A Heuristic for Load Distribution on Data Center Hierarchy: A MEC Approach

RAFAEL F. VIEIRA¹, DANIEL DA S. SOUZA¹, MARCELINO S. DA SILVA¹, AND DIEGO L. CARDOSO.¹
¹Post-Graduate Program in Electrical Engineering, Federal University of Pará, Belém 66075-110, Brazil
Corresponding author: Rafael F. Vieira (e-mail: fogarollirafael@gmail.com)

This work was supported in part by the National Council for Scientific and Technological Development (CNPq) and in part by the Federal University of Pará (UFPA).

ABSTRACT
Mobile Edge Computing (MEC) extends Cloud Computing to the edge of the network, creating a hierarchy of data centers. This paradigm provides computing capacity close to final users, relieves backhaul and the leading network, and serves latency-sensitive applications. When providing computing services at different network levels, it becomes necessary to carry out a more efficient distribution of the resources that come to coexist. A random allocation of these resources can lead to a low service acceptance rate and backhaul overhead problems. Problems like these can be solved with MEC. To maximize the service acceptance and ensure a fair distribution of services in the kinds of servers guaranteeing their QoS requirements, we propose a MILP (Mixed Integer Linear Programming). The model performs an optimal allocation of applications in a two-level hierarchy of data centers: (i) MEC and (ii) cloud computing. On a large scale, the use of MILP becomes unfeasible due to the high computational cost, so we propose a heuristic based on the application profiles. We compare the proposed heuristic with two metaheuristics: (i) Genetic Algorithm and (ii) Particle Swarm Optimization. The solutions are compared in terms of service acceptance rate, large-scale performance, and efficient use of available resources. Results show that the proposed heuristic reaches 91% of the optimal solution and over 140% compared to AG and PSO solutions.

INDEX TERMS Mobile Edge Computing, Cloud Computing, Data Center Hierarchy, Mixed Integer Linear Programming, Genetic Algorithm, Particle Swarm Optimization.

I. INTRODUCTION
Seven times more mobile traffic compared to 2017 is expected for 2022 [1]. This growth will be a consequence of the new applications that the fifth generation of mobile networks (5G) will enable [2]. Internet of Things (IoT), education 4.0, and industry 4.0 are some examples of applications that will increase the demand for [3] resources. High demand volume and pressure on high network costs will drive the need for changes to maintain quality of service (QoS) and experience (QoE) for end-users [4], generate revenue for service providers’ service, and optimize [5] operations.

The increasing dependence of users to perform intensive operations, both computing and storage, requires the download of applications to the [6] cloud. In addition, ultra-low latencies and high bandwidth availability will be standard requirements among new applications. Therefore, applications will often not be able to be served by the cloud due to the latency required to reach these servers (round trip time, router delays, queue delays, bandwidth bottlenecks, etc.). Sophisticated planning and operation approaches should be adopted by service providers considering the inherent heterogeneity of new [7] applications. Maintaining good levels of QoS would be difficult and expensive without bringing the cloud closer to the edge of the network and the end-users [6].

Mobile Edge Computing (MEC) supports these new [8] applications. MEC provides an Information Technology (IT) service environment and computing resources on the Radio Access Network (RAN) and in proximity to mobile subscribers [8]. They are implemented directly on cellular base stations (BSs) or local wireless access points (APs) using a generic [9] computing platform. Unlike the traditional cloud computing system, which uses remote public servers, MEC servers belong to the network operator. Fig. 1 illustrates the MEC architecture.
MEC is a crucial architecture and technology concept to enable the evolution to 5G, as it contributes to satisfying the demanding requirements in terms of QoS [5]. According to [10] and [11], some of the key features of MEC are proximity, low latency, high bandwidth, flexible deployment, and real-time response. Therefore, with new and heterogeneous requirements, the new set of applications can benefit from the MEC implementation.

The resources of the MEC platform are scarce compared to the cloud and can support a limited number of applications. Thus, cloud servers or even regional MEC servers can serve applications with less stringent latency requirements. Therefore, edge servers do not exclude the cloud; they work in [12] collaboration, creating a data center (DC) hierarchy. The orderly distribution of processing and storage powers diminishes as you approach the edge of the network.

There are several solutions to meet application requirements and provide a good quality of service to users, such as (i) deploying MEC servers at all access points and (ii) increasing the number of servers in the cloud and the capacity of backhaul links. However, these solutions are not ideal for hardware consumption or operating and planning costs. Therefore, it is necessary to have adequate load distribution solutions that respect application requirements and balance the available servers, maximizing the use of available resources.

Resource allocation planning is a necessary step to achieving better QoS levels. Aiming at this state, it presents a Mixed Integer Linear Programming (MILP) model to maximize the number of allocated applications. This work considers a fixed set of application demands and a static scenario with steady-state traffic. Due to the high complexity problem of linear solutions, a new heuristic with low computational cost is proposed. The heuristic is based on the known behavior of different application profiles. To validate the quality of the solutions found by the heuristic, we compare their results with the optimal solution of the MILP. Furthermore, we compare the heuristic with two meta-heuristics: the Genetic Algorithm (GA) [13] and the Particle Swarm Optimization (PSO) [14]. It is possible to improve QoS levels by making the most efficient use of available resources.

The rest of this paper is organized as follows. In section 2, some related works are presented. In section 3, the system model is given, and the load distribution problem is formulated. The heuristics are presented in section 4; the numerical results are discussed in section 5, and conclude the paper in section 6.

II. RELATED WORKS

Recently, the literature discussed the MEC architecture approach and the challenges faced to maintain the QoE and QoS levels of the end-users [15]–[17]. A common need for all network operators is to investigate the allocation of resources distributed across the network to meet new application profiles (i.e., IoT, smart home, and the increasing volume of data). Therefore, in this section, we present some related works to the problem of using MEC resources for new application profiles.

The authors in [18] considered a scalable Software Defined Networking (SDN) scenario where Cloud, Edge, and IoT resources are connected and intend to minimize different network metrics according to specific requirements and provide more efficient solutions; consider a Dijkstra algorithm that produces routing tables capable of reducing network latency. The authors in [19] proposed a reinforcement learning method to minimize the delay of NOMA-MEC (Non-Orthogonal Multiple Access) systems with multiple users and one BS (integrated with a MEC server). By applying NOMA to the MEC offload, they effectively reduced system latency.

In [20], an algorithm is proposed to minimize the total energy consumption considering the latency requirements of time-sensitive computing tasks for users. The simulation results demonstrate that the proposed algorithm can effectively reduce the total consumption of computing tasks and outperform reference schemes with better computational performance. In [10], an ideal offloading method for MEC computing smart mobile devices with latency restriction and searching to achieve energy efficiency is proposed. The results show that energy efficiency is achieved when offloading occurs as the number of devices increases.

In [21], the authors present a set iterative algorithm of offloading decision, cooperative relay selection, and resource allocation based on double Lagrangian decomposition, Sheng-Jing Formula method, and monotonic optimization method where the objective is to minimize execution latency of the task. The simulation results show that the proposed Cooperative Offloading Decision Method (CODM) algorithm can obtain the lowest execution latency for different network parameters, surpassing the existing schemes. In [22], an
offloading mechanism is proposed to maximize the total data rate of a multiuser MEC system in a heterogeneous 5G network, ensuring that the maximum delay time of the applications is respected. The proposed method guarantees the quality of the communication, increase in transfer rate, and spectrum use.

In [23], the authors propose an auction theory that assigns edge servers to mobile device tasks and defines payments for computing services performing the process through a pair of deep neural networks and effectively improving edge server utility, causing the allocation to satisfy the delay in completing the task. Virtual machine (VM) placement and workload assignment in a collaborative MEC system are discussed in [24], where a MILP is proposed to minimize hardware consumption to deploy VMs, meeting latency requirements heterogeneous from different applications. The authors in [25] propose two mechanisms for resource allocation in edge computing systems achieving results close to optimal.

In [9], the authors design a holistic solution for the joint downloading of tasks and the allocation of resources in a MEC network assisted by several servers to maximize the download gains of users. In [26], the authors formulate a problem of a geographic grouping of the MEC system through MILP and a heuristic based on graphs to find the partitions of MEC areas to maximize the use of the servers. In [27], the allocation of resources in a mutative MEC environment is investigated. An intelligent Deep Reinforcement Learning based Resource Allocation (DRLRA) algorithm based on the emerging DRL technology is proposed that obtains better performance than the classic Open Shortest Path First (OSPF) algorithm.

In [12], the orchestration of resources at the edge, caching, collaborative processing, and interference cancellation are addressed to show the benefits of having real-time collaboration, sensitive to the context of MEC systems and mobile devices. In this work, we present a heuristic based on the behavior of applications in [28], [29], which differs from the other papers given. The distribution of available resources on the network is optimized, maximizing the number of requests met with the latency requirement respected.

### III. MEC LOAD BALANCING

This section presents the load distribution problem in scenarios with smart devices, and we give the corresponding mathematical model. We adapted the model from work introduced in [28] and [29]. Fig. 2 illustrates a hierarchy of DC, where there are three layers of computational resources. The first layer represents end devices with extremely limited computing resources. The second is the MEC layer which has network edge resources associated with a BS or AP. The third layer is cloud computing, where available resources are at the network’s core. Compared to mobile devices, servers have more computing resources and greater energy efficiency. The hardware on a server comprises CPU, memory, and storage.

![Figure 2](image_url)

**FIGURE 2.** The computational hierarchy, which is the orderly distribution of powers, which decreases as the edge of the network approaches. The first layer of the model is made up of end users who have limited computing resources and a low power reserve; The second layer is the MEC layer, where the servers are associated with the base stations or access points and have limited computational power but larger than the fan devices; The third layer is cloud computing, which is at the core of the network and has unlimited computing resources.

**A. SYSTEM MODEL**

The model consists of two sets. A set of devices, $D$, where each has a request, $R$, and a topology, $T(N; L)$, with nodes attributes and associated links.

1) Topology Profile

The model considers two layers of computational resources available to fulfill requests. Available resources consist of $N$ servers, divided into MEC servers, $E$, and cloud servers, $C$. Therefore, $E \in N$ and $C \in N$. $CP_n$, where $n = \{1, 2, ..., N\}$. The resources of the first layer consist of $E$ MEC servers, whose computing capacities are $CP_{n,e}$, where $e = \{1, 2, ..., E\}$. The second layer of resources consists of $C$ cloud servers, whose computing capabilities are $CP_{n,c}$ gives the hardware capacity, where $c = \{1, 2, ..., C\}$. The topology parameters are presented in table 1.

| Parameters | Description                  |
|-----------|------------------------------|
| $N$       | Number of Server             |
| $SiD$     | Server Identifier            |
| $CP_n$    | Server Capacity              |
| $L_{n_1,n_2}$ | Link Topology              |
| $DL_{n_1,n_2}$ | Networking Propagation Delay |
| $WL_{n_1,n_2}$ | Networking Links Capacity |

**TABLE 1.** Description of the network topology parameters used. The parameters are used as input to the proposed model and the discussed heuristics in this paper.

$L_{n_1,n_2}$ is a binary variable that takes the value 1 when there is a link between nodes $n_1$ and $n_2$, and 0 otherwise. $DL_{n_1,n_2}$ represents the network propagation delay between the links $n_1$ and $n_2$. $WL_{n_1,n_2}$ is the maximum traffic capacity that
the link between \( n_1 \) and \( n_2 \) can support without generating obstructions or communication failures.

2) Device Profile
There are \( D \) mobile devices making service requests. Each \( i \) device, where \( i \in D \), has a task request called \( R \). Each \( i \) device consists of a set of parameters presented in Table 2. The fixed set of application demands is formed by the set of requests \( RiD \).

\[
R_i = \text{the type of application that the } i \text{ device is requesting. } \Theta_p^i, \quad \text{where } p \in R \text{ represents the maximum delay allowed by the application profile in device } i \in D. \text{ The delay considered does not define total end-to-end latency. Only the latency from the source node of the application } S_i \text{ to its destination node, where the application will be processed or stored. } \gamma_p^i \text{ represents the cost that an application profile must travel through the link between nodes } n_1 \text{ and } n_2, \text{ where } n \in N. \quad \phi_p^i \text{ defines the hardware cost that an application profile needs to meet 1 Megabit per second (Mbps). } \beta_p^i \text{ represents the packet size, in Mb, from the device application profile } i.
\]

3) Variable Definition
The application can be run on a MEC server, \( E \), or a cloud server, \( C \), for each mobile device that performs the offloading. The model then needs to decide where the application \( p \) from device \( i \) will be processed. Once the destination is decided, the application needs to go through the topology to reach the server \( n \). Thus, the model consists of two binary variables. The \( i \) variable is \( A_{i,n} \) which represents which server \( n \) is serving request \( p \) from device \( i \). \( A_{i,n} \) takes the value 1 if request \( R_i \) will be served by the server \( n \) and \( A_{i,n} \) takes the value 0 otherwise. The second, \( X_{n_1,n_2}^i \), is a binary variable that defines the path of the application \( p \) from device \( i \) of its source \( S_i \) to the destination server \( n \), where \( A_{i,n} \) takes 1. So \( X_{n_1,n_2}^i \) takes 1 for every application \( p \) from device \( i \) that travels between nodes \( n_1 \) and \( n_2 \), and \( X_{n_1,n_2}^i \) take 0 otherwise.

B. MIXED INTEGER LINEAR MODEL FOR MEC LOAD BALANCING

Each mobile user request is associated with only one BS to formulate the load distribution problem. Thus, removing the UE-BS connection part of our modeling is possible. As the MEC servers can coexist with the BSs on the same site, the source node of a request can also be the destination node. Equation (1) describes the objective function of maximizing the number of applications allocated, making a fairer distribution of the available resources. The formulation of the distribution problem is presented below:

**Objective Functions:**

\[
MAX \sum \sum_{i \in D} A_{i,n} \quad (1)
\]

**Subject to:**

\[
\sum_{n \in N} A_{i,n} \leq 1, \forall i \in D \quad (2)
\]

\[
\sum_{i \in D} (A_{i,n} \Phi_p^i \times \beta_p^i) \leq CP_n, \forall n \in N, p \in R \quad (3)
\]

\[
\sum_{n_2 \in N} X_{n_1,n_2}^i + A_{i,n_1} - X_{n_2,n_1}^i \leq 1 \mid n_1 = S_i, \forall i \in D \quad (4)
\]

\[
\sum_{n_2 \in N} X_{n_1,n_2}^i + A_{i,n_1} - X_{n_2,n_1}^i = 0 \mid n_1 \neq S_i, \forall i \in D \quad (5)
\]

\[
X_{n_1,n_2}^i = 0 \mid L_{n_1,n_2} = 0, \forall i \in D, [n_1,n_2] \in N \quad (6)
\]

\[
X_{n_1,n_2}^i - X_{n_2,n_1}^i \leq 1, \forall i \in D, [n_1,n_2] \in N \quad (7)
\]

\[
\sum_{n_1 \in N} \sum_{n_2 \in N} X_{n_1,n_2}^i \times Dist_{n_1,n_2} \leq \Theta_p^i, \forall i \in D, p \in R \quad (8)
\]

\[
\sum_{i \in D} (X_{n_1,n_2}^i + X_{n_2,n_1}^i) \times (\gamma_p^i \times \beta_p^i) \leq W_{l,n_1,n_2}, \forall [n_1,n_2] \in N \quad (9)
\]

The equation (2) guarantees that requests can be served exclusively by a server. The equation (3) indicates that the resources used by requests are limited by the maximum number of available hardware capacity. The variable \( A_{i,n} \) is defined in III-A3. The equation (4) and (5) guarantee the flow of the request from its source \( S_i \) to the destination \( n \), where \( A_{i,n} \) takes on the value 1. When the application of device \( i \) passes through the nodes \( n_1 \) and \( n_2 \) to the variable \( X_{n_1,n_2}^i \) takes on the value 1.

The equation (6) enforces that a request from the device \( i \) cannot travel between two nodes, \( n_1 \) and \( n_2 \), that do not have links connecting them. The equation (7) guarantees that the device request \( i \) will not loop between two nodes, \( n_1 \) and \( n_2 \), wasting resources. The equation (8) ensures that the maximum delay accepted by the application \( p \), where \( p \in R \) of the device \( i \) will be respected. The equation (9) limits the
number of requests \( p \) that can travel between nodes \( n_1 \) and \( n_2 \). Note that we do not consider node affinity, i.e., the ability to define the node at which an application will be executed. However, this can be included in the model by extending the application definition and including a few constraints.

### IV. OPTIMIZATION METHODS

In this session, we introduce the proposed heuristic, with low computational cost, to solve the problem of heterogeneous load distribution in a hierarchy of DCs, objective function, and the routing algorithm used. AG and PSO are also presented.

#### A. OBJECTIVE FUNCTION AND SOLUTION ENCODING

This subsection will present how the objective function value is calculated for the heuristics proposed in this work. We will first present how the solutions are coded in the proposed techniques. It is worth mentioning that a solution in AG is called a chromosome, while in PSO, it is a particle. The equation 10 describes the vector of possible solutions \( PS \).

\[
PS_i = x \forall i \in D, n \in N
\]  

(10)

A vector of \( D \) positions represents a solution. The index \( i \) of the vector \( PS_i \) represents the request identifier, \( RiD \), and \( x \) represents the server identifier, \( SiD \), where \( x \in N \). The value of \( x \) is defined according to the individual strategies of each of the proposed heuristics, presented below. The solution encoding method guarantees the constraint presented by the equation (2). Fig. 3 represents the coding model for a solution to our problem.

![FIGURE 3. The solution vector represents a possible allocation scenario for existing requests. The \( PS_i \) vector serves as input to the objective function, where we have the index as the request identifier and the value assigned to that position, the destination server.](image)

Having the \( PS \) vector, a possible solution to the problem, the heuristic can then call the objective function to check the quality of that solution. The algorithm 1 presents the sequential FO scheme for the algorithms.

For each of the positions of the solution vector, the first step is to check if there is enough computational capacity in the destination DC to meet the request. After the verification, it is necessary to verify if there is a possible route that meets the application’s latency requirements. The values are updated if all conditions are met and the application can reach its destination.

#### B. DIJKSTRA’S ALGORITHM FOR ROUTING

The Dijkstra algorithm is an approach to finding the shortest paths between a specific source node and a destination node in a network [18], [30]. Vertices represent network devices in a graph, edges represent network links, and edge weights represent different network parameters. A full-duplex connection between two network devices can be arranged as a pair of simplex connections. Since we assume at least one path between one network device and the other, the derived graph is connected.

Dijkstra’s algorithm has a simple procedure. The essence of the algorithm is to find the shortest path between two nodes and a graph \( T(N, L) \). Following the connected graph model, it is easy to organize the weight of the links according to different evaluation parameters (such as communication latency and maximum link capacity) that can also meet the requirements of applications in MEC scenarios. The algorithm 2 presents the pseudocode.

#### Algorithm 1: Objective Function Algorithm

**Input:** \( PS \)  
**Output:** \( FO = 0 \)

\[
\text{begin} \\
\text{for } i \leftarrow 1 \text{ to } D \text{ do} \\
\text{ } n = PS_i \\
\text{if } CP_n > \Phi_i \text{ then} \\
\text{ } \text{if } Dijkstra’s(T(N, L), S_i, n) = True \\
\text{ } \text{Update } CP_n \\
\text{ } \text{FO } + + \\
\text{end if} \\
\text{end if} \\
\text{end for} \\
\text{end} 
\]

#### Algorithm 2: Dijkstra Algorithm

**Input:** \( T(N, L), D_i, n \)  
**Output:** \( T(N, L) \)

\[
\text{begin} \\
\Theta = \text{Dijkstra’s}(T(N, L), S_i, n) \\
\text{if } \Theta \leq \Theta_0 \text{ then} \\
\text{Update } T(N, L) \\
\text{return } True \\
\text{end if} \\
\text{return } False \\
\text{end} 
\]
The proposed heuristic to solve the resource allocation problem is based on the probability of a MEC server serving an application profile \( R \). The probability \( P_r \) is calculated based on the average result of the optimal solutions found by the model described in load balancing MEC. After simulating the experiments, it was verified the probability, in the optimal solution, of a particular application being allocated to the edge or core of the network. Therefore, we probability that a request that specifies \( R \) will be served on a MEC server.

We use this value as the maximum threshold for the number of requests of a given type to be served by MEC servers.

We assume that each MEC server has an intelligent system that knows the proportion of each type of application that it can handle without compromising the network. It can determine whether to accept a given application \( R \) based on its current state. The strategy is to limit applications with lower sensitivities from being served at the network’s edge, taking these applications to the central cloud. So, when an application enters the network, it looks for an available server to serve it, whether it is a MEC server or a cloud server, but MEC servers prefer to serve the most latency-sensitive requests. The first step of the proposed heuristic is to create an empty \( PS_i \) vector, where each of the destinations of the existing requests will be stored. Pseudocode 3 presents the flow of creation of the possible heuristic solution.

Where \( O_{ri} \) is the occupancy rate of the server \( n \) by the application profile \( R \). Therefore, if the occupancy of the server \( n \) by the application profile \( r \) is less than the proportion \( P \) calculated previously, applications can be allocated to it. When serving a request, the server updates its current \( O_{ni} \) state. After deciding where the application will be allocated, when creating \( PS_i \), the heuristic calls the objective function 1 to measure the fitness level of the solution. The calculation is performed according to IV-A. As long as the stopping criterion is not satisfied, the solution goes through an update process in search of better results. The solution’s update is done by changing the destination of applications for a new one, following the proportions, or being sent to the cloud. The stopping criterion used is the number of generations.

It is worth mentioning that the heuristic does not prevent applications from being served on a given node; it only limits the number of \( R \) applications of a specific type that can be served on the MEC servers. The 3 algorithm presents the proposed solution that is based on the behavior of the application profiles that are discussed in section V.

D. GENETIC ALGORITHM FOR MEC LOAD DISTRIBUTION

A Genetic Algorithm is a meta-heuristic for solving optimization problems inspired by the process of natural selection. In the algorithm, a population of possible solutions, called chromosomes, evolves within the optimization problem’s domains towards an optimal solution [31]. Each individual corresponds to a chromosome, which is the coded representation of the solution. Each chromosome can be represented by a vector, with values representing the problem domain. Each chromosome is associated with a fitness level, calculated by the objective function presented in 1. The GA is an iterative process where a set of processes, called generation, creates a new population through the random recombination and mutation of selected individuals from the current population to generate the next population. Individuals are stochastically selected, with those with better fitness being favored over those with lower fitness. Generally, the evolution process ends up reaching a certain number of generations or finding a satisfactory fitness level.

The first step of GA is initializing the population, which is a set of vectors with random values. Each vector generated represents a chromosome. The population is randomly initialized using a linear distribution. The proposed chromosomal (C) representation is presented in section IV-A.

The equation 10 represents how chromosomes are generated. \( C \) is a vector of \( D \) positions, where each position has a random value of [1, N]. After initializing generations, the objective function evaluates each individual in the pop-
ulation. Each of the subsequent generations, called daughter generations, is created by natural selection methods, whereas the previous generation, called the parent generation, uses the selection, crossover, and mutation operators. This process is carried out until the predetermined stopping criterion is satisfied.

The parents must be chosen from the current population to generate the child population in each generation. The proposed algorithm uses the roulette method, where individuals from a generation are chosen through a roulette drawing. In this method, each population member is represented on the roulette wheel according to their fitness level. Thus, individuals with high fitness are given a more significant portion of the roulette wheel. In contrast, those with lower fitness are given a relatively minor portion of the roulette wheel. Finally, the roulette wheel is spun a certain number of times, depending on the size of the population, and those drawn on the roulette wheel are chosen as parent individuals.

Once the parents are selected, the crossover process can be performed with a probability rate called the crossover rate. In this study, we used a one-point crossover. A crossing point is chosen, and from this point, the genetic information of the parents will be exchanged. Information prior to this point in one parent is linked to information after this point in the other parent, generating a new individual. After crossover, the mutation occurs. The mutation operator is necessary for the introduction and maintenance of the genetic diversity of the population, arbitrarily altering one of the genes of the chosen individual, thus providing means for introducing new elements into the population. In this way, the mutation ensures that the probability of reaching any point in the search space will never be zero, in addition to circumventing the problem of local minima since, with this mechanism, the direction of the search is slightly altered. The mutation operator is applied to individuals with a probability named mutation rate. Table 3 presents the parameters used by the algorithm that were empirically chosen through exhaustive tests.

| Parameters             | Value   |
|------------------------|---------|
| Number of Generations  | 1000    |
| Size of Population     | 100     |
| Crossover Rate         | 0.8     |
| Mutation Rate          | 0.1     |
| Selection              | Roulette|
| Stop Condition         | Number of Generations |

TABLE 3. Configuration of simulation parameters used in GA.

E. PARTICLE SWARM OPTIMIZATION FOR MEC LOAD DISTRIBUTION

The term "Swarm Intelligence" is used to describe algorithms inspired by the collective behavior of colonies and other animal societies [32]. The particle swarm optimization (PSO) is then an algorithm based on self-organizing systems’ collective and decentralized behavior. Thus, the PSO simulates a flock of birds or a shoal of fish in search of food. It treats each solution to the optimization problem as a bird that flies at a certain speed in the search space, and its speed is dynamically adjusted [14]. So, a swarm solution is a particle in a multidimensional search space. Consider that a solution is a vector of D positions, where each position represents a request and its value is its destination, that is, the address of the MEC or Cloud server. The encoding is discussed in IV-D.

Each particle has a flight acceleration that determines its direction and speed, so they move within the search space at speed adjusted to each iteration according to cognitive and social factors. Similar to that discussed in IV-D, the population is initialized at random through a linear distribution. The solution, once created, is evaluated using the 1 equation, like in IV-A. After each particle has a fitness level, the particles move in the search space, looking for better solutions. With each movement that is performed by the particles, their fitness level is updated. The particle’s motion is described by the equation 11.

\[
\text{Particle}_i(t + 1) = \text{Particle}_i(t) + V_i(t + 1) \quad (11)
\]

\(\text{Particle}_i\) is an element of a set of solutions. \(\text{Particle}_i\) is the same as \(C_1\). \(V_i\) is the speed at which the particle moves in the search space, at time \(t\). \(V\) is presented in the equation 12.

\[
V_i(t + 1) = W(t) + V_i(t) + C_1(t) \times \text{rand}(C_1(t) - \text{Particle}_i(t)) + C_2(t) \times \text{rand}(C_2(t) - \text{Particle}_i(t))
\]

Where \(W\) is inertia at time \(t\), that is, the tendency of the particle to continue in the same direction. \(C_1\) is the cognitive factor, the tendency of the particle to move from past learning. \(C_2\) is the social factor, the tendency of the particle to move from the learning obtained with the neighboring particles. \(\text{rand}_1\) and \(\text{rand}_2\) assume a random value between 0 and 1. The 4 pseudocode describes the application of the functions.

Algorithm 4: Particle Update

\begin{verbatim}
Input: Particle\(_i\), V\(_i\)(t + 1)
Output: Particle\(_i\)(t + 1)
begin
for i ← 1 to D do
    Particle\(_i\)(t + 1) = Particle\(_i\)(t) + V\(_i\)(t + 1)
    Round(Particle\(_i\)(t + 1))
while Particle\(_i\)(t + 1) > N do
    Particle\(_i\)(t + 1) = Particle\(_i\)(t + 1) - N
end while
end
\end{verbatim}

The first step of the update is to add the velocity vector to the particle vector, updating each of the \(i\) elements within
the search space. After that, the elements that are outside the sample space undergo a correction to make the solution valid. After the solutions are updated, they are evaluated again until the stopping criterion is satisfied, either the number of moves or a determined fitness level. The parameters used are presented in Table 4 and were empirically chosen through exhaustive tests.

| Parameters                  | Value             |
|-----------------------------|-------------------|
| Number of Generations       | 1000              |
| Size of Population          | 100               |
| Cognitive Factor            | 1.8               |
| Social Factor               | 2.2               |
| Inertia                     | 2                 |
| Stop Condition              | Number of Generations |

**TABLE 4.** Configuration of simulation parameters used in PSO.

**V. PERFORMANCE EVALUATION**

In this section, the numerical results are presented, and the performance of the proposed algorithm is analyzed under different traffic conditions. For each scenario profile, thirty experiment runs were performed. We consider a MEC-enabled mobile network system in which each BS is equipped with a MEC server that assists mobile users in performing computational tasks. To achieve realistic propagation delays, seven cities in the eastern United States were chosen as edge nodes, which host small DCs, and two cities in the west as core nodes, which host cloud DCs. The latency of network propagation between each DC is obtained in [33]. The details of the edge and cloud nodes are listed in Table 5. The total load in each city is defined randomly using a linear distribution. The average of the various experiments was calculated to minimize the effect of outliers and draw a common behavioral profile in each scenario. The confidence interval used is 95%. The average results are shown. The proposed model can be used for systems with different computational hierarchies. The execution time for MILP to find the optimal solution is extremely long for $\Omega \gg 10$. The machine used has an Intel I7 8750H processor with six cores - 4.1 GHz, 16GB DDR4.

| Application Profile          | $\Theta$(ms) | $\phi$(Unit) |
|------------------------------|--------------|--------------|
| AR/VR                        | 10           | 0.03         |
| Factory Automation [FA]      | 20           | 0.009        |
| Data Backup [DB]             | 1000         | 0.0001       |
| Smart Grid [SG]              | 50           | 0.007        |
| Smart Home [SH]              | 60           | 0.0001       |
| Medical [M]                  | 40           | 0.02         |
| Environmental Monitoring [EM]| 1000         | 0.02         |
| Tactile Internet [TI]        | 1            | 0.005        |

**TABLE 6.** Configuration of application parameters used in the simulation. It presents the type of application, with the maximum latency allowed and the number of computational resources that it needs to be processed. [7]

The proposed heuristic (IV-C) has an acceptance rate of 40% and 82% higher than AG and PSO, respectively. Regarding the optimal solution, the acceptance rate of the proposed heuristic is 91%. The good results are a reflection of the optimization strategy used by the heuristic, which preferentially allocates applications that allow high latency values in cloud servers further away from the edge of the network. In addition to maximizing the acceptance rate, it implies a better use of available network resources, minimizing the unnecessary use of these resources. Fig. 5 presents the acceptance rate of the different techniques presented. It is also noted that with the saturation of the network system, the presented heuristics present a similar drop behavior, with the proposed heuristic being the smoothest. In a high-density scenario, the proposed heuristic has an acceptance rate of 60% higher than the GA and 130% higher than the PSO.

In the case of uncontrolled growth in the number of requests existing on the network, the computer system could collapse and not be able to meet all requests. This would occur due to the overload that the network is exposed to and the lack of an adequate solution that optimizes the use of available resources. As it was observed, with better solutions such as the proposed heuristic, the gain in the adequate use of resources can be up to approximately two times when compared to other techniques proposed in the literature. Without having a good resource allocation and distribution strategy, especially the edge system, it would not be able to maintain QoS levels for end-users.

The proposed heuristic consumes resources from cloud servers up to 3.5 times more than AG and 11.7 times more than PSO in dense scenarios. These high values are directly related to the strategy’s higher acceptance rate index. Fig.

| City            | HW Capacity ($CP_{lu}$) | Type  |
|-----------------|-------------------------|-------|
| Atlanta         | 110                     | Edge  |
| Boston          | 180                     | Edge  |
| Detroit         | 170                     | Edge  |
| Washington      | 120                     | Edge  |
| Manhattan       | 480                     | Edge  |
| Miami           | 70                      | Edge  |
| Orlando         | 600                     | Edge  |
| Oklahoma City   | 2741                    | Cloud |
| Kansas City     | 2069                    | Cloud |

**TABLE 5.** Configuration of scenario simulation parameters. Cities chosen to simulate the position of servers [24], including their capabilities and server type.

We consider that requests always come from an edge node, a base station, or an access point, which has an associated MEC server. A parameter $\Omega$ is defined as a scale factor to expand and reduce the number of applications, linearly, between the seven cities and to see the performance of the proposed algorithms. For $\Omega = 1$, there are 500 service requests. $\Omega = \{1, 10\}$. The sets are formed by applications drawn at random following a linear distribution. Eight application profiles are used as input to the model, and their latency and hardware requirements are listed in Table 6 [7]. The results are compared in terms of service acceptance rate with the optimal solution found by MILP and the meta-heuristics presented.
As latency sensitivity increases, applications tend to look for servers closer to the edge of the network. Fig. 7 presents the behavior of the application profiles for the optimal solution in low and high density ($\Omega = 1$) and high ($\Omega = 10$) scenarios. It is observed that more latency-sensitive applications, such as Tactile Internet and AR/VR, focus exclusively on edge servers at MEC. Applications that do not require real-time responses, such as Data Backup, tend to be served frequently on servers at the center of the network, in the CC. Thus, the influence of latency on the destination of different application profiles is clear. Therefore, requests that are more sensitive to latency seek servers at the edge of the network; and the less the sensitivity becomes, they tend to move away. This behavior occurs to allow more sensitive applications to be able to be met with their requirements being respected.

VI. CONCLUSIONS

MEC is an efficient technology to support the growing demand for communication and computing from future mobile networks. Service providers are adapting their architectures and looking to offer better levels of QoE and QoS to end-users, so it is necessary to have an adequate operating plan for future architectures, including the MEC system. The MEC will enable the distribution of computing resources from the cloud to the edge of the network, allowing delay-sensitive applications to run close to end-users while eliminating the use of backhaul links.
In this work, we propose an optimal model and a new low computational cost heuristic for allocating network applications, maximizing the use of available computational resources. The proposed heuristic is based on the behavior of the application profiles. The results reach, on average, 91% of the optimal solution found by MILP, while AG and PSO get 65% and 50%, respectively. The allocations made meet the application’s QoS requirements, such as latency and capacity of available resources. The results show that the heuristic proposed in this article can effectively improve service quality and user experience. From this study, challenges are still open, such as the mobility of users and the exchange of MEC servers during the session.

ACKNOWLEDGMENT
This study was financed in part by the Federal University of Para (UFPA) and National Counsel of Technological and Scientific Development (CNPq).

REFERENCES

[1] G. Forecast et al., “Cisco visual networking index: global mobile data traffic forecast update, 2017–2022,” Update, vol. 2017, p. 2022, 2019.
[2] A. Cheri and G. Jean, “Optimal matching between energy saving and traffic load for mobile multimedia communication,” Concurrency and Computation: Practice and Experience, vol. 33, no. 4, p. e5035, 2021.
[3] E. Borgia, R. Bruno, M. Conti, D. Mascitti, and A. Passarella, “Mobile edge clouds for information-centric iot services,” in 2016 IEEE symposium on computers and communication (ISCC). IEEE, 2016, pp. 422–428.
[4] Y. Xu, G. Gui, H. Gacanin, and F. Adachi, “A survey on resource allocation for 5g heterogeneous networks: current research, future trends and challenges,” IEEE Communications Surveys & Tutorials, 2021.
[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile edge computing—a key technology towards 5g,” ETSI white paper, vol. 11, no. 11, p. 1–16, 2015.
[6] L. Gupta, R. Jain, and H. A. Chan, “Mobile edge computing—an important ingredient of 5g networks,” IEEE Software Defined Networks Newsletter, 2016.
[7] E. Sturzinger, M. Tornatore, and B. Mukherjee, “Application-aware resource provisioning in a heterogeneous internet of things,” in 2017 International Conference on Optical Network Design and Modeling (ONDM), IEEE, 2017, pp. 1–6.
[8] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, “A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art,” IEEE Access, vol. 8, pp. 116974–117017, 2020.
[9] T. X. Tran and D. Pompili, “Joint task offloading and resource allocation for multi-server mobile-edge computing networks,” IEEE Transactions on Vehicular Technology, vol. 68, no. 1, pp. 856–868, 2018.
[10] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, “Mobile edge computing and networking for low and low-latency internet of things,” IEEE Communications Magazine, vol. 56, no. 5, pp. 39–45, 2018.
[11] G. Orsini, D. Bade, and W. Lamersdorf, “Computing at the mobile edge: Designing elastic android applications for computing offloading,” in 2015 8th IFIP wireless and mobile networking conference (WMNC). IEEE, 2015, pp. 112–119.
[12] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, “Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges,” IEEE Communications Magazine, vol. 55, no. 4, pp. 54–61, 2017.
[13] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, “An improved multi-objective genetic algorithm with heuristic initialization for service placement and load distribution in edge computing,” Computer Networks, vol. 194, p. 108146, 2021.
[14] N. Geng, Z. Chen, Q. A. Nguyen, and D. Gong, “Particle swarm optimization algorithm for the optimization of rescue task allocation with uncertain time constraints,” Complex & Intelligent Systems, vol. 7, no. 2, pp. 873–890, 2021.
[15] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, “On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1657–1681, 2017.
[16] P. Mach and Z. Bencvar, “Mobile edge computing: A survey on architecture and computation offloading,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628–1656, 2017.
[17] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450–465, 2017.
[18] A. Buzachis, A. Celesti, A. Galletta, J. Wan, and M. Fazio, “Evaluating an application aware distributed dijkstra shortest path algorithm in hybrid cloud/edge environments,” IEEE Transactions on Sustainable Computing, 2021.
[19] P. Yang, L. Li, W. Liang, H. Zhang, and Z. Ding, “Latency optimization for multi-user noma-mec offloading using reinforcement learning,” in 2019 28th Wireless and Optical Communications Conference (WOCC). IEEE, 2019, pp. 1–5.
[20] M. Zhao, J.-J. Yu, W.-T. Li, D. Liu, S. Yao, W. Feng, C. She, and T. Q. Quek, “Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems,” IEEE Transactions on Vehicular Technology, vol. 70, no. 10, pp. 10925–10940, 2021.
[21] Z. Kuan, Z. Ma, Z. Li, and X. Deng, “Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing,” Journal of Systems Architecture, vol. 118, p. 102167, 2021.
[22] J. Xue, H. Shao, and Q. Ma, “Resource allocation for system throughput maximization based on mobile edge computing,” in Proceedings of the 2018 International Conference on Electronics and Electrical Engineering Technology, 2018, pp. 177–181.
[23] L. Liu, H. Ge, S. Li, X. Chen, H. Gong, and Y. Cui, “Resource allocation strategy based on improved auction algorithm in mobile edge computing environment,” in 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA). IEEE, 2021, pp. 355–363.
[24] W. Wang, Y. Zhao, M. Tornatore, A. Gupta, J. Zhang, and B. Mukherjee, “Virtual machine placement and workload assignment for mobile edge computing,” in 2017 IEEE 6th International Conference on Cloud Networking (CloudNet). IEEE, 2017, pp. 1–6.
[25] T. Bahrreini, H. Badri, and D. Grou, “Mechanisms for resource allocation and pricing in mobile edge computing systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 3, pp. 667–682, 2021.
[26] M. Bouet and V. Conan, “Mobile edge computing resources optimization: A geo-clustering approach,” IEEE Transactions on Network and Service Management, vol. 15, no. 2, pp. 787–796, 2018.
[27] J. Wang, L. Zhao, J. Liu, and N. Kato, “Smart resource allocation for mobile edge computing: A deep reinforcement learning approach,” IEEE Transactions on Emerging topics in computing, 2019.
[28] R. Vieira, C. A. de Mattos Teixeira, and D. Cardoso, “Optimizing resource allocation in hierarchically distributed data centers,” in Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. SBC, 2019, pp. 556–565.
[29] R. F. Vieira, P. H. A. Pereira, and D. L. Cardoso, “Resource allocation optimization for hierarchical cloud data centers,” in 2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech). IEEE, 2018, pp. 1–6.
[30] M. Luo, X. Hou, and J. Yang, “Surface optimal path planning using an extended dijkstra algorithm,” IEEE Access, vol. 8, pp. 147 827–147 838, 2020.
[31] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” 1988.
[32] J. Kennedy, “Swarm intelligence,” in Handbook of nature-inspired and innovative computing. Springer, 2006, pp. 187–219.
[33] “Network testing solutions.” [Online]. Available: https://wondernetwork.com/
RAFAEL F. VIEIRA Received the Bachelor’s in computer engineering in 2016 and master’s in electrical engineering with emphasis in applied computing from the Federal University of Pará (UFPA) in 2018, where he is currently studying for a doctorate. He is currently a member of the Operational Research Laboratory (LPO). His research topics are focused on evolutionary computing, telecommunication networks and computer networks. He is an Associate Member of the Brazilian Computer Society (SBC).

DANIEL DA SILVA SOUZA Received the Bachelor’s degree in Information Systems in 2016 and master’s degree in electrical engineering with emphasis in applied computing from Federal University of Pará (UFPA) in 2018, where he is currently pursuing a Doctoral degree. He is currently a member of the Operational Research Laboratory (LPO). His research topics are focused on Human-Computer Interaction, User Experience, Software Engineering, and Computer Networks. He is an Associate Member of the Brazilian Computer Society (SBC).

MARCELINO SILVA DA SILVA received the bachelor’s degree in computer engineering from the Federal University of Pará (UFPA), Belém, Brazil, in 2006, and the master’s and Ph.D. degrees in electrical engineering from UFPA, in 2008 and 2014, respectively. He is currently a Professor with the Telecommunication and Computer Engineering Department, UFPA. His research interests include optimization, performance evaluation, Markov processes, and numerical models.

DIEGO LISBOA CARDOSO He holds a Bachelor’s degree in Computer Science from the University of Amazônia (2002), a Master’s degree (2005) and a PhD degree (2010) in Electrical Engineering from the Federal University of Pará and a post-doctoral fellowship at the Royal Institute of Technology of Sweden (KTH). He was Director of Technology at the State Department of Education of the Government of the State of Pará (2008-2009). He was Pro-Rector of Undergraduate Education at the Federal University of South and Southeast Pará (UNIFESSPA) (2014). He works as an associate professor at the Federal University of Pará in the School of Computer Engineering and Telecommunications and the Graduate Program in Electrical Engineering (PPGEE). He has experience in Computer Science and Computer Engineering, with emphasis on Performance Evaluation, working mainly on the following topics: Digital TV, Access Technologies, Markovian performance and simulation models, applied computational intelligence and optimization techniques.