Tasks Scheduling with Load Balancing in Fog Computing: a Bi-level Multi-Objective Optimization Approach

Najwa Kouka*
najoua.kouka@unimi.it
Università degli Studi di Milano
Dipartimento di Informatica
Milano, Italy

Vincenzo Piuri
vincenzo.piuri@unimi.it
Università degli Studi di Milano
Dipartimento di Informatica
Milano, Italy

Pierangela Samarati
pierangela.samarati@unimi.it
Università degli Studi di Milano
Dipartimento di Informatica
Milano, Italy

ABSTRACT

Fog computing is characterized by its proximity to edge devices, allowing it to handle data near the source. This capability alleviates the computational burden on data centers and minimizes latency. Ensuring high throughput and reliability of services in Fog environments depends on the critical roles of load balancing of resources and task scheduling. A significant challenge in task scheduling is allocating tasks to optimal nodes. In this paper, we tackle the challenge posed by the dependency between optimally scheduled tasks and the optimal nodes for task scheduling and propose a novel bi-level multi-objective task scheduling approach. At the upper level, which pertains to task scheduling optimization, the objective functions include the minimization of makespan, cost, and energy. At the lower level, corresponding to load balancing optimization, the objective functions include the minimization of response time and maximization of resource utilization. Our approach is based on an Improved Multi-Objective Ant Colony algorithm (IMOACO). Simulation experiments using iFogSim confirm the performance of our approach and its advantage over existing algorithms, including heuristic and meta-heuristic approaches.

CCS CONCEPTS

• **Optimization with randomized search heuristics → Evolutionary algorithms**; • **Scheduling algorithms**;

KEYWORDS

Task scheduling, Load-balancing, fog computing, multi-objective optimization problem, and ant colony optimization

ACM Reference Format:
Najwa Kouka, Vincenzo Piuri, and Pierangela Samarati. 2024. Tasks Scheduling with Load Balancing in Fog Computing: a Bi-level Multi-Objective Optimization Approach . In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638529.3654069

1 INTRODUCTION

The rise of the Internet of Things (IoT) and the widespread use of diverse mobile devices and sensors are posing new challenges for traditional cloud computing solutions [9]. In response, fog computing was proposed [18] to address these challenges. By definition, fog computing is a novel decentralized paradigm that provides computational services at the network edge, enabling the development of innovative services and applications for the future of the Internet [1]. Fog computing has several unique characteristics, including proximity to edge devices, a distributed computing model, heterogeneous devices, and a strong focus on security [2, 3]. Effective resource management within the fog environment plays a crucial role in minimizing costs, processing and communication delays.

Two of the main categories of resource management are task scheduling and load-balancing [5]. The scheduling involves searching for optimal solutions that organize a set of scheduled tasks on available resources with the best Quality of Service (QoS) requirements, such as time, deadline, and cost. These tasks are to be scheduled on a set of computing nodes with varying capabilities, including network usage, memory usage, and processing power. The load balancing aims to reduce response time and energy consumption while increasing throughput. In fact, the overload of fog nodes not only consumes more energy but also results in prolonged response times and increased costs.

Several approaches have separately solved these two problems (task scheduling and load-balancing) in a fog-cloud environment. These approaches include heuristic algorithms and meta-heuristic algorithms [4]. Unlike heuristic approaches, which typically rely on specific rules for particular problems, meta-heuristic algorithms are problem-independent, having the ability to explore a wide range of problems. In this context, various meta-heuristic algorithms have been proposed that solve task scheduling or load-balancing as a Single Objective Optimization (SOP) or as a Multi-Objective Optimization (MOP). The most common optimized metrics [6] are related to makespan, delay, and energy consumption, ignore important metrics related the load-balancing optimization. Imbalance in the QoS metrics can greatly affect the overall system performance. For instance, tasks might be scheduled on overloaded nodes because load balancing is not considered in the scheduling performance. Thus, the integration of load balancing optimization in the main process of task scheduling can achieve better performance, satisfying the requirements of each aspect. Such dependency between task scheduling and load balancing can be regarded as a Bi-level Multi-objective Optimization Problem (BMOP) [16]. This category of optimization problem involves two interconnected optimization tasks, each assigned to a distinct decision level (upper level and lower level). Consequently, assessing a solution at the upper level necessitates evaluating the lower level. The BMOP is a

GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0494-9/24/07.
https://doi.org/10.1145/3638529.3654069

suitable method for formulating various real-world scenarios, such as feature selection [13].

To the best of our knowledge, this work represents the first that optimizes task scheduling and load balancing as a bi-level multi-objective optimization problem, which is denoted the Bi-level Multi-Objective Task Scheduling-based Load balancing problem (BMO-TSLB). The contributions of this paper are the following:

(1) We formulate task scheduling and load-balancing as a bi-level multi-objective optimization problem. In BMO-TSLB, we consider task scheduling at the upper level and load balancing at the lower level. The objective set to be optimized at the upper level includes minimizing cost, energy, and makespan. The decision variables at the upper level encompass the optimal solution of the lower level, where the objectives are to minimize response time and maximize resource utilization.
(2) We propose a new Improved Multi-Objective Ant Colony Optimization (IMOACO) to address the BMO-TSLB. In IMOACO, the dominance comparator is employed to update the memory of each ant and the global optimal solution.
(3) We conduct an experimental study with a large number of tasks and heterogeneous nodes using the iFogSim simulator.

The remainder of this paper is structured as follows: Section 2 discusses related work. Section 3 illustrates our proposed formulation of task scheduling optimization with load balancing and the IMOACO algorithm. Section 4 reports the experimental study using iFogSim. Section 5 concludes the paper and outlines future directions.

2 RELATED WORK

In this section, a brief review of meta-heuristic algorithms for optimizing task scheduling and load balancing is illustrated. For instance, task scheduling and resource allocation for the Internet of Medical Things (IoMT) have been tackled using a Modified Particle Swarm Optimization (MPSO) [8]. The task scheduling problem is formulated as SOP, with the objective function as a linear combination aiming to minimize execution delay, execution cost, energy consumption, and network bandwidth usage. This approach entails mapping tasks to fog servers for executing IoMT tasks, while cloud servers manage more complex operations that surpass the capabilities of fog servers. A method known as Fog-Adaptive Multi-Objective Optimization Task Scheduling (FOG-AMOSM) is introduced in [21]. This method optimizes both total execution time and cost consumption in fog computing. Experimental results conducted on the CloudSim simulator showcase the enhanced performance of FOG-AMOSM in addressing task scheduling, particularly with a limited number of tasks. In another study [12], the Whale Optimization Algorithm (WOA) is applied to optimize the task scheduling problem in fog computing, with the two objectives of (1) reducing power consumption and (2) minimizing costs. However, it is important to note that the algorithm's performance has not been evaluated in the context of large-scale task scheduling. In [19] Energy-Efficient Task Scheduling based on Particle Swarm Optimization (EETSPSO) is proposed. In this algorithm, the fitness function is a linear equation derived from considerations of makespan and energy efficiency.

In addition, Ant Colony Optimization (ACO) is widely utilized for scheduling problems and has demonstrated good performance in addressing resource management challenges in both fog and cloud computing. A novel variant of ACO, called Load Balancing Ant Colony Optimization (LBACO) [14], incorporates considerations of the load on each virtual machine alongside core ACO concepts to expedite task execution. In this algorithm, a load balance factor, derived from response time and execution time, is employed to select the optimal node for task execution in each solution (ant). While numerous algorithms exist for task scheduling [6], they often overlook the crucial aspect of balancing load distribution alongside efficient task scheduling. These algorithms come with limitations such as improper scheduling, non-consideration of large-scale tasks, and neglecting key cooperative objectives essential for enhancing fog computing performance.

3 PROPOSED APPROACH

To make a clear presentation of the different abbreviations, the notations of problem formulation, and their corresponding descriptions are provided in Table 1.

The task scheduling problem in the context of fog computing involves allocating IoT tasks to suitable fog nodes from a pool of candidate fog nodes to optimize overall QoS. Assuming a collection of $|T|$ independent tasks, denoted as: $T = \{t_1, t_2, t_3, \ldots, t_{|T|}\}$. Each task t_i is characterized by a set of attributes including:

- $Length(t_i)$: is quantified by the number of instructions, with the unit expressed in million instructions (MI).
- $Size_{in}(t_i)$: is the input data size, represented, which signifies the relevant size of the input data.
- $Size_{out}(t_i)$: is the output data size presents the size of the output data.
- $Mem_{size}(t_i)$: is the required memory size to execute the task.

In addition, assume that the fog system comprises a set of $|N|$ computing nodes, denoted as $N = \{n_1, n_2, n_3, \ldots, n_{|N|}\}$. Each one n_j possesses distinctive attributes including:

- $CPU(n_j)$: is the CPU processing rate.
- $Mem_{size}(n_j)$: is the memory size.
- $Bw(n_j)$: is the network bandwidth.
- $STR(n_j)$: is the storage capacity.
- $CPU_{cost}(n_j)$: is the cost of CPU usage.
- $Bw_{cost}(n_j)$: is the cost of bandwidth usage.
- $Mem_{cost}(n_j)$: is the cost memory usage.

The decision variables for task scheduling represent the scheduled tasks and are denoted by X. Each variable of X presents the state of task t_i if it is allocated to n_j: $X_{i,j}$. The decision variable $X_{i,j}$ can be given as:

$$X_{i,j} \begin{cases} 1 & if\ t_i\ allocated\ to\ n_j \\ 0 & otherwise \end{cases} \tag{1}$$

Table 1: Description of mathematical notations

Notation	Description
$T = \{t_1, t_2, t_3, \ldots, t_{\lvert T\rvert}\}$	Tasks
$N = \{n_1, n_2, n_3, \ldots, n_{\lvert N\rvert}\}$	Nodes
X	Task Schedule
$X_{i,j}$	Scheduling of t_i on n_j
MS	Makespan
$Length(t_i)$	Length of t_i
$Size_{in}(t_i)$	Input size of t_i
$CPU(n_j)$	CPU processing rate of n_j
$Mem_{size}(n_j)$	Memory size of n_j
$Mem_{size}(t_i)$	Memory size of t_i
$Bw(n_j)$	Network bandwidth of n_j
$Bw(t_i)$	Required bandwidth to execute t_i
$STR(n_j)$	Storage capacity of n_j
$CPU_{cost}(n_j)$	Cost of CPU usage of n_j
$Bw_{cost}(n_j)$	Cost of bandwidth usage of n_j
$Mem_{cost}(n_j)$	Cost of memory usage of n_j
$p_{power(n_j)}$	Processing power of n_j
$RU(n_j)$	Resource utilization of n_j
$ETC(t_i, n_j)$	Expected computation time of t_i on n_j
$Penum(n_j)$	Number of CPUs for node n_j
$Cost_{comp}(t_i, n_j)$	Computational cost to execute t_i on n_j
$Cost_{comm}(t_i, n_j)$	Communication cost to execute t_i on n_j
$Cost(t_i, n_j)$	Cost to execute t_i on n_j
$Cost(X)$	Cost to execute all tasks
$E_p(t_i, n_j)$	Energy to execute t_i on n_j
$E_{trans}(t_i, n_j)$	Energy to transmit a t_i on n_j
$Energy(t_i, n_j)$	Energy ($E_{trans} + E_p$) to execute t_i on n_j
$Energy(X)$	Energy to execute all tasks
$RT(t_i, n_j)$	Response time to execute t_i on n_j
$RT(X)$	Response time to execute all tasks
$fitness^{LL}$	Fitness function of the lower level
$fitness^{UL}$	Fitness function of the upper level
PM^{RT}	Pheromone matrix of RT
PM^{ETC}	Pheromone matrix of ETC
PM^{Cost}	Pheromone matrix of $Cost$
PM^{Energy}	Pheromone matrix of $Energy$

3.1 Problem Formulation

The bi-level optimization problems BMO-TSLB involve two interconnected optimization tasks, with each assigned to a distinct decision level (i.e., upper and lower levels) [15, 16]. Consequently, the assessment of an upper-level solution necessitates the evaluation of the lower level. In our scenario, the optimization of task scheduling is closely linked to load-balancing optimization.

The proposed task scheduling optimization is formulated as BMOP, where the lower level consists of Multi-Objective Load-Balancing Optimization (LL-MOLBO) and the upper level consists of Multi-Objective Task Scheduling Optimization (UL-MOTSO). The set of objective functions for UL-MOTSO is denoted by F, which includes the minimization of $Cost$, the minimization of $Energy$, and the minimization of MS. The set of objective functions for LL-MOLBO is denoted by f, which includes the minimization of response time RT and the maximization of resource utilization RU. The general proposed formulation of BMO-TSLB is defined as follows:

$$\min F = \{Cost\ (X, y^*), Energy\ (X, y^*), MS\ (X, y^*)\} \tag{2}$$

subject to:

$$y^* \in argmin\,\{RT\ (X), RU\ (X)\} \tag{3}$$

Each objective function will be detailed below.

3.1.1 *Objective functions of UL-MOTSO.*

The proposed task scheduling optimization consists of assigning tasks to appropriate n_j. The initial search space for this level is based on LL-MOLBO, where $n_j \in y^*$. The descriptions of makespan, cost, and energy are given below:

- **Makespan** (MS) is defined as the total time taken to complete the entire task T. The minimization of MS is the first objective to optimize. The MS is calculated as follows:

$$MS(X) = max_{\,\forall j \in |N|} \sum_{i=1}^{|T|} ETC(t_i, n_j) * X_{i,j} \tag{4}$$

  where $ETC(t_i, n_j)$ is the expected time of computation of task t_i in n_j as presented in following equation:

$$ETC(t_i, n_j) = \frac{Length(t_i)}{p_{power}(n_j)} * X_{i,j} \tag{5}$$

  where $p_{power}(n_j)$ is the processing power of n_j (given in MIPS), which is defined as follows:

$$p_{power(n_j)} = Mips(n_j) * Penum(n_j) \tag{6}$$

  in the formula, $Mips(n_j)$ represents the computing power of n_j, and $Penum(n_j)$ represents the number of CPUs for node n_j.

- **Cost consumption**: in general, the cost of executing tasks includes both the cost of computation and the cost of communication, which are detailed as follows:
  - **Cost of computation**: the computation cost for a specific t_i consists of two components: processing cost and memory cost. These costs can be estimated using the following expressions:

$$Cost_{comp}(t_i, n_j) = CPU_{cost}(n_j) * ETC(t_i, n_j) + Mem_{cost}(n_j) * Mem_{size}(t_i) \tag{7}$$

  - **Cost of communication**: depends on the file size and bandwidth usage per transmitted data unit per node. The necessity of bandwidth amount for t_i is $Bw(t_i)$. The communication cost ($Cost_{comm}(t_i, n_j)$) for a particular t_i is calculated as follows:

$$Cost_{comm}(t_i, n_j) = Bw_{cost}(n_j) * Bw(t_i) \tag{8}$$

  The cost consumption for executing t_i on node n_j is measured with Equation 9

$$Cost(t_i, n_j) = Cost_{comp}(t_i, n_j) + Cost_{comm}(t_i, n_j) \tag{9}$$

  To this end, the total cost of scheduled task X is measured by Equation 9.

$$Cost(X) = \sum_{j=1}^{|N|} \sum_{i=1}^{|T|} Cost(t_i, n_j) * X_{i,j} \tag{10}$$

- **Energy Consumption**: energy consumption is composed of two components: (1) the energy spent on transmitting a task to a computing node denoted as E_{trans}; (2) the energy spent on executing the task denoted as E_p.
  The energy E_{trans} required to transmit t_i to n_j is calculated by multiplying the transmission time by a constant coefficient as defined in Equation 11.

$$E_{trans}(t_i, n_j) = \lambda * Trans(t_i, n_j) \tag{11}$$

  where λ is a constant related to the wireless interface.
  $Trans(t_i, n_j)$ is the transmission time for task t_i to fog node n_j [20], which is determined as follows:

$$Trans(t_i, n_j) = \frac{Size_{in}(t_i) + Size_{out}(t_i)}{Bw(n_j)} \tag{12}$$

  The energy consumption E_p for task processing is defined as follows:

$$E_p(t_i, n_j) = \mu * ETC(t_i, n_j) \tag{13}$$

  where μ is the coefficient denoting the energy consumption per CPU cycle. The energy consumption for executing task t_i on node n_j is determined by Equation 14, while the overall energy for execution of the scheduled task X is determined with 15.

$$Energy(t_i, n_j) = E_{trans}(t_i, n_j) + E_p(t_i, n_j) \tag{14}$$

$$Energy(X) = \sum_{j=1}^{|N|} \sum_{i=1}^{|T|} Energy(t_i, n_j) * X_{i,j} \tag{15}$$

3.1.2 *Objective functions of LL-MOLBO.*

In the proposed load balancing optimization, the solution comprises an optimal set of nodes that optimize the conflicting objectives: RT and RU.

- **Response time** RT**:** the response time of a task t_i is the sum of the specified node's execution time plus the task's transmission time from the source to the destination. The next formula is utilized to compute the response time of ti that is handled at computing node n_j:

$$RT(t_i, n_j) = ETC(t_i, n_j) + Trans(t_i, n_j) \tag{16}$$

  The total RT of scheduled tasks X is determined by $RT(X)$ (Equation 17).

$$RT(X) = \sum_{j=1}^{|N|} \sum_{i=1}^{|T|} RT(t_i, n_j) * X_{i,j} \tag{17}$$

- **Resource utilization** RU: the resource utilization of node n_j is the optimal utilization of resources, with a crucial link between efficiency and the reduction of MS. Therefore, these two concepts exhibit an inverse relationship. The resource utilization is determined as follows:

$$RU(n_j) = \sum_{i=1}^{|T|} \frac{ETC(t_i, n_j)}{MS(X)} \tag{18}$$

  The total resource utilization of scheduled tasks is defined by Equation 19.

$$RU(X) = \sum_{j=1}^{|N|} RU(n_j) \tag{19}$$

3.2 IMOACO: Improved Multi-Objective Ant Colony Optimization

Task scheduling problem has been established as an NP-Hard problem, highlighting the potential for significant improvements in scheduling efficiency through the use of ACO in fog computing task scheduling. Since ACO is initially designed for SOP, it is inadequate when addressing the MOP. Several multi-objective ACO algorithms have been proposed to solve MOP [7, 11, 22], by employing a single colony with several pheromone trace matrices τ^m, where $m \in [1, M]$ (where M represents the number of objectives), and a single heuristic information matrix η. Additionally, multi-colony ACO algorithms consist of several colonies of ants. Each colony uses separate pheromones to maximize the explored search area. Taking inspiration from these works, the proposed IMOACO employs two colonies for task scheduling and load balancing, respectively. IMOACO utilizes several pheromone trace matrices τ^m and a unified heuristic information matrix η. To elaborate, for the UL-MOTSO, there are three pheromone trace matrices (PM^{ETC}, PM^{Cost}, and PM^{Energy}) and a single heuristic information matrix η. In contrast, for the lower-level load balancing, a single pheromone trace matrix is associated with the RT, since the RU can be updated after completing the path of each ant. For each colony, every ant selects the next node to visit according to the probability distribution. In the upper-level colony, the search process is initiated by utilizing the optimal solutions $n_j \in y^*$ from the lower-level colony. The flowchart of our proposed IMOACO is outlined in Figure 1 and the main search process in each colony follows the steps outlined below.

3.2.1 *LL-MOLBO optimization colony.*

The optimization process of the LL-MOLBO colony is outlined as follows:

- **Step 1: Initialization**: at the beginning of the process, the expected response time of the task t_i on n_j is represented by the matrix PM^{RT}.

$$PM^{RT} = \begin{bmatrix} RT(t_1, n_1) & RT(t_1, n_2) & \dots & RT(t_1, n_{|N|}) \\ RT(t_2, n_1) & RT(t_2, n_2) & \dots & RT(t_2, n_{|N|}) \\ \dots & \dots & \dots & \dots \\ RT(t_{|T|}, n_1) & RT(t_{|T|}, n_2) & \dots & RT(t_{|T|}, n_{|N|}) \end{bmatrix} \tag{20}$$

  During the initial iteration of the lower-level optimization, each ant_k ($k = 1..A$, A is the number of the ants) initialize the first pheromone $\tau_{i,j}^{LL,k}(0)$ with the value of index i, j of PM^{RT} (Equation 21).
  Subsequently, each ant is randomly deployed on a task node t_i.

$$\tau_{i,j}^{LL,k}(0) = PM_{i,j}^{RT} \tag{21}$$

- **Step 2: Calculation of heuristic information**: each ant constructs a tour by executing $|T|$ tasks with a probabilistic transition rule.
  Every ant_k selects the next n_j to visit according to the probability given by the following equation:

$$p_{i,j}^{LL,k}(t) = \frac{(\eta_{i,j}^{LL,k})^{\beta} (\tau_{i,j}^{LL,k})^{\alpha}}{\sum_{j \in \Omega} (\eta_{i,j}^{LL,k})^{\beta} (\tau_{i,j}^{LL,k})^{\alpha}} \tag{22}$$

  In this context, β and α represent the respective weightings assigned to the heuristic information and pheromone trace. The

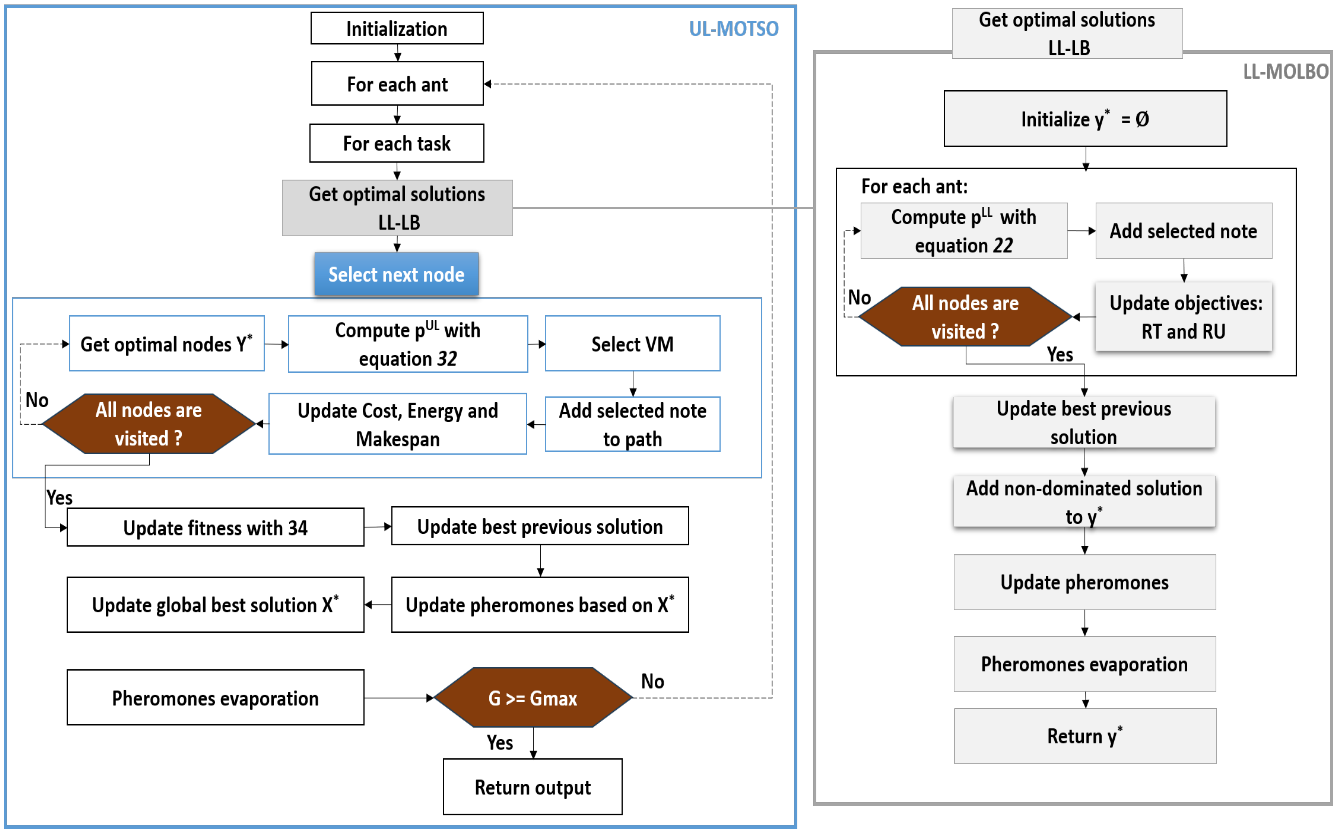
UL-MOTSO
Initialization
For each ant
For each task
Get optimal solutions LL-LB
Select next node
Get optimal nodes Y*
Compute p^UL with equation 32
Select VM
No
All nodes are visited ?
Update Cost, Energy and Makespan
Add selected note to path
Yes
Update fitness with 34
Update best previous solution
Update global best solution X*
Update pheromones based on X*
Pheromones evaporation
G >= Gmax
No
Yes
Return output
Get optimal solutions LL-LB
LL-MOLBO
Initialize y* = Ø
For each ant:
Compute p^LL with equation 22
Add selected note
No
All nodes are visited ?
Update objectives: RT and RU
Yes
Update best previous solution
Add non-dominated solution to y*
Update pheromones
Pheromones evaporation
Return y*

Figure 1: Flowchart of IMOACO based on BMO-TSLB optimization

heuristic function of ant_k is denoted by $\eta_{i,j}^{LL,K}$, indicating the response time of the node n_j to execute t_i.
The calculation method is as follows:

$$\eta_{i,j}^{LL,K} = \frac{1}{fitness_{i,j}^{LL}} \tag{23}$$

where the $fitness^{LL}$ is based on the RT, which is defined by Equation 24.

$$fitness_{i,j}^{LL} = RT(t_i, n_j) \tag{24}$$

The smaller the $fitness^{LL}$ value is, the larger the probability of selecting the current node n_j for t_i since it has a minimum RT.

- **Step 3: Local pheromone update**: after each iteration, the pheromone of each ant is updated using the following formulation:

$$\tau_{i,j}^{LL,k}(t) = (1-p)\tau_{i,j}^{LL,k}(t-1) + \Delta\tau_{i,j}^{LL,k} \tag{25}$$

where p is the pheromone evaporation rate and the quantity of pheromone left by each ant_k $Delta\tau_{i,j}^{LL,k}$ calculated as follows:

$$\Delta\tau_{i,j}^{LL,k} = \frac{Q}{RT(t_i, n_j)} \tag{26}$$

Here Q is a constant related to the quantity of pheromone left by the ants.

- **Step 4: Update global pheromone**: When all the ants have constructed their solution, the best solution is selected from ants based on fitness function ($fitness^{LL} = RT(X)$). Then, the updating of the global pheromone is performed on the solution $X_{i,j}$ as follows:

$$\tau_{i,j}^{LL,k}(t) = \tau_{i,j}^{LL,k}(t-1) + \Delta\tau_{i,j}^{LL,k} \tag{27}$$

where the $\Delta\tau_{i,j}^{LL,k}(t) = \sum_{k=1}^{A} \Delta\tau_{i,j}^{LL,k}(t)$.

- **Step 5: Update non-dominated solutions**: the optimal y^* is updated based on the non-dominated solutions found by different ants, utilizing the RT (Equation 17) and RU (Equation 18) objective functions, respectively.

3.2.2 *UL-MOTSO optimization colony*.

- **Step 1: Initialization**: at the beginning of the optimization process in the upper level, the different objective functions are measured and expressed as a matrix: (1) PM^{ETC}; (2) PM^{Cost}; (3) PM^{Energy} (determined as PM^{RT}). Also, for each ant_k the three pheromones are initialized with matrix PM^{ETC}, PM^{Cost}, and

PM^{Energy}, respectively.

$$\tau_{i,j}^{UL,k,1}(0) = PM_{i,j}^{ETC} \tag{28}$$

$$\tau_{i,j}^{UL,k,2}(0) = PM_{i,j}^{Cost} \tag{29}$$

$$\tau_{i,j}^{UL,k,3}(0) = PM_{i,j}^{Energy} \tag{30}$$

- **Step 2: Calculation of heuristic information**: the heuristic function of the $UL - MOTSO$ of each ant_k is denoted by $\eta_{i,j}^{UL,k}$, and it is determined by:

$$\eta_{i,j}^{UL,k} = \frac{1}{fitness_{i,j}^{UL}} \tag{31}$$

  where $fitness_{i,j}^{UL} = ETC(t_i, n_j) + Cost(t_i, n_j) + Energy(t_i, n_j)$. Based on the different pheromones concentration and heuristic function, each ant_k chooses node j for task i with the probability $p(i,j)^{UL,k}$is measured as follows:

$$p_{i,j}(t)^{UL,k} = \frac{(\eta_{i,j}^{UL,k})^{\beta} \prod_{m=1}^{M} (\tau_{i,j}^{UL,k,m})^{\alpha_m}}{\sum_{j \in y^*} (\eta_{i,j}^{UL,k})^{\beta} \prod_{m=1}^{M} (\tau_{i,j}^{UL,k,m})^{\alpha_m}} \tag{32}$$

  where α_m represents the respective weightings assigned to the pheromone trace of objective m.

- **Step 3: Update objectives functions**: for each ant_k, the scheduled task is evaluated with MS, $Cost$, and $Energy$, which are measured by equations 4, 10, and 15. Once the objective functions are evaluated, the fitness function will be updated.

- **Step 4: Local pheromone update**: Once the solution is updated, each pheromone m trace is updated as follows:

$$\tau_{i,j}^{UL,k,m}(t) = \tau_{i,j}^{UL,k,m}(t-1) + \Delta\tau_{i,j}^{UL,k,m} \tag{33}$$

  where $\Delta\tau_{i,j}^{UL,k,m}$ value represents the change in pheromone level of ant_k on edge (t_i, n_j for objective m), which measured as follows:

$$\Delta\tau_{i,j}^{UL,k,m} = \frac{Q}{F_k}, \tag{34}$$

  where $F_k \in (ETC(t_i, n_j), Cost(t_i, n_j), Energy(t_i, n_j))$.

- **Step 5: Update global pheromone**: after each iteration, the global pheromones for each edge $X_{i,j}$ are updated as follows:

$$\tau_{i,j}^{UL,k,m}(t) = (1 - p_m)\tau_{i,j}^{UL,k,m}(t-1) + \Delta\tau_{i,j}^{UL,k,m} \tag{35}$$

  where p_k is the pheromone evaporation rate for objective m, $p \in [0,1]$. $\Delta\tau_{i,j}^{UL,m}(t) = \sum_{k=1}^{A} \Delta\tau_{i,j}^{UL,k,m}(t)$.

- **Step 6: Update the best global optimal solution**: the update of the best global solution is conducted using a dominance comparator that includes the objective functions $MS(X)$, $Cost(X)$, and $Energy(X)$.

- **Step 7: Verification of stopping criteria** if the current iteration G reaches the maximum number of iterations (G_{max}), the best global solution is returned, otherwise, it proceeds to step 2.

4 EXPERIMENTAL STUDIES

This experiment aims to demonstrate how the problem formulation of task scheduling impacts QoS parameters such as time, cost, and energy, while also considering the load balance of nodes.

However, the proposed IMOACO is compared against heuristics and meta-heuristics algorithms:

- **FCFS: First Come First Serve** algorithm, is a heuristic algorithm that schedules the first process to arrive and allows it to run to completion [10].

- **RR: Round Robin** algorithm is a type of CPU scheduling algorithm where each process is allocated a fixed time quantum for its execution [10].

- **SJF: Shortest Job First** algorithm. It's a scheduling policy that chooses the waiting process with the shortest execution time [10].

- **LBACO: Load Balancing Ant Colony Optimization** is an improved version of the ACO algorithm, where the probability for selecting the optimum node for each task is based on excessive virtual memory and the predicted execution time [14].

- **ACO**: is a standard ACO algorithm where the objective is minimizing the makespan of a given task set. The task is allocated to the resource possessing the highest processing speed to ensure that all tasks are completed in the shortest possible time [17].

Within our simulations, we have manipulated the volume of incoming tasks, ranging from a modest 100 to a substantial 1000. The performance of the compared algorithms is evaluated in terms of total cost computation, total energy consumption, makespan, and total resource utilization.

4.1 Parameter Settings

For the conducted experiments, the simulator iFogSim is employed for modeling and simulating the fog environment for task scheduling. The CPU of the experiment processor is an Intel(R) Core(TM) i5-9300HF CPU @ 2.40GHz with 16 GB of memory, the operation system is Windows 11 64-bit, and the development tool is Eclipse. In this section, we present all parameter settings in Table 2, encompassing parameters relevant to tasks, fog nodes, and common parameters for ACO-based algorithms (ACO, LBACO, and IMOACO).

4.2 Results and Analysis

In this section, the comparative results are presented. Tables 3-6 display the mean and standard deviation values of cost, energy, makespan, and resource utilization across 20 independent runs. In each table, blue and light blue colors are used to color the first and second-best algorithm's results. Table 7 presents the average rankings of each compared algorithm based the Friedman Test.

The performance of the proposed algorithm is assessed by varying the task count from 100 to 1000. Observing Figures 2 and 3, it is evident that increasing the task count impacts system performance by escalating the burden. With a higher number of tasks, both makespan and resource utilization increase. Consequently, the service time of tasks on fog nodes also rises.

Table 2: Experiment parameters setting

Entity type	Parameter	Value
Tasks	Length of task	$[9,000, 15,000]$ *MI*
	File size	300
	Output size	300
	Number of tasks	100 – 1000
Fog nodes (VM)	Processing rate	512 – 1024 MIPS
	Penum per VM	1
	Bandwidth	500-1200
	Memory	512-2048
	Storage	100000-800000
	Unit cost memory	0.05
	Unit cost storage	0.001
	Unit cost bandwidth	0.1
	Number of VM	50
ACO algorithms	Number of ants	10
	G_{max}	50
	α	1
	β	1
	Q	100
IMOACO	Number of ants in each colony	5

Figure 3 illustrates the results obtained by the proposed approach in terms of resource utilization. The figure indicates that IMOACO achieved maximum resource utilization across all ranges of tasks, from small to large, when compared with other algorithms. As confirmed by results reported in the tables, the IMOACO outperforms the other algorithms concerning different metrics.

The observed results indicate that the makespan time of the proposed IMOACO algorithm is shorter compared to other algorithms. This is attributed to its exploiting the load-balancing to select optimal nodes for scheduling tasks.

5 CONCLUSIONS

In this paper, the primary contribution lies in the novel formulation of task scheduling optimization, aiming to address most QoS aspects. This formulation introduces a bi-level multi-objective task scheduling approach, to simultaneously optimize scheduling and load balancing. In the optimization process, cost, energy, makespan, response time, and resource utilization are considered as criteria to seek an optimal solution that meets user requirements. To address the formulated problem, we introduce an improved multi-objective ACO algorithm that utilizes a dominance comparator to assess solutions discovered during the search process. We conducted a comparative analysis between our proposed IMOACO approach and existing scheduling algorithms. The experimental results illustrate that our proposed task scheduling mechanism surpasses the compared algorithms in terms of cost, energy, makespan, and resource utilization. The proposed algorithms demonstrate stability in resource utilization even as task numbers increase. This capability ensures effective handling of the substantial surge in request generation from edge devices, thereby preventing resource overload. However, a primary limitation of this work lies in the necessity to accommodate dynamic environments where there is a lack of prior information regarding task properties or available resources.

In the future, the developed algorithm can be extended to address real-time scheduling by extending optimization techniques to solve dynamic MOP. Additionally, our forthcoming research will focus on addressing the application of our approach to real-time health scenarios.

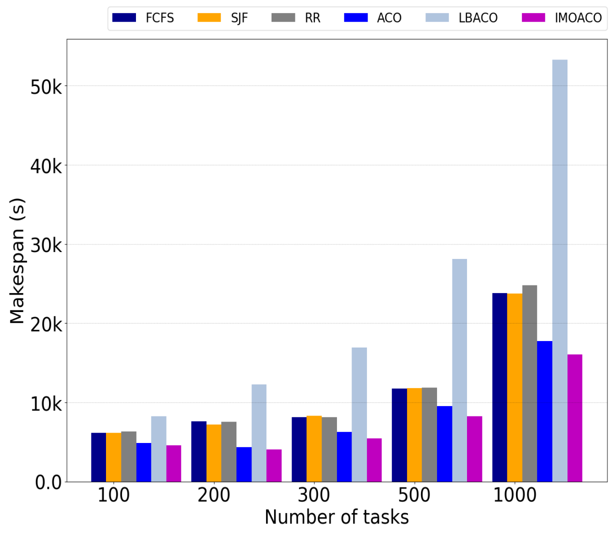

Figure 2: Comparison of makespan performances

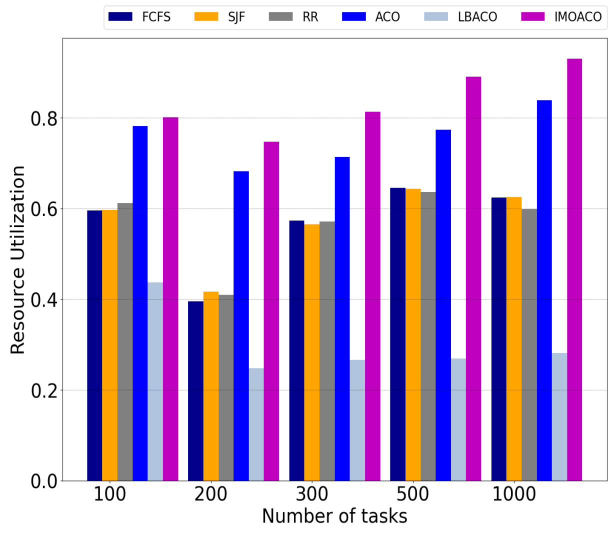

Figure 3: Comparison of resource utilization performances

Table 3: Cost (mean and standard deviation)

N	FCFS	SJF	RR	ACO	LBACO	IMOACO
100	$2.2271e+05_{1.36e+04}$	$2.2534e+05_{1.29e+04}$	$2.2343e+05_{9.51e+03}$	$2.2790e+05_{1.41e+04}$	$2.2573e+05_{1.04e+04}$	$2.2079e+05_{8.86e+03}$
200	$4.4788e+05_{1.66e+04}$	$4.4160e+05_{1.58e+04}$	$4.5001e+05_{1.66e+04}$	$4.4667e+05_{1.62e+04}$	$4.4571e+05_{1.54e+04}$	$4.4146e+05_{1.59e+04}$
300	$6.6669e+05_{1.72e+04}$	$6.6668e+05_{3.21e+04}$	$6.6497e+05_{1.63e+04}$	$6.7019e+05_{1.76e+04}$	$6.6843e+05_{2.15e+04}$	$6.6393e+05_{2.53e+04}$
500	$1.1111e+06_{2.69e+04}$	$1.1130e+06_{3.26e+04}$	$1.1075e+06_{2.38e+04}$	$1.1062e+06_{2.27e+04}$	$1.1168e+06_{1.42e+04}$	$1.1042e+06_{2.41e+04}$
1000	$2.2155e+06_{3.35e+04}$	$2.2157e+06_{3.36e+04}$	$2.2163e+06_{4.17e+04}$	$2.2321e+06_{3.88e+04}$	$2.2365e+06_{2.95e+04}$	$2.2148e+06_{4.01e+04}$

Table 4: Energy (mean and standard deviation)

N	FCFS	SJF	RR	ACO	LBACO	IMOACO
100	$2.7466e+05_{1.68e+04}$	$2.7791e+05_{1.59e+04}$	$2.7554e+05_{1.17e+04}$	$2.8106e+05_{1.74e+04}$	$2.7838e+05_{1.28e+04}$	$2.7229e+05_{1.09e+04}$
200	$5.5235e+05_{2.05e+04}$	$5.4461e+05_{1.95e+04}$	$5.5498e+05_{2.05e+04}$	$5.5086e+05_{2.00e+04}$	$5.4967e+05_{1.89e+04}$	$5.4443e+05_{1.96e+04}$
300	$8.2220e+05_{2.12e+04}$	$8.2219e+05_{3.96e+04}$	$8.2007e+05_{2.01e+04}$	$8.2652e+05_{2.17e+04}$	$8.2434e+05_{2.66e+04}$	$8.1880e+05_{3.12e+04}$
500	$1.3703e+06_{3.32e+04}$	$1.3726e+06_{4.03e+04}$	$1.3658e+06_{2.94e+04}$	$1.3642e+06_{2.80e+04}$	$1.3772e+06_{1.75e+04}$	$1.3618e+06_{2.97e+04}$
1000	$2.7323e+06_{4.13e+04}$	$2.7325e+06_{4.14e+04}$	$2.7333e+06_{5.14e+04}$	$2.7527e+06_{4.79e+04}$	$2.7582e+06_{3.64e+04}$	$2.7315e+06_{4.94e+04}$

Table 5: Makespan (mean and standard deviation)

N	FCFS	SJF	RR	ACO	LBACO	IMOACO
100	$6.5658e+03_{1.72e+03}$	$6.5040e+03_{1.20e+03}$	$6.3185e+03_{1.28e+03}$	$4.8691e+03_{3.85e+02}$	$9.0164e+03_{2.11e+03}$	$4.6173e+03_{3.61e+02}$
200	$7.6086e+03_{6.78e+02}$	$7.2057e+03_{1.11e+03}$	$7.5290e+03_{1.28e+03}$	$4.3735e+03_{2.33e+02}$	$1.2260e+04_{1.77e+03}$	$4.0153e+03_{3.49e+02}$
300	$8.1602e+03_{1.93e+03}$	$8.2903e+03_{1.89e+03}$	$8.1152e+03_{1.80e+03}$	$6.2682e+03_{3.30e+02}$	$1.6959e+04_{2.10e+03}$	$5.4561e+03_{3.25e+02}$
500	$1.1773e+04_{1.85e+03}$	$1.1816e+04_{1.89e+03}$	$1.1870e+04_{1.83e+03}$	$9.5386e+03_{3.13e+02}$	$2.8117e+04_{3.76e+03}$	$8.2710e+03_{3.24e+02}$
1000	$2.3827e+04_{2.11e+03}$	$2.3750e+04_{1.98e+03}$	$2.4790e+04_{1.68e+03}$	$1.7748e+04_{3.61e+02}$	$5.3248e+04_{4.20e+03}$	$1.5797e+04_{4.76e+02}$

Table 6: Resource Utilization (mean and standard deviation)

N	FCFS	SJF	RR	ACO	LBACO	IMOACO
100	$5.9568e-01_{1.26e-01}$	$5.9637e-01_{1.09e-01}$	$6.1188e-01_{1.18e-01}$	$7.8165e-01_{3.11e-02}$	$4.3755e-01_{9.04e-02}$	$8.0073e-01_{5.82e-02}$
200	$3.9527e-01_{3.55e-02}$	$4.1659e-01_{5.54e-02}$	$4.0937e-01_{6.68e-02}$	$6.8220e-01_{3.37e-02}$	$2.4740e-01_{3.69e-02}$	$7.3734e-01_{5.69e-02}$
300	$5.7310e-01_{1.24e-01}$	$5.6526e-01_{1.30e-01}$	$5.7079e-01_{1.14e-01}$	$7.1413e-01_{3.01e-02}$	$2.6646e-01_{3.12e-02}$	$8.1313e-01_{4.17e-02}$
500	$6.4504e-01_{1.02e-01}$	$6.4314e-01_{9.81e-02}$	$6.3665e-01_{9.67e-02}$	$7.7377e-01_{2.58e-02}$	$2.6924e-01_{3.39e-02}$	$8.9085e-01_{2.62e-02}$
1000	$6.2432e-01_{5.12e-02}$	$6.2554e-01_{4.49e-02}$	$5.9852e-01_{3.90e-02}$	$8.3852e-01_{1.13e-02}$	$2.8193e-01_{2.47e-02}$	$9.3510e-01_{1.94e-02}$

Table 7: Average Rankings of the Algorithms for Performance Metrics

Algorithm	Cost	Energy	Makespan	Resource utilization
FCFS	3.40	3.40	4.20	3.00
SJF	3.40	3.40	3.80	3.20
RR	3.59	3.59	4.00	2.80
ACO	4.60	4.60	2.00	5.00
LBACO	5.00	5.00	6.00	1.00
IMOACO	1.00	1.00	**1.00**	**6.00**

ACKNOWLEDGMENTS

This work was supported in part by the EC under projects Chips JU EdgeAI (101097300) and GLACIATION (101070141), and by project SERICS (PE00000014) under the MUR NRRP funded by the EU - NextGenerationEU. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor Italian MUR can be held responsible for them.

REFERENCES

[1] Resul Das and Muhammad Muhammad Inuwa. 2023. A review on fog computing: Issues, characteristics, challenges, and potential applications. *Telematics and Informatics Reports* 10 (2023), 100049. https://doi.org/10.1016/j.teler.2023.100049

[2] Sabrina De Capitani di Vimercati, Sara Foresti, Giovanni Livraga, Vincenzo Piuri, and Pierangela Samarati. 2021. Security-Aware Data Allocation in Multicloud Scenarios. *IEEE Transactions on Dependable and Secure Computing* 18, 5 (2021), 2456–2468. https://doi.org/10.1109/TDSC.2019.2953068

[3] Sabrina De Capitani Di Vimercati, Sara Foresti, Giovanni Livraga, Vincenzo Piuri, and Pierangela Samarati. 2021. Supporting User Requirements and Preferences in Cloud Plan Selection. *IEEE Transactions on Services Computing* 14, 1 (2021), 274–285. https://doi.org/10.1109/TSC.2017.2777977

[4] Shally Gupta and Nanhay Singh. 2023. Heuristics and Meta-Heuristics based Algorithms for Resource Optimization in Fog Computing Environment: A Comparative Study. In *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. 271–276. https://doi.org/10.1109/IDCIoT56793.2023.10053388

[5] Cheol-Ho Hong and Blesson Varghese. 2019. Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms. *ACM Comput. Surv.* 52, 5 (sep 2019). https://doi.org/10.1145/3326066

[6] Mehdi Hosseinzadeh, Elham Azhir, Jan Lansky, Stanislava Mildeova, Omed Hassan Ahmed, Mazhar Hussain Malik, and Faheem Khan. 2023. Task Scheduling Mechanisms for Fog Computing: A Systematic Survey. *IEEE Access* 11 (2023), 50994–51017. https://doi.org/10.1109/ACCESS.2023.3277826

[7] Tiansheng Huang, Weiwei Lin, Chennian Xiong, Rui Pan, and Jingxuan Huang. 2021. An Ant Colony Optimization-Based Multiobjective Service Replicas Placement Strategy for Fog Computing. *IEEE Transactions on Cybernetics* 51, 11 (2021), 5595–5608. https://doi.org/10.1109/TCYB.2020.2989309

[8] Bushra Jamil, Humaira Ijaz, Mohammad Shojafar, Kashif Munir, and Rajkumar Buyya. 2022. Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions. *ACM Comput. Surv.* 54, 11s (sep 2022). https://doi.org/10.1145/3513002

[9] Fatemeh Khoda Parast, Chandni Sindhav, Seema Nikam, Hadiseh Izadi Yekta, Kenneth B. Kent, and Saqib Hakak. 2022. Cloud computing security: A survey of service-based models. *Comput. Secur.* 114, C (mar 2022). https://doi.org/10.1016/j.cose.2021.102580

[10] Khaled Matrouk and Kholoud Alatoun. 2021. Scheduling Algorithms in Fog Computing: A Survey. *International Journal of Networked and Distributed Computing* 9 (2021), 59–74. Issue 1. https://doi.org/10.2991/ijndc.k.210111.001

[11] Leonor Albuquerque Melo. 2009. Multi-colony ant colony optimization for the node placement problem. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1570256.1570391

[12] K.P. N. Jayasena and B.S. Thisarasinghe. 2019. Optimized task scheduling on fog computing environment using meta heuristic algorithms. In *2019 IEEE International Conference on Smart Cloud (SmartCloud)*. 53–58. https://doi.org/10.1109/SmartCloud.2019.00019

[13] Rihab Said, Maha Elarbi, Slim Bechikh, Carlos Artemio Coello Coello, and Lamjed Ben Said. 2023. Discretization-Based Feature Selection as a Bilevel Optimization Problem. *IEEE Transactions on Evolutionary Computation* 27, 4 (2023), 893–907. https://doi.org/10.1109/TEVC.2022.3192113

[14] Neetu Sharma, Sonal, and Puneet Garg. 2022. Ant colony based optimization model for QoS-based task scheduling in cloud computing environment. *Measurement: Sensors* 24 (2022), 100531. https://doi.org/10.1016/j.measen.2022.100531

[15] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. 2017. Approximated set-valued mapping approach for handling multiobjective bilevel problems. *Computers Operations Research* 77 (2017), 194–209. https://doi.org/10.1016/j.cor.2016.08.001

[16] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. 2017. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research* 257, 2 (2017), 395–411. https://doi.org/10.1016/j.ejor.2016.08.027

[17] Medhat A. Tawfeek, Ashraf El-Sisi, Arabi E. Keshk, and Fawzy A. Torkey. 2013. Cloud task scheduling based on ant colony optimization. In *2013 8th International Conference on Computer Engineering Systems (ICCES)*. 64–69. https://doi.org/10.1109/ICCES.2013.6707172

[18] Luis M. Vaquero and Luis Rodero-Merino. 2014. Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. 44, 5 (oct 2014), 27–32. https://doi.org/10.1145/2677046.2677052

[19] Shilpa Dinesh Vispute and Priyanka Vashisht. 2023. Energy-Efficient Task Scheduling in Fog Computing Based on Particle Swarm Optimization. *SN Computer Science* 4 (2023). https://doi.org/10.1007/s42979-022-01639-3

[20] Jiuyun Xu, Xiaoting Sun, Ruru Zhang, Hongliang Liang, and Qiang Duan. 2020. Fog-cloud task scheduling of energy consumption optimisation with deadline consideration. *International Journal of Internet Manufacturing and Services* 7, 4 (2020), 375–392. https://doi.org/10.1504/IJIMS.2020.110228

[21] Ming Yang, Hao Ma, Shuang Wei, You Zeng, Yefeng Chen, and Yuemei Hu. 2020. A Multi-Objective Task Scheduling Method for Fog Computing in Cyber-Physical-Social Services. *IEEE Access* 8 (2020), 65085–65095. https://doi.org/10.1109/ACCESS.2020.2983742

[22] Celal Özkale and Alpaslan Fığlalı. 2013. Evaluation of the multiobjective ant colony algorithm performances on biobjective quadratic assignment problems. *Applied Mathematical Modelling* 37, 14 (2013), 7822–7838. https://doi.org/10.1016/j.apm.2013.01.045