 2018	Balance workload and resource distribution.	Improve the convergence speed.	Reduces energy consumption with better performance.
Hybrid WOA	Strumberger, I., Bacanin, N., Tuba, M., and Tuba, E. (2019)	Load balancing in the cloud environment.	Fast convergence.	The trade between exploitation and exploration in simple WO.
Optimized WOA	FHemasian-Etefagh, F., and Safi-Esfahani, F., 2019	Load balancing in the cloud with improved quality parameters.	Minimize the average execution time, response time, and throughput.	The early convergence issue.
Improved WOA	Xuan Chen, Long Cheng, Cong Liu, Qingzhi Liu, Jinwei Liu, Ying Mao., 2020	Cloud task scheduling	Optimizes the makespan, price, and load.	To boost the optimal solution exploration capability.
		Social Spider Algorithm (SSA)		
Spider Mesh Overlay	Mahato, D. P., and Singh, R. S., 2017	Load balancing pre-emptive jobs.	Improved fitness with fast convergence.	Enhance performance with improved efficiency of cloud-based environments.
Chaotic SSA	Arul Xavier, V. M., and Annadurai, S., 2018	To promote load transfer for effective scheduling.	Effective performance improvements.	To solve job planning problems in the cloud.
SS Cloud Web Algorithm	Preeto Abrola, Savita Gupta, Sukhwinder Singh, 2019	Load balancing non pre-emptive jobs.	Improve QoS requirements.	Create the population of tasks and resources with quality constraints.

Table 6. *Cont.*

Algorithm	Authors	Main Objective	Area of Application	Targeted Issue(s)
Dragonfly Optimization Algorithm (DOA)				
Constraint Measure DOA	Polepally, V., and Shahu Chatrapati, K., 2017	Load balancing in cloud.	Improve Performance.	Delegated job scheduling.
Resource allocation based DOA	Zahra Amini, Mehrdad Maeen, Mohammad Reza Jahangir., 2018	Resource Management to balance the load.	Substantial performance increase in scheduling tasks, balancing load, and allocating resources.	Task scheduling issues in data centers.
Adaptive DOA	Neelima, P., and Reddy, A. R. M., 2020	To avoid under and overutilization.	Optimal utilization of processing time.	The capability of minimization of execution time and cost.
Raven Roosting Algorithm (RRO)				
Basic RRO	E. Rani and H. Kaur., 2017	Efficient task scheduling.	Improves makespan, average response, and waiting time.	To find the capacity of VMs.
Improved RRO	Torabi, S., & Safi-Esfahani, F., 2018	Effective load scheduling.	Improve performance, avg. response, and waiting time.	To resolve premature convergence.
Hybrid IRRO	S. Torabi and F. Safi-Esfahani, 2018	Improve the allocation of tasks.	The optimal solution for load balancing.	To overcome premature convergence and lacking performance.
Reinforcement Learning with RRO Policy	Bhargavi, K., and Babu, B. S., 2019	Load balancing in cloud	Proactive adjustment to the complex data center.	All the challenges and QoS parameters.

6. Comparative Performance Analysis of Various SI Algorithms Used for Load Balancing

This part consists of two types of comparative analysis that have been surveyed. In the Section 1, performance analysis is presented with respect to some discussed algorithms, while the Section 2 is composed of some quality parameters achieved by the discussed algorithms.

6.1. Comparative Performance Analysis of SI Algorithms in Cloud Computing

6.1.1. Comparative Performance Analysis of Average Response Time (TRT) for Different User Bases (UB) and Data Centers (DC)

Tables 7–10 present the comparative performance analysis among GA, ant colony, and particle swarm optimization techniques used for load balancing based on average response time for different UBs and DCs. This comparative study is supported by the graphs presented in [112].

Table 7. Performance analysis of avg. TRT for 10, 20, 30, and 50 UB with 5 DC.

Algorithms	Avg. TRT for 10 UB (ms)	Avg. TRT for 20 UB (ms)	Avg. TRT for 30 UB (ms)	Avg. TRT for 50 UB (ms)
GA	48.4	49.2	52.3	56.4
ACO	46.7	49	52.6	58.2
PSO	45.6	47.2	50.3	53.2

Table 8. Performance analysis of avg. TRT for 10, 20, 30, and 50 UB with 10 DC.

Algorithms	Avg. TRT for 10 UB (ms)	Avg. TRT for 20 UB (ms)	Avg. TRT for 30 UB (ms)	Avg. TRT for 50 UB (ms)
GA	45.6	46	48.4	52.8
ACO	44.6	46.9	49	52.8
PSO	43.2	45.2	47.8	50.8

Table 9. Performance analysis of avg. TRT for 10, 20, 30, and 50 UB with 15 DC.

Algorithms	Avg. TRT for 10 UB (ms)	Avg. TRT for 20 UB (ms)	Avg. TRT for 30 UB (ms)	Avg. TRT for 50 UB (ms)
GA	42.1	43.2	45.8	47.9
ACO	41.2	42.6	44	46.2
PSO	40.2	42.2	44	47.6

Table 10. Performance analysis of avg. TRT for 10, 20, 30, and 50 UB with 20 DC.

Algorithms	Avg. TRT for 10 UB (ms)	Avg. TRT for 20 UB (ms)	Avg. TRT for 30 UB (ms)	Avg. TRT for 50 UB (ms)
GA	39.6	41.3	43.9	46.4
ACO	38.7	39.8	41.2	43.9
PSO	37.8	38.4	39.8	41.2

Table 7 compares the TRT for 10, 20, 30, and 50 UB with five DC. It can be concluded that PSO provides reduced average TRT than compared to GA and ACO. Table 8 increases DC to 10. PSO's performance for reduced average TRT in this particular condition is still the best than compared to GA and ACO. In the case of 15 DC (Table 9), the average TRT of GA is the greatest than compared to ACO and PSO. The same trend is observed in Table 10.

If we make comparisons, then PSO's response time is better than GA and ACO. After the comparative results obtained from Table 5, it can be concluded that the response time of PSO is approximately 4.5% better than ACO and GA.

As far as overall progress is concerned, all algorithms show promising results with 20 Data Centers than compared to five DCs.

6.1.2. Comparative Performance Analysis of Average Data Center Processing Time (DCPT) for Different User Bases (UB) and Data Centers (DC)

Tables 11–14 present the comparative performance analysis among GA, ant colony, and particle swarm optimization techniques used for load balancing based on average data center processing time for different UBs and DCs.

Table 11. Performance analysis of avg. DCPT for 10, 20, 30, and 50 UB with 5 DC.

Algorithms	Avg. DCPT for 10 UB (ms)	Avg. DCPT for 20 UB (ms)	Avg. DCPT for 30 UB (ms)	Avg. DCPT for 50 UB (ms)
GA	6.4	6.1	6	5.6
ACO	5.6	5.2	5.1	4.8
PSO	5.9	5.1	4.9	4.6

Table 12. Performance analysis of avg. DCPT for 10, 20, 30, and 50 UB with 10 DC.

Algorithms	Avg. DCPT for 10 UB (ms)	Avg. DCPT for 20 UB (ms)	Avg. DCPT for 30 UB (ms)	Avg. DCPT for 50 UB (ms)
GA	5.8	5.6	5.5	5.2
ACO	5.2	5	4.8	4.6
PSO	5.4	4.8	4.6	4.3

Table 13. Performance analysis of avg. DCPT for 10, 20, 30, and 50 UB with 15 DC.

Algorithms	Avg. DCPT for 10 UB (ms)	Avg. DCPT for 20 UB (ms)	Avg. DCPT for 30 UB (ms)	Avg. DCPT for 50 UB (ms)
GA	5.6	5.3	5.1	4.7
ACO	5	4.8	4.7	4.4
PSO	5.1	4.6	4.5	4.2

Table 14. Performance analysis of avg. DCPT for 10, 20, 30, and 50 UB with 20 DC.

Algorithms	Avg. DCPT for 10 UB (ms)	Avg. DCPT for 20 UB (ms)	Avg. DCPT for 30 UB (ms)	Avg. DCPT for 50 UB (ms)
GA	5.3	5.1	4.9	4.5
ACO	4.8	4.6	4.5	4.2
PSO	4.9	4.5	4.3	4

The average data center processing time (DCPT) for various UBs is shown in Table 11 with five DC. The average DCPT of GA is the greatest than compared to ACO and PSO, which ultimately shows that ACO has the least average DCPT. The results for 10, 15, and 20 DC are shown in Tables 12–14 with the same observation.

The comparative result analysis from Tables 9–12 showed that the DCPT of GA is higher than ACO and PSO. It can be concluded that as the number of user bases increased from 10 to 50, the lowest DCPT is produced by PSO than compared to GA and ACO.

As far as overall progress is concerned, all algorithms show slightly better results with the increased numbers of UBs.

6.1.3. Comparative Performance Analysis Based on Time and Cost to Complete Tasks

Tables 15 and 16 present the comparative performance analysis among GA, ant colony, and particle swarm optimization techniques used for load balancing based on average data center processing time for different UBs and DCs.

Table 15. Approximate comparison of time to complete tasks.

No of Tasks	3	4	5	6
GA (time)	2.1	3	5.75	6
PSO (time)	1.5	2.5	5	5.5

Table 16. Approximate comparison of cost to complete tasks.

No of Tasks	3	4	5	6
GA (cost)	20	42	90	112
PSO (cost)	10	25	80	100

It is inferred from the table that no matter how much the number of tasks increased, PSO is significantly better than GA. As the number of tasks increases, the cost of PSO and GA also increases; but PSO's cost is considerably lower than GA.

Comparative result analysis from Tables 13 and 14 showed that the PSO is better than compared to GA based on time and cost. It consumes less time and has less cost.

By utilizing comparative studies, we have analyzed that the GA experiences the highest DCPT. Another key observation is for DCPT, which shows that when DC is set to five, ACO achieves higher DCPT, and PSO produces the very same DCPT. It has been discovered that GA achieves the highest DCPT.

6.1.4. Comparative Performance Analysis Based on Makespan

Table 17 presents the comparative analysis of numerous algorithms based on quality metrics that are used to evaluate the performance of algorithms. The key parameters are defined below:

Response Time (RT): This is the time that the device takes to respond to the request of the client. To provide good user experiences, a quick response is preferable.

Throughput (T): This is the rate at which requests from customers are handled for processing.

Makespan (MS): This is the amount of time needed to process the specified collection of tasks.

Energy conservation (EC): This illustrates the reliability and efficacy of the use of electrical resources for various data center services, e.g., providing the desirable power to servers and cooling systems.

Scalability (S): This reflects the ability of the SI algorithm to manage the rising demands of the user effectively.

Resource utilization (RU): In the cloud datacenter, it evaluates the amount of resource utilization of computing resources.

In Table 17, the tick mark indicates that the mentioned SI algorithm for load balancing enhances the parameter of the corresponding load balance effect assessment.

Table 17. Comparative analysis based on improved quality parameters.

Algorithms (with Reference)	RT	T	MS	EC	S	RU
GA (2017, [18,25])	✗	√	√	✗	√	✗
IGA (2017, [44])	✗	√	√	✗	✗	√
Fuzzy based GA (2020, [46])	√	√	√	✗	✗	√
Multi Agent GA (2020, [113])	✗	√	√	✗	✗	√
PSO (2019, [55])	√	✗	✗	✗	✗	√
IPSO (2019, [58])	√	✗	√	√	✗	√
Hybrid PSO (2017, [114])	√	✗	√	✗	✗	√
ACO (2015, [63])	✗	✗	✗	√	√	✗
Improved ACO (2016, [65])	✗	✗	✗	√	√	√
LBA_Honey Bee (2017, [74])	√	√	√	✗	✗	✗
Enhanced Bee colony (2016, [73])	✗	✗	√	✗	✗	√
ABC (2018, [20])	✗	✗	√	✗	✗	✗
Improved ABC (2020, [71])	√	✗	√	✗	✗	√
Fuzzy GWO (2017, [82])	✗	✗	√	✗	✗	√
GWO (2020, [81])	√	✗	√	✗	✗	✗
Improved GWO (2020, [80])	√	√	√	✗	✗	√
BAT (2016, [84])	√	✗	✗	✗	✗	√
iBAT (2017, [62])	✗	✗	√	✗	✗	√
microBAT (2018, [90])	✗	✗	✗	✗	√	√
Improved BAT (2020, [115])	√	√	✗	✗	✗	√
Chaotic WOA (2018, [94])	√	✗	✗	✗	√	✗
Hybrid WOA (2019, [93])	√	✗	✗	✗	√	✗
Optimized WOA (2019, [95])	√	√	✗	√	√	✗
Improved WOA (2020, [96])	√	√	√	√	√	✗
Spider Mesh Overlay (2017, [99])	✗	✗	✗	√	√	√
Chaotic SSA (2018, [101])	✗	✗	✗	√	√	√
SS Cloud Web Algorithm (2019, [102])	√	✗	✗	√	√	√
Constraint Measure DOA (2017, [105])	√	√	✗	√	√	√
Resource allocation based DOA (2018, [104])	√	✗	✗	√	√	√
Adaptive DOA (2020, [106])	√	✗	✗	√	√	√
Basic RRO (2017, [109])	√	√	√	✗	✗	√
Improved RRO (2018, [108])	√	√	√	√	✗	√
Hybrid IRRO (2017, [110])	√	√	✗	✗	√	√
Reinforcement Learning with RRO Policy (2019, [111])	√	√	√	✗	✗	√

The improved GA targets throughput, makespan, and availability, while the improved versions of GA also target improved response time and optimized resource utilization. PSO algorithms focus on reducing response time with efficient resource utilization while the hybrid and improved approaches of PSO facilitate the improvement of makespan and energy conservation. Simple ACO targets the performance parameters of energy conservation and scalability, while the improved hybridized variations of ACO also facilitate efficient resource utilization. Artificial honey bee incorporates the reduction in response time, throughput, and makespan while its improved versions also facilitate efficient utilization of available resources. Basic GWO further provides load balancing with reduced response time with better scalability while their improved versions also target at reducing the response time with better resource utilization. The simple BAT algorithm for load balancing reduces the response time with better utilization of resources but the proposed improved variations not only support resource optimization but also facilitate makespan with high scalability. The Modern Chaotic Wave algorithm is introduced to support reduced response time with better scalability while the improved and optimized variations of whale optimization support reduced response time, better throughput, makespan, conservation of energy, and better scalability. Spider Mesh Overlay supports load balancing by providing better energy conservation, scalability, and optimized resource utilization. Dragonfly optimization is another swarm-based load balancing technique that targets at improving scalability, energy conservation with optimized resources, and reduced response time.

The basic raven roosting algorithm focuses on improving the performance parameters of response time, throughput, makespan, and resource utilization, while its improved version further provides better energy conservation with improved scalability.

From the above comparison, it is concluded that the overall performance of the dragonfly optimization algorithm and raven roosting algorithm is better than the other surveyed algorithms. They improve scalability, energy conservation with optimized resources, and reduced response time.

7. Future Directions

It is inferred from the analysis performed during this research process that a range of problems is still open in the process of load balancing [116]. In this review paper, we have discussed various SI load balancing algorithms along with certain variations. It is surveyed that these algorithms contribute to improving quality parameters and QoS in cloud computing. Despite having many advantages, there are some loopholes as well, such as inadequate frequency regulation, power loss, slow convergence rate, complexity, low efficiency, no accurate method to estimate execution time, throughput being low in dynamic load balancing, etc. Table 18 indicates the research challenges that are faced in load balancing.

Table 18. Key challenges.

Sr. No.	Challenges
1	The backup program even system does not fail completely
2	Maintenance of system regularly
3	Resources must be used competently under load conditions

In the future, they can be resolved by applying some innovative and advanced load balancing algorithms, particularly along with additional QoS metrics and algorithm complexity assessment dimensions [117]. As the emergence of cloud computing deals with more and more data, the swarm load balancing algorithms also need to evolve in the fields of machine learning, artificial intelligence, IoT, blockchain, etc. Furthermore, in most of the review techniques, some important cloud computing factors such as security, cost of service, storage space, and carbon emission were not considered, which are more qualitative and quantitative attributes related to selecting the suitable service. Maintaining the network' self-organization as everything is adapting to the concept of the Internet of things and managing the load efficiently by keeping all the quality parameters in view can be other future research possibilities. Therefore, there is an absolute demand for techniques that efficiently deal with the complex nature of load without bypassing any of the quality parameters.

8. Conclusions

This research paper emphasizes the dynamic solution of one of the most highlighted challenges of cloud computing, i.e., load balancing. Cloud computing is something that we all use the entire day without realizing. This tremendous increase in use adds exponential load relative to the cloud due to which its performance can suffer.

LB's primary objective is to fulfill user needs by spreading the workload across several network nodes and optimizing the usage of resources and increasing the performance of devices. To overcome this challenge, a comprehensive survey is presented, focusing on some traditional and modern techniques of SI based algorithms for load balancing in the cloud. These techniques include GA, PSO, ACO, BAT, ABC, GWO, WOA, RRO, DO, and SSO and their variants for performing load balancing in cloud computing more efficiently. A comparative analysis based on the performance and quality parameters of the algorithms surveyed is also provided. It is noted that the algorithms surveyed usually work to boost QoS, the response time, utilization of resources, throughput, makespan, scalability, and fast convergence.

Apart from all of the improved advantages, these algorithms have some discrepancies as well, such as resource and energy overutilization, insufficient control rates, and static thresholds. Therefore, in the future, more enhancements are required in the field of load balancing by using swarm intelligence algorithms to boost the quality. Close comparisons are performed on the surveyed algorithms. The actual quality of the raven roosting and dragonfly algorithms is found to be higher than the other implementations after evaluating all of the surveyed techniques. It has been discovered by the observation that both dragonfly and raven roosting algorithms perform load balancing more effectively than other approaches. Although some of the strategies are ineffective with mediocre results, others are not worse. The function and application are completely reliant on the cloud environment and quality-control criteria.

This survey considers the most crucial consideration for supplying cloud services adequately, which is load balancing. Therefore, this survey paper provides a proper breakdown of SI based cloud load balancing algorithms and the issues these algorithms pose when applied in a cloud context in this research study. The entire set of algorithms investigated in this study has been presented with pros and cons. As a result, the algorithms still have the potential for development. As a result, the plan is to develop our swarm intelligence-based methodology in the near future in order to confront the concerns that have been raised and discussed. This survey will also make it easier for investigators to compare quality characteristics and contribute to this critical area of attention, which will more effectively help to balance the load in cloud infrastructure by using more optimized state-of-the-art techniques.

Due to the tremendous benefits in practical problems [118], several industries are adopting dynamic load balancing techniques:

- For software or hardware maintenance, the complete network does not need to be taken down or offline;
- It can be performed on one server at a time while other servers' services are still functioning;
- When more storage or processing capability is needed, businesses may simply ask their service provider to swiftly and seamlessly deploy additional servers to the connection;
- In the case of a catastrophe at the primary data center, distributed load balancing can provide disaster restoration services by diverting user connection requests in a disaster recovery data center;
- Massive enterprises with resource-intensive services, large amounts of constantly expanding data, and continual traffic want network connectivity, reliability, and flexibility in order to ensure that consumers can access their services or products at any location worldwide.

Author Contributions: Conceptualization, M.A.E. and D.S.; methodology, M.A.E. and A.S.; validation, N.I., A.A. and S.R.; formal analysis, A.S. and A.A.; investigation, M.A.E. and S.R.; resources, D.S. and S.R.; data curation, A.S. and N.I.; writing—original draft preparation, M.A.E. and A.S.; writing—review and editing, S.R., N.I. and A.A.; visualization, S.R. and A.A.; supervision, A.S.; project administration, S.R.; funding acquisition, M.A.E. All authors have read and agreed to the published version of the manuscript.

Funding: The authors express their gratitude to the ministry of education and the Deanship of Scientific Research of Najran University, Kingdom of Saudi Arabia, for financial and technical support under code number NU/-/SERC/10/555.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. DeStefano, T.; Kneller, R.; Timmis, J. *Cloud Computing and Firm Growth*; 2020; p. 8306. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3618829 (accessed on 15 July 2021).
2. Chen, Y.; Li, X.; Chen, F. Overview and analysis of cloud computing research and application. In Proceedings of the 2011 International Conference on E-Business and E-Government (ICEE), Shanghai, China, 6–8 May 2011.

3. Shahid, M.A.; Islam, N.; Alam, M.M.; Mazliham, M.S.; Musa, S. Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment. *Comput. Sci. Rev.* **2021**, *40*, 100398. [CrossRef]
4. Langmead, B.; Nellore, A. Cloud computing for genomic data analysis and collaboration. *Nat. Rev. Genet.* **2018**, *19*, 208–219. [CrossRef] [PubMed]
5. Velte, A.T.; Velte, T.J.; Elsenpeter, R. Cloud Computing: A Practical Approach. *ISSN* **2019**, *2278*, 0181.
6. Kumar, P.; Kumar, R. Issues and challenges of load balancing techniques in cloud computing: A survey. *ACM Comput. Surv. (CSUR)* **2019**, *51*, 1–35. [CrossRef]
7. Gutierrez-Garcia, J.O.; Ramirez-Nafarrate, A. Collaborative agents for distributed load management in cloud data centers using live migration of virtual machines. *IEEE Trans. Serv. Comput.* **2015**, *8*, 916–929. [CrossRef]
8. Chen, S.-L.; Chen, Y.-Y.; Kuo, S.-H. CLB: A novel load balancing architecture and algorithm for cloud services. *Comput. Electr. Eng.* **2017**, *58*, 154–160. [CrossRef]
9. Kaur, K.; Kumar, Y. Swarm Intelligence and its applications towards Various Computing: A Systematic Review. In Proceedings of the 2020 International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 17–19 June 2020.
10. Princess, G.A.P.; Radhamani, A. A Hybrid Meta-Heuristic for Optimal Load Balancing in Cloud Computing. *J. Grid Comput.* **2021**, *19*, 1–22.
11. Chien, N.K.; Son, N.H.; Loc, H.D. Load balancing algorithm based on estimating finish time of services in cloud computing. In Proceedings of the 2016 18th International Conference on Advanced Communication Technology (ICACT), Pyeongchang, Korea, 31 January–3 February 2016.
12. Xavier, M.G.; Neves, M.V.; Rossi, F.D.; Ferreto, T.C.; Lange, T.; Rose, C.A.F. Performance evaluation of container-based virtualization for high performance computing environments. In Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Belfast, UK, 27 February–1 March 2013.
13. Soltesz, S.; Pötzl, H.; Fiuczynski, M.E.; Bavier, A.; Peterson, L. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, Lisbon, Portugal, 21–23 March 2007.
14. Santos, J.; Wauters, T.; Volckaert, B.; Truck, F.D. Towards network-aware resource provisioning in Kubernetes for fog computing applications. In Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 24–28 June 2019.
15. Fazio, M.; Celesti, A.; Ranjan, R.; Liu, C.; Chen, L.; Villari, M. Open issues in scheduling microservices in the cloud. *IEEE Cloud Comput.* **2016**, *3*, 81–88. [CrossRef]
16. Burns, B.; Beda, J.; Hightower, K. *Kubernetes: Up and Running: Dive into the Future of Infrastructure*; O'Reilly Media: Sebastopol, CA, USA, 2019.
17. Mishra, S.K.; Sahoo, B.; Parida, P.P. Load balancing in cloud computing: A big picture. *J. King Saud Univ.-Comput. Inf. Sci.* **2020**, *32*, 149–158. [CrossRef]
18. Makasarwala, H.A.; Hazari, P. Using genetic algorithm for load balancing in cloud computing. In Proceedings of the 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Ploiesti, Romania, 30 June–2 July 2016.
19. Elmagzoub, M.A.; Shaikh, A.; Alghamdi, A.; Rajab, K. A review on MIMO wireless signals over fibre for next generation fibre wireless (FiWi) broadband networks. *Electronics* **2010**, *9*, 2014. [CrossRef]
20. Mosa, M.A.; Anwar, A.S.; Hamouda, A. A survey of multiple types of text summarization with their satellite contents based on swarm intelligence optimization algorithms. *Knowl.-Based Syst.* **2015**, *163*, 518–532. [CrossRef]
21. Junaid, M.; Sohail, A.; Ahmed, A.; Baz, A.; Khan, I.A.; Alhakami, H. A hybrid model for load balancing in cloud using file type formatting. *IEEE Access* **2020**, *8*, 118135–118155. [CrossRef]
22. Gźsior, J.; Seredyński, F. Decentralized Job Scheduling In The Cloud Based On A Spatially Generalized Prisoner's Dilemma Game. *Int. J. Appl. Math. Comput. Sci.* **2015**, *25*, 737–751. [CrossRef]
23. Kennedy, J. Swarm Intelligence. In *Handbook of Nature-Inspired and Innovative Computing*; Springer: Boston, MA, USA, 2006; Volume 1, pp. 187–219. ISBN 978-0387-27705-9.
24. Tan, Y.; Shi, Y.; Tuba, M. *Advances in Swarm Intelligence, In Proceedings of the 11th International Conference ICSI 2020, Belgrade, Serbia, 14–20 July 2020*; Springer Nature: Basingstoke, UK, 2020; Volume 12145.
25. Fahad, M.; Aadil, F.; Khan, S.; Shah, P.A.; Muhammad, K.; Lloret, J.; Wang, H.; Lee, J.W.; Mehmood, I. Grey wolf optimization based clustering algorithm for vehicular ad-hoc networks. *Comput. Electr. Eng.* **2018**, *70*, 853–870. [CrossRef]
26. Sun, W.; Tang, M.; Zhang, L.; Huo, Z.; Shu, L. A survey of using swarm intelligence algorithms in IoT. *Sensors* **2020**, *20*, 1420. [CrossRef] [PubMed]
27. Xu, M.; Tian, W.; Buyya, R. A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurr. Comput. Pract. Exp.* **2017**, *29*, e4123. [CrossRef]
28. Chakraborty, T.; Datta, S.K. Application of swarm intelligence in internet of things. In Proceedings of the 2017 IEEE International Symposium on Consumer Electronics (ISCE), Kuala Lumpur, Malaysia, 14–15 November 2017.
29. Houssein, E.H.; Gad, A.G.; Wazery, Y.M.; Suganthan, P.N. Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends. *Swarm Evol. Comput.* **2021**, *62*, 100841. [CrossRef]
30. Ojha, V.K.; Abraham, A.; Snášel, V. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Eng. Appl. Artif. Intell.* **2017**, *60*, 97–116. [CrossRef]

31. Zhang, B.; Lin, C.; Huo, L.; Wang, Z.; Chan, C.K. A simple high-speed WDM PON utilizing a centralized supercontinuum broadband light source for colorless ONUs. In Proceedings of the 2006 Optical Fiber Communication Conference and the National Fiber Optic Engineers Conference, Anaheim, CA, USA, 5–10 March 2006.
32. Buyya, R.; Srirama, S.N.; Casale, G.; Calheiros, R.; Simmhan, Y.; Varghese, B.; Gelenbe, E.; Javadi, B.; Vaquero, L.M.; Netto, M.A.; et al. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–38. [CrossRef]
33. Ebadifard, F.; Babamir, S.M. Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. *Clust. Comput.* **2021**, *24*, 1075–1101. [CrossRef]
34. Singh, S.; Chana, I. A survey on resource scheduling in cloud computing: Issues and challenges. *J. Grid Comput.* **2016**, *14*, 217–264. [CrossRef]
35. Vakili, A.; Navimipour, N.J. Comprehensive and systematic review of the service composition mechanisms in the cloud environments. *J. Netw. Comput. Appl.* **2017**, *81*, 24–36. [CrossRef]
36. Hota, A.; Mohapatra, S.; Mohanty, S. Survey of different load balancing approach-based algorithms in cloud computing: A comprehensive review. *Comput. Intell. Data Min.* **2019**, *711*, 99–110.
37. Jyoti, A.; Shrimali, M.; Tiwari, S.; Singh, H.P. Cloud computing using load balancing and service broker policy for IT service: A taxonomy and survey. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 4785–4814. [CrossRef]
38. Ghomi, E.J.; Rahmani, A.M.; Qader, N.N. Load-balancing algorithms in cloud computing: A survey. *J. Netw. Comput. Appl.* **2017**, *88*, 50–71. [CrossRef]
39. Sa, P.K.; Sahoo, M.N.; Murugappan, M.; Wu, Y.; Majhi, B. (Eds.) *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications: Proceedings of ICACNI 2016, Volume 2*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 719.
40. Kabir, M.S.; Kabir, K.M.; Islam, D.R. Process of load balancing in cloud computing using genetic algorithm. *Electr. Comput. Eng. Int. J. (ECIJ)* **2015**, *4*, 57–65. [CrossRef]
41. Shafiq, D.A.; Jhanjhi, N.; Abdullah, A. vLoad balancing techniques in cloud computing environment: A review. *J. King Saud Univ. -Comput. Inf. Sci.* **2021**. [CrossRef]
42. Miao, Z.; Yong, P.; Mei, Y.; Quanjun, Y.; Xu, X. A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment. *Future Gener. Comput. Syst.* **2021**, *115*, 497–516. [CrossRef]
43. Dam, S.; Mandal, G.; Dasgupta, K.; Dutta, P. Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing. In Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT), Hooghly, India, 7–8 February 2015.
44. Kaur, S.; Sengupta, J. Load balancing using improved genetic algorithm (iga) in cloud computing. *Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET)* **2017**, *6*, 2278-1123.
45. Basu, S.; Kannayaram, G.; Ramasubbareddy, S.; Venkatasubbaiah, C. Improved Genetic Algorithm for Monitoring of Virtual Machines in Cloud Environment. In *Smart Intelligent Computing and Applications*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 319–326.
46. Saadat, A.; Masehian, E. Load Balancing in Cloud Computing Using Genetic Algorithm and Fuzzy Logic. In Proceedings of the 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 5–7 December 2019.
47. Goar, V.; Kuri, M.; Kumar, R.; Senjyu, T. *Advances in Information Communication Technology and Computing*; Springer: Berlin/Heidelberg, Germany, 2021.
48. Jafarnejad Ghomi, E.; Masoud Rahmani, A.; Nasih Qader, N. Service load balancing, task scheduling and transportation optimisation in cloud manufacturing by applying queuing system. *Enterp. Inf. Syst.* **2019**, *13*, 865–894. [CrossRef]
49. Vidya, S.H.; Prakash, R.M. Response time analysis of dynamic load balancing algorithms in Cloud Computing. In Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 27–28 July 2020.
50. Dashti, S.E.; Rahmani, A.M. Dynamic VMs placement for energy efficiency by PSO in cloud computing. *J. Exp. Theor. Artif. Intell.* **2016**, *28*, 97–112. [CrossRef]
51. Mapetu, J.P.B.; Chen, Z.; Kong, L. Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing. *Appl. Intell.* **2019**, *49*, 3308–3330. [CrossRef]
52. Ebadifard, F.; Babamir, S.M. A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. *Concurr. Comput. Pract. Exp.* **2018**, *30*, e4368. [CrossRef]
53. Singh, A.B.; Bhat, S.; Raju, R.; D'Souza, R. Survey on various load balancing techniques in cloud computing. *Adv. Comput.* **2017**, *7*, 28–34.
54. Gaidhane, P.J.; Nigam, M.J. A hybrid grey wolf optimizer and artificial bee colony algorithm for enhancing the performance of complex systems. *J. Comput. Sci.* **2018**, *27*, 284–302. [CrossRef]
55. Alguliyev, R.M.; Imamverdiyev, Y.N.; Abdullayeva, F.J. PSO-based load balancing method in cloud computing. *Autom. Control. Comput. Sci.* **2019**, *53*, 45–55. [CrossRef]
56. Agarwal, R.; Baghel, N.; Khan, M.A. Load balancing in cloud computing using mutation based particle swarm optimization. In Proceedings of the 2020 International Conference on Contemporary Computing and Applications (IC3A), Lucknow, India, 5–7 February 2020.

57. Jordehi, A.R.; Jasni, J. Particle swarm optimisation for discrete optimisation problems: A review. *Artif. Intell. Rev.* **2015**, *43*, 243–258. [CrossRef]
58. Golchi, M.M.; Saraeian, S.; Heydari, M. A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation. *Comput. Netw.* **2019**, *162*, 106860. [CrossRef]
59. Pan, K.; Chen, J. Load balancing in cloud computing environment based on an improved particle swarm optimization. In Proceedings of the 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 September 2015.
60. Ghumman, N.S.; Kaur, R. Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system. In Proceedings of the 2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Dallas-Fortworth, TX, USA, 13–15 July 2015.
61. Gao, R.; Wu, J. Dynamic load balancing strategy for cloud computing with ant colony optimization. *Future Internet* **2015**, *7*, 465–483. [CrossRef]
62. Raj, B.; Ranjan, P.; Rizvi, N.; Pranav, P.; Paul, S. Improvised Bat Algorithm for Load Balancing-Based Task Scheduling. In *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 521–530.
63. Li, G.; Wu, Z. Ant colony optimization task scheduling algorithm for SWIM based on load balancing. *Future Internet* **2019**, *11*, 90. [CrossRef]
64. Gupta, A.; Garg, R. Load balancing based task scheduling with ACO in cloud computing. In Proceedings of the 2017 International Conference on Computer and Applications (ICCA), Doha, Qatar, 6–7 September 2017.
65. Ragmani, A.; Elomri, A.; Abghour, N.; Moussaid, K.; Rida, M. An improved hybrid fuzzy-ant colony algorithm applied to load balancing in cloud computing environment. *Procedia Comput. Sci.* **2019**, *151*, 519–526. [CrossRef]
66. Pourghaffari, A.; Barari, M.; Sedighian Kashi, S. An efficient method for allocating resources in a cloud computing environment with a load balancing approach. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e5285. [CrossRef]
67. Tang, L.; Pan, J.S.; Hu, Y.; Ren, P.; Tian, Y.; Zhao, H. A novel load balance algorithm for cloud computing. In *International Conference on Genetic and Evolutionary Computing*; Springer: Berlin/Heidelberg, Germany, 2015.
68. Kumar, A.; Kumar, D.; Jarial, S. A review on artificial bee colony algorithms and their applications to data clustering. *Cybern. Inf. Technol.* **2017**, *17*, 3–28. [CrossRef]
69. Rajan, C.; Geetha, K.; Priya, C.R.; Sasikala, R. Investigation on bio-inspired population based metaheuristic algorithms for optimization problems in ad hoc networks. *Int. J. Math. Comput. Phys. Electr. Comput. Eng.* **2015**, *9*, 163–170.
70. Pruitt, J.N.; Avilés, L. Social spiders: Mildly successful social animals with much untapped research potential. *Anim. Behav.* **2018**, *143*, 155–165. [CrossRef]
71. Muthsamy, G.; Ravi Chandran, S. Task scheduling using artificial bee foraging optimization for load balancing in cloud data centers. *Comput. Appl. Eng. Educ.* **2020**, *28*, 769–778. [CrossRef]
72. Li, J.-Q.; Han, Y.-Q. A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. *Clust. Comput.* **2020**, *23*, 2483–2499. [CrossRef]
73. Babu, K.R.; Samuel, P. Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud. In *Innovations in Bio-Inspired Computing and Applications*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 67–78.
74. Hashem, W.; Nashaat, H.; Rizk, R. Honey bee based load balancing in cloud computing. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 5694–5711.
75. Patel, D.; Patra, M.K.; Sahoo, B. GWO Based Task Allocation for Load Balancing in Containerized Cloud. In Proceedings of the 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–28 February 2020.
76. Abed-alguni, B.H.; Alawad, N.A. Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments. *Appl. Soft Comput.* **2021**, *102*, 107113. [CrossRef]
77. Mousavi, S.; Mosavi, A.; Varkonyi-Koczy, A.R.; Fazekas, G. Dynamic resource allocation in cloud computing. *Acta Polytech. Hung.* **2017**, *14*, 83–104.
78. Faris, H.; Aljarah, I.; Al-Betar, M.A.; Mirjalili, S. Grey wolf optimizer: A review of recent variants and applications. *Neural Comput. Appl.* **2018**, *30*, 413–435. [CrossRef]
79. Niu, P.; Niu, S.; Chang, L. The defect of the Grey Wolf optimization algorithm and its verification method. *Knowl.-Based Syst.* **2019**, *171*, 37–43. [CrossRef]
80. Natesan, G.; Chokkalingam, A. An improved grey wolf optimization algorithm based task scheduling in cloud computing environment. *Int. Arab. J. Inf. Technol.* **2020**, *17*, 73–81. [CrossRef]
81. Gohil, B.N.; Patel, D.R. A hybrid GWO-PSO algorithm for load balancing in cloud computing environment. In Proceedings of the 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT), Bangalore, India, 16–18 August 2018.
82. Xingjun, L.; Zhiwei, S.; Hongping, C.; Mohammed, B.O. A new fuzzy-based method for load balancing in the cloud-based Internet of things using a grey wolf optimization algorithm. *Int. J. Commun. Syst.* **2020**, *33*, e4370. [CrossRef]
83. Ouhame, S.; Hadi, Y.; Arifullah, A. A hybrid grey wolf optimizer and artificial bee colony algorithm used for improvement in resource allocation system for cloud technology. *Int. J. Online Biomed. Eng.* **2020**, *16*, 4–17. [CrossRef]

84. Chételat, J.; Hickey, M.B.C.; Poulain, A.J.; Dastoor, A.; Ryjkov, A.; McAlpine, D.; Vanderwolf, K.; Jung, T.S.; Hale, L.; Cooke, E.L.; et al. Spatial variation of mercury bioaccumulation in bats of Canada linked to atmospheric mercury deposition. *Sci. Total Environ.* **2018**, *626*, 668–677. [CrossRef]
85. Ullah, A.; Nawi, N.M.; Khan, M.H. BAT algorithm used for load balancing purpose in cloud computing: An overview. *Int. J. High Perform. Comput. Netw.* **2020**, *16*, 43–54. [CrossRef]
86. Jayabarathi, T.; Raghunathan, T.; Gandomi, A. The bat algorithm, variants and some practical engineering applications: A review. In *Nature-Inspired Algorithms and Applied Optimization*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 313–330.
87. Kotteeswaran, R.; Sivakumar, L. A Novel Bat algorithm based re-tuning of PI controller of coal gasifier for optimum response. In *Mining Intelligence and Knowledge Exploration*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 506–517.
88. Kalra, M.; Singh, S. A review of metaheuristic scheduling techniques in cloud computing. *Egypt. Inform. J.* **2015**, *16*, 275–295. [CrossRef]
89. Shaddad, R.Q.; Mohammad, A.B.; Al-Gailani, S.A.; Al-Hetar, A.M. Optical frequency upconversion technique for transmission of wireless MIMO-type signals over optical fiber. *Sci. World J.* **2014**, *2014*, 170471. [CrossRef]
90. Fahim, Y.; Rahhali, H.; Hanine, M.; Benlahmar, E.H.; Labriji, E.H.; Hanoune, M.; Eddaoui, A. Load balancing in cloud computing using meta-heuristic algorithm. *J. Inf. Process. Syst.* **2018**, *14*, 569–589.
91. Bhargavi, K.; Babu, B.S.; Pitt, J. Performance Modeling of Load Balancing Techniques in Cloud: Some of the Recent Competitive Swarm Artificial Intelligence-based. *J. Intell. Syst.* **2021**, *30*, 40–58. [CrossRef]
92. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
93. Strumberger, I.; Bacanin, N.; Tuba, M.; Tuba, E. Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. *Appl. Sci.* **2019**, *9*, 4893. [CrossRef]
94. Kaur, G.; Arora, S. Chaotic whale optimization algorithm. *J. Comput. Des. Eng.* **2018**, *5*, 275–284. [CrossRef]
95. Hemasian-Etefagh, F.; Safi-Esfahani, F. Dynamic scheduling applying new population grouping of whales meta-heuristic in cloud computing. *J. Supercomput.* **2019**, *75*, 6386–6450. [CrossRef]
96. Chen, X.; Cheng, L.; Liu, C.; Liu, Q.; Liu, J.; Mao, Y.; Murphy, J. A woa-based optimization approach for task scheduling in cloud computing systems. *IEEE Syst. J.* **2020**, *14*, 3117–3128. [CrossRef]
97. James, J.; Li, V.O. A social spider algorithm for global optimization. *Appl. Soft Comput.* **2015**, *30*, 614–627.
98. Evangeline, D.; Abirami, T. Social spider optimization algorithm: Theory and its applications. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *8*, 327–332.
99. Usurelu, C.C.; Nita, M.C.; Istrate, R.; Pop, F.; Tapus, N. Spider mesh overlay for task load balancing in cloud computing. In Proceedings of the 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 3–5 September 2015.
100. Mahato, D.P.; Singh, R.S. Balanced task allocation in the on-demand computing-based transaction processing system using social spider optimization. *Concurr. Comput. Pract. Exp.* **2017**, *29*, e4214. [CrossRef]
101. Xavier, V.A.; Annadurai, S. Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. *Clust. Comput.* **2019**, *22*, 287–297.
102. Abrol, P.; Gupta, S.; Singh, S. QoS aware social spider cloud web algorithm: Analysis of resource placement approach. In Proceedings of the International Conference on Advancements in Computing & Management (ICACM), Jaipur, India, 13–14 April 2019.
103. Rahman, C.M.; Rashid, T.A. Dragonfly algorithm and its applications in applied science survey. *Comput. Intell. Neurosci.* **2019**, *2019*, 9293617. [CrossRef] [PubMed]
104. Amini, Z.; Maeen, M.; Jahangir, M.R. Providing a load balancing method based on dragonfly optimization algorithm for resource allocation in cloud computing. *Int. J. Netw. Distrib. Comput.* **2018**, *6*, 35–42. [CrossRef]
105. Polepally, V.; Chatrapati, K.S. Dragonfly optimization and constraint measure-based load balancing in cloud computing. *Clust. Comput.* **2019**, *22*, 1099–1111. [CrossRef]
106. Neelima, P.; Reddy, A.R.M. An efficient load balancing system using adaptive dragonfly algorithm in cloud computing. *Clust. Comput.* **2020**, *23*, 2891–2899. [CrossRef]
107. Brabazon, A.; Cui, W.; O'Neill, M. The raven roosting optimisation algorithm. *Soft Comput.* **2016**, *20*, 525–545. [CrossRef]
108. Torabi, S.; Safi-Esfahani, F. Improved raven roosting optimization algorithm (IRRO). *Swarm Evol. Comput.* **2018**, *40*, 144–154. [CrossRef]
109. Rani, E.; Kaur, H. Efficient Load Balancing Task Scheduling in Cloud Computing using Raven Roosting Optimization Algorithm. *Int. J. Adv. Res. Comput. Sci.* **2017**, *8*, 2419–2424.
110. Torabi, S.; Safi-Esfahani, F. A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing. *J. Supercomput.* **2018**, *74*, 2581–2626. [CrossRef]
111. Bhargavi, K.; Babu, B.S. Load Balancing Scheme for the Public Cloud using Reinforcement Learning with Raven Roosting Optimization Policy (RROP). In Proceedings of the 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), Bengaluru, India, 20–21 December 2019.
112. Arulkumar, V.; Bhalaji, N. Performance analysis of nature inspired load balancing algorithm in cloud environment. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 3735–3742. [CrossRef]

113. Li, Z.; Yi, L.; Zhang, Y.; Dong, Y.; Xiao, S.; Hu, W. Compatible TDM/WDM PON using a single tunable optical filter for both downstream wavelength selection and upstream wavelength generation. *Photonics Technol. Lett. IEEE* **2012**, *24*, 797–799. [CrossRef]
114. Valarmathi, R.; Sheela, T. Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing. *Clust. Comput.* **2019**, *22*, 11975–11988. [CrossRef]
115. Al-Shargabi, M.A.; Shaikh, A.; Ismail, A.S. Enhancing the quality of service for real time traffic over Optical Burst Switching (OBS) networks with ensuring the fairness for other traffics. *PLoS ONE* **2016**, *11*, e0161873. [CrossRef]
116. Sethi, S.; Sahu, A.; Jena, S.K. Efficient load balancing in cloud computing using fuzzy logic. *IOSR J. Eng.* **2012**, *2*, 65–71. [CrossRef]
117. Afzal, S.; Kavitha, G. Load balancing in cloud computing–A hierarchical taxonomical classification. *J. Cloud Comput.* **2019**, *8*, 1–24. [CrossRef]
118. Dey, R.K.; Roy, S.; Bose, R.; Sarddar, D. Assessing Commercial Viability of Migrating On-Premise Mailing Infrastructure to Cloud. *Int. J. Grid Distrib. Comput.* **2021**, *14*, 1–10.