

Load Balancing Based on Optimization Algorithms: An Overview

Fatma Mbarek and Volodymyr Mosorov

*Institute of Applied Computer Science at the Faculty of Electrical, Electronic, Computer and Control Engineering,
Lodz University of Technology, Lodz, Poland*

<https://doi.org/10.26636/jtit.2019.131819>

Abstract—Combinatorial optimization challenges are rooted in real-life problems, continuous optimization problems, discrete optimization problems and other significant problems in telecommunications which include, for example, routing, design of communication networks and load balancing. Load balancing applies to distributed systems and is used for managing web clusters. It allows to forward the load between web servers, using several scheduling algorithms. The main motivation for the study is the fact that combinatorial optimization problems can be solved by applying optimization algorithms. These algorithms include ant colony optimization (ACO), honey bee (HB) and multi-objective optimization (MOO). ACO and HB algorithms are inspired by the foraging behavior of ants and bees which use the process to locate and gather food. However, these two algorithms have been suggested to handle optimization problems with a single-objective. In this context, ACO and HB have to be adjusted to multi-objective optimization problems. This paper provides a summary of the surveyed optimization algorithms and discusses the adaptations of these three algorithms. This is pursued by a detailed analysis and a comparison of three major scheduling techniques mentioned above, as well as three other, new algorithms (resulting from the combination of the aforementioned techniques) used to efficiently handle load balancing issues.

Keywords—ant colony optimization, honey bee, load balancing, multi-objective optimization.

1. Introduction

Load balancing is an important element in distributed and parallel environments, as it is used to achieve the maximum utilization of resources and to avoid the overwhelming of nodes. It is also used to provide scalability and availability of systems, to reduce response time and to avoid network bottlenecks. Therefore, it is important to improve the performance of load balancing schemes in order to ensure that good overall performance of the system may be achieved.

Several techniques are used to implement load balancing, depending on the current status of the system. These techniques can be categorized into two types: static and dynamic algorithms. In the case of static algorithms, prior knowledge of the system is needed, while dynamic algorithms depend on the current system state [1], [2]. Ant

colony optimization (ACO), honey bee (HB) and multi-objective optimization (MOO) are considered to be three dynamic load balancing algorithms. The ACO algorithm relies on a meta-heuristic approach that is designed to solve the most challenging concern of combinatorial optimization problems, as well as of network applications, such as routing and load balancing [3]. Thus, the aim of the meta-heuristic algorithm is to supply high quality solutions within a feasible lead time. ACO algorithms may be applied to solve different problems, such as the traveling salesman problem (TSP), vehicle routing, sequential ordering, scheduling, quadratic assignment and so on. The HB algorithm is a search method that is used for both continuous and combinatorial optimization. The MOO algorithm is known as Pareto optimization and is classified as a multiple-criteria decision analysis (MCDA). The MOO algorithm is used in connection with optimization problems that include more than one conflicting objective function to be optimized simultaneously [4]. Multi-objective problems, such as routing in communication networks, compressor design, engineering and logistics, require a number of objective functions for simultaneous optimization.

In this paper, we describe the use of three main optimization algorithms, i.e. ACO, MOO and HB, as a solution for load balancing. The adaptation of these three algorithms is demonstrated as well, resulting in three new techniques. These new techniques include multi-objective ant colony optimization (MOACO), multi-objective honey bee (MOHB) and ANT-BEE algorithms. We also provide a detailed comparison and analysis of previous works involving these six algorithms.

The remaining parts of this paper are organized as follows. Section 2 gives an overview of related work. Section 3 demonstrates six optimization techniques used for load balancing. Section 4 presents the qualitative parameters and policies for load balancing techniques, while Section 5 contains a theoretical and experimental comparison of optimization algorithms. Finally, conclusions are given in Section 6.

2. Related Work

The existing research concerning load balancing solutions and based on optimization algorithms can be classified into

three categories: methods based on ACO, HB and MOO algorithms.

Several research projects have been conducted with regard to load balancing – a phenomenon that constitutes an important aspect in a distributed system. These projects aim to solve optimization problems using ACO. Zhang *et al.* [5] proposed a load balancing mechanism based on the ant colony and complex network theories for cloud environments. They described underload and overload balancing methods. Kumar *et al.* [6] implemented the ant colony algorithm in cloud computing using artificial ants in order to understand how they find the shortest path facilitating better performance. In [30], an improved ant colony algorithm was proposed to solve the main issues of ACO, such as ACO's dependence on the initial conditions. This dependence can affect the convergence speed and the final optimal solution.

In [7], an effective load balancing algorithm was developed using the ACO algorithm in order to maximize or minimize different performance parameters in cloud environments, such as CPU load, network delay and memory capacity. Kun *et al.* [8] proposed a task scheduling policy based on a load balancing ACO (LBACO) algorithm for achieving optimized resource allocation to each task in a dynamic cloud system. This scheduling strategy is to balance the entire load system while minimizing the completion span of a given set of tasks.

Scheduling methods based on the MOO algorithm constitute another category. Arun *et al.* [9] proposed a new load balancing algorithm for mobile cloud networks founded on the MOO algorithm and the genetic algorithm. This algorithm proposes a model for efficient resource allocation in cloud environments. Li *et al.* [10] suggested a hybrid VM live migration algorithm based on the MOO approach, in order to speed up the virtual machine load rebalancing process.

In addition to solving load balancing issues in distributed systems, other efforts based on the HB algorithm have been taken as well. Paper [11] presented a load balancing method based on the BC algorithm. This method employs a resource cost model in which the cloud is partitioned into several sectors.

3. Load Balancing Optimization Techniques

3.1. Optimization Algorithms

In this section, we present a review of existing optimization methods inspired by the behavior of insect colonies, such as ACO and HB algorithms. The MOO algorithm is described as well.

3.1.1. Ant Colony Optimization Algorithm

The ACO approach [12]–[14] is based on the principle of controlling the behavior of a natural system. ACO is

a meta-heuristic algorithm that is presented in Algorithm 1. It is inspired by the observation of real ant colonies during foraging, and by their innate ability to find the optimal path between their nest and the food source. Ants find the shortest path using stigmergy. It is a consensus social network mechanism where individual organisms interact with each other through the modifications of their environment. Thus, it is a form of self-organization that can produce complex structures and behaviors without the need for planning or direct communication.

Algorithm 1. The ACO meta-heuristic algorithm

```

1: procedure ACO-Metaheuristic
2:   Initialization;
3:   while stopping_criterion_is_not_met do
4:     Construct_Ant_Solutions();
5:     Pheromone_evaporation();
6:     Daemon_actions(); {optional}
7:   end while
8: end procedure

9: procedure Construct_Ant_Solutions
10:  Initialization;
11:  ant  $k = 1, \dots, n_k$ ;
12:   $t = 0$ ;
13:   $x = origin\_node$ ;
14:   $Dest = target\_node$ ;
15:   $S_{best}(t) = 0\{Short\_path\}$ ;
16:   $L_{best}(t) = 0\{Short\_path\_Length\}$ ;
17:   $M = Manage\_private\_ant\_memory$ ();
18:  while  $x \neq Dest$  do
19:    for all ant  $k = 1, \dots, n_k$  do
20:       $S_k \leftarrow origin\_node$  of ant  $k$ ;
21:       $x \leftarrow origin\_node$  of ant  $k$ ;
22:      repeat
23:        Build set of neighbors for  $x$  using  $M$ ;
24:        Select next node  $y$  and assign the probability  $p_{xy}$ ;
25:        Add  $y$  to the list  $S_k(t)$ ;
26:         $x = y$ ;
27:      until Full path construction
28:      if  $L_k(t) < L_{best}(t)$  then
29:         $\triangleright L_k(t)$  length of the tour defined in  $S_k(t)$ 
30:         $S_{best}(t) \leftarrow S_k(t)$ ;
31:      end if
32:    end for
33:    for each edge  $(x, y)$  do
34:      Pheromone_evaporation();
35:    end for
36:     $t \leftarrow t + 1$ ;
37:  end while
38:  Return  $S_{best}(t)$ ;
39: end procedure

```

In artificial swarm intelligence, the optimization problem is concerned with the collective behaviors of ants looking

for food. This means that food needs to be collected while minimizing the energy required to accomplish that task. The biological analogy of the ACO algorithm consists in the fact that ants are considered to be simple agents for an iterative process, used to construct potential solutions, and are termed artificial ants. The indirect communication within a colony of simple agents relies on artificial pheromone trails.

There are two main phases in the ACO algorithm: tour construction and pheromone updates.

Tour construction: Suppose an artificial ant k builds a tour from node x to y at time t . At each construction step, a probabilistic action choice is applied by the ant k , which is termed *the transition probability* [3]. The transition probability of the route (x, y) is noted as $P_{xy}^k(t)$ and is defined as follows:

$$P_{xy}^k(t) = \begin{cases} \frac{[\tau_{xy}(t)]^\alpha \cdot [\eta_{xy}(t)]^\beta}{\sum_{z \in J_x^k} [\tau_{xz}(t)]^\alpha \cdot [\eta_{xz}(t)]^\beta}, & y \in J_x^k \\ 0, & y \notin J_x^k \end{cases}, \quad (1)$$

where J_x^k is the set of possible displacements that remain to be visited by ant k positioned at node x , η_{xy} is a heuristic that represents the *a priori desirability* of the move from x to y , typically $\eta_{xy} = \frac{1}{d_{xy}}$, where d is the distance of the route (x, y) , τ_{xy} is the pheromone intensity of the move from x to y , α and β are two positive parameters used to control the influence of pheromone concentrations and heuristic information. As given in Eq. (1), the transition probability of path (x, y) rises with the values of heuristic information η_{xy} and pheromone trail τ_{xy} .

Pheromone trails update: Once the edge tour is constructed, each ant has deposited a certain amount of pheromone. When all ants have built their tours, the pheromone is updated at all edges due the evaporation mechanism which is presented by:

$$\tau_{xy}(t) \leftarrow (1 - \rho) \tau_{xy}(t), \quad (2)$$

where $1 - \rho$ is the residual coefficient of the pheromone decay rate, $0 < \rho \leq 1$. Pheromone evaporation enables to avoid bad decisions taken beforehand. After evaporation, all ants drop the pheromone at the edge (x, y) at iteration $t + 1$:

$$\tau_{xy}(t+1) \leftarrow \tau_{xy}(t) + \Delta\tau_{xy}(t), \quad (3)$$

$$\Delta\tau_{xy}(t) = \sum_{k=1}^m \Delta\tau_{xy}^k(t), \quad (4)$$

with

$$\Delta\tau_{xy}^k(t) = \begin{cases} \frac{Q}{L_k}, & \text{if ant } k \text{ passes } (x, y) \\ 0, & \text{otherwise} \end{cases}, \quad (5)$$

where $\tau_{xy}(t)$ is the pheromone concentration on the edge (x, y) and $\Delta\tau_{xy}(t)$ represents the total amount of pheromones dropped by all ants on edge (x, y) , as given

in Eq. (4), with m being the number of ants used for iteration t . $\Delta\tau_{xy}^k(t)$, defined as in Eq. (5), is the pheromone rate deposited by ant k on the visited edge (x, y) . L_k is the path length performed by ant k , and Q is a constant.

3.1.2. Honey Bee Algorithm

The honey bee approach is a search algorithm for load balancing. The bee algorithm (BA) is designed based on the food foraging behavior of honey bee colonies. In nature, bees are social and domestic insects. Honey bee colonies are well-run organizations [15] which live on nectar serving as their source of energy. The structure of a single colony of bees includes a queen (a single fertile female within the colony), drones (thousands of males to fertilize the queen's eggs), workers (the largest population of sterile females within the colony), and broods (young bee larvae) [16]. The bees perform several functions within the group, such as rearing the young, maintaining the hive or collecting nectar. They search for the best food source, selecting it from many sites, taking into consideration both speed and accuracy. In beehives, honey bees are categorized into two classes:

- scout bees: are the workers performing random searches for new flower patches. When they find the food source, they come back to the hive for depositing their nectar. Then they inform the forager bees by performing a waggle dance in front of the hive;
- forager bees: are the bees which follow the scout bees to the food location and begin to harvest the nectar. The foragers may waggle dance to recruit other bees to proceed to abundant food patches.

Algorithm 2. The bee algorithm

- 1: Initialization;
 - 2: $i = 0$;
 - 3: $MaxIr = 0; \{Number_iterations\}$
 - 4: $FitnessValue(i) = 0; \{Fitness_value_result\}$
 - 5: Place all bees, $b = 1, \dots, N$;
 - 6: \triangleright Generate the initial population;
 - 7: Evaluate fitness of the population;
 - 8: Sort the initial population based on the fitness result;
 - 9: $MaxIr \leftarrow MaxIr + 1$;
 - 10: **while** $i \leq MaxIr$ or $FitnessValue(i) - FitnessValue(i - 1) \leq Error$ **do**
 - 11: $i \leftarrow i + 1$;
 - 12: Select elite patches and non-elite patches for neighborhood search;
 - 13: Recruit the forager bees for selected patches;
 - 14: Evaluate the fitness value of each patch;
 - 15: Select the fittest bee from each patch;
 - 16: Assign remaining bees to search randomly for elite patch;
 - 17: Evaluate the fitness value of elite patch;
 - 18: **end while**
-

The communication language used by bees is fascinating. It is based on two types of dances: the round dance and the waggle dance. The round dance is short and attracts other foragers to learn the approximate distance from the hive to the food site. This distance is approximately 50 m [17]. The main purpose of the waggle dance is to advertise news about the quality, quantity, direction of the food source and the distance between the beehive and the food, provided it is greater than 50 m [18]. The pseudo-code of bee algorithm is presented as Algorithm 2.

3.1.3. Multi-Objective Optimization Algorithm

ACO considers optimization problems with one objective function which is addressed to a unique optimal solution, while MOO gives a countless combination of solutions [19], [20]. In MOO, the solution is termed *Pareto optimal* or *non-dominated*.

The general single-objective optimization problem is presented by Definition 1. Definition 2 describes multi-objective optimization problems.

Definition 1: A single-objective optimization problem [21]–[23] is defined as:

$$\left. \begin{array}{l} \text{Minimize/Maximize} \quad f(x), \quad x = (x_1, \dots, x_n) \\ \text{Subject to} \quad \left. \begin{array}{l} g_i(x) \leq 0, \quad i = \{1, \dots, m\} \\ h_j(x) = 0, \quad j = \{1, \dots, l\} \end{array} \right\}, \quad (6) \end{array} \right\}$$

where f is the objective function that is discrete or continuous. $x \in \Omega$ is an n -dimensional vector of decision variables from the universe Ω . Constraints $g_i(x) \leq 0$ and $h_j(x) = 0$ must be achieved while optimizing $f(x)$.

Definition 2: A multi-objective optimization problem [24], [25]. The goal of MOO problems is to optimize, simultaneously, p objective functions by setting up a vector function $F(x)$ that designated: $f_1(x), f_2(x), \dots, f_p(x)$, where p is the number of objective functions. $g_i(x)$ and $h_j(x)$ are the achieved constraints for optimizing $F(x)$:

$$F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_p(x) \end{bmatrix}. \quad (7)$$

Obviously, this can be written as:

$$F(x) = [f_1(x), f_2(x), \dots, f_p(x)]^T.$$

The general multi-objective optimization problem (MOOP) is defined as:

$$\text{MOOP :} \quad \left\{ \begin{array}{l} \text{Minimize/Maximize} \quad F(x), \quad x = [x_1, \dots, x_n] \\ \text{Subject to} \quad g_i(x) \leq 0, \quad i = \{1, \dots, m\} \\ h_j(x) = 0, \quad j = \{1, \dots, l\} \end{array} \right., \quad (8)$$

where $x \in \Omega$, $x = [x_1, \dots, x_n]$ is an n -dimensional decision variables vector (also called *design variables*) from the universe Ω . A solution of MOOP minimizes or maximizes the components of the vector $F(x)$.

3.2. Load Balancing Based on Optimization Methods

This section presents the results of combining the three algorithms mentioned in the Subsection 3.1.

3.2.1. Multi-Objective Honey Bee Algorithm

The MOHB algorithm is proposed by Soni *et al.* [26]. This algorithm optimizes system performance based on multi-objective requirements, by optimizing the fitness value of task priority, load and execution error. The proposed technique pursues the steps shown in the flow chart of Fig. 1.

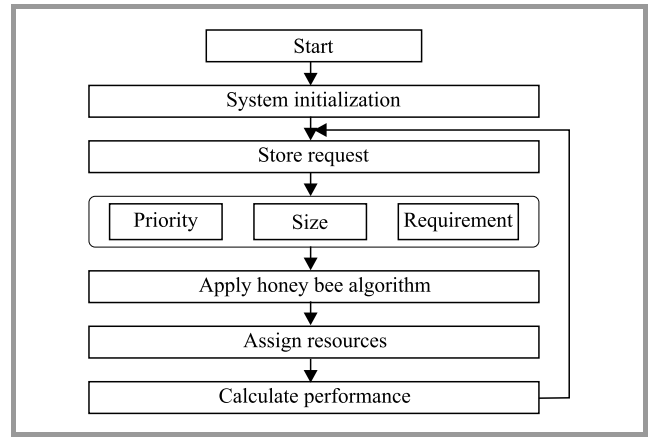


Fig. 1. MOHB algorithm flow chart.

3.2.2. Multi-Objective Ant Colony Optimization Algorithm

In this section, we define the MOACO method (summarized as Algorithm 3) that extends Eq. (1) to obtain multiple pheromone matrices [27], [28]. The transition probability of Eq. (1) has shifted to:

$$P_{xy}^k(t) = \begin{cases} \frac{[\eta_{xy}(t)]^\beta \prod_{l=1}^n [\tau_{xy}^l(t)]^{\alpha_l}}{\sum_{z \in J_x^k} [\eta_{xz}(t)]^\beta \prod_{l=1}^n [\tau_{xz}^l(t)]^{\alpha_l}}, & y \in J_x^k \\ 0, & y \notin J_x^k \end{cases}, \quad (9)$$

where τ_{xy}^l is the amount of l -pheromone at edge (x, y) for l -th objective. Supposing that n sub-objectives require to be optimized. η_{xy} is the attractiveness of the move from node x to node y , J_x^k is the feasible nodes for ant k at node x . β sets out the relative heuristic information of η_{xy} while $\alpha_l \in \mathbb{R}_0^+(l = 0, 1, \dots, n)$ are heuristic parameters to control the influence of the appropriate objective.

Algorithm 3. The MOACO algorithm

```

1: Place all ants,  $k = 1, \dots, n_k$ ;
2: Initialize parameters, i.e.  $\alpha, \beta, \rho, \tau_0, n_k, Q$ ;
3:  $x = origin\_node$ ;
4:  $S_{best}(t) = 0 \{Short\_path\}$ ;
5:  $L_{best}(t) = 0 \{Short\_path\_Length\}$ ;
6:  $t = 0$ ;
7: for each edge  $(x, y)$  do
8:    $\tau_{xy}(t) = \tau_0$ ;
9:    $\triangleright$  Initialize pheromones to  $\tau_0$ 
10:   $\eta_{xy}(t) = \frac{1}{d_{xy}}$ ;
11:   $\triangleright d_{xy}$  is the distance between the link  $(x, y)$ 
12: end for
13: while stopping\_criterion\_is\_not\_met do
14:   for all ant  $k = 1, \dots, n_k$  do
15:     $S_k \leftarrow origin\_node$  of ant  $k$ ;
16:     $x \leftarrow origin\_node$  of ant  $k$ ;
17:    repeat
18:      Calculate  $P_{xy}^k(t)$  using Eq. (9);
19:      Add  $y$  to the list  $S_k(t)$ ;
20:       $x = y$ ;
21:    until Full path construction
22:    if  $L_k(t) < L_{best}(t)$  then
23:       $\triangleright L_k(t)$  length of the tour defined in  $S_k(t)$ 
24:       $S_{best}(t) \leftarrow S_k(t)$ ;
25:    end if
26:  end for
27:  for each edge  $(x, y)$  do
28:    Apply evaporation mechanism using Eq. (10);
29:    Calculate  $\Delta\tau_{xy}^l(t)$  using Eq. (11);
30:    Update the pheromone trail using Eq. (3);
31:  end for
32:   $t \leftarrow t + 1$ ;
33: end while
34: Return  $S_{best}(t)$ ;

```

After each cycle, the pheromone level of each objective is updated at edge (x, y) due to evaporation, using the following formula:

$$\tau_{xy}^l(t) \leftarrow (1 - \rho_l) \tau_{xy}^l(t), \quad (10)$$

where $\rho_l \leq 1$ ($l = 0, 1, \dots, n$) is the pheromone decay rate for the l -th objective ($1 - \rho_l$ is the persistence trail).

At every cycle for each objective, the ants left a certain quantity of pheromones through link (x, y) using the following rule:

$$\Delta\tau_{xy}^l(t) = \sum_{l \in \Omega} \frac{Q}{f^l T_k}, \quad (11)$$

where $\Delta\tau_{xy}^l(t)$ is the quantity of pheromone dropped by ants. Ω is the set of ants that passed through path (x, y) . Q is a constant, T_k is the non-dominated solution (in each iteration, non-dominated solutions are stocked in archive Γ) and f^l is the objective value for l -th objective functions.

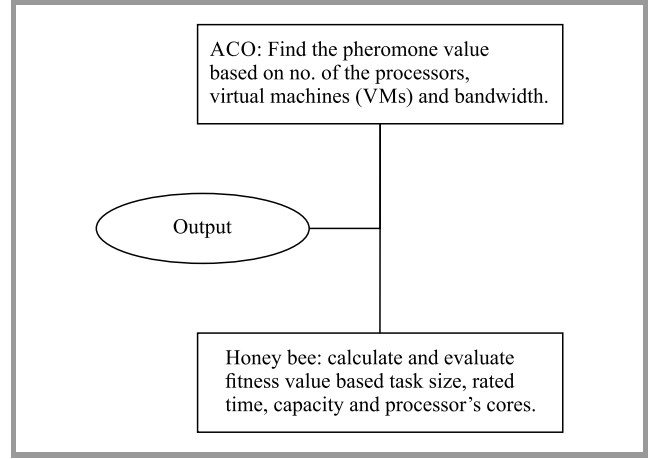


Fig. 2. Flow chart of ant-bee algorithm.

Algorithm 4. The ant-bee algorithm

```

1: Initialization;
2:    $\triangleright$  Initially all virtual machines (VMs) possess request 0;
3: Maintain an index table with VM_id and VM_request;
4: Schedule new request to VM;
5:    $VM \leftarrow current\_request0$ ;
6: Update the index table;
7: if VMs are not available then
8:   Create random population of ants with same pheromone value and place parameter;
9:   Sort the ants randomly for search;
10:  Store the current value of optimal solution;
11:  Update pheromone table;
12: end if
13: if all ants complete their tour then
14:   Compare updated pheromone table;
15:   Select the best solution;
16:   Choose the optimal node based on pheromone table;
17:   Assign best solution to optimal node;
18:   Calculate fitness function;
19:   if fitness_function < threshold then
20:     Migrate task to optimal node;
21:   else
22:     Find next optimal solution;
23:   end if
24: end if
25: Return best solution;

```

3.2.3. Ant-Bee Algorithm

Gothi *et al.* [29] suggested the ant-bee algorithm for load balancing in cloud computing. This algorithm combines both ACO and HB algorithms to obtain an efficient and feasible optimization algorithm. The generic pseudo-code of this proposed technique is presented in Algorithm 4, while Fig. 2 contains the corresponding flow chart.

4. Load Balancing Policies and Performance Parameters

4.1. Dynamic Load Balancing Policies

Dynamic load balancing algorithms can be defined by their implementation of strategies or policies, such as:

- **Load estimation policy.** It determines how to estimate the workload of a particular node. The load may vary depending on the remaining service time, which can be calculated based on:
 - setting up the periodic CPU state check timer (idle/busy),
 - the number of CPU cycles executed per unit of real time,
 - speed of the CPU node,
 - resource demands;
- **Process transfer policy.** It determines if the process will be executed remotely or locally;
- **Location policy.** It selects which node will run the remote process;
- **Priority assignment policy.** It determines which local or remote processes have a higher priority on a particular node;
- **State information exchange policy.** It is dynamic load balancing information strategy which collects information about all nodes in the system;
- **Migration limiting policy.** It helps to control trashing and it sets the number of times that the process may migrate from one node to another;
- **Selection policy.** It is a processor matching policy which selects the processors involved in the load exchange;
- **Transfer policy.** It determines if the process will be migrated remotely or locally.

4.2. Performance Measurement

Load balancing techniques are particularly useful in high-speed networks required by emerging high performance applications, as they enable to distribute the workload evenly across the resources. Performance metrics used in load balancing algorithms include the following parameters [30], [31]:

- **Nature.** Determines the type of load balancing algorithms: static or dynamic;
- **Overload rejection.** It is a parameter used to decide the maximum load supported by any server [23]. Overload rejection measures are discontinued if the overload condition ends. Then, load balancing is shut down after a short period of time;
- **Mean response time.** It refers to the period of time that needs to elapse in order for a process to be

executed in full. To achieve a good response time, an appropriate load balancing algorithm must be used to distribute processes among all nodes within a system. The response time can be affected by several factors, such as number of users, network bandwidth, number of submitted requests and mean thinking time. The average response time can be calculated using the following formula:

$$T_{response} = \frac{n}{p} - T_{think} , \quad (12)$$

where:

- $T_{response}$ is the response time in seconds,
 - n is the number of users,
 - p is the number of processes per second the server receives,
 - T_{think} is the mean of thinking time (which the time between one request and the next) in seconds;
- **Resource utilization.** It includes automatic load balancing. In a distributed environment, the number of processes might be large and that demands more processing power. Resource utilization is relied upon more frequently in dynamic algorithms than in static ones;
 - **Processor utilization** refers to the amount of time over which a workload is handled by the central processing unit (CPU);
 - **Fault tolerance.** This parameter enables the algorithm to work properly if a failure occurs. It determines the value of the system's performance. If the performance of an algorithm degrades when a fault of any type occurs, it causes a total failure in load balancing;
 - **Predictability.** It determines the degree of conformity of the calculated outcome of the algorithm. In static load balancing algorithms, the outcome is predictable because it is fixed before the execution stage, while the outcome is unpredictable in dynamic load balancing, as assumptions are made at run time;
 - **Stability.** It is the delay required for transferring information among processors and obtaining faster performance. In static load balancing algorithms, there is more stability than in their dynamic load balancing counterparts. This is explained by the information exchanged between processors in dynamic load balancing algorithms;
 - **Adaptability.** It checks if an algorithm is capable of facing a change in the number of processes. Dynamic algorithms are perfectly adaptive, while static algorithms are not;
 - **Process migration.** This parameter determines when a system should forward a process. That means whether it should be created locally or remotely.

5. Comparison and Analysis of Optimization Algorithms

In this section, a study was performed regarding various load balancing techniques based on ACO, MOO, HB, MOACO, MOHB and ant-bee algorithms in a distributed system.

The analysis and the comparison of the above mentioned algorithms were performed based on their purpose, the methods used, their advantages and disadvantages, or experimental results.

Table 1 presents a comparison between ACO, MOO and MOACO algorithms, while the comparison of HB, MOHB and ant-bee algorithms is shown in Table 2.

In [32], the proposed SALB algorithm has better performance than the ACO algorithm. The MBBO/DE algorithm offers better results than the ACO algorithm and the genetic algorithm [33]. As can be seen in Table 1, the MOACO algorithm provides better results than ACO, first fit (FF) and greedy scheduling (GS) algorithms [34]. From the comparison shown in Table 2, it can be concluded that the LBA-HB algorithm is more efficient than round robin, modified throttled, ACO, artificial bee colony (ABC) and honey bee

Table 1
Load balancing based on ACO, MOO, and MOACO algorithms

Reference title	Purpose	Methods	Advantages	Disadvantages
ACO algorithm				
Effective scheduling algorithm for load balancing (SALB), using ant colony optimization in cloud computing [32].	To balance the load of the entire system and to maximize or minimize different parameters, such as performance and energy use.	Scheduling algorithm for load balancing (SALB), using ACO.	<ul style="list-style-type: none"> • It is efficient in finding the overloaded node within a minimum time period, • Good response time, • Less energy consumption. 	Poorer performance. It means that the performance can be increased by varying different parameters.
MOO algorithm				
Multi-objective optimization algorithm based on BBO for the problem of virtual machine consolidation [33].	To achieve good load balancing, better performance and to reduce power consumption.	Multi-objective optimization algorithm named MBBO/DE.	<ul style="list-style-type: none"> • Lower power consumption, • Ability to find optimal solutions, • Acceptable time cost. 	<ul style="list-style-type: none"> • For load balancing, MBBO/DE algorithm provides same results as ACO algorithm in scenario 1, • Migration time is longer than in other algorithms simulated in scenario 2.
MOACO algorithm				
Multi-objective virtual machine placement for load balancing [34].	To solve the virtual machine placement problem.	Multi-objective Ant Colony Optimization (MOACO) algorithm.	Optimize multiple objectives to balance the load in different scales.	The proposed MOACO algorithm provides poor results for the two objectives OBD and IBD.

Table 2
Load balancing based on HB, MOHB and ant-bee algorithms

Reference authors	Purpose	Methods used	Experimental results
HB algorithm			
W. Hashem <i>et al.</i> [35]	To avoid overutilization of resources and to minimize execution time.	Load balancing algorithm based on honey bee behavior (LBA_HB).	<ul style="list-style-type: none"> • Shorter execution time, • Efficient resources utilization, • Lower degree of imbalance, • Lower makespan.
MOHB algorithm			
A. Soni <i>et al.</i> [26]	To schedule workload and minimize the total processing cost.	Bee colony based multi-objective algorithm.	The execution time is longer due to loop parameters.
Ant-bee algorithm			
M. Kashefikia <i>et al.</i> [36]	To find different routes in order to avoid network congestion.	Multiple ant bee colony optimization (MABC) algorithm.	<ul style="list-style-type: none"> • Offers better effects during network failures, • More suitable for unstable networks.

behavior load balancing (HBB-LB) algorithms [35]. And the bee colony-multi-objective algorithm provides better results than the single objective bee colony algorithm [26]. From the experimental results shown in [36], the multiple ant bee colony optimization (MABC) algorithm is significantly more effective in an unstable network than the multiple ant colony optimization (MACO) algorithm.

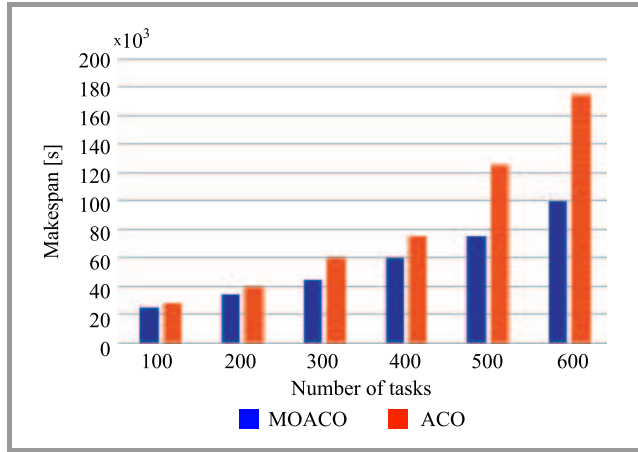


Fig. 3. Mean response time.

The experimental results of ACO and MOACO algorithms [37], related to qualitative performance parameters, are shown in Figs. 3 and 4.

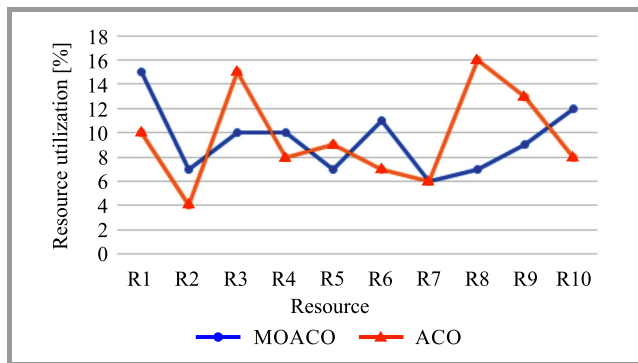


Fig. 4. Resource utilization.

Figure 3 shows that the performance of MOACO is the best when the number of tasks is bigger than in ACO. Figure 4 shows that MOACO offers a lower resource utilization rate with different resources.

In [38], an experiment was performed using ACO and HB algorithms and evaluating the execution time covering 10, 50, 200 and 1000 iterations. Its results are shown in Fig. 5. ACO offers a better execution time than HB for a small number of iterations. However, when the number of iterations increased, both ACO and HB algorithms were almost equal in terms of the computation time.

In this paper, the significance of load balancing optimization algorithms is discussed through an analytic previous works. The results are summed up in Table 3, showing the conclusions concerning three main optimization algo-

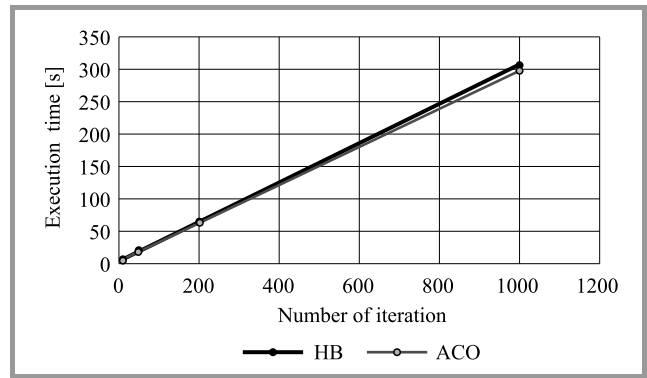


Fig. 5. Resource utilization.

Table 3
Comparative analysis of optimization algorithms

Parameters	ACO	HB	MOO
Nature	Dynamic	Dynamic	Dynamic
Throughput	High	High	High
Fault tolerance	High	Moderate	Low
Response time	High	High	High
Resource utilization	Moderate	High	High
Migration time	Low	Low	High
Complexity	Moderate	High	High
Speed	Low	Moderate	Low
Adaptability	More	More	More
Predictability	Low	Low	Low
Stability	Low	Low	Low
Waiting time	Low	Low	Low

gorithms: ACO, HB and MOO, as well as their qualitative, performance-related parameters.

6. Conclusions

This paper addresses load balancing-related optimization problems impacting the performance of entire systems. The research provides a comprehensive review of the optimization algorithms used to balance system loads.

Furthermore, it describes three well known algorithms used in engineering applications, namely ACO, HB and MOO. A consolidation of the three methods has been performed in order to present MOACO, MOHB and ant-bee algorithms. The advantages and disadvantages of the different algorithms are discussed as well in the paper. Some optimization techniques suffer from specific limitations, e.g. the HB algorithm offers the best results only at a particular distance (short path), while ACO suffers from speed-related problems and convergence.

As far as theoretical considerations are concerned, it can be concluded that the three new algorithms based on the combination of optimization methods are self-adaptive and more feasible than the original techniques.

Acknowledgements

This work was financed by the Lodz University of Technology, Faculty of Electrical, Electronic, Computer and Control Engineering, and was conducted as part of its statutory activity (project no. 501/12- 24-1-5418).

References

- [1] F. Mbarek and V. Mosorov, "Load balancing algorithms in heterogeneous Web cluster", in *Proc. Intern. Interdiscipl. PhD Worksh. IIPhDW 2018*, Swinoujście, Poland, 2018, pp. 205–208 (doi: 10.1109/IIPhDW.2018.8388358).
- [2] A. A. Rajguru and S. S. Apte, "A comparative performance analysis of load balancing algorithms in distributed system using qualitative parameters", *Int. J. of Recent Technol. and Engin. (IJRTE)*, vol. 1, pp. 175–179, 2012 (ISSN: 2277-3878).
- [3] C. Blum, "Ant Colony Optimization: Introduction And Recent Trends", *Phys. of Life Rev.*, vol. 2, no. 4, pp. 353–357, 2005 (doi: 10.1016/j.phrev.2005.10.001).
- [4] D. Constantinou, "Ant colony optimisation algorithms for solving multi-objective power-aware metrics for mobile ad hoc networks", Ph.D. Thesis, University of Pretoria, Hatfield, Pretoria, South Africa, August 2010 [Online]. Available: <http://hdl.handle.net/2263/25981>
- [5] Z. Zhang and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation", in *Proc. 2nd Int. Conf. on Industr. Mechatron. and Autom. ICIMA 2010*, Wuhan, China, 2010, pp. 240–243 (doi: 10.1109/ICINDMA.2010.5538385).
- [6] R. Kumar and G. Sahoo, "A load balancing using ant colony in cloud computing", *Int. J. of Inform. Technol. Conver. and Serv. (IJITCS)*, vol. 3, no. 5, pp. 1–5, 2013 (doi: 10.5121/ijitcs.2013.3501).
- [7] R. Mishra and A. Jaiswal, "Ant colony Optimization: a solution of load balancing in cloud", *Int. J. of Web and Semantic Technol. (IJWesT)*, vol. 3, no. 2, pp. 33–50, 2012 (doi: 10.5121/ijwest.2012.3203).
- [8] L. Kun, X. Gaochao, Z. Guangyu, D. Yushuang, and W. Dan, "Cloud task scheduling based on load balancing ant colony optimization", in *Proc. 6th Ann. ChinaGrid Conf. ChinaGrid 2011*, Liaoning, China, 2011, pp. 3–9 (doi: 10.1109/ChinaGrid.2011.17).
- [9] E. Arun, A. Reji, P. M. Shameem, and R. S. Shaji, "Novel algorithm for load balancing in mobile cloud networks: multi-objective optimization approach", *Wirel. Personal Commun.*, vol. 97, no. 2, pp. 3125–3140, 2017 (doi: 10.1007/s11277-017-4665-6).
- [10] R. Li, Q. Zheng, X. Li, and J. Wu, "A novel multi-objective optimization scheme for rebalancing virtual machine placement", in *Proc. IEEE 9th Int. Conf. on Cloud Comput. CLOUD 2016*, San Francisco, CA, USA, 2016, pp. 1–7 (doi: 10.1109/CLOUD.2016.0099).
- [11] P. Ehsanimoghadam and M. Effatparvar, "Load balancing based on bee colony algorithm with partitioning of public clouds", *Int. J. of Adv. Comp. Sci. and Appl. (IJACSA)*, vol. 9, no. 4, pp. 450–455, 2018 (doi: 10.14569/IJACSA.2018.090462).
- [12] M. Dorigo, "The ant colony optimization meta-heuristic: algorithms, applications, and advances", in *Handbook of Metaheuristics*, M. Gendreau, J.-Y. Potvin, Eds. Springer, 2003, pp. 251–285 (doi: 10.1007/0-306-48056-5_9).
- [13] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization", *Artif. Life*, vol. 5, no. 2, pp. 137–172, 1999 (doi: 10.1162/106454699568728).
- [14] D. Darquennes, "Implementation and applications of ant colony algorithms", Master Thesis, Facultés Universitaires Notre-Damedela Paix, Namur, Institut d'Informatique, 2005 [Online]. Available: <http://www.swarm-bots.org/~mdorigo/HomePageDorigo/thesis/master/DarquennesMASTER.pdf>
- [15] C. Jankowski, "Social Structure of the Honey Bee" [Online]. Available: <http://animals.mom.me/social-structure-honeybee-7317.html> (accessed on Feb. 24, 2018).
- [16] S. Bitam, "Bees life algorithm for job scheduling in cloud computing", in *Proc. 3rd Int. Conf. on Commun. and Inform. Technol. ICCIT 2012*, Hammamet, Tunisia, 2012, pp. 186–191 [Online]. Available: <https://pdfs.semanticscholar.org/1823/27d9c30c4970313704c53701100771d85bed.pdf>
- [17] O. Bin Hassan and A. S. Ahmad, "Optimum load balancing of cloudlets using honey bee behavior load balancing algorithm", *Int. J. of Adv. Res. in Comp. Sci. and Manag. Studies*, vol. 3, pp. 334–339, 2015 (ISSN: 2321-7782).
- [18] B. Yuze, M. S. Packianather, E. Mastrocinque, D. C. Pham, and A. Lambiasi, "Honey bees inspired optimization method: the bees algorithm", *Insects 2013*, vol. 4, pp. 646–662, 2013 (doi: 10.3390/insects4040646).
- [19] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms For Solving Multi-Objective Problems*, 2nd ed. Springer, 2007 (doi: 10.1007/978-0-387-36797-2).
- [20] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering", *Struct. Multidisc. Optim.*, vol. 26, no. 6, pp. 369–395, 2004 (doi: 10.1007/s00158-003-0368-6).
- [21] A. Mukerjee, R. Biswas, K. Deb, and A. P. Mathur, "Multi-objective evolutionary algorithms for the risk return trade-off in bank loan management", KanGAL Report Number 2001005, 2001 [Online]. Available: <https://www.iitk.ac.in/kangal/reports.shtml#2001>
- [22] K. Deb, "Multi-objective optimization using evolutionary algorithms: an introduction", KanGAL Report Number 2011003, pp. 1–24, 2011 [Online]. Available: <https://www.egr.msu.edu/~kdeb/papers/k2011003.pdf>
- [23] C. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer, 2002, pp. 4–13 (ISBN: 0306467623).
- [24] G. Chiandussi, M. Codegone, S. Ferrero, and F. E. Varesio, "Comparison of multi-objective optimization methodologies for engineering applications", *Comp. and Mathem. with Appl.*, vol. 63, no. 5, pp. 912–942, 2012 (doi: 10.1016/j.camwa.2011.11.057).
- [25] E. Zitzler, "Evolutionary algorithms for multi-objective optimization: methods and applications". Ph.D. Thesis, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology Zurich, 1999 [Online]. Available: <https://sop.tik.ee.ethz.ch/publicationListFiles/zitz1999a.pdf> (doi: 10.3929/ethz-a-003856832).
- [26] A. Soni, G. Vishwakarma, and Y. K. Jain, "A bee colony based multi-objective load balancing technique for cloud computing environment", *Int. J. of Comp. Appl.*, vol. 114, no. 4, pp. 19–25, 2015 (doi: 10.5120/19967-1825).
- [27] S. Srichandan, T. A. Kumar, and S. Bibhudatta, "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm", *Future Comput. and Inform. J.*, vol. 3, no. 2, pp. 1–21, 2018 (doi: 10.1016/j.fcij.2018.03.004).
- [28] P. Cardoso, M. Jesus, and A. Marquez, "MONACO – multi-objective network optimization based on an ACO", in *Proc. of 10th Enguentros de Geometria Computacional*, Sevilla, Spain, 2003, pp. 1–10, 2003.
- [29] P. B. Gothi and D. V. Vekariya, "An efficient approach for load balancing using dynamic AB algorithm in cloud computing", *Int. J. of Innov. Res. in Comp. and Commun. Engin.*, vol. 4, no. 4, pp. 7767–7773, 2016 (doi: 10.15680/IJIRCC.2016.0404283).
- [30] F. Mbarek and V. Mosorov, "A load balancing system to protect servers against DDoS attacks", in *Algorithms, Networking and Sensing for Data Processing, Mobile Computing and Applications*, A. Romanowski, D. Sankowski, and J. Sikora, Eds. Łódź: Lodz University of Technology Press, 2016, pp. 55–74 (ISBN: 9788372837387).
- [31] P. Kanungo, "Measuring performance of dynamic load balancing algorithms in distributed computing applications", *Int. J. of Adv. Res. in Comp. and Commun. Engin.*, vol. 2, no. 10, pp. 4063–4066, 2013 [Online]. Available: <https://pdfs.semanticscholar.org/ee91/3f3d20f107ae269f66adc72c4b4f6fa71993.pdf>
- [32] S. Khan and N. Sharma, "Effective scheduling algorithm for load balancing (SALB) using ant colony optimization in cloud computing", *Int. J. of Adv. Res. in Comp. Sci. and Softw. Engin.*, vol. 4, no. 2, pp. 966–973, 2014 [Online]. Available: https://pdfs.semanticscholar.org/e2cc/4722d826943c99a3bdb5eb7dde8797516a25.pdf?_ga=2.168973801.915765179.1571657419-1047092990.1571657419

- [33] Q. Zheng *et al.*, “Multi-objective optimization algorithm based on BBO for virtual machine consolidation problem”, in *Proc. 21st Int. Conf. on Paralle. and Distrib. Sys. ICPADS 2015*, Melbourne, VIC, Australia, 2015, pp. 414–421 (doi: 10.1109/ICPADS.2015.59).
- [34] F. Fang and B. B. Qu, “Multi-objective virtual machine placement for load balancing”, in *Proc. Int. Conf. on Inform. Science and Technol. IST 2017*, Wuhan, Hubei, China, 2017, pp. 1–9 (doi: 10.1051/itmconf/20171101011).
- [35] W. Hashem, H. Nashaat, and R. Rizk, “Honey bee based load balancing in cloud computing”, *KSI Trans. on Internet and Inform. Syst.*, vol. 11, no. 12, pp. 5694–5711, 2017 (doi: 10.3837/tis.2017.12.001).
- [36] M. Kashefikia, N. Nematbakhsh, and R. A. Moghadam, “Multiple ant-bee colony optimization for load balancing in packet-switched networks”, *Int. J. of Comp. Netw. and Commun. (IJCNC)*, vol. 3, no. 5, pp. 107–117, 2011 (doi: 10.5121/ijcnc.2011.3508).
- [37] L. Zuo, P. Shu, S. Dong, C. Zhu, and T. Hara, “A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing”, *Big Data Services and Computational Intelligence for Industrial Systems, IEEE Access*, vol. 3, pp. 2687–2699, 2015 (doi: 10.1109/ACCESS.2015.2508940).
- [38] M. B. Jasser, M. Sarmini, and R. Yaseen, “Ant colony optimization (ACO) and a variation of bee colony optimization (BCO) in solving TSP problem, a comparative study”, *Int. J. of Comp. Appl.*, vol. 96, no. 9, pp. 1–8, 2014 (doi: 10.5120/16819-6587).



Fatma Mbarek received her B.Sc. degree in Computer Science in 2012 and her M.Sc. degree in Computer Systems and Network Security in 2014 from the Faculty of Sciences of Gabes, Tunisia. She is currently a Ph.D. student at the Institute of Applied Computer Science at the Faculty of Electrical, Electronic, Computer and Control

Engineering in Lodz, Poland. She was an Erasmus grant-holder under the Erasmus Mundus EGOV-TN Project at Lodz University of Technology. Her current research focuses on load balancing, optimization algorithms, network security, and web servers performance.


 <https://orcid.org/0000-0002-5545-3918>

E-mail: fmbarek@kis.p.lodz.pl
Institute of Applied Computer Science
Lodz University of Technology
Stefanowskiego 18/22
Lodz, Poland



Volodymyr Mosorov received his M.Sc. and the Ph.D. degrees in Telecommunications from the Lviv Polytechnic National University, Ukraine in 1983 and 1998, respectively. In 2009 he received the D.Sc. degree (Habilitation) in Computer Science from AGH University of Science and Technology in Cracow, Poland. He has been

working at the Institute of Applied Computer Science (previously – Computer Engineering Department), at the Faculty of Electrical, Electronic, Computer and Control Engineering, TUL since 2000, currently as a Professor of the Lodz University of Technology.

 <https://orcid.org/0000-0001-6016-8671>

E-mail: mosorow@kis.p.lodz.pl
Institute of Applied Computer Science
Lodz University of Technology
Stefanowskiego 18/22
Lodz, Poland