Load Balancing Algorithms in Cloud Computing inspired on Animal Behaviour

Dalia Camacho García-Formentí,
Instituto Tecnológico Autónomo de México (ITAM)
Mexico City, Mexico

***Abstract*—Load balancing is one of the most important topics on cloud computing, because by having balanced machines execution time decreases, and energy waste can be avoided. Many load balancing algorithms are inspired on animal behaviour. In this work we review articles containing algorithms inspired on animal behaviour. We show the frequency of algorithms inspired on each specie and how this has evolved from 2012 to 2019. We show the variables used for optimisation in each case, as well as evaluation metrics, experiment platform, and algorithms used to compare the performance. We also provide different classification methods based on the balancing objective whether it is done on virtual machines or physical machines, and on characteristics of the algorithms such as hybrids and individual behaviour.**

***Index Terms*—Load balancing, cloud computing, bioinspired algorithms.**

I. Introduction

Cloud computing provides online services on demand that enables users to access computational resources without the need of owning them. Cloud computing services can be focused on infrastructure as a service (IaaS), platform as a service (PaaS), or software as a Service (SaaS).

Load balancing has been one of the research topics within cloud computing. It focuses on distributing work properly in order to avoid overloaded hosts, which can be either virtual machines (VM) or physical machines (PM). Thus, the aim of load balancing can be balancing VM by either task scheduling or task migration. Or balancing PM by VM allocation or VM migration. In both cases migrations are triggered when there is an overloaded host.

Load balancing is known to be an NP problem, therefore, several heuristics and metaheuristics have been implemented to get approximate solutions. Some examples of the different approaches that have been taken are first come first served (FCFS), round robin (RR) [1], particle swarm optimization (PSO) [2], genetic algorithms (GA) [3], swarm intelligence [4], [5], and statistical methods [6] such as local regression (LR), interquartile range (IQR), and median absolute regression (MAD), among others.

Within this work we focus on load balancing with swarm intelligence algorithms inspired on animal behaviour. We reviewed a total of 32 articles from 2012 to 2019, whose algorithms are inspired mainly on ants, bees, and fireflies, but other animals are also included. For the 32 articles reviewed we show the variables that each author chose to optimise, the metrics used, and the algorithms to which they compared.

We also present two different ways to classify the different works. The first is focused on the objective, which can be balancing VMs or balancing PMs. Even though the aim is similar it is attained by either task scheduling or VM allocation and VM migration. The other classification distinguishes the algorithms in terms of hybridisation with others and in terms of differences in indivuals' behaviour within the population.

This article is organised in five sections. In section II we present similar articles that have reviewed the work done on bioinspired algorithms for load balancing in cloud computing.

In section III we present the basic ideas that gave rise to the fundamental algorithms for each of the species considered.

In section IV we do a general overview of the articles reviewed for this work, including a timeline of the appearance and continuity of the different species that inspired load balancing algorithms from 2012 to 2019.

In section V we give a classification in terms of the objective of load balancing which may be on VM or PM and evaluate the usage of optimisation variables, evaluation metrics. We also provide a classification that depends on the characteristics of the optimisation algorithms used in terms of the individuals behaviour.

In section VI we show the gaps that were found and possible future lines of research. In section VII we give our final conclusions.

II. Related Work

Existing research on load balancing algorithms is huge, only in the first ten months of 2019 more than $10,000$ research articles containing the words "load balancing" and "cloud" were published, according to Google scholar. Therefore, several surveys have been done on this topic trying to organise the research.

For example Ullah *et al.* [7] focused on algorithms that used different modified versions of the artificial bee colony algorithm (ABC) and categorised them depending on the load balancing area in which they were used.

Milan *et al.* [8] did a systematic review of nature inspired algorithms, they categorised the articles in terms of their inspiration and for those that were inspired on ant colony optimisation they listed the advantages and disadvantages.

Hota *et al.* [9] reviewed different load balancing algorithms including FCFS, RR, GA, and several algorithms inspired on animal behaviour including hybrids. They evaluated advantages, disadvantages, optimisation metrics, and experiment platform.

Kumar and Bawa [10] proposed a taxonomy based on characteristics that inspired the different algorithms, these include bio-simulated, nature-inspired, physics based, evolutionary and swarm based. They identified the basic underlying algorithms, they used these algorithms to optimise different functions and evaluated their performance.

These four reviews were all published in 2019, and there are certainly more of them. In this work we focus on those algorithms inspired on animal behaviour, this selection is independent of Kumar's taxonomy. We differentiate between algorithms whose aim is balancing VM or PM. We find the important variables considered for optimisation, we show the metrics used to evaluate the different algorithms and we show the algorithms which they compare against. Moreover, we further classify algorithms depending on hybridisation and differences on behaviour between individuals.

III. Overview of the algorithms that originated these approaches

In this section we describe the underlying ideas of animal behaviour that led to the different optimisation algorithms. The animal species used throughout the 32 articles are ants, bees, fireflies, cuckoo birds, bats, krill, lions, social spiders, dragonflies, crows, moths, and butterflies. Some of these algorithms involve different behaviours for different individuals, movements induced by external forces, different communication schemes, and randomness.

A. Ants

The ant colony optimisation algorithm (ACO) was first proposed by Dorigo *et al.* in 1996 [11]. The idea is that ants leave a pheromone trace as they walk, and they tend to follow the path where the pheromone is the strongest. In terms of the algorithm ants are placed randomly. They move to other possible solutions with a certain probability that depends on the pheromone concentration at each node (τ_i) and on another variable that provides information of the problem at hand (η_i). In general τ_i decreases by pheromone evaporation and increases if an ant moved to that node in the current step. This process is repeated several times before choosing a solution. In load balancing, ACO has been used to find overloaded and underloaded hosts as ants traverse the different VM or PM accordingly [12], [13], [14], [15]. Also, the variable η can be a function of cost, execution time, transfer time, network bandwidth, load [16], [13], [17].

B. Bees

In 2005 Karaboga proposed an algorithm inspired by the foraging of honey bees. The algorithm is better known as artificial bee colony (ABC). It distinguishes three categories of bees that are employed, unemployed scouts, and unemployed onlookers. Employed bees exploit food sources and update the other bees of the status of the food source when they return to the beehive. Scout bees explore new areas to find new food sources, and they also give updates when returning to the hive. Onlooker bees wait in the hive and receive information from scout bees and employed bees. Information is communicated via waggle dances. Communication between individuals is one of the most relevant features of their behaviour and has also been taken for hybrid implementations [16]. In the algorithm employed bees take advantage of food sources, after that they update the knowledge base. Then scout bees explore other sources and update the knowledge base. Finally onlooker bees may become employed if food sources are sufficient. In the particular case of load balancing VM (or PM) are considered as the food sources; and in some cases tasks are seen as bees that change their category [18], [19], [20], [21].

C. Fireflies

The firefly algorithm (FA) was proposed by Thilagavathi *et al.* in 2014 inspired on the movement of fireflies while foraging. Fireflies communicate with others by showing a bright light, if there is more food they show a brighter light and fireflies move towards the brightest firefly they see. A firefly's appreciation of the brightest firefly depends on the attractiveness of other fireflies and its distance to them. The parameter of attractiveness contains information about the problem that is being optimised, which in this particular case can be constructed from CPU, memory, and bandwidth utilisation, as well as execution time [22], [23], [24].

D. Cuckoo Bird

Cuckoo birds are known for laying their eggs on nests from other bird species. If eggs are similar to the original birds' eggs, then the chance that cuckoo birds hatch is higher. Therefore, cuckoo birds try to identify areas that have more nests with similar eggs. Once a nest was successful, cuckoo birds only explore areas close to the successful nest. This behaviour was implemented as an optimisation algorithm by Rajabioun in 2011 [25] and was used in 2015 by Yakhci *et al.* [26] to locate overloaded hosts from their CPU utilisation. In this case areas were considered as nearby hosts.

E. Bats

Yang proposed a metaheuristic for optimisation problems based on bats' foraging [27]. Bats locate their food via echolocation in which they emit sonar waves and sense the echo produced by them. As they come closer to their prey they increment the frequency of the waves and decrease the volume. This behaviour along with their flight patterns is taken in consideration in the optimisation algorithm. Initially bats are placed randomly and they adjust their position and flight in terms of the value of the objective function and the best solution found up to that moment. Sharma *et al.* [28] used the BAT algorithm for load balancing via task scheduling, where the VM for a task was selected in such a way that overloaded hosts were avoided.

F. Krill

The algorithm inspired on krill herd behaviour was proposed by Hassan in 2017 for load balancing in cloud [29]. It is based on krill movement which is provoked by diffusion, foraging

movement, inertia, and induced movement from the rest of the individuals in the herd. Foraging movement is directly related to the best solution at a given moment. For the presented algorithm the fitness function considered task request speed and task cost, in the first implementation the solution was chosen as the krill with the best fitness value, but in the second one energy consumption and execution time were added to the final selection.

G. Lions

Lions' behaviour within a population is different for males and females and it also depends on their membership to a pride or if they are loners. Several behaviours are considered within the lions optimisation algorithm (LOA) which include roaming, mating, hunting, and fighting. Each of the lions moves towards solutions in different fashions depending on their role within the population and they can also change their role from loners to members of a pride depending on which male lion has the best solution to the optimisation problem. By considering different behaviours within the population the algorithm can exploit the best solutions and explore new ones at the same time. LOA was proposed by Yazdani and Jolai in 2016 [30]. Almezeini and Hafez implemented LOA for load balancing [31], and their objective was to minimize the makespan.

H. Social Spider

The chaotic social spider algorithm (CSSA) was proposed by Xavier and Annadurai in 2019 [4], it reflects the way in which spiders move towards food based on their communication via vibrations on the spiderweb. Communication is simulated with broadcast messages and movement is influenced by the best solution received, their inertia and a random value. For their fitness function they considered makespan and cost.

I. Dragonflies and Crows

Dragonflies have two ways in which they move depending on the objective. When they are hunting they stay within an area in which they move erratically. And when they migrate they move together maintaining separation to avoid collision, cohesion in order to maintain together, and alignment that allows them to follow the same direction. Hunting can be compared to exploitation and migration to exploration. Thus, these two behaviours were considered within the dragonfly algorithm (DA) proposed by Mirjalili in 2016 [32].

The crow search algorithm (CSA) proposed by Askarzadeh in 2016 [33] considers that crows follow each other in order to steal from others. If a crow notices it is being followed it does not return to its nest, but moves randomly instead. Moreover, crows remember where they leave their belongings so they can go back for them. The memory aspect is considered within the optimisation algorithm to save the best solutions, flying randomly helps exploration and following another crow improves exploitation. After several iterations of crows following each other a solution is selected.

To our knowledge there are no load balancing algorithms based on the behaviour of crows and dragonflies independently, but More and Ingle considered the implementation of a hybrid specie [34]. To update the position of individuals, they consider both mathematical models DA and CSA and add them to obtain the new position. They use this algorithm for migrating VM once overloaded hosts are detected.

J. Moths

Moths tend to move towards light sources, this is known as photo-taxis. Once moths are close to the light source they fly around it with Lévy flights. The algorithm proposed by Wang in 2018 [35], the moth search algorithm (MSA), considers both movements. For this the population is sorted from the best solution to the worst and is divided into two groups. The individuals on the best group move with Lévy flights, while the other individuals move towards the best solution. This process is repeated on every iteration, included the sorting of the solutions. Elaziz *et al.* defined their fitness function as the makespan and implemente MSA for load balancing via task scheduling [36].

K. Monarch Butterflies

Wang also proposed the monarch butterfly optimisation (MBO) in 2015 (published in 2019) [37]. Monarch butterflies migrate from one location to another. This is represented in the algorithm by dividing the butterfly population into two different groups whose movements are determined by a random number that reflects migration of one group or the other, and the best solution in each population is maintained on every iteration. Strumberger *et al.* implemented MBO, and modified version of MBO in which the behaviour of one group is either random or towards the best solution, and the other group moves either randomly or with Lévy flights [38]. Their ojective function is defined as the makespan.

IV. General Comparison and Classification

For this survey we selected 32 articles on load balancing in cloud computing, whose algorithms are inspired on animal behaviour. The articles range from 2012, when modified ACO [12] and ABC [39] algorithms were first reported to be used in load balancing, to 2019. Fifteen of these articles were published between 2017 and 2019. The distribution of the articles is shown in Figure 1 and in Table I, which also contains the main characteristics of each of the articles considered in this work, such as the level in which load balancing is performed, the animal whose behaviour inspired the algorithm, and the experiment platform among other characteristics.

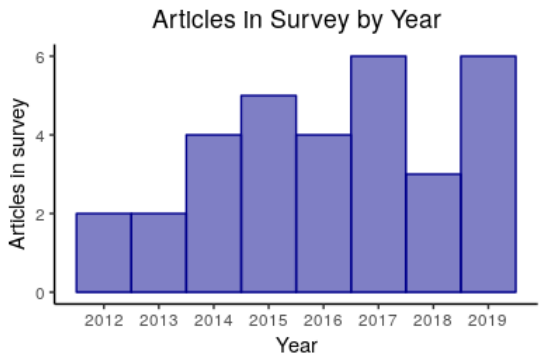

Fig. 1. Distribution of articles in survey in terms of publication year.

Most of the articles are inspired on bees, ants and fireflies as shown in Figure 2. However, from 2015 on wards, algorithms inspired on another 8 different species have been implemented, this can be seen in Figure 3.

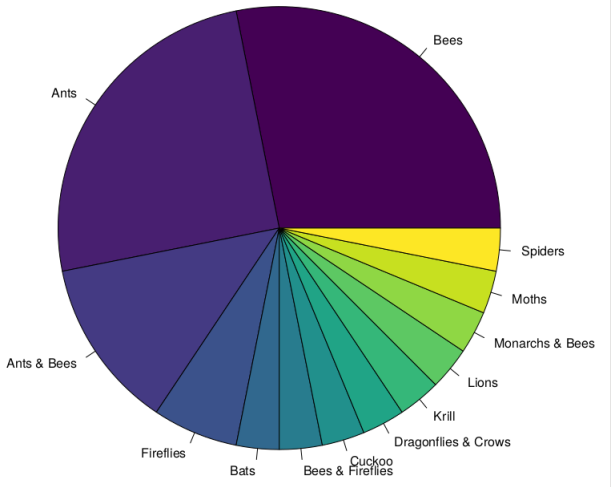

Fig. 2. Distribution of animal species within articles in survey.

From Figure 3 we may also infer that algorithms based solely on honey bee foraging have lost their relevance, but algorithms involving bees and other species have gained importance. This may be due to the high complexity of ABC which is $O(n^3)$ [5] compared to $O(n^2)$, the complexity of ACO [5]. But properties such as having a shared knowledge base [16] and a parameter that measures exhaustion when a better solution is not achieved are desirable characteristics for other algorithms [38].

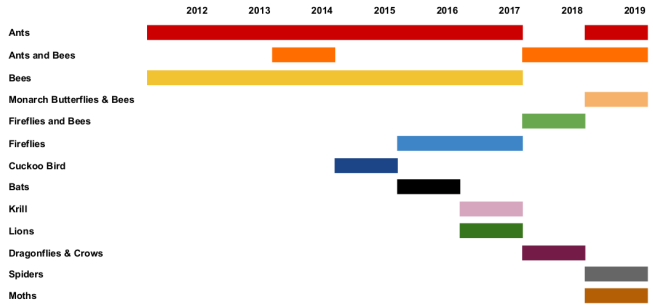

Fig. 3. Timeline of the species that inspired load balancing algorithms throughout the years.

In terms of the experimental set up, most experiments were carried out in CloudSim toolkit [40], which allows to perform different simulations. To our knowledge only Kansal and Chana in [24] performed their experiments on a real data center (Bharat Sanchar Nigam Limited). Other platforms used were Cloud Analyst and Matlab. This information can be seen in Table I.

In the following section we will address more specific details in terms of the characteristics considered for optimisation, on evaluation metrics, and the most common algorithms that authors compare to. All these are related to the classification in terms of either task scheduling or VM allocation and migration.

We also describe a possible classification in terms of the characteristics of the algorithms used which include hybridisation and population behaviour.

V. Classification

In this section we evaluate two different ways in which animal inspired load balancing can be classified. The first is related to the kind of load balancing problem that is being addressed, which may be balancing VM or balancing PM and on the different approaches to solve it. The second classification distinguishes hybrid algorithms and also distinguishes populations by the groups of individuals with different behaviour, which we call inequity .

A. Objective

As we have mentioned previously load balancing in cloud refers to two different objectives. The first objective consists on avoiding overloaded VM. This can be achieved by defining a task scheduling algorithm that distributes tasks considering VM capacity [41], [42], [29]. The other way to achieve balanced VM is by identifying overloaded VM and migrating tasks from overloaded VM to underloaded VM once a threshold is exceeded [18], [19], [20]. This threshold can depend on capacity, standard deviation of the load (SD) or degree of imbalance (DI).

The other objective of load balancing is to avoid overloaded PM and underloaded PM. Overloaded PM take more time to process clients' requests and underloaded PM waste as much energy as balanced PM, but being mostly idle [22]. The solutions try to migrate all VM from underloaded PM and

turning them off, and migrating some VM from overloaded hosts to other PM. Another approach to load balancing on PM is through VM allocation algorithms that consider load on the different hosts, that is the case of [43].

In the context of load balancing task scheduling and VM allocation have a similar underlying idea, which is defining properly which task goes to which VM or which VM goes to which PM before load imbalance occurs. On the other hand task migration and VM migration share the idea of identifying load imbalance and migrating tasks or VMs to achieve balance. Due to these similarities VM balancing and PM balancing fall into the category of load balancing in cloud.

From Table I we can see that the first VM balancing algorithms consisted mainly on task migration, but in the most recent years this has changed and load balancing is achieved from task scheduling. From 2017 on wards, only [13] considered task migration. This may indicate that achieving VM balancing from scheduling is more cost effective than migrating tasks. On the other hand most PM allocation algorithms consider VM migration, even though live migration is known to be costly [43]. Following the trend that is seen on VM balancing, future works on PM balancing may give higher importance on optimising VM allocation in order to prevent VM migrations as much as possible.

Since the aim of VM balancing and PM balancing is different the characteristics considered during optimisation vary. For example in VM balancing: VM capacity, priority, makespan, and cost were the most important aspects to consider, this can be seen in Table II. Note that even though balancing on VM is highly related to task scheduling, dependency between tasks is not one of the most important aspects. Just [44] and [13] consider dependency between tasks. Moreover, none of the proposed algorithms considers preemptive execution.

For PM balancing energy, CPU load, network load, memory load and number of migrations were the most important aspects during the definition of objective functions. This is shown in Table III.

Similar differences can be seen on the evaluation metrics, where makespan, DI, cost, and execution time were the most used metrics for load balancing in VM. In contrast for PM the most used metrics were number of migrations, energy, and service level agreement violations (SLAV). For detailed information on all the metrics used see Table IV and Table V.

In terms of comparison to other algorithms ACO is the most used algorithm to compare against in both scenarios. But in VM balancing algorithms such as FCFS and ABC are widely used, but they also compare to a wide range of bioinspired algorithms including GA and even whale optimisation algorithm (WOA) [45] which is also inspired on animal behaviour, which has been considered for task scheduling, but without taking into consideration load balancing [46].

On the other hand PM balancing compares to different implementations of IQR, LR, and MAD. Some compare to previous works on load balancing on animal inspired algorithms. For example [26] compares to [47] and [48] to [14].

The algorithms used for comparison for each of the articles reviewed are on Tables VI and VII for VM balancing and PM balancing respectively.

We have seen that identifying whether VM or PM are being balanced is an important aspect to consider when defining objective functions, evaluation metrics, and comparison with other algorithms. However, this classification is a general one that must be present on any approach, not specific to animal behaviour inspired algorithms. The following section deepens on the algorithms' characteristics.

B. Algorithms' Characteristics

Algorithms inspired on animal behaviour are considered as swarm intelligence algorithms, because individuals behaviour is influenced by others. Individuals from different species interact with others and with their environment in different ways. Here we consider hybridisation and inequity as elements that can be found.

Hybrid algorithms not only include those in which two species interact [49], [5], [15], [24], or in which one of the species is given additional attributes that the other animal has [16], [34], [38]. Hybrid algorithms may also include other heuristics or algorithms involved, such as minimal migration time (MMT) as is the case of [47], [26], genetic algorithms (GA) as in [48], support vector regression (SVR) in [34], or Predict Earliest Finish Time (PEFT) and Heterogeneous Earliest Finish Time (HEFT) as in [13]. Even chemical processes such as osmosis has been added to an already hybrid algorithm of ants and bees [5]. From the 32 articles reviewed 13 have some type of hybridisation from those seven consider the behaviour of two different species. Four of them consider two coexisting species [15], [24], [49], [5] and three a hybrid specie [16], [34], [38]. This is shown in detail on Table I.

The algorithms can also be distinguished by inequity. We consider three levels of inequity. On level zero all individuals have the same behaviour, this is the case of non-hybrid algorithms inspired on ants, fireflies, cuckoo birds and others [11], [50], [25]. On level one there are two groups of individuals with different behaviour patterns, as examples we have MBO and MSA [37], [35]. Finally, on level three there are more than two different groups of individuals, this is the case of LOA which considers at least four different behaviours and of ABC in which there are three different kinds of bees[51], [30].

On Table I we show the level of inequity of each of the algorithms on every article. Notice that some algorithms using hybridisation of different species are on inequity level one [16], [34]. This occurs, because individuals share traits from both species, but interacting species are not considered. This also occurs in [38] whose algorithm is in level two, the level two is due to the sub populations of monarch butterflies, and not to hybridisation with bees.

Algorithms on levels two and three consider explicitly that on every iteration there will be exploration and exploitation.[51], [30], [37], [35] This is a consequence of dividing the population into different activities. Algorithms on level three can consider different exploration and exploitation approaches such as in LOA. Or they can join information from different individuals before exploiting as in ABC.

Algorithms such as ACO and BAT [11], [27], that do not differentiate between individuals, assign exploration or

exploitation randomly for each individual on each iteration. On the case of ACO as iterations go by, the probability inclines towards nodes with high pheromone concentration, this helps convergence to a solution, but exploration loses importance. In some cases this may be a good approach, but it may lead to local optimisation rather than global optimisation. However, ACO has a lower complexity than ABC and LOA, and is easier to implement.

Most of the algorithms inspired in species different than bees belong to level one, probably, because of the latter. However, in 2019 MSDE [36] and MBO [38] have been implemented for load balancing, which are the only two algorithms on level two. If these two show to have a good performance in practice, which may be likely since MSDE is $O(n^2)$, then future research on load balancing may include more algorithms on level two. Considering that these algorithms are easier to implement than those on level 3 and they also take into account exploration and exploitation explicitly.

VI. Future lines of research

In this section we present the gaps that we found on the articles discussed; some recommendations on the implementation and evaluation; and lastly possible future lines of research considering hybridisation and behaviour.

From the first classification we see that load balancing on VM has transitioned from task migration to scheduling considering load balancing. This evolution of load balancing in VM may also occur in the PM context, since live migrations are costly. As general observations on algorithms based on task scheduling we noted that not all of them consider priority [52], [28], [41], [31], [4], [49], [23], [17]; only two[44] [13] consider dependency within tasks, and none of them consider preemptive execution. Even though authors since 2014 [21] consider interdependent tasks within their future work section.

We also found that authors tend to compare with a huge variety of algorithms. But comparisons against other algorithms within the category of load balancing with animal inspired algorithms is not done as often. Specially it is not done against recent works, unless they have direct inspiration from previous article. Thus, comparisons are often done against the basic ACO and ABC algorithms, instead of their modified versions. Therefore, we cannot really know if the new algorithms are actually better than other previously existing ones. Also, complexity should be considered more often for a quick referral of comparison, along with the simulations. Because execution time and makespan may decrease as technology improves.

In terms of the experimental setup we would suggest performing experiments on real cloud environments, this is also a common line within the future work sections of various authors [16], [43], [22].

Also, to our knowledge based on the reviewed works there are no standard workloads to perform the experiments. Developing standard workloads for both VM balancing and PM balancing would help evaluate the performance on different scenarios. Some of these workloads may involve having highly dependent workflows, workloads containing dependent and independent tasks, different request ratios from clients, having clients whose VM have different priority and the cost is higher when SLAVs occur for high priority clients.

The analysis done for the hybrid and non hybrid classification and to the inequity level suggest that future algorithms will tend to hybridise desired characteristics of different species, but probably maintaining an inequity level of two. This would mean having two subgroups one centred on exploration, the other on exploitation and combining desirable behaviours such as exhaustion in which a new random solution is explored when no progress is made.

VII. Conclusions

Within this work we have compared 32 articles on load balancing whose algorithms are inspired on animal behaviour. As a more general classification we found that load balancing can be done at the level of assigning tasks to VM or on allocating VMs to PM. In both cases load balancing can be done by either migrating work from overloaded hosts to underloaded hosts, or by considering load balancing as tasks are scheduled or as VMs are allocated. We also proposed distinguishing algorithms as hybrid or non hybrid which may be related to coexisting species, joining different heuristics and metaheuristics to the basic algorithms, or giving a specie characteristics from another one. Finally, we distinguished algorithms by the behaviour of individuals considering if it was the same for all individuals or if there were different groups.

For each article we distinguished the objective, the approach taken to achieve it, the variables considered during optimisation, evaluation metrics used and the experiment platform. Also, we showed that most recent algorithms tend to hybridise species or merge different algorithms to take advantage of the strengths of each approach.

Moreover, we proposed a classification based on differences of individuals behaviour. If there are different behaviours, exploration and exploitation are considered explicitly. Furthermore, exploration can be done with different approaches when more than two behaviours are considered.

Lastly, we exposed some of the gaps encountered, such as a lack of standardisation in experiment evaluation and the urge to consider dependent tasks and preemptive execution when balancing on VM.

References

[1] D. Nusrat Pasha, A. Agarwal, and R. Rastogi, "Round robin approach for vm load balancing algorithm in cloud computing environment," *International Journal*, vol. 4, no. 5, pp. 34–39, 2014.

[2] N. Sethi, S. Singh, and G. Singh, "Improved mutation-based particle swarm optimization for load balancing in cloud data centers," in *Harmony Search and Nature Inspired Optimization Algorithms* (N. Yadav, A. Yadav, J. C. Bansal, K. Deep, and J. H. Kim, eds.), (Singapore), pp. 939–947, Springer Singapore, 2019.

[3] B. Wang and J. Li, "Load balancing task scheduling based on multi-population genetic algorithm in cloud computing," in *2016 35th Chinese Control Conference (CCC)*, pp. 5261–5266, July 2016.

[4] V. M. Arul-Xavier and S. Annadurai, "Chaotic social spider algorithm for load balance aware task scheduling in cloud computing," *Cluster Computing*, vol. 22, pp. 287–297, Jan 2019. https://doi.org/10.1007/s10586-018-1823-x.

[5] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic bio-inspired load balancing algorithm in cloud computing," *IEEE Access*, vol. 7, pp. 42735–42744, 2019.

[6] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[7] A. Ullah, N. M. Nawi, J. Uddin, S. Baseer, and A. H. Rashed, "Artificial bee colony algorithm used for load balancing in cloud computing," vol. 8, no. 2, pp. 156–167, 2019.

[8] S. T. Milan, L. Rajabion, H. Ranjbar, and N. J. Navimipour, "Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments," *Computers & Operations Research*, vol. 110, pp. 159 – 187, 2019.

[9] A. Hota, S. Mohapatra, and S. Mohanty, "Survey of different load balancing approach-based algorithms in cloud computing: A comprehensive review," in *Computational Intelligence in Data Mining* (H. S. Behera, J. Nayak, B. Naik, and A. Abraham, eds.), (Singapore), pp. 99–110, Springer Singapore, 2019.

[10] A. Kumar and S. Bawa, "A comparative review of meta-heuristic approaches to optimize the sla violation costs for dynamic execution of cloud services," *Soft Computing*, Jun 2019.

[11] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, pp. 29–41, Feb 1996.

[12] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, Nitin, and R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization," in *2012 UKSim 14th International Conference on Computer Modelling and Simulation*, pp. 3–8, March 2012.

[13] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid heuristic-metaheuristic techniques in cloud environment," *Journal of King Saud University - Computer and Information Sciences*, 2019. http://www.sciencedirect.com/science/article/pii/S1319157818309820.

[14] W. Wen, C. Wang, D. Wu, and Y. Xie, "An aco-based scheduling strategy on load balancing in cloud computing environment," in *2015 Ninth International Conference on Frontier of Computer Science and Technology*, pp. 364–369, Aug 2015.

[15] A. Tripathi, S. Shukla, and D. Arora, "A hybrid optimization approach for load balancing in cloud computing," in *Advances in Computer and Computational Sciences* (S. K. Bhatia, K. K. Mishra, S. Tiwari, and V. K. Singh, eds.), (Singapore), pp. 197–206, Springer Singapore, 2018.

[16] R. Madivi and S. S. Kamath, "An hybrid bio-inspired task scheduling algorithm in cloud environment," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–7, July 2014.

[17] Q. Guo, "Task scheduling based on ant colony optimization in cloud environment," *AIP Conference Proceedings*, vol. 1834, no. 1, pp. 040039–1–040039–11, 2017. https://aip.scitation.org/doi/abs/10.1063/1.4981635.

[18] D. B. L.D. and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput.*, vol. 13, pp. 2292–2303, May 2013. http://dx.doi.org/10.1016/j.asoc.2013.01.025.

[19] K. R. R. Babu, A. A. Joy, and P. Samuel, "Load balancing of tasks in cloud computing environment based on bee colony algorithm," in *2015 Fifth International Conference on Advances in Computing and Communications (ICACC)*, pp. 89–93, Sep. 2015.

[20] K. R. Remesh Babu and P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud," in *Innovations in Bio-Inspired Computing and Applications* (V. Snášel, A. Abraham, P. Krömer, M. Pant, and A. K. Muda, eds.), (Cham), pp. 67–78, Springer International Publishing, 2016.

[21] Y. S. Sheeja and S. Jayalekshmi, "Cost effective load balancing based on honey bee behaviour in cloud environment," in *2014 First International Conference on Computational Systems and Communications (ICCSC)*, pp. 214–219, Dec 2014.

[22] N. J. Kansal and I. Chana, "Energy-aware virtual machine migration for cloud computing - a firefly optimization approach," *Journal of Grid Computing*, vol. 14, pp. 327–345, Jun 2016. https://doi.org/10.1007/s10723-016-9364-0.

[23] G. Kaur and K. Kaur, "An adaptive firefly algorithm for load balancing in cloud computing," in *Proceedings of Sixth International Conference on Soft Computing for Problem Solving* (K. Deep, J. C. Bansal, K. N. Das, A. K. Lal, H. Garg, A. K. Nagar, and M. Pant, eds.), (Singapore), pp. 63–72, Springer Singapore, 2017.

[24] N. J. Kansal and I. Chana, "An empirical evaluation of energy-aware load balancing technique for cloud data center," *Cluster Computing*, vol. 21, pp. 1311–1329, Jun 2018. https://doi.org/10.1007/s10586-017-1166-z.

[25] R. Rajabioun, "Cuckoo optimization algorithm," *Appl. Soft Comput.*, vol. 11, pp. 5508–5518, Dec. 2011. http://dx.doi.org/10.1016/j.asoc.2011.05.008.

[26] M. Yakhchi, S. M. Ghafari, S. Yakhchi, M. Fazeli, and A. Patooghi, "Proposing a load balancing method based on cuckoo optimization algorithm for energy management in cloud computing infrastructures," in *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, pp. 1–5, May 2015.

[27] X.-S. Yang, *A New Metaheuristic Bat-Inspired Algorithm*, pp. 65–74. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. https://doi.org/10.1007/978-3-642-12538-6_6.

[28] S. Sharma, A. K. Luhach, and S. A. Sinha, "An optimal load balancing technique for cloud computing environment using bat algorithm," *Indian Journal of Science and Technology*, vol. 9, no. 28, pp. 1–4, 2016.

[29] H. R. Abdulkareem and M. M. N, "A krill herd behaviour inspired load balancing of tasks in cloud computing," *Studies in Informatics and Control*, vol. 26, no. 4, pp. 413–424, 2017.

[30] M. Yazdani and F. Jolai, "Lion optimization algorithm (loa): A nature-inspired metaheuristic algorithm," *Journal of Computational Design and Engineering*, vol. 3, no. 1, pp. 24 – 36, 2016. http://www.sciencedirect.com/science/article/pii/S2288430015000524.

[31] N. Almezeini and A. Hafez, "Task scheduling in cloud computing using lion optimization algorithm," *algorithms*, vol. 5, pp. 77–83, 2017.

[32] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, pp. 1053–1073, May 2016. https://doi.org/10.1007/s00521-015-1920-1.

[33] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Computers & Structures*, vol. 169, pp. 1 – 12, 2016. http://www.sciencedirect.com/science/article/pii/S0045794916300475.

[34] N. S. More and R. B. Ingle, "Energy-aware vm migration using dragonfly–crow optimization and support vector regression model in cloud," *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 09, no. 06, pp. 1850050–1–1850050–24, 2018. https://doi.org/10.1142/S1793962318500502.

[35] G.-G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, pp. 151–164, Jun 2018.

[36] M. A. Elaziz, S. Xiong, K. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Systems*, vol. 169, pp. 39 – 52, 2019. http://www.sciencedirect.com/science/article/pii/S0950705119300322.

[37] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Computing and Applications*, vol. 31, pp. 1995–2014, Jul 2019.

[38] I. Strumberger, M. Tuba, N. Bacanin, and E. Tuba, "Cloudlet scheduling by hybridized monarch butterfly optimization algorithm," *Journal of Sensor and Actuator Networks*, vol. 8, no. 3, 2019. https://www.mdpi.com/2224-2708/8/3/44.

[39] Jing Yao and Ju-hou He, "Load balancing strategy of cloud computing based on artificial bee algorithm," in *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, vol. 1, pp. 185–189, April 2012.

[40] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, pp. 23–50, Jan. 2011. http://dx.doi.org/10.1002/spe.995.

[41] A. Gupta and R. Garg, "Load balancing based task scheduling with aco in cloud computing," in *2017 International Conference on Computer and Applications (ICCA)*, pp. 174–179, Sep. 2017.

[42] W. Hashem, H. Nashaat, and R. Rizk, "Honey bee based load balancing in cloud computing.," *KSII Transactions on Internet & Information Systems*, vol. 11, no. 12, pp. 5694–5711, 2017.

[43] N. J. Kansal and I. Chana, "Artificial bee colony based energy-aware resource utilization technique for cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 5, pp. 1207–1225, 2015. https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3295.

[44] S. Xue, M. Li, X. Xu, J. Chen, and S. Xue, "An aco-lb algorithm for task scheduling in the cloud environment," *Journal of Software*, vol. 9, no. 2, pp. 466–473, 2014.

[45] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.

[46] K. Sreenu and M. Sreelatha, "W-scheduler: whale optimization for task scheduling in cloud computing," *Cluster Computing*, vol. 22, pp. 1087–1098, Jan 2019.

[47] S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, "Bee-mmt: A load balancing method for power consumption management in cloud computing," in *2013 Sixth International Conference on Contemporary Computing (IC3)*, pp. 76–80, Aug 2013.

[48] K. Kaur and A. Kaur, "A hybrid approach of load balancing through vms using aco, minmax and genetic algorithm," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pp. 615–620, Oct 2016.

[49] M. Gamal, R. Rizk, H. Mahdi, and B. Elhady, *Bio-inspired Based Task Scheduling in Cloud Computing*, pp. 289–308. Cham: Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-02357-7_14.

[50] D. Thilagavathi and A. S. Thanamani, "Scheduling in high performance computing environment using firefly algorithm and intelligent water drop algorithm," *Int. J. Eng. Trends Technol*, vol. 14, no. 1, pp. 8–12, 2014.

[51] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer . . . , 2005.

[52] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, "An ant colony based load balancing strategy in cloud computing," in *Advanced Computing, Networking and Informatics- Volume 2* (M. Kumar Kundu, D. P. Mohapatra, A. Konar, and A. Chakraborty, eds.), (Cham), pp. 403–413, Springer International Publishing, 2014.

[53] A. Soni, G. Vishwakarma, and Y. K. Jain, "A bee colony based multi-objective load balancing technique for cloud computing environment," *Int J Comput Appl*, vol. 114, no. 4, pp. 19–25, 2015.

Year	Algorithm	Author	Animal Behaviour	Hybrid	Coesxisting Species	Individuals	Inequity Level	VM Balancing	PM Balancing	Migrations	Experiment Platform
2012	modified ACO [12]	Nishant et al.	Ants			Same	1	X		X	None
2012	improved ABC [39]	Yao & He	Bees			Different	3	X		X	Repast.NET
2013	HBB-LB [18]	Babu & Krishna	Bees			Different	3	X		X	CloudSim
2013	Bee-MMT [47]	Ghafari et al.	Bees	X		Different	3		X	X	CloudSim
2014	proposed ACO [52]	Dam et al.	Ants			Same	1	X			Cloud Analyst
2014	ACO-LB [44]	Xue et al.	Ants			Same	1	X		X	CloudSim
2014	HBBLBP [21]	Sheeja & Jayalekshmi	Bees			Different	3	X		X	CloudSim
2014	Hybrid	Madivi [16] & Kamath	Ants & Bees	X		Same	1	X			CloudSim
2015	ERU [43]	Kansal & Chana	Bees			Different	3		X		CloudSim
2015	ACO-VMM [14]	Wen et al.	Ants			Same	1		X	X	CloudSim
2015	BCBLB [19]	Babu et al.	Bees			Different	3	X		X	CloudSim
2015	ABC-MO [53]	Soni et al.	Bees			Different	3	X			Matlab
2015	COA-MMT [26]	Yakhchi et al	Cuckoo bird	X		Same	1		X	X	CloudSim
2016	EBC [20]	Babu & Samuel	Bees			Different	3	X		X	Simulated Environment
2016	ACO-MMAS-GA [48]	Kaur & Kaur	Ants	X		Same	1		X	X	CloudSim
2016	FFO-EVMM [22]	Kansal & Chana	Fireflies	X		Same	1		X	X	CloudSim
2016	BAT [28]	Sharma et al.	Bats			Same	1	X			Matlab
2017	LB-ACO [41]	Gupta & Garb	Ants			Same	1	X			CloudSim
2017	LBA_HB [42]	Hashem et al.	Bees			Different	3	X			CloudSim
2017	Krill-LB [29]	Raed Abdulkareem & Muamer	Krill			Same	1	X			CloudSim
2017	ADF	Kaur & Kaur	Fireflies			Same	1	X			Cloud Analyst
2017	MO-ACO [17]	Guo	Ants			Same	1	X			CloudSim
2017	LOA [31]	Almezeini & Hafez	Lions			Different	3	X			CloudSim
2018	Hybrid2 [15]	Tripathi et al.	Ants & Bees	X	X	Different	3	X			Cloud Analyst
2018	D-Crow [34]	More & Ingle	Dragonflies & Crows	X		Same	1		X	X	Java tool
2018	ELB [24]	Kansal & Chana	Bees & Fireflies	X	X	Different	3		X	X	BSNL data center
2019	CSSA [4]	Xavier & Annadurai	Spiders			Same	1	X			CloudSim
2019	H-BAC [49]	Gamal et al.	Ants & Bees	X	X	Different	3	X			CloudSim
2019	HPA & HHA [13]	Kaur & Kaur	Ants	X		Same	1	X		X	Cloud Workflow Simulator
2019	OH_BAC [5]	Gamal et al.	Ants & Bees	X	X	Different	3		X	X	CloudSim
2019	MSDE [36]	Elaziz et al.	Moths	X		Different	2	X			CloudSim
2019	MBO [38] & MBO-ABC	Strumberger et al.	Monarch Butterfly & bees	X		Different	2	X			CloudSim

TABLE I
GENERAL CHARACTERISTICS OF ARTICLES IN SURVEY

Year	Algorithm	Dependent Tasks	Priority	Makespan	Execution Time	Transfer Time	Cost	Energy	CPU Load	Memory Load	Network Load	VM Capacity
2012	modified ACO [12]											
2012	improved ABC [39]			X								
2013	HBB-LB [18]		X	X								
2014	proposed ACO [52]						X		X		X	
2014	ACO-LB [44]	X		X								
2014	HBBLBP [21]		X	X			X					
2014	Hybrid [16]		X		X				X	X	X	
2015	BCBLB [19]		X									
2015	ABC-MO [53]		X				X					X
2016	EBC [20]		X									
2016	BAT [28]											X
2017	LB-ACO [41]			X			X					
2017	LBA_HB [42]		X	X								X
2017	KrilI-LB [29]		X				X	X				
2017	ADF [23]								X	X	X	
2017	MO-ACO [17]			X			X					
2017	LOA [31]			X								
2018	Hybrid2[15]				X				X	X	X	X
2019	CSSA [4]			X			X					X
2019	H-BAC [49]		X		X	X						X
2019	HPA & HHA [13]	X	X		X		X					X
2019	MSDE [36]		X	X								
2019	MBO & MBO-ABC [38]		X	X								

TABLE II
CHARACTERISTICS CONSIDERED DURING OPTIMISATION FOR VM LOAD BALANCING

Year	Algorithm	None	Makespan	No. of Migrations	Execution Time	Deegree of Imbalance	SD	Energy	Workload %	Cost	Priority	Request Drop Ratio	Load Execution Ratio	Waiting Time	Processing Time	Response Time	Latency	Throughput	Resource Utilization
2012	modified ACO [12]	X																	
2012	improved ABC [39]																	X	
2013	HBB-LB [18]		X	X	X	X													
2014	proposed ACO [52]															X			
2014	ACO-LB [44]				X				X										
2014	HBBLBP [21]				X	X				X									
2014	Hybrid [16]				X														
2015	BCBLB [19]		X			X													
2015	ABC-MO [53]										X	X	X						
2016	EBC [20]		X	X															
2016	BAT [28]				X	X													
2017	LB-ACO [41]		X			X													
2017	LBA_HB [42]		X			X	X												
2017	Krill-LB [29]				X		X	X						X			X	X	
2017	ADF [23]														X	X			
2017	MO-ACO [17]		X			X				X									
2017	LOA [31]		X			X				X									X
2018	Hybrid2 [15]									X					X	X			X
2019	CSSA [4]		X			X				X						X			X
2019	H-BAC [49]		X		X		X									X			X
2019	HPA & HHA [13]		X							X									
2019	MSDE [36]		X		X	X													
2019	MBO & MBO-ABC [5]		X			X													

TABLE IV
EVALUATION METRICS USED FOR VM LOAD BALANCING

Year	Algorithm	Execution Time	No. of Migrations	Energy Load	CPU Load	Memory Load	Network
2013	Bee-MMT [47]		X				
2014	ERU [43]	X		X			
2015	ACO-VMM [14]				X	X	X
2015	COA-MMT [26]		X	X			
2016	ACO-MMAS-GA [48]		X				
2016	FFO-EVMM [22]			X			
2018	D-Crow [34]	X			X	X	X
2018	ELB [24]	X		X	X	X	
2019	OH_BAC [5]				X	X	X

TABLE III
CHARACTERISTICS CONSIDERED DURING OPTIMISATION FOR PM LOAD BALANCING

Year	Algorithm	No. of Migrations	Execution Time	Degree of Imbalance	SD	SLATAH	PDM	SLAV	Energy	Workload %	Cost	Priority	Hosts per VM	CPU Usage	Memory Usage	Shutdowns	Complexity	No. of Nodes
2013	Bee-MMT [47]	X				X	X	X	X									
2014	ERU [43]		X						X									
2015	ACO-VMM [14]	X			X			X										
2015	COA-MMT [26]	X				X	X	X	X									
2016	ACO-MMAS-GA [48]	X						X										
2016	FFO-EVMM [22]	X							X				X					
2018	D-Crow [34]			X					X		X	x						
2018	ELB [24]	X							X	X				X	X			X
2019	OH_BAC [5]	X				X	X	X	X							X	X	

TABLE V
EVALUATION METRICS USED FOR PM LOAD BALANCING

Year	Algorithm	None	FCFS	RR	SJF	Throttled	BRS	DLB	SHC	NSDA-II	ACO	ACO-RE	PB-ACO	ABC	HBB-LB	Hybrid	Krill	MSDE	Fuzzy-GSO	WOA	MSA	GA	HFKCS	PSO	Min-Min
2012	modified ACO [12]	X																							
2012	improved ABC [39]						X							X											
2013	HBB-LB[18]		X	X				X																	
2014	proposed ACO [52]		X						X		X											X			
2014	ACO-LB [44]		X								X	X													
2014	HBB-LBP [21]		X												X										
2014	Hybrid [16]		X								X			X											
2015	BCB-LB [19]		X																						
2015	ABC-MO [53]		X											X											
2016	EBC [20]														X										
2016	BAT [27]					X													X						
2017	LB-ACO [41]									X															
2017	LBA_HB [42]		X	X		X					X			X	X										
2017	Krill-LB [29]					X									X		X								
2017	ADF [23]										X														
2017	MO-ACO [17]										X														X
2017	LOA [31]																					X		X	
2018	Hybrid2 [15]										X														
2019	CSSA[4]										X											X	X	X	
2019	H-BAC[49]										X			X		X									
2019	HPA & HHA [13]	X																							
2019	MSDE[36]			X	X															X	X			X	
2019	MBO & MBO-ABC [38]		X	X							X		X					X							X

TABLE VI
ALGORITHMS EACH ARTICLE COMPARED TO IN THE CASE OF BALANCING VM

Year	Algorithm	RR	ACO	ACO-VMM	ABC	Bee-MMT	H_ABC	FFD	DA	DVFS	LR-MMT	IQR-MMT	MAD-MMT	LR-RS	IQ-RS	MAD-RS	THR-RS	LR	MEGSA-VMM
2013	Bee-MMT[47]									X	X	X	X						
2014	ERU [43]		X					X											
2015	ACO-VMM [14]													X	X	X	X		
2015	COA-MMT [26]					X					X	X	X						
2016	ACO-MMAS-GA [48]			X															
2016	FFO-EVMM [22]							X											
2018	D-Crow[34]		X						X									X	X
2018	ELB [24]	X	X					X											
2019	OH_BAC [5]		X		X		X												

TABLE VII
ALGORITHMS EACH ARTICLE COMPARED TO IN THE CASE OF BALANCING PM