UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Optimizing the Fog Service Placement with R3GP: a Rotation-Guided Greedy Genetic Particle algorithm

Dissertação de Mestrado

Jonathan Santos Cunha

São Cristóvão – Sergipe

2023

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Jonathan Santos Cunha

Optimizing the Fog Service Placement with R3GP: a Rotation-Guided Greedy Genetic Particle algorithm

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Rubens Matos de Souza Júnior

São Cristóvão – Sergipe

2023

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: Jonathan Santos Cunha

Em 17 dias do mês de agosto do ano de dois mil e vinte três, com início às 09h00min, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Jonathan Santos Cunha**, que desenvolveu o trabalho intitulado: **" *Optimizing Fog Service Placement with R3GP: A Rotation-Guided Greedy Genetic Particle algorithm*",** sob a orientação do Prof. Dr. **Rubens de Souza Matos Junior**. A Sessão foi presidida pelo Prof. Dr. **Rubens de Souza Matos Junior** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Jamilson Ramalho Dantas** (UFPE) e, em seguida, o Prof. Dr. **Ricardo josé Paiva de Britto Salgueiro** (Dcomp/UFS) e o Prof. Dr. **André Britto de Carvalho** (Procc/UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) aprovada "*(aprovado/reprovado)*". Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos*", 17 de agosto de 2023.

Prof. Dr. Rubens de Souza Matos Júnior
(PROCC/UFS)
Presidente

Prof. Dr. André Britto de Carvalho
(PROCC/UFS)
Examinador Interno

Prof. Dr. Jamilson Ramalho Dantas
(UFPE)
Examinador Externo

Prof. Dr. Ricardo José Paiva de Britto Salgueiro
(Dcomp/UFS)
Examinador Externo ao programa

Jonathan Santos Cunha
Candidato

Documento assinado digitalmente
gov.br ANDRE BRITTO DE CARVALHO
Data: 01/12/2023 16:00:06-0300
Verifique em https://validar.iti.gov.br

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Cunha, Jonathan Santos
C972o Optimizing the fog service placement with r3gp: a rotation-guided greedy genetic particle algorithm / Jonathan Santos Cunha ; orientador Rubens Matos de Souza Júnior. - São Cristóvão, 2023.
130 f. : il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2023.

1. Internet das coisas. 2. Computação em nuvem. I. Souza Júnior, Rubens Matos de orient. II. Título.

CDU 004.73

À minha família. Aos meus avós Arnaldo, Izaura, José Vicente e Maria Rosilva. À minha namorada Franciely. Ao meu irmão Igor Nathan. Em especial, dedico e deixo aqui registrado o meu "Muito obrigado!" às duas pessoas que fizeram de tudo para que tivéssemos a melhor educação: meu pai Silvânio e minha mãe Maria Elizia.

Agradecimentos

Agradeço ao professor Rubens que iluminou todas as minhas dúvidas e ajudou a guiar este trabalho pelos melhores caminhos.

Agradeço aos professores Ricardo Salgueiro e Edilayne Salgueiro por me acolherem e por todas as oportunidades e ajuda que me forneceram durante toda a minha trilha acadêmica.

Agradeço a Itauan e Wesley que foram essenciais em minha evolução acadêmica. Os dois me acolheram, tiveram paciência e ensinaram-me tudo que sei hoje. Parte deste trabalho foi inspirado nos trabalhos de vocês. Super agradeço todo o conhecimento compartilhado.

Agradeço a todos que fizeram e fazem parte do ELAN, pela amizade e apoio nessa minha jornada acadêmica.

Grande abraço a todos!

"Como faz pra colocar
um elefante na geladeira?
Você abre a geladeira,
coloca o elefante dentro
e fecha a geladeira.
E como faz pra colocar
uma girafa na geladeira?
Você abre a geladeira,
tira o elefante,
coloca a girafa
e fecha a geladeira."
Autor(a) desconhecido(a).

Resumo

O paradigma Fog Computing surgiu como uma solução complementar à Cloud Computing para levar o processamento de aplicações para dispositivos da borda da rede (edge computing devices), que interligam-se aos dispositivos típicos da Internet das Coisas (IoT - Internet of Things). Entretanto, a capacidade limitada dos nós *edge* lança alguns desafios no gerenciamento dos recursos disponíveis para as aplicações distribuídas. O service placement em Fog Computing é um problema NP-completo que consiste no gerenciamento da decisão sobre em qual nó da Fog o serviço de uma aplicação IoT será executado. Se não houver recurso suficiente na Fog, a aplicação é enviada para a Cloud. Este trabalho consiste na otimização do Fog Service Placement Problem para execução de aplicações IoT, empregando um estudo de caso referente a sistemas de prevenção de colisões de veículos em vias urbanas. O problema é formulado como um modelo de satisfação de restrições para otimização de cinco funções objetivos: makespan, energy consumption gap, CPU load-balancing, memory load-balancing e bandwidth load-balancing. Neste trabalho é proposto um algoritmo para otimização do problema, denominado Rotation-Guided Greedy Genetic Particle (R3GP). O estudo é conduzido com um experimento *in silico* que compara o algoritmo com outros encontrados na literatura. Os resultados estatísticos mostram que o R3GP consegue superar os algoritmos comparados, principalmente, na otimização da métrica energy consumption gap.

Palavras-chave: service placement. fog computing. cloud computing. edge computing. optimization. internet of things.

Abstract

The Fog Computing paradigm emerged as a complementary solution to the Cloud Computing to bring application processing to edge computing devices, which interconnect with typical Internet of Things (IoT) devices. However, the limited capacity of edge nodes poses some challenges in managing the resources available to distributed applications. Service placement in Fog Computing is an NP-complete problem that consists of managing the decision on which Fog node the service of an IoT application will run. If there is not enough resource in the Fog, the application is sent to the Cloud. This work consists of optimizing the Fog Service Placement Problem for the execution of IoT applications, applying a case study regarding vehicle collisions on urban roads. The problem is formulated as a Constraint Satisfaction Problem for optimization of five objective functions: makespan, energy consumption gap, CPU load-balancing, memory load-balancing and bandwidth load-balancing. In this work, an algorithm for optimization of the problem, named Rotation-Guided Greedy Genetic Particle (R3GP), is proposed. The study is conducted with an *in silico* experiment that compares the algorithm with others found in the literature. Statistical results show that R3GP can outperform the compared algorithms, mainly in optimizing the energy consumption gap metric.

Keywords: service placement. fog computing. cloud computing. edge computing. optimization. internet of things.

List of Figures

List of Tables

List of Algorithms

List of abbreviations and acronyms

DAG	Directed Acyclic Graph
DEBTS	Delay Energy Balanced Task Scheduling
DMS	Double-Matching Strategy
I2I	Infrastructure-to-Infrastructure
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ITS	Intelligent Transportation System
FFD	First-Fit Decreasing
FSPP	Fog Service Placement Problem
GA	Genetic Algorithm
GLS	Guided-Local Search
GQM	Goal Question Metric
GUI	Graphical User Interface
HC	Hill Climbing
HDD	Hard Disk Drive
LTE-A	Long-Term Evolution Advance
MOPSO	Multi-Objective Particle Swarm Optimization
MOWOA	Multi-Objective Whale Optimization Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm 2
OBU	On Board Unit
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RFID	Radio Radio Frequency IDentification

RSU	Road Side Unit
R3GP	Rotation-Guided Greedy Genetic Particle
SFC	Service Function Chain
SOA	Service-Oriented Architecture
SPEAII	Strength Pareto Evolutionary Algorithm 2
TTY	Teletypewriter
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicle Ad-hoc NETwork
VLC	Visible Light Communication
WiMAX	Worldwide Interoperability for Microwave Access

List of symbols

$\in$	Element of
$=$	Equal
μ	Lowercase *mu*
$\neq$	Not equal
$\mid$	Such that

Contents

1 Introduction

Internet of Things (IoT) is a computing paradigm that refers to interconnected objects that use sensors capable of collecting data from the physical environment around them and, based on the analysis of the information, acts on it using actuators. These objects intercommunicate by exchanging knowledge over the Internet or a private network (CHOI; AHN, 2021). However, the limitations of these types of devices, such as hardware capacity constraints, bring some challenges in data processing. In this way, the Cloud Computing paradigm emerges as an option to overcome the restrictions of IoT devices, providing data centers with robust servers for massive data processing and management and deployment of applications in a large-scale environment (NATH et al., 2019).

In contrast to its benefits, Cloud Computing has some disadvantages such as, for example, difficulty in meeting IoT applications that require low communication latency. The paradigm has a high response time for applications that are sensitive to latency, such as communication between smart vehicle networks, patient monitoring, industrial robot monitoring and online games (KIM; CHUNG, 2018b). For example, to avoid a traffic collision, an application for self-driving cars requires real-time response to calculate the reaction decision. Therefore, it is not recommended to deploy these types of applications in the Cloud.

In order to overcome the drawbacks of Cloud Computing, (BONOMI et al., 2012) proposed Fog Computing, a paradigm that emerges as a complementary solution to Cloud. Designed to bring processing power to edge devices, close to IoT devices, acting as a bridge between Edge and Cloud Computing. Fog's ability to bring application processing closer to the end devices layer makes it possible to reduce service latency and network overload (MARTIN; KANDASAMY; CHANDRASEKARAN, 2020). However, running services on edge devices poses some challenges such as controlling and managing the availability of resources for distributed applications, given that these nodes have limited computing resources (MEHRAN; KIMOVSKI; PRODAN, 2019).

Some of the challenges of the state-of-the-art fog computing paradigm include device heterogeneity, security, network latency, dynamic behaviors and fault tolerance. As an example use case, we have VANETs (Vehicle Ad-Hoc Networks), in which intelligent vehicles can share information with the infrastructure as they transit between access points along highways (EYCKERMAN et al., 2020). In order to deal with these problems efficiently, one of the measures to be taken is the application of better placement strategies for application services along the Fog infrastructure.

Service placement in a Fog Computing environment, formally known as Fog Service Placement Problem (FSPP) (LIU et al., 2022; AYOUBI; RAMEZANPOUR; KHORSAND, 2021), is a problem that consists of an entity called Fog Controller responsible for generating the placement plan of IoT device application requests. This placement plan contains decisions about which Fog node an application's service will run on. If no Fog node has enough resources to process the requested application services, the controller offloads the application to the Cloud (SKARLAT et al., 2017a). The service placement problem is an NP-complete problem derived from the Knapsack problem. For this reason, many studies use heuristic and meta-heuristic approaches to find a solution close to the optimum in a short time (MEHRAN; KIMOVSKI; PRODAN, 2019).

As fog computing is an emerging paradigm, some gaps are found in the literature, such as solving the FSPP for more than three objectives (AYOUBI; RAMEZANPOUR; KHORSAND, 2021) and considering the load-balancing function in the mathematical model (EYCKERMAN et al., 2020; YADAV; NATESHA; GUDDETI, 2019). Furthermore, few works are found in the literature involving FSPP and applications of Intelligent Transportation Systems (ITS) and vehicular networks. In order to explore these gaps, this work considers a mission-critical application study of vehicular networks in Intelligent Transportation Systems. As in (EYCKERMAN et al., 2020; MSEDDI et al., 2019; DONASSOLO et al., 2019b), the motivating scenario consists of detecting and preventing possible vehicle collisions against pedestrians, animals, other vehicles, or objects that pose a risk to the safety of passengers and the driver.

1.1 Objective

This work is an explanatory research that aims to solve the Fog Service Placement Problem to help prevent vehicle collisions on highways in urban cities. The problem is formulated as a constraint satisfaction model for optimizing five objective functions: makespan, energy consumption gap, CPU load-balancing, memory load-balancing and bandwidth load-balancing.

Modeling with multiple objectives is necessary to meet multiple system requirements that are not strictly correlated. These requirements have varied purposes such as application functionality, financial expenditure and environmental impact. From the point of view of application functionality, a mission-critical application, to save lives, requires a time deadline

objective function to provide quality of service when executing the application. From a financial aspect, it is important to have CPU, memory and bandwidth optimization load balancing functions, as monetary expenses are linked to the consumption of infrastructure resources. In relation to environmental impact, aiming for less or no degradation of the environment, an optimization function for energy consumption is required, for example.

In order to conduct experimental evaluations in a structured way, the GQM (Goal-Question-Metric) method was used in this work. Proposed by Basili, Caldiera e Rombach (1994), the GQM methodology was used a fundamental pillar to guide and structure the research process, due to the systematic approach in defining objectives, deriving specific research questions and selecting of appropriate metrics for evaluation. By employing GQM, this work aims to provide a rigorous and well-founded methodology to evaluate and advance knowledge in the field of computer science with credibility. Thus, this study was conducted in an *in silico* experiment with the objective of comparing the performance of the proposed algorithm against the algorithms found in the literature for solving the service placement problem in Fog Computing.

The main objective of this work was to compare the performance of the proposed algorithm, named R3GP (Rotation-Guided Greedy Genetic Particle), against the algorithms found in the literature, for solving the service placement problem in Fog Computing. The finality is to prevent vehicle collisions in urban cities through Intelligent Transportation System networks. In order to achieve the main objective, the following secondary objectives were covered: systematic mapping of the literature to obtain the most used optimization and performance algorithms and metrics; creation and validation of a continuous-time Fog Computing simulator for performance analysis according to M/M/1 queuing theory; and, algorithm validation in static analysis of Fog infrastructure.

The main contributions of this study are:

- Study of application of Intelligent Transportation Systems for preventing vehicle collisions against pedestrians, animals, other vehicles, or objects on urban roads;
- Creation of a metaheuristic algorithm inspired by the main strategies found in the literature for solving the Fog Service Placement Problem efficiently and effectively for the placement of mission-critical and real-time applications;
- Creation of a methodology with planning and executing experiments in a structured way, using the GQM framework;
- Execution of a comparative experiment between the proposed algorithm and the main types of algorithms found in the literature, such as First-Fit Decreasing, Genetic Algorithm and ϵ-Greedy;

- FSPP modeling minimizing the Euclidean distance between five objective functions, for fast problem resolution, taking into account load-balancing, as one of the gaps found in the literature, and energy consumption, in order to implement Green Computing;
- Systematic mapping of the literature with information from algorithms, mathematical models, optimization metrics, tools and devices used in solving the service placement problem in Fog Computing;
- Creation of a simulator of Fog Computing and distributed systems to solve the service placement problem;
- Simulator validation using queuing theories; Validation of the hypotheses of the algorithm comparison experiment with statistical tests;
- Creation of a framework for the implementation of optimization problems and a programming library with implementation of heuristic and metaheuristic algorithms.

1.2 Document Structure

The remainder of this work is structured in two parts. Part I discusses the Theoretical Background of the study, where the basic concepts and related works found in the literature are presented, in addition to the description of the simulator and the proposed algorithm. Part II presents the planning and evaluation of the experiments, consisting of the experimental validation of the simulator and the proposed algorithm, and the experimental evaluation of the case study.

Specifically, Part I is organized as follows: literature review with explanation of methodology, evaluation of results, and presentation of the related works is shown in Chapter 2; the conceptual basis of Fog Computing, Intelligent Transportation System, Kintoun simulator, and the R3GP are explained in Chapter 3; in Chapter 4 are the service placement problem in Fog Computing is modeled and mathematically formulated as an optimization problem.

Finally, Part II is organized as follows: the simulator created in this work is presented in Chapter 5; in Chapter 6, the algorithm is validated in a simple Fog topology experiment with static simulations of mission-critical application placement requests; Chapter 7 presents the case study, the experimental method and the statistical analysis of the results using a more complex Fog topology with dynamic requests of two Intelligent Transportation Systems applications; at last, in Chapter 8, the final considerations are made about the conduction of this study, as well as the conclusion of the main contributions.

Part I

Theoretical Background

2

Literature Review

This chapter aims to present a descriptive quantitative research about service placement in Fog Computing in the context of the Internet of Things. The objective is to evidence the state-of-the-art elements addressed in the literature, such as optimization strategies, mathematical models, optimization metrics, performance metrics, case studies, test-bed environments, and experimentation tools.

The main contributions of this literature review are:

- A systematic mapping of the state-of-the-art about service placement in Fog Computing in the context of the Internet of Things;
- Individual statistical analyses of the following collected variables: optimization strategies, optimization metrics, types of case studies, types of test-bed environments, and experimentation tools;
- Correlated statistical analyses between the variables mentioned above;
- Identification and analysis of the trends about Fog Service Placement Problem;
- Presentation of the related works about Fog Service Placement Problem.

This literature review chapter is organized as follows: Section 2.1 gives detail of the methodology used to make the systematic mapping, such as research questions, search strategy, data extraction variables, and data analysis strategy; Section 2.2 presents the summary of the selected works, the results and discussions about data analyzed, and the answers to the formulated research questions; lastly, Section 2.5 explains the threats to validity of this research and summarizes the main points addressed along the article.

2.1 Methodology

This systematic mapping was built based on the PICOC (Population, Intervention, Comparison, Outcome, Context) method (PETTICREW; ROBERTS, 2006), using research questions and a search string created as guidance. Table 1 presents the terms formulated using PICOC. The following steps describe the conduction method of this work: subsection 2.1.1 shows the formulation of the research questions; 2.1.2 shows the search strategy, which includes the base search string and the search bases used; 2.1.3 presents the variables extracted from the works to answer the research questions; and, finally, the subsection 2.1.4 explains how was performed data analysis for the values collected.

Table 1 – PICOC result.

Population	Service Placement in Fog Computing.
Intervention	Optimization algorithm.
Comparison	–
Outcome	Methodologies, techniques, case studies, testbed scenarios and metrics.
Context	Internet of Things

Source: Author.

2.1.1 Research questions

Based on terms presented in Table 1, the general research question (RQG) formulated is: Which techniques and methodologies solve service placement problems in a Fog Computing environment in the context of the Internet of Things? In order to answer this question, this study needs to answer the following specific questions:

- **RQ1:** Which are the mathematical model types?
- **RQ2:** Which optimization strategies solve the FSPP?
- **RQ3:** Which metrics are works used to optimize the mathematical model?
- **RQ4:** Which application areas do the researchers use in the case studies?
- **RQ5:** Which test-bed environment types do the studies use in experiments?

2.1.2 Search strategy

Based on terms present in Table 1 and formulated research questions, it was refined and created a generic search string to help locate desired articles. Table 2 presents the keywords and their synonyms.

Table 2 – Search string keywords and synonyms.

Keyword	**Synonym**
placement	–
fog computing	–
optimization	optimized, optimizing, optimal
Internet of Things	IoT

Source: Author.

This research used the following research bases to select works in the literature: Scopus, IEEE Xplore Digital Library, Web of Science, Science Direct, and ACM Digital Library. By adequating the base search string for each search base was possible finding studies according to title, abstract, or keywords. Table 3 presents the base search string formulated based on terms in Table 2.

Table 3 – Base string used to search studies on the research bases.

(placement AND "fog computing" AND (optimization OR optimized OR optimizing OR optimal) AND ("Internet of Things" OR IoT)))

Source: Author.

Furthermore, formulated inclusion and exclusion criteria helped to filter relevant studies. The Inclusion Criteria comprise works that address: cloudlet, edge computing, edge-cloud computing, or a mathematical the model of service placement problem. On the other hand, the Exclusion Criteria adopted were: before 2016; do not address a optimization of service placement in Fog Computing; duplicated; unreachable by the account on site of papers; uncompleted; address node placement; address data placement.

2.1.3 Data extraction variables

This study uses the following variables to extract values that help answer the research questions:

- **Mathematical model**: the mathematical model that describes the service placement problem;
- **Service placement strategy**: the strategy used to solve the service placement problem. It can be an algorithm, a framework, or any else;
- **Optimization metric**: the metric used in the mathematical model to be optimized;
- **Application area**: the IoT field of the case study used in the experiments;
- **Test-bed environment**: the environment where researchers executed the experiments. It can be a simulator, the real world, or any other;

As Fog Computing is a novel paradigm, to extract the maximum information presented in the literature, this study accepts works having missing values for any variable (except for the service placement problem variable). Missing values means the variable is either: not applied in work, used but not explained, or applied and explained but not recognized in this study.

2.1.4 Data analysis strategy

In order to answer the research questions, the analysis of the data extracted variables presented in subsection 2.1.3 follows these strategies: **independent analysis**, in which each variable is analyzed separately from the others; **dependent analysis**, in which the correlation of two variables is analyzed using the contingency data evaluation. This second does not account works with missing values for any of the paired variables.

2.2 Results and Discussion

This section presents a summary of the works and results found in the literature. Also, it answers the research questions formulated in 2.1.1, analyzing and explaining the results. Furthermore, it examines the correlation between the variables.

2.2.1 Works summary

This subsection presents the works found after applying the search strategy described in 2.1.2. The first search admitted articles published from 2016/Jan until 2021/Jun. Later, a second search was performed to update the collection until 2022/Jun. As follows, Table 4 outlines the studies discovered. It shows the search string found 379 papers of which 186 were duplicated and 91 were discarded according to the exclusion criteria. Thus, this study accepted a total of 102 scientific research articles.

Table 4 – The selection process of the research works.

Study selection		**Partial**	**Total**
Works found	ACM Digital Library	15	379
	IEEE Digital Library	88	
	Web of Science	124	
	Science Direct	21	
	Scopus	131	
Duplicated works			186
Works excluded by the exclusion criteria			91
Works included by the inclusion criteria			102

Source: Author.

2.2.2 Answers to research questions

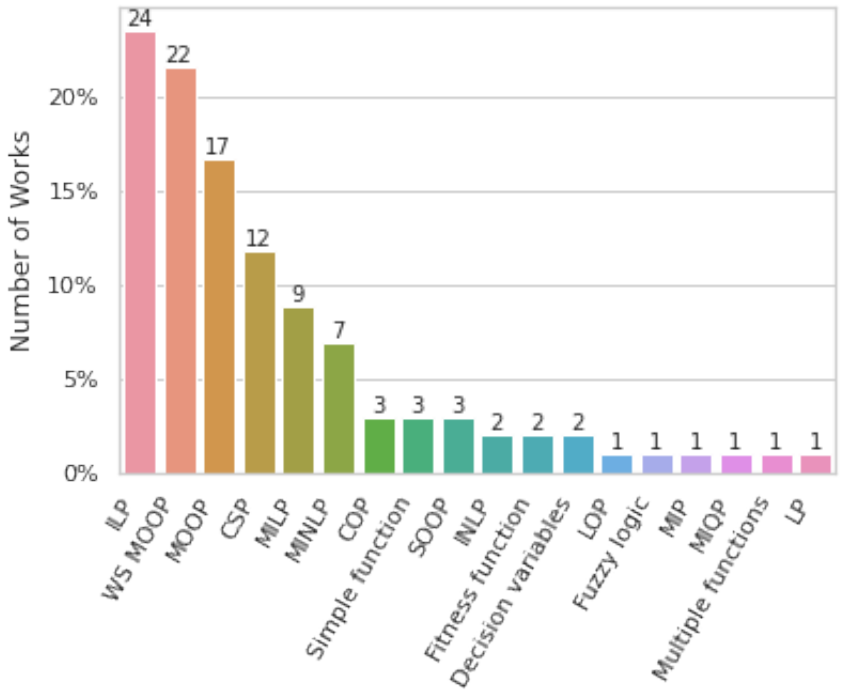
This subsection shows the independent analysis of the values extracted from the selected works. For each research question formulated in Section 2.1.1, an answer is given based on the appropriate variable defined in 2.1.3.

Concerning data analysis of the variables, this study counts duplicated values once to the same research; one research could appear in different values of the same variable. In this way, the paper shows bar plots with absolute values on top of each bar representing the number of studies. At the same time, the vertical axis denotes the relative frequency of that value about the number of total works.

RQ1 – Which are the mathematical model types?

Using the *Mathematical model* variable, Fig. 1 shows how the authors represent the service placement problem in the Fog Computing. Notice that was opted to do not subdivide the types of mathematical representation because researchers do not distinguish your models by categories.

Figure 1 – Mathematical models.



Source: Author.

In the Fig. 1 Integer Linear Programming (ILP), Weighted-Sum Multi-objective Optimization Problem (WS MOOP), and Multi-objective Optimization Problem (MOOP) are the 3 mathematical models most used in the literature, appearing in 24, 22, and 17 works, respectively. Most works like (SAHOO, 2021; VIJOUYEH et al., 2020; NATH et al., 2019)

model the problem directly as an ILP, but some other authors model the problem as a MINLP (Mixed-Integer Non-Linear Programming) and relaxes the constraints using relaxed variables to reduce it to an ILP, as in (FATICANTI et al., 2020; KIM; CHUNG, 2018b). Works such as (SALIMIAN; GHOBAEI-ARANI; SHAHIDINEJAD, 2022; FARZIN et al., 2022; SALIMIAN; GHOBAEI-ARANI; SHAHIDINEJAD, 2021; YAO; ANSARI, 2019), represent the FSPP as ILP and WS MOOP, aggregating multiple variables into a single optimization function. Others, such as (LIU et al., 2022; ALMURSHED; RANA; CHARD, 2022; AL-TARAWNEH, 2022; HUANG et al., 2020; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020; HUSSAIN et al., 2020; MORKEVICIUS et al., 2021; DONASSOLO et al., 2019b), treat the variables in multiple optimization functions and figure out the solutions in a Pareto front result.

Problems designed as ILP, MILP or MINLP are NP-hard complex to solve, i. e. finding the best solution demands spending much more time than usual for traditional computers. Avoiding this requires strategies that solve the Fog Service Placement Problem approximating the results in polynomial time.

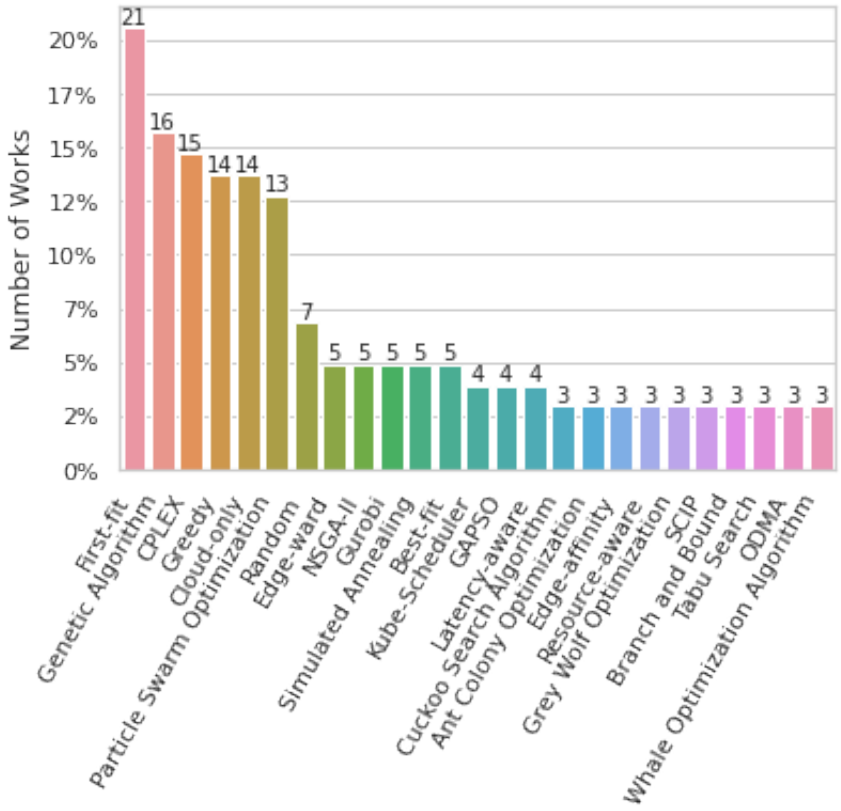
RQ2 – Which optimization strategies solve the service placement problem in fog computing?

The *Placement strategy* variable defined in the Methodology supports answering to this question. Fig. 2 exhibits the top 25 values. The other 140 strategies were omitted due to the size of the figure. In this analysis the counting procedure considers derived strategies equal to the original ones, for example, GA-based algorithms counts as GA (Genetic Algorithm). On the other hand, it was assumed hybridization as a distinct strategy, such as GAPSO (Genetic Algorithm + Particle Swarm Optimization).

The first position of the First-Fit reflect its frequent usage as control group in the experiments (TAVOUSI; AZIZI; GHADERZADEH, 2022; PATRO et al., 2021; NEZAMI et al., 2021; RAHBARI; NICKRAY, 2020; GILL; SINGH, 2020). The high usage of the CPLEX reflects it as a control group too, as in (HUSSAIN et al., 2020; MSEDDI et al., 2019; BOURHIM; ELBIAZE; DIEYE, 2019), but also shows that some authors focused in the mathematical modeling and uses the solver just to find the solutions, as in (YOSUF et al., 2021; ALQAHTANI et al., 2021b; SANTOS et al., 2017). Its popularity for solving placement and scheduling problems in the cloud and its combinatorial-like feature by means of crossover attract researchers to develop GA-inspired strategies. Its broad diffusion comes with fine-tuning, modifications, and also hybridization with other meta-heuristics, i. e. GAPSO (YADAV; TRIPATHI; SHARMA, 2022; NATESHA; GUDDETI, 2022; NATESHA; GUDDETI, 2021; YADAV; NATESHA; GUDDETI, 2019), that visioning to improve the execution time of the GA to fit with the requirements of fog computing such as the service latency.

The top 3 of the Fig. 2 represent the three major types of strategies used to solve the Fog Service Placement Problem: solver, heuristic, and meta-heuristic. But, some authors had applied Machine Learning techniques to figure out the placement plan, as in (GOUDARZI;

Figure 2 – Service placement strategies.



Source: Author.

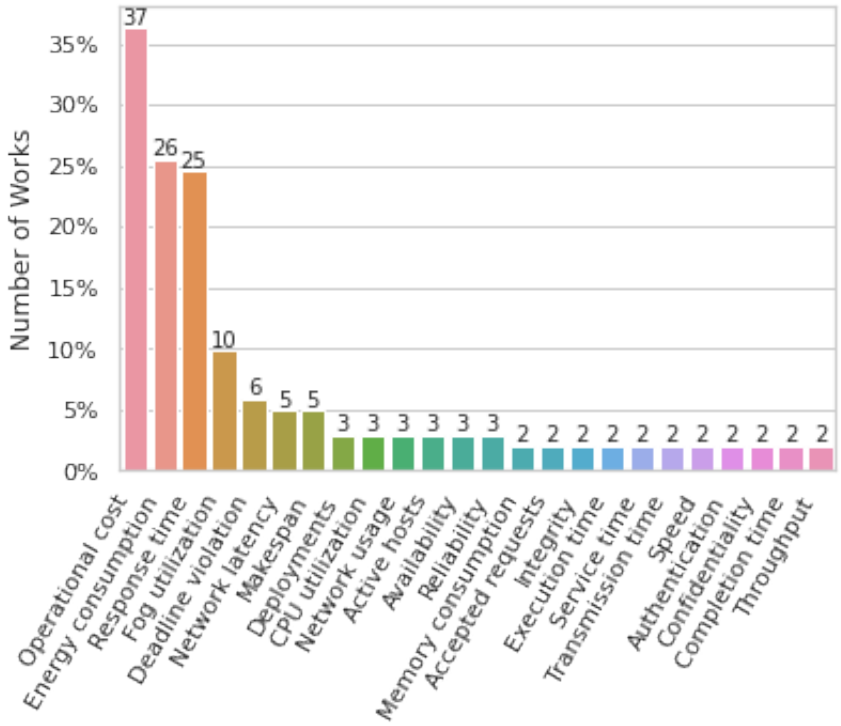
PALANISWAMI; BUYYA, 2021) that performs Reinforcement Learning variations to optimize the WS MOOP concerns to the energy consumption and execution time variables. In (MANIHAR; PATEL; AGRAWAL, 2018) they use a Feed Forward Neural Network to place applications of different domains, such as Hospital, Smart Home, and Smart Surveillance. In (RAGHAVENDRA; CHAWLA; NARASIMHULU, 2021) they use clustering-inspiration algorithms, such as K-means, to optimize energy consumption and response time. And, (PATRO et al., 2021; RAHBARI; NICKRAY, 2020) that propose Decision Tree-based algorithms to decrease the response time and the energy consumption using decision variables.

Executing placement strategies requires objective functions that are part of the mathematical model. These functions are built based on variables that need to be optimized. Choosing the best optimization metrics is essential to model the problem and determining the best strategy.

RQ3 – Which metrics do works use to optimize the mathematical model?

Understanding the metrics used to model the service placement problem in Fog Computing is crucial to apply the best strategy. In such a way, the *Optimization metric* variable supports getting this information from the works. This study grouped similar optimization metrics meanings and presented them in Fig. 3. It exhibits the top 24 values, while the other 44 values were omitted.

Figure 3 – Optimization metrics.



Source: Author.

Fig. 3 shows a significant difference between the first three and the remaining optimization metrics. Overall, the operational cost, energy consumption, and response time metrics have at least ten times more occurrences than 80% of the metrics. In addition, this graph presents fog utilization, a customized metric for the area, appearing in about 10% of the works.

The operational cost optimization metric represents any unit cost related to the usage of the resources along the IoT, fog and cloud devices. In (ALI et al., 2022; BARANWAL; VIDYARTHI, 2022; BROGI et al., 2019; YOUSEFPOUR et al., 2019; REZAZADEH; RAHBARI; NICKRAY, 2018; ARKIAN; DIYANAT; POURKHALILI, 2017), it appears in the format of monetary value. In (ALI et al., 2022; BARANWAL; VIDYARTHI, 2022; BROGI et al., 2019; YOUSEFPOUR et al., 2019; REZAZADEH; RAHBARI; NICKRAY, 2018; ARKIAN; DIYANAT; POURKHALILI, 2017), as the sum of network communication and processing costs. This wide usage reflects a worry in turning Fog Computing monetary cost-friendly, with regards to Cloud Computing, in order to make it attractive to the market and industry (FARZIN et al., 2022; HAPP; BAYHAN; HANDZISKI, 2021).

As important as the first, energy consumption gained much more attention over the last years in the industry (DJEMAI et al., 2021; ALQAHTANI et al., 2021a; YOSUF et al., 2020; KIM; CHUNG, 2018a). Fig. 4 shows the usage tendency of this optimization metric since 2018 and a half of 2022. It is an effort that vision Green Computing (DASH; AHMAD; IQBAL, 2021) by decreasing the rate of carbon dioxide (CO_2) emitted to the atmosphere by IoT devices, as

presented in (HUSSAIN; BEG et al., 2019; YOSUF et al., 2020; YADAV; NATESHA; GUDDETI, 2019).

Response time, also known as service latency or service delay (AYOUBI; RAMEZANPOUR; KHORSAND, 2021; REDDY et al., 2020; YOUSEFPOUR et al., 2019), is the first response that IoT devices receive after application request (CHEKIRED; KHOUKHI, 2018). It is one of the main metrics in Fog Computing, once end devices have several challenges with high communication with cloud servers (AMARASINGHE et al., 2018). Due this drawbacks of Cloud Computing and the advantage of the location of the Fog devices closest to the end devices, many studies (HERRERA et al., 2021; KARAMOOZIAN; HAFID; ABOULHAMID, 2019; CHEKIRED; KHOUKHI, 2018; AMARASINGHE et al., 2018; NATESHA; GUDDETI, 2018) propose optimize the response time aiming decrease much more the communication latency to returns the result of the application processing.

Fig. 3 also presents another evident metric, fog utilization. It means the percentage of services placed in fog computing, i.e., the usage of the resources of the fog devices in regards to the cloud resources (LIU et al., 2022; SALIMIAN; GHOBAEI-ARANI; SHAHIDINEJAD, 2022). The adoption of fog utilization grew over the years, with 40% of studies (LIU et al., 2022; SALIMIAN; GHOBAEI-ARANI; SHAHIDINEJAD, 2022; PALLEWATTA; KOSTAKOS; BUYYA, 2022; ZHAO; ZOU; ZADEH, 2022) concentrated in half of 2022 and the remaining distributed from 2017 to 2021 (AYOUBI; RAMEZANPOUR; KHORSAND, 2021; GODINHO; CURADO; PAQUETE, 2019; TRAN et al., 2019; MAHMUD; RAMAMOHANARAO; BUYYA, 2018; SKARLAT et al., 2017b; MINH et al., 2017).

Some other metrics can be understood as complements to the others. For example, deployments, active hosts, CPU utilization, and memory consumption, as used in (HOSSEINPOUR et al., 2021; MORKEVICIUS et al., 2021), reflect the usage of the device resources represented by fog utilization metric. Execution time, service time, makespan, and completion time are correlated to the response time, as in (GILL; SINGH, 2020). Furthermore, other metrics are related to the privacy, security and quality of the service, such as deadline violation, availability, reliability, integrity, authentication, and confidentiality.

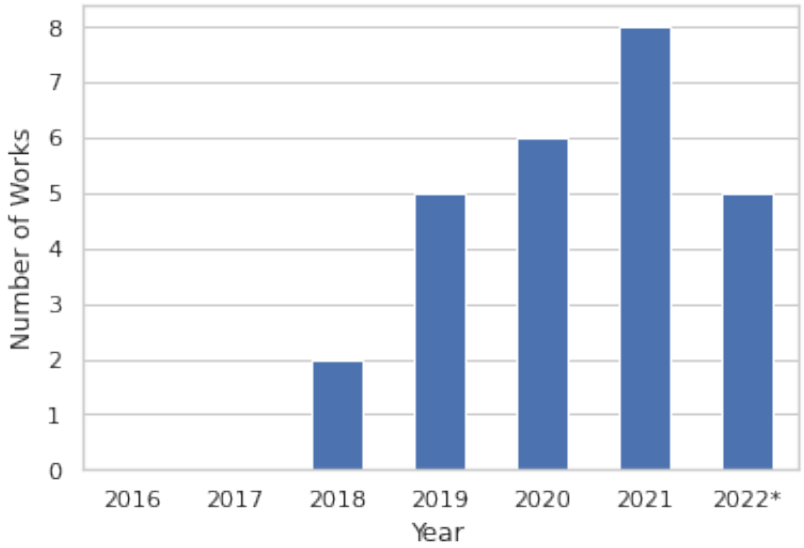
Despite the optimization metrics, discovering the best placement strategy also requires knowing the case studies, as can be seen in the next question.

RQ4 – Which application areas do the researchers use in the case studies?

Understanding how to model the Fog Service Placement Problem requires knowing which field involves it. Thus, the *Application area* variable helps to answer this question. Fig. 5 presents all the 17 application areas explored in the research experiments.

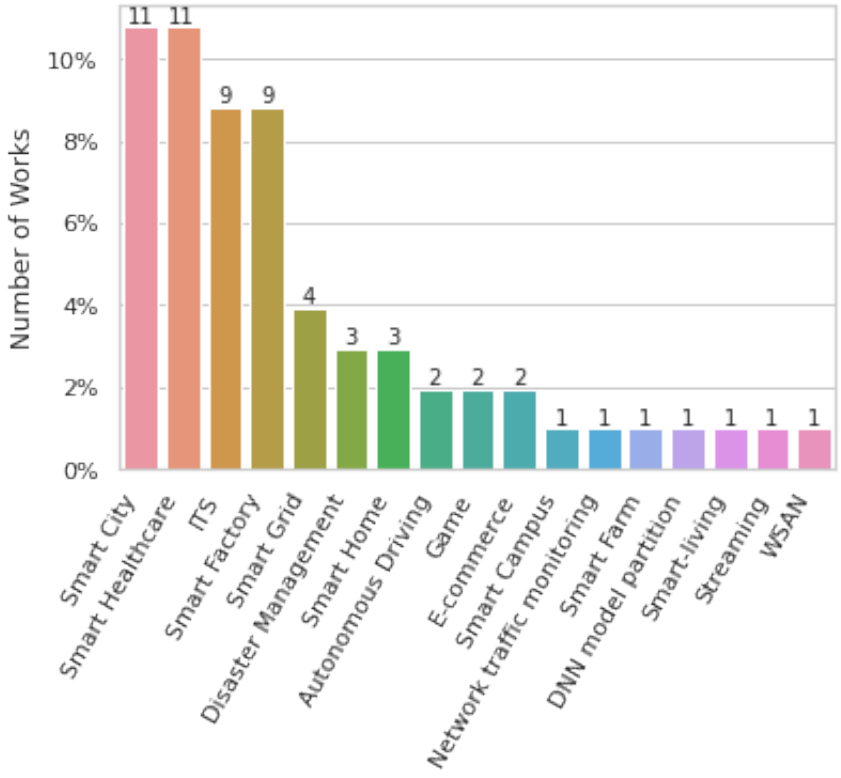
Overall, Fig. 5 show there are four predominant areas of study in FSPP: Smart City and Smart Healthcare tied with 11 works and Intelligent Transportation Systems (ITS) and Smart

Figure 4 – Usage tendency of the energy consumption optimization metric.



Source: Author.

Figure 5 – Application areas.



Source: Author.

Factory tied with 9 works.

The Smart City adoption, as in (SHAIK; BASKIYAR, 2022; DJEMAI et al., 2021; NIKOUI et al., 2020; VIJOUYEH et al., 2020; SANTOS et al., 2020b; SANTOS et al., 2017), could be explained by its distributed-like environment characteristic, covering a wide range area in a region, an aspect of the fog computing paradigm. Another explanation is Smart City definition encapsulates other kinds of Smart fields, such as Smart Vehicles, Smart Traffic, and Smart Water, which could be a not well-defined case study by authors.

The Smart Healthcare study cases, as used in (ALI et al., 2022; PALLEWATTA; KOSTAKOS; BUYYA, 2022; ZHAO; ZOU; ZADEH, 2022; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020; NIKOUI et al., 2020; APAT et al., 2020; MEHRAN; KIMOVSKI; PRODAN, 2019; MOURADIAN et al., 2019; REZAZADEH; REZAEI; NICKRAY, 2019; MAHMUD; RAMAMOHANARAO; BUYYA, 2018; REZAZADEH; RAHBARI; NICKRAY, 2018), explore applications focusing in provide better quality treatment to the patients health. For example, (ALI et al., 2022) implement a ILP model in a Fog-assisted infrastructure to provide a secure and reliable deployment for IoT applications in healthcare (IoTH). In (MARTIN; KANDASAMY; CHANDRASEKARAN, 2020), they deploy an wearable electrocardiogram (ECG) monitoring sensor application composed of micro-services that monitor the ECG , analyze the data to detect anomalies, and take actions in order to keep the stability of the patient health if necessary.

The relevance of Intelligent Transportation System (ITS) (SHARMA; BUTLER; JENNINGS, 2022; NTUMBA; GEORGANTAS; CHRISTOPHIDES, 2021; HAPP; BAYHAN; HANDZISKI, 2021; EYCKERMAN et al., 2020; MSEDDI et al., 2019; TRAN et al., 2019; DONASSOLO et al., 2019b; XIA et al., 2018b; KHARE et al., 2018) is related to the traffic congestion in the urban locations of the cities. In (TRAN et al., 2019), for example, the authors built a real-world service deployment environment by using analyzing traffic flow and traffic light control ITS applications.

Smart Factory, or Smart Manufacturing, is the field of the Industry 4.0 and Industrial IoT (IIoT) (NATESHA; GUDDETI, 2021) that is composed by intelligent IoT devices and systems that monitor and control a set of industry machines (GODINHO; CURADO; PAQUETE, 2019). Authors of (GHOBAEI-ARANI; SHAHIDINEJAD, 2022; MAHMUD et al., 2019b), for example, propose a placement algorithms to deploy industry 4.0-oriented applications (I4OAs), such as image processing for robot navigation assistance in a manufacturing.

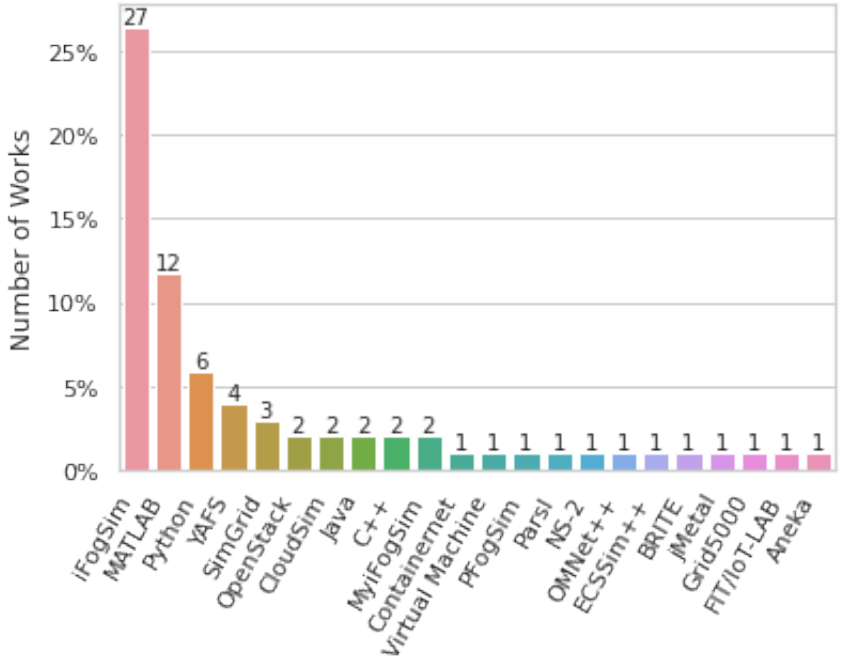
RQ5 – Which test-bed environment types do the studies use in experiments?

The *Test-bed environment* variable identifies where researchers executed your experiments to help answer this question. It was found that 82 works use simulator, 22 use real-world, and only one use emulator tools to create the test-bed environment. An explanation for the large

usage of simulator tools is that Fog Computing is an emerging paradigm. Also, its distributed characteristic requires many devices, which makes it hard to reproduce it in several real-world scientific research experiments.

Fig. 6 shows all simulator tools adopted in the experiments in the literature. The iFogSim simulator has the most popularity, presenting in nearly 27% of the studies, such as (ALGHAMDI; ALZAHRANI; THAYANANTHAN, 2021; REDDY et al., 2020; MAHMUD et al., 2019a; MAHMUD; RAMAMOHANARAO; BUYYA, 2019; DJEMAI et al., 2019; NATESHA; GUDDETI, 2018). The MATLAB and Python programming languages means the authors, as in (MAITI et al., 2022; BARANWAL; YADAV; VIDYARTHI, 2020; MOALLEMI; BOZORGCHENANI; TARCHI, 2019), use these tools to developed from scratch they simulated fog environment required for their experiments. The YAFS (Yet Another Fog Simulator) has gained popularity, used in (SHARMA; BUTLER; JENNINGS, 2022; SAMANI; SAURABH; PRODAN, 2021; APAT et al., 2020; LERA; GUERRERO; JUIZ, 2018), emerging as an alternative to iFogSim, focused on Fog Computing and IoT environments. At last, (EYCKERMAN et al., 2020; XIA et al., 2018b; XIA et al., 2018a) use the SimGrid, a generic simulator that runs distributed-like computing environments, such as the Fog paradigm.

Figure 6 – Simulation test-bed tools.

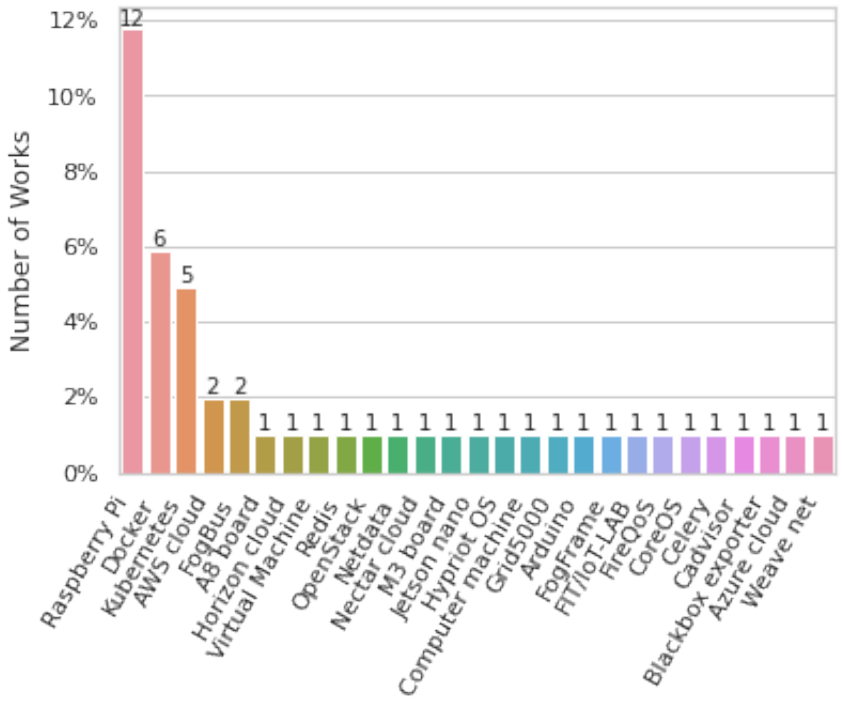


Source: Author.

In regards to real-world tools, Fig. 7 shows the most frequents in the literature. At first, the Raspberry Pi (LLORENS-CARRODEGUAS et al., 2021; SKARLAT; SCHULTE, 2021; KHOSROABADI; FOTOUHI-GHAZVINI; FOTOUHI, 2021; SANTOS et al., 2020a; SAMI; MOURAD, 2020; VENTICINQUE; AMATO, 2019; SKARLAT et al., 2018) comes up as a hardware infrastructure tool. Docker (LLORENS-CARRODEGUAS et al., 2021; MEHRAN; KIMOVSKI;

PRODAN, 2019; DONASSOLO et al., 2019a; SKARLAT et al., 2018) comes as a container engine for encapsulation of application services. Kubernetes (LLORENS-CARRODEGUAS et al., 2021; SAMI; MOURAD, 2020; SANTOS et al., 2020b; SANTOS et al., 2020a; FATICANTI et al., 2019), appears as an orchestrator of the services in containers. At last, FogBus (MAHMUD et al., 2019b; MAHMUD; RAMAMOHANARAO; BUYYA, 2019) emerges as a framework with components to build specifically Fog Computing components and environments.

Figure 7 – Real-world test-bed tools.



Source: Author.

2.2.3 Correlations

This subsection presents a dependent analysis, such as shown in the Data analysis strategy in the Methodology, helping as an additional analysis to reinforce some factors of the variables analyzed in 2.2.2. Thus, for better knowledge of the scientific research directions about optimization of service placement in fog computing, this work analysis the associations between the variables computing the correlations using the contingency table approach. Then, each examination shows a ranking table with the top 3 most frequent combinations.

Placement strategy and Mathematical model correlation

This association reveals which placement strategy the works use to solve the mathematical models of the Fog Service Placement Problem. Table 5 shows the 3 most recurring associations.

It reinforce the variety of the most used values as analyzed individually and shown in Fig. 1 and Fig. 2.

Table 5 – Top 3 most used combinations between algorithm and mathematical model.

Rank	**Placement Strategy**	**Mathematical Model**	f_{abs}
1º	CPLEX	ILP	10
2º	GA	WS MOOP	9
3º	Greedy	ILP	6

Source: Author.

Table 5 shows the CPLEX solver is the strategy most used to solve ILP models, as in (SAHOO, 2021; ALQAHTANI et al., 2021b; HUSSAIN et al., 2020; YAO; ANSARI, 2019; MSEDDI et al., 2019; BOURHIM; ELBIAZE; DIEYE, 2019; MOURADIAN et al., 2019; HIESSL et al., 2019; MOURADIAN; KIANPISHEH; GLITHO, 2018; SKARLAT et al., 2017b; SANTOS et al., 2017). Most of the studies use this combination as a control group in their experiments, which supports the analysis of the placement strategy in 2.2.2.

Mathematical model and optimization metric correlation

The objective of this analysis is to understand which metrics the studies use to create the optimization function of an FSPP mathematical model. Table 6 shows the three most used associations between the mathematical model and optimization metric variables.

Table 6 – Top 3 most used combinations between mathematical model and optimization metric.

Rank	**Mathematical Model**	**Optimization Metric**	f_{abs}
1º	WS MOOP	Operational cost	14
2º	ILP	Operational cost	9
	MOOP	Operational cost	
3º	WS MOOP	Energy consumption	8

Source: Author.

Overall, Table 6 reveals a combination of the three most used optimization metrics with the two most used mathematical models. This evidence also shows a strongest correlation with the operational cost. Around 63% of the works that model the FSPP as a WS MOOP use the operational cost as an optimization metric, such as in (YADAV; TRIPATHI; SHARMA, 2022; SALIMIAN; GHOBAEI-ARANI; SHAHIDINEJAD, 2022; FARZIN et al., 2022; SALIMIAN; GHOBAEI-ARANI; SHAHIDINEJAD, 2021; NEZAMI et al., 2021; NIKOUI et al., 2020; YAO; ANSARI, 2019; MOURADIAN et al., 2019; BROGI et al., 2019). About 53% of the works that model the FSPP as a MOOP use the operational cost as an optimization metric, such as in (KOCHOVSKI et al., 2022; LIU et al., 2022; AYOUBI; RAMEZANPOUR; KHORSAND, 2021; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020; HUANG et al., 2020; MEHRAN;

KIMOVSKI; PRODAN, 2019). At last, more than 30% of the studies that address energy consumption model the FSPP as a WS MOOP, as in (GHOBAEI-ARANI; SHAHIDINEJAD, 2022; ZHAO; ZOU; ZADEH, 2022; DJEMAI et al., 2021; TULI et al., 2021; DJEMAI et al., 2020; KAYAL; LIEBEHERR, 2019; YADAV; NATESHA; GUDDETI, 2019).

Optimization metric and application area correlation

This correlation helps to understand which metrics the works use to optimize the mathematical model of the IoT application areas. Table 7 shows the most used combinations between the optimization metric and application area.

Table 7 – Top 3 most used combinations between optimization metric and application area.

Rank	Optimization Metric	Application Area	f_{abs}
1º	Operational cost	Smart Healthcare	6
2º	Energy consumption	Smart Factory	5
3º	Operational cost	Smart City	4
	Response time	Smart Factory	

Source: Author.

Overall, Table 7 reveals a combination of the most used optimization metrics with the most applied application areas. The Smart Healthcare, for example, has a high correlation with the operational cost, once that this optimization metric appears in about 54% of the use cases in that field, as in (ZHAO; ZOU; ZADEH, 2022; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020; NIKOUI et al., 2020; MEHRAN; KIMOVSKI; PRODAN, 2019). High correlation also appears in the analysis of the combination of Smart Factory and energy consumption, which this optimization metric happen in about 55% of the use cases in that field, as in (GHOBAEI-ARANI; SHAHIDINEJAD, 2022; NATESHA; GUDDETI, 2022; TULI et al., 2021; NATESHA; GUDDETI, 2021).

2.3 Trending topics

As discussed on the analysis of 2.2.2, using the data extraction variables presented in 2.1.3, the results reveal a growth tendency in the usage of the energy consumption optimization metric. As a complement, this section discuss other three trending topics exposed in the works analyzed.

The first topic is the mobility of the data sources. Overall, studies in the literature solve FSPP using a static Fog environment. The mobility of the IoT objects brings several challenges for the management of the services and resources in order to preserve the Quality of Service (QoS) on a fog computing (DJEMAI et al., 2021). Works that address this point need to provide solutions to service migration issues such as start time, transfer path, and extra

resource consumption (DJEMAI et al., 2020). In this way, several works, such as (PALLEWATTA; KOSTAKOS; BUYYA, 2022; NEZAMI et al., 2021; GOUDARZI; PALANISWAMI; BUYYA, 2021; KHOSROABADI; FOTOUHI-GHAZVINI; FOTOUHI, 2021), pointed the treatment of mobility of devices as future work.

Another topic is the serverless computing (PALLEWATTA; KOSTAKOS; BUYYA, 2022; TULI et al., 2021; GOUDARZI; PALANISWAMI; BUYYA, 2021). This is a mechanism that gained popularity with the cloud computing paradigm offering pay-per-use service computation, memory, storage, and auto-scaling functionality in order to provide better use of the resources (GHOBAEI-ARANI; SHAHIDINEJAD, 2022).

At last, another interesting topic, found in works of 2022 only (GHOBAEI-ARANI; SHAHIDINEJAD, 2022; ZHAO; ZOU; ZADEH, 2022; LIU et al., 2022; SALIMIAN; GHOBAEI-ARANI; SHAHIDINEJAD, 2022), is the usage of the Monitoring, Analysis, Decision-making, and Execution phases shared with a knowledge-base (MADE-k) (JACOB et al., 2004). It is an autonomous framework that consists in managing the IoT services and fog and cloud resources in the monitoring phase; prioritizing the services based on their deadline in the analysis phase; generating a placement plan of the services in the fog and cloud resources in the decision-making phase; and deploying the services according the placement plan in the execution phase (LIU et al., 2022; SALIMIAN; GHOBAEI-ARANI; SHAHIDINEJAD, 2022).

2.4 Related Works

This section presents the main related works, found in the literature, which were the basis for the construction of this study. The following works were selected according to the algorithms used, the optimization metrics, the performance metrics and the case study.

The work of Natesha e Guddeti (2021) aimed to analyze the proposed algorithm, called Elitism-based Genetic Algorithm (EGA), in solving the container placement problem. The problem was modeled as multi-objective optimization using weighted-sum. The performance of the EGA was evaluated against the DEBTS (Delay Energy Balanced Task Scheduling), DMS (Double-Matching Strategy), First-Fit, Branch and Bound, and GAPSO algorithms. The treatments were analyzed in relation to the metrics service time, service cost, energy consumption, and CPU utilization. The experiment conducted was a case study of Smart Manufacturing in the context of Fog Computing in a real infrastructure with Docker container and Raspberry Pi devices.

In Martin, Kandasamy e Chandrasekaran (2020), the authors modeled service placement in the form of a multi-objective optimization problem. The objective was to analyze the performance of the proposed algorithm, called CREW (an Eagle strategy-based multi-objective whale optimization), against NSGA-II, MOWOA (Multi-Objective Whale Optimization), and FFD-latency (First-Fit Decreasing-latency). The algorithms were compared in terms of reliability,

monetary cost, and response time. The case study used was based on e-commerce and Smart Healthcare applications. The experiments were conducted in a simulated environment using iFogSim.

In Ayoubi, Ramezanpour e Khorsand (2021), the IoT service placement problem was implemented as a multi-objective optimization model using Pareto-bound. The proposal was to analyze the performance of the SPEAII, MOPSO, and NSGA-II algorithms in relation to the metrics service latency, fog utilization, monetary cost, time violation, and count violation. The case study experiment was also conducted via simulation in iFogSim.

The work of Yadav, Natesha e Guddeti (2019) aimed to analyze the proposed hybrid algorithm, called GA-PSO (Genetic Algorithm + Particle Swarm Optimization-based), in solving the service placement problem. The problem was modeled as multi-objective optimization using weighted-sum. The performance of the GA-PSO was evaluated against the GA (Genetic Algorithm) and PSO (Particle Swarm Optimization) algorithms. The treatments were analyzed in relation to the metrics makespan, energy consumption, and fitness value. The experiments were conducted in a simulated Fog Computing infrastructure developed in C++.

In Nath et al. (2019), the authors formulated the microservice placement problem as a mathematical model of the Integer Linear Programming type. The objective was to analyze the performance of the algorithm proposed by the authors, a Bayesian Optimization-based reinforcement learning, against Best-Fit, First-Fit, and strategies called Foglets and mobility-based. The algorithms were compared in terms of response time, memory utilization, CPU utilization, bandwidth, and number of migrations of the microservices. The experiments were conducted in a case study of Fog Computing simulated in iFogSim and in a real infrastructure with Docker container and Raspberry Pi devices.

In Djemai et al. (2019) the authors aimed to minimize the energy consumption and the application deadline violation of the placement of IoT services of real-time, streaming and mission-critical applications in Fog Computing infrastructure. They proposed the optimization of the service placement problem in Fog computing as a Constraint Satisfaction Problem. To optimize, they proposed the Discrete Particles Swarm Optimization algorithm compared with the Binary Particle Swarm optimization, Dicothomous Module Mapping algorithms, and placement strategies called IoTFogOnly placement, IoTCloud placement and FogCloud placement. The study was conducted using the iFogSim simulator. In the simulation, the authors considered only five Fog Computing devices and four IoT devices making service requests. In addition, despite using different types of IoT applications, the number of services for each application was only three.

In Pallewatta, Kostakos e Buyya (2022) the researchers' objective was to minimize the metrics makespan violation, budget violation, computation resource usage and network resource usage in the execution of microservice-based application placement in Fog Computing. For this, the authors modeled the problem as a Lexicographic Multi-objective Combinatorial Optimization

Problem. The authors proposed a placement algorithm based on the Set Comprehensive Learning Particle Swarm Optimization (S-CLPSO) and evaluated the performance against algorithms from the literature in an iFogSim simulation. The motivating study scenario was based on a Smart Healthcare application for patient monitoring and a Smart City application for parking occupancy detection. Both applications have only three microservices.

Regarding the applications of Intelligent Transportation Systems used for placement in Fog Computing, the works found in the literature are related to accident prevention. In Donassolo et al. (2019b) the authors study a Smart Light Traffic Application scenario. The problem is formulated as an Integer Linear Programming (ILP) and solved with proposed algorithms based on GRASP (Greedy Randomized Adaptive Search Procedures). In Eyckerman et al. (2020) DRACO (Distributed Reconnaissance Ant Colony Optimization) is proposed for multi-objective optimization of energy consumption, network usage and network latency metrics. The case study described by the researchers was the monitoring of infrastructure cameras to detect dangerous situations at road crossings. The application uses the combination of vehicle data with information from road infrastructure sensors as data sources and returns an alert to the driver's screen in case of a safety risk. In Mseddi et al. (2019) the researchers implement the simulation of a Smart Roadside System in a Fog infrastructure with real data from the circulation of vehicles and pedestrians in the cities of Manhattan and Rome. In the work, the solution of the placement problem was proposed considering the dynamicity of the Fog Computing platform. When running the simulation, the authors consider a collision avoidance application that detects and tracks vehicles and pedestrians so that it can prevent accidents, and another license plate recognition application in real time.

Like the works Almurshed, Rana e Chard (2022), Donassolo et al. (2019b), Mouradian et al. (2019), this study proposes the modeling of the Fog Service Placement Problem as a Constraint Satisfaction Problem. Unlike studies found in the literature, which optimize a maximum of four objective functions in weighted-sum form, this work minimizes five objective functions using the euclidean distance strategy. Like the works Nezami et al. (2021), Santos et al. (2020a), this one proposes the optimization of load-balancing, however, in this work the load-balancing of CPU, memory and bandwidth utilization is considered. In addition to these, as well as Ghobaei-Arani e Shahidinejad (2022), Yosuf et al. (2020), Hussain et al. (2020), Mehran, Kimovski e Prodan (2019) the application service response time, makespan, and energy consumption metrics are optimized. Of the latter, different from Yosuf et al. (2020), in this work, instead of considering the sum of the total energy consumption of the devices, it is considered the sum of the difference in the energy consumption of the devices added to the system with the placement of services. In this way, it is possible to make a comparison of the additional energy consumption for different numbers of nodes in the Fog.

As in the works of Djemai et al. (2019) and Pallewatta, Kostakos e Buyya (2022), this one proposes the use of real-time and mission-critical IoT applications, however, differently, in

this work an application with 10 services is used, in a topology with 10, 25, 50 and 100 Fog nodes and 18 IoT nodes. Also, in this work Fog nodes with more restricted capacities are used. Also, unlike the authors, who use robust devices, in this one Raspberry Pi models are used as Fog nodes. As a case study, this work proposes to use two applications of Intelligent Transportation Systems based on the work of Eyckerman et al. (2020), Mseddi et al. (2019), Donassolo et al. (2019b), one for detection of heavy vehicles on the road to control intelligent traffic lights and another to avoid collisions between vehicles, pedestrians and objects in transit.

In addition, this work proposes a Fog Computing and distributed systems simulator based on queuing theory, called Kintoun. Finally, the R3GP (Rotation-Guided Greedy Genetic Particle) algorithm is proposed, inspired by the strategies used in the Grover, Guided Local Search (GLS), Greedy, Genetic Algorithm and Particle Swarm Optimization algorithms.

2.5 Considerations

The number of IoT devices connected to the Internet is growing rapidly and bringing IoT applications that require Quality of Service metrics that IoT devices cannot handle themselves because of their hardware constraints. Unfortunately, the Cloud Computing paradigm despite useful and widely adopted in this context, also demands some concerns to execute these applications due to its disadvantages, such as high response time and network congestion tendency. Thus, the Fog Computing paradigm emerged to cover these drawbacks scenarios.

This investigation proposed a systematic mapping to reveals which models, metrics, tools, and algorithms are addressed in the state-of-the-art of the literature. Overall, studies present an Integer Linear Programming model of the Fog Service Placement Problem and optimize the operational cost, response time, and energy consumption metrics.

In order to solve the FSPP mathematical model, the First-Fit, CPLEX, and Cloud-only strategies are used as control groups in the experiments. Overall, new solutions are inspired in the Genetic Algorithm and used as treatment groups. At last, the researchers use the iFogSim simulator to create virtual Fog infrastructure environments to execute their experiments, such as Smart Cities, Smart Healthcare, Intelligent Transportation Systems, or Smart Factories.

In regards to the treatments to validity, this study considers the search string a threat to the construction validity because it probably did not reach some other works. The threat to internal validity was the aggregation of the values to analyze the variables. The threat to external validity was the usage of five research bases. Collecting works from more research bases helps to mitigate this bias. Concerning the threat to conclusion validity, the number of works selected from the literature and the statistical evaluation using correlations mitigate the bias in the conclusion of the answer to questions.

As future works it is suggested a systematic review with meta-analysis to identify which

algorithms have better performance regards to the operational cost, energy consumption, and response time metrics. In addition, it is suggested to investigate which technologies could be used to enable serverless computing in a fog computing landscape. Finally, a deep investigation of the state-of-the-art of the literature to reveal the mathematical parameters, technologies, and issues that are addressed when modeling mobility in fog computing environments.

3
Conceptual Base

The objective of this chapter is to present the conceptual basis for Fog Computing and Intelligent Transportation System. Furthermore, the basic concepts of the Kintoun simulator and the R3GP algorithm, proposed in this work to solve the service placement problem in Fog infrastructure, are described.

3.1 Fog Computing

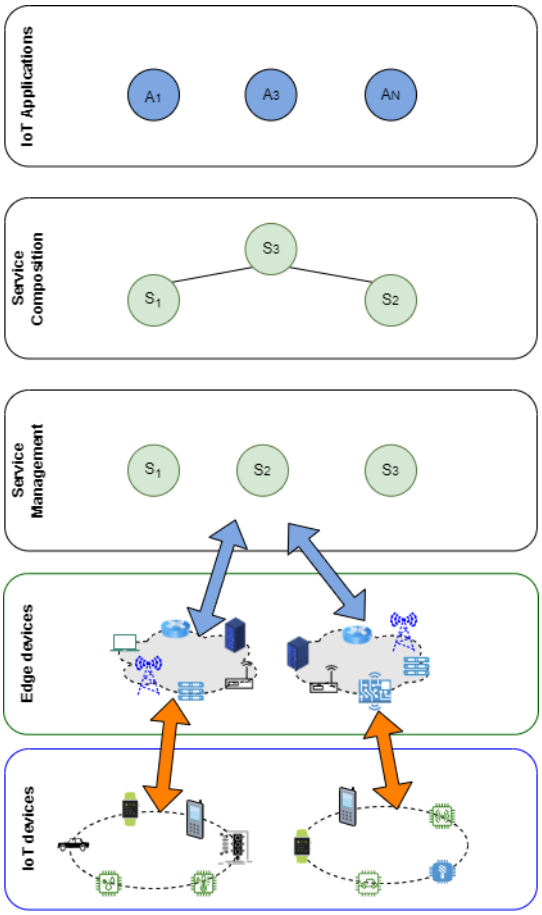
Fog computing is a paradigm introduced by Cisco (BONOMI et al., 2012) that complements the cloud computing using edge devices to mitigate the problems of high latency and network usage, by bringing the processing power and intelligence close to IoT devices (DESIKAN; KOTAGI; MURTHY, 2020).

In order to standardize the architecture to handle IoT applications, one of the most known choices is the Service-Oriented Architecture (SOA). This is motivated by your popularity due to similarity of IoT application service and applications designed for SOA. This architecture includes the following layers: end devices (IoT devices), end devices abstraction (edge devices), service management, service composition, and IoT applications (SANTOS et al., 2017), as can be seen in Figure 8.

In the literature, related to IoT and fog computing, many works propose a multi-tier architecture for this paradigm. The most common is the 3-tier architecture that is composed by: device/group-of-device Tier, that includes the IoT devices; regional tier, that consist of the fog computing elements; and global tier, containing the cloud computing services (MANOGARAN; RAWAL, 2021; SANTOS et al., 2020b; TRAN et al., 2019). The 3-tier architecture based on the works is shown in Figure 9.

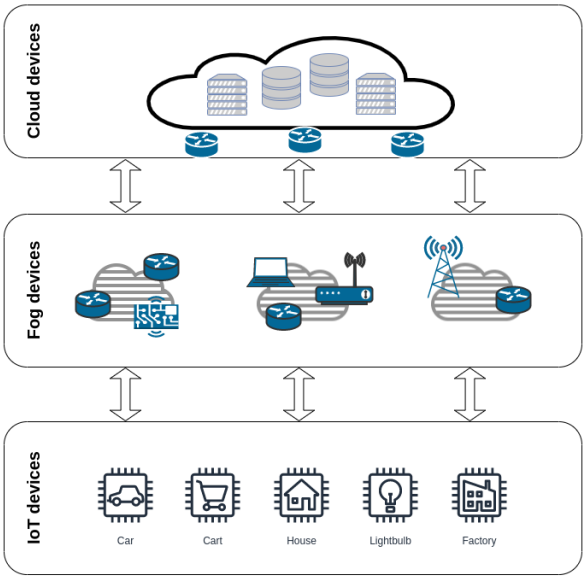
Because it is a recent theme, fog computing has some different proposed definitions, but one of the state-of-art standards known in the literature is the fog landscape concept. Proposed by

Figure 8 – SOA standard.



Source: Adapted from (MINH et al., 2017)

Figure 9 – 3-tier fog architecture.



Source: Author.

OpenFog consortium, fog landscape consists of a set of computational power resources organized in an hierarchical structure that places and executes application services. The components of a fog landscape are fog gateway, fog cell, fog computing node, fog controller, and fog colony (SKARLAT et al., 2018).

Fog Colony

A fog colony is a group of fog cells and fog nodes sets, and only one fog control. It's the basic concept in a fog landscape, which is used as a dynamic distributed micro-data center providing edge resources for IoT application requests deployments to serve IoT devices (TRAN et al., 2019; SKARLAT et al., 2018).

Fog Cell

Fog cell is a software running in an edge device with power processing, network, and storage resources constraints that manages IoT devices such as sensors and actuators. Typically, a set of fog cells forms a fog colony which only one of them (generally the cell with most resources) is chosen to become the fog control node (VENTICINQUE; AMATO, 2019; PHAM-NGUYEN; TRAN-MINH, 2019; SKARLAT et al., 2018).

Fog Computing Node

A fog computing node is a fog cell with more features providing storage, processing, and networking functionalities acting as access points to other fog cells. They are also considered resources for placement, hosting services in the form of unikernels, containers, or virtual machines. Some examples of fog nodes are gateways, switches, routers, firewalls, dedicated servers, etc (YOUSEFPOUR et al., 2019; MAHMUD et al., 2019a; SKARLAT et al., 2018).

Fog Controller

The fog controller is the main component of the fog, it's a node responsible for discovering and managing other fog cells and nodes. Also, it is responsible for receives IoT application requests, solves the service placement problem and deploys them. For each fog colony there is only one fog controller (YOUSEFPOUR et al., 2019; SKARLAT et al., 2018).

Fog Gateway

A fog gateway is the first fog node which communicates directly to IoT devices such as sensors and actuators. It is responsible for subscribing IoT devices in the fog colony. Some examples of fog nodes are on-premise devices like modems, smartphones, and raspberry boards (APAT et al., 2020; BARANWAL; YADAV; VIDYARTHI, 2020; MAHMUD et al., 2019a).

3.2 Intelligent Transportation System

One of the most important fields of Smart City is Intelligent Transportation System (ITS) (QIAO, 2022). With the increase in the volume of vehicles traveling on city roads, there is a need to equip vehicles and infrastructure with IoT devices with sensors and actuators to collect information and make decisions about road traffic (HARVEY; KUMAR, 2020). Also known as Smart Transport or Traffic Management System, ITS is an area that consists of applying intelligent solutions to improve the mobility and safety conditions of urban roads for vehicles and pedestrians (MARCOS et al., 2022; OMONIWA et al., 2018).

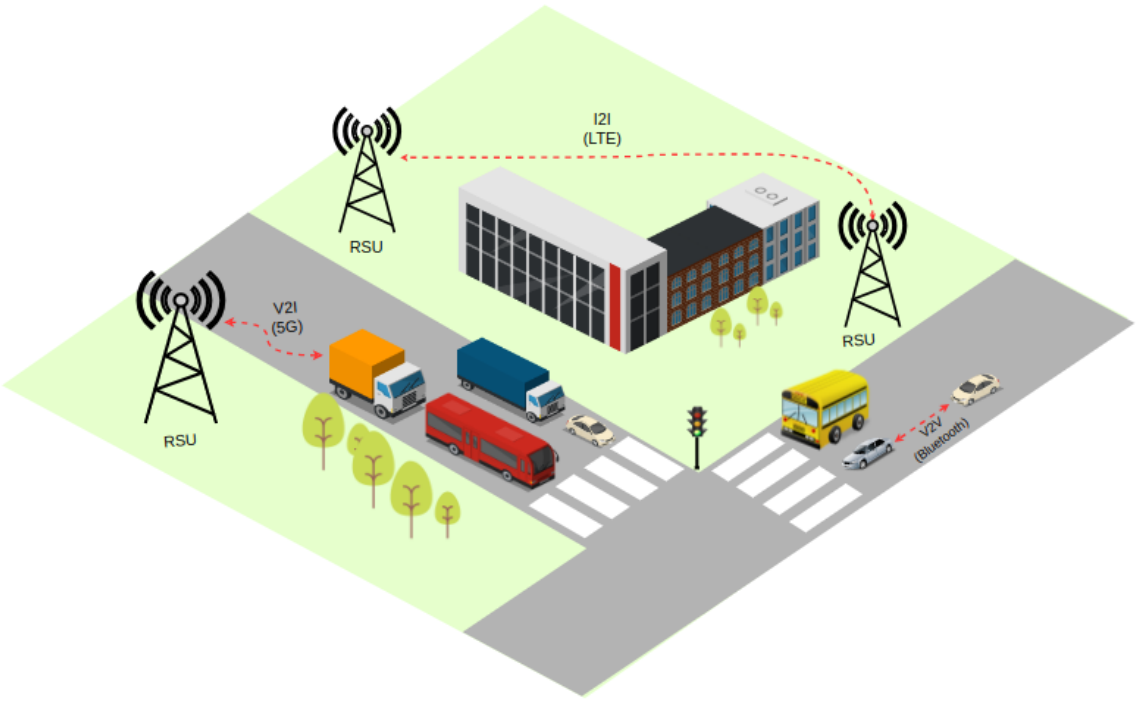
Among the main solutions are the monitoring and management of the traffic flow on the roads, with the aim of reducing congestion and preventing traffic accidents. Smart traffic light, for example, is an ITS application that aims to control the flow of urban roads and also prevent accidents through the control of intelligent traffic lights (OMONIWA et al., 2018). Through a real system IoT control system, sensors and cameras collect the volume of the road to be analyzed with big data strategies and provide information that regulate the signaling of traffic lights in the region (RABBY; ISLAM; IMON, 2019).

Other ITS applications include Smart Parking, an intelligent system that allows drivers to reserve a parking area during the traveling and Smart Assistance, a system of cameras and sensors spread across the road for automatic emergency activation in the detection of traffic

accidents (RABBY; ISLAM; IMON, 2019).

One of the great challenges of ITS is managing traffic efficiently and with low energy consumption, less time and high security. For this, one of the focuses is the collection of data in real time. Multiple stationary sensors such as infrared, Radio Radio Frequency IDentification (RFID), camera and accelerator, are installed along the roads and are identified as a Road Side Unit (RSU) or belonging to the vehicle, called On Board Unit (OBU) (ROY; PATNAIK; DUTTA, 2021), as shown in Figure 10. The architecture illustrated in Figure 10 represents a Vehicular Ad-hoc NETwork (VANET) environment composed of three types of communications Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Infrastructure-to-Infrastructure (I2I). In a VANET, vehicles can obtain and share accurate information about road traffic, infrastructures can share data with vehicles and other infrastructures in order to maintain urban mobility (LIANG et al., 2020).

Figure 10 – VANET architecture.



Source: Author.

In V2V communication, vehicles communicate with other vehicles exchanging information about their states and warning about traffic jams and possible dangers of accidents. The exchange of information in this type of communication is fast for nearby vehicles, however, to reach distant vehicles it is necessary to use V2I communication, due to the communication overhead that exists in the jump between the vehicular devices. In V2I, vehicles transmit information to nearby RSU infrastructures that are scattered on the roads. RSUs receive and retransmit data to other vehicles, infrastructures or analyzes on ITS servers via I2I communication (CHATTERJEE et al., 2022; ULLAH et al., 2019).

In order to perform V2V, V2I and I2I communications there are wireless communication protocols established in the literature. Some of them are IEEE 802.11 (Wi-Fi), IEEE 802.16 (WiMAX - Worldwide Interoperability for Microwave Access), IEEE 802.15.7 based on Visible Light Communication (VLC), Long-Term Evolution Advance (LTE-A), 5G , Bluetooth, ZigBee and satellite communication. These protocols were established with the aim of standardizing VANET communication networks due to the relevance obtained in the academic world and in government agencies (HUSSEIN et al., 2022).

Despite the relevance and advantages that VANETs present for ITS, there are challenges in the implementation of applications that require attention from researchers for improvements, such as low latency requirements, communication scalability, high vehicles dynamism, data security and privacy, failure detection and diagnostics, network resources scarcity, application scheduling, infrastructure deployment costs, computational complexity, different Quality of Service (QoS) requirements, and comprehensive Quality of Experience (QoE) modeling (HUSSEIN et al., 2022).

3.3 Kintoun Simulator

Created in this work, Kintoun[1] is a resource simulation tool for Fog Computing and distributed systems. Developed in Python 3, it is a continuous-time asynchronous simulator that aims to study the service placement of IoT applications. It is a tool inspired by the YAFS (Lera; Guerrero; Juiz, 2019) and iFogSim (GUPTA et al., 2017) simulators, however, unlike them, Kintoun executes the services placed on a device using the Round-Robin strategy to be able to process several services in a way close to the real thing. Furthermore, the proposed simulator is validated based on M/M/1 queuing theory as presented in Chapter 5.

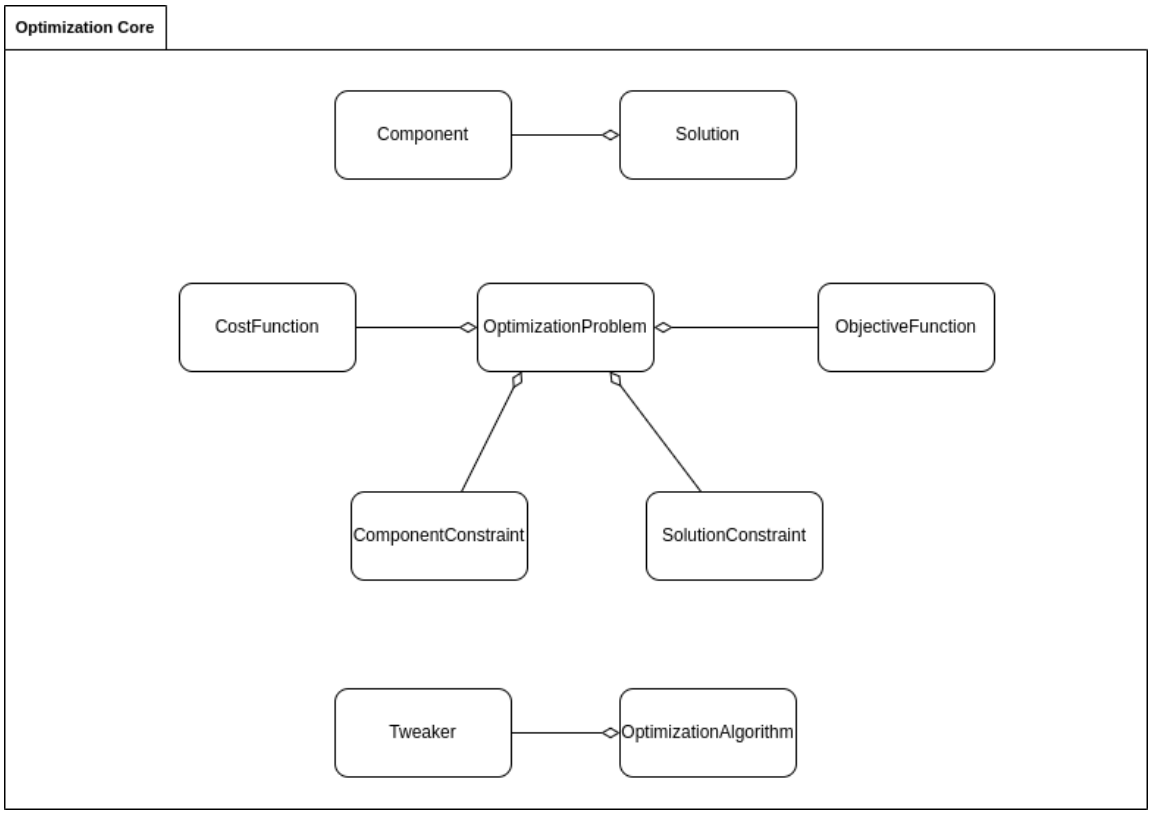
Kintoun simulator consists of three cores: Optimization core, Application core and Infrastructure core. Optimization core contains frameworks for modeling the optimization problem, Figure 11 presents the reduced model diagram. Application core contains frameworks for customizing the services and applications that will be generated, Figure 12 shows the reduced model diagram. Finally, Infrastructure core contains frameworks for customizing infrastructure devices, Figure 13 shows the reduced model diagram.

In the optimization core there are the *Component* and *Solution* entities that represent the modeling of the problem solution (see Figure 18). For example, in the Fog Service Placement Problem, a *Component* can represent the tuple (service, node) that indicates the service placement plan on a given node. The set of *Component* entities form a *Solution*. For example, the mapping set (service, node) forms a solution of the service placement plan of an application in an infrastructure of Fog nodes. In the core, we also have the *OptimizationProblem* entity, which represents the optimization problem, for example, the Fog Service Placement Problem. An

[1] Available in: <https://github.com/jonathan-cunha/kintoun>

OptimizationProblem contains one or many objective functions (*ObjectiveFunction*) and may contain constraints (*SolutionConstraint*). If the optimization problem is combinatorial, as in the case of FSPP, it may contain cost functions (*CostFunction*) and component constraints (*ComponentContraint*). Finally, the optimization core has the representation of the algorithm that will solve the optimization problem (*OptimizationAlgorithm*) and an entity for adjusting the solution (*Tweaker*), if necessary.

Figure 11 – Kintoun optimization core module.

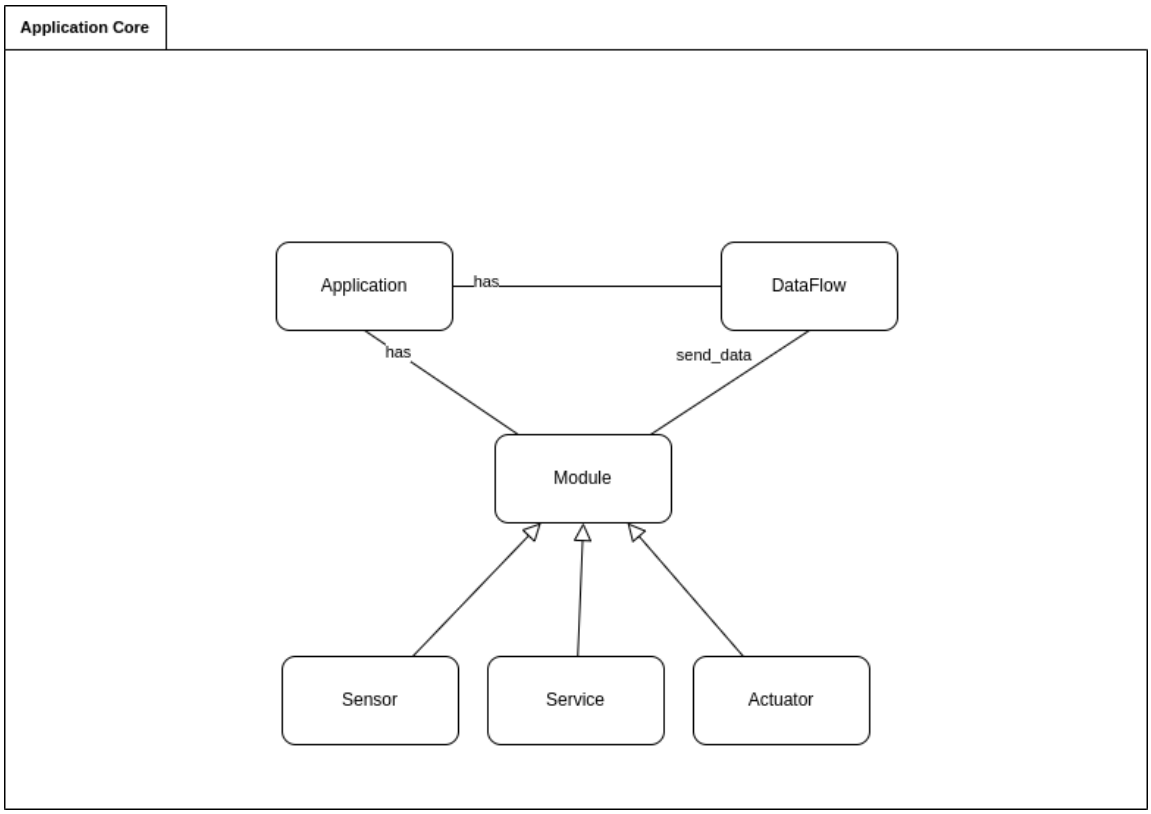


Source: Author.

The application core is represented by an Application entity that contains modules (*Module*) and data flow between modules (*DataFlow*). A module can be of type *Sensor*, *Actuator* or *Service*. Modules of type *Sensor* and *Actuator* represent a sensor and an actuator of the end device or IoT, consequently, these modules only run on these types of devices. The *Service* module, on the other hand, represents the services executed in the fog or cloud nodes, this module is responsible for processing the data requested by the end devices layer devices. Finally, sending data between modules is represented by the *DataFlow* entity that connects two modules, one of origin (who sends the data) and one of destination (who receives the data). The set of *Module* and *DataFlow* entities form a DAG (Directed Acyclic Graph).

Finally, the infrastructure core has entities that represent the devices, communication links and the modeled infrastructure. In a *Fog Colony* infrastructure are *Fog Gateway*, *Fog Computing Node* and *Fog Controller* devices. Extending across the entire Fog Landscape we

Figure 12 – Kintoun application core module.



Source: Author.

also have *IoT devices* and *Cloud devices*. All infrastructure devices are connected by a network through *Wired* or *Wireless* links.

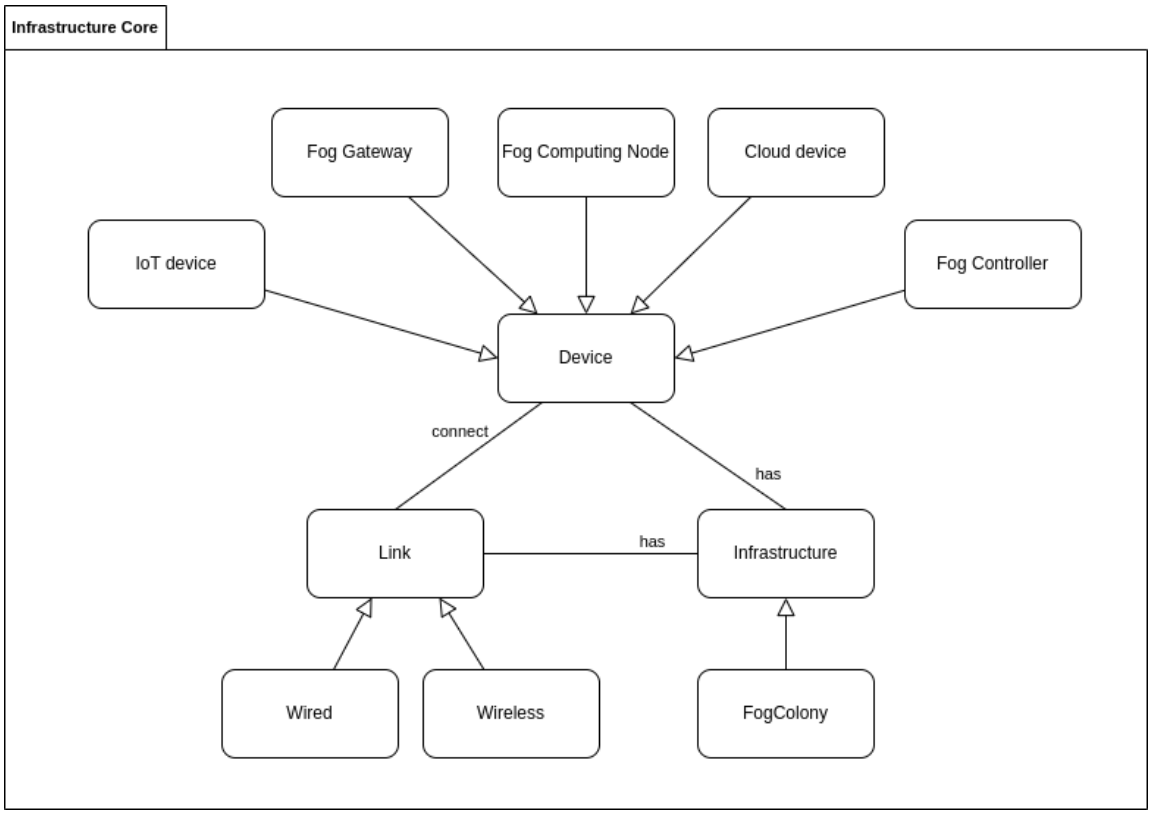
The main features of Kintoun are:

- Device customization: processing nodes, gateways, links, routers, and any other type;
- Customization of device models and features;
- Customization of services and applications;
- Network customization and distributed infrastructure;
- Customization of service placement optimization problem;
- Customization of algorithms for solving the placement problem;
- Customization and analysis of performance metrics.

3.4 R3GP – Rotation-Guided Greedy Genetic Particle

In this section, the functioning of the algorithm proposed for the calculation of the placement plan in Fog Computing will be explained. Named Rotation-Guided Greedy Genetic

Figure 13 – Kintoun infrastructure core module.



Source: Author.

Particle (R3GP), the proposed algorithm is based on (PLATEL; SCHLIEBS; KASABOV, 2008) and in characteristics present in the Grover, Guided Local Search, Greedy, Genetic Algorithm and Particle Swarm Optimization algorithms. The Algorithm 1 presents an overview of R3GP.

The general idea of the algorithm is to guide the current best solution from new components (particles) with greater probability of success. For this, a temporary solution T loaded with new particles based on the repellent pheromone is generated. In this way, the combination between the new particles and the best solution is made to generate a new individual J and attract it to S with the particles with the best global probability, if necessary. If the new individual J is the best solution found, rotate the particles of J by $+\theta$ degrees to increase the probability of being selected in the next iterations, otherwise rotate $-\theta$ to decrease the probability. Finally, the individual S synchronizes with J to obtain the particles with the highest probability and updates the pheromones of the particles of T and $Best$ so that they are avoided in the next iterations.

The main element of R3GP is the Particle. The Particle is a data structure that represents a component of the solution — a tuple (service, server), for example —, it stores information about the quantum state of the component and the repulsion pheromone. Through the quantum state it is possible to determine the probability of the component being accepted as part of the final solution. The repulsion pheromone is a memory of the number of times the component was used as part of a candidate solution, with it it is possible to determine the probability of the

Data: Services, Nodes
Result: $Best$
1 $P \leftarrow |Services| \times |Nodes|$ matrix of particles
2 Particles pheromones of P initialization with value 1
3 Particles Superposition(P)
4 $\theta \leftarrow arcsin(\frac{1}{\sqrt{|Nodes|}})$
5 $S \leftarrow$random solution
6 Evaluate fitness (S)
7 $Best \leftarrow S$;
8 **while** *a stop condition is not satisfied* **do**
9 $T \leftarrow$ PheromoneTournementSelection(P)
10 $J \leftarrow$ AdjustedGreedyCrossover(T, $Best$, P)
11 $J \leftarrow$ Attract(J, T, S, $Best$, P)
12 Evaluate fitness(J)
13 **if** *Fitness(J) is better than Fitness($Best$)* **then**
14 Rotate(J, $Best$, θ)
15 $Best \leftarrow J$
16 **else**
17 Rotate(J, $Best$, -θ)
18 **end**
19 $S \leftarrow$ Synchronize(S, J)
20 UpdatePheromones(T)
21 UpdatePheromones($Best$)
22 **end**

Algorithm 1: Rotation-Guided Greedy Genetic Particle.

component being accepted as part of the current solution.

It is possible to apply a rotation and a synchronization operation to any Particle. Rotating a Particle means changing its quantum state, consequently increasing or decreasing the probability of the Particle being accepted as part of the final solution to the problem. Synchronizing a Particle is the act of checking whether a particle has the largest current probabilistic amplitude and, if so, adding it to a temporary solution that has the components most likely to be accepted as part of the final solution.

In detail, regarding Algorithm 1, in line 1, the repulsion pheromones of P particles are initialized with the value 1, to avoid division by zero. Inspired by Quantum Computing algorithms, such as Grover's algorithm, in line 2, the particles are placed in superposition of equal probability. Algorithm 2 presents the superposition formulation. Basically, the vector of each particle, initially in state $|0\rangle = [1, 0]$, is multiplied by the Hadamard matrix $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, resulting in a new superposition vector called plus state given by $|+\rangle = \frac{1}{\sqrt{2}}[1\ 1]$.

In line 9 of Algorithm 1, a temporary solution is selected with the best particles according to the pheromones. The selection is made in a 2-by-2 tournament between particles that can

Data: P
Result:
1 $H \leftarrow \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
2 **foreach** *particle* $p \in P$ **do**
3 $\quad |p\rangle \leftarrow H|p\rangle$
4 **end**

Algorithm 2: Particles Superposition

compose the solution. Algorithm 3 presents this tournament. While the solution is not complete, randomly, two particles are selected. Then a random draw of a number between 0 and 1 is done with the $random()$ function. If the drawn number is within the range of one of the particles, the particle is selected to compose the solution. The higher the pheromone value, the lower the chance of the particle being selected. This is a strategy based on the Guided–Local Search algorithm, which also uses repulsion pheromones to be able to select new components.

Data: P
Result: T
1 **while** *Temporary solution T not completed* **do**
2 $\quad T = []$
3 $\quad p1 \leftarrow$ random selection from P
4 $\quad p2 \leftarrow$ random selection from P
5 $\quad r \leftarrow \frac{Pheromone(p1)}{Pheromone(p1)+Pheromone(p2)}$
6 $\quad$ **if** $r \leq random()$ **then**
7 $\quad\quad$ Add $p1$ to T
8 $\quad$ **else**
9 $\quad\quad$ Add $p2$ to T
10 $\quad$ **end**
11 **end**

Algorithm 3: Pheromone Tournement Selection

In line 10 of Algorithm 1, a new solution is created based on a crossover between the components of the temporary solution T and the best solution found so far (Best). Algorithm 4 presents the crossover steps. In lines 4 and 5, measurements of the probabilistic amplitudes of the T and Best particles are made. The measurement is made using Algorithm 9. Then these amplitudes are combined with the cost values of the components (particles) of T and B to generate a new cost with adjusted values. This is also a technique inspired by the adjustments of the GLS fitness function values. The greater the probabilistic amplitude, the greater the chance of the component being chosen – depending on the cost of the component – to compose the new solution. Whoever has the best cost adjustment will be chosen to compose the J solution.

In line 10 of Algorithm 1, the attraction function is used to correct possible repetitions of the solution generated by the crossover. Algorithm 5 presents the solution correction. Basically, if the solution J generated by the crossover is a repetition of the Best solution, then the solution J is modified. If the global synchronization solution S is also equal to Best, then J is assumed

Data: T, Best, P
Result: J
while *New solution J not completed* **do**
  $J = []$
  foreach *paired* $(t_i, b_i) \in (T, Best)$ **do**
    $A_t \leftarrow Measure(t_i)$
    $A_b \leftarrow Measure(b_i)$
    $t_i^{adjusted\ cost} \leftarrow (2 - A_t)Cost(t_i)$
    $b_i^{adjusted\ cost} \leftarrow (2 - A_b)Cost(b_i)$
    if $t_i^{adjusted\ cost} \leq b_i^{adjusted\ cost}$ **then**
      Add t_i to J
    else
      Add b_i to J
    end
  end
end

Algorithm 4: Adjusted Greedy Crossover

to be all particles from the temporary solution T generated by the pheromone tournament. If the synchronization solution S is different from Best, then J assumes the attraction particles of T for the particles of S that have the largest probabilistic amplitudes. Attraction is performed using Algorithm 4. This attraction is based on the attraction process used by Particle Swarm Optimization (PSO).

Data: J, T, S, Best, P
Result: J
if $J = Best$ **then**
  if $S = Best$ **then**
    $J \leftarrow T$
  else
    $J \leftarrow$ AdjustedGreedyCrossover(T, S, P)
  end
end

Algorithm 5: Attract

From lines 12 to 16 of Algorithm 1, a check is made whether the new solution J has a better fitness value than the current Best solution. If J is better than Best, the probabilistic amplitude of the particles are rotated $+\theta$ degrees toward J and $-\theta$ degrees away from Best, if not the inverse of the rotation. Algorithm 6 presents the rotation mechanism. Based on Grover's algorithm and the work of (PLATEL; SCHLIEBS; KASABOV, 2008), the rotation of the particle vector is done using a Rotation Matrix. The idea is to amplify the probabilistic amplitude of the components of the best solution and decrease those of the losing solution.

Finally, from lines 18 to 20 of Algorithm 1, the synchronization of the particles with the best probabilistic amplitudes between J and S, and the update of the pheromones of the particles

Data: J, $Best$, θ
1 $R \leftarrow \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$
2 $R_{inverse} \leftarrow \begin{bmatrix} cos(-\theta) & -sin(-\theta) \\ sin(-\theta) & cos(-\theta) \end{bmatrix}$
3 **foreach** *paired* $(j_i, b_i) \in (J, Best)$ **do**
4 **if** $j_i \neq b_i$ **then**
5 $|j_i\rangle \leftarrow R|j_i\rangle$
6 $|b_i\rangle \leftarrow R_{inverse}|b_i\rangle$
7 **end**
8 **end**

Algorithm 6: Rotate

in T and in $Best$ are performed. Synchronization is performed using Algorithm 7, which evaluates the particles in J and S in pairs and updates S with the particles with the highest probabilistic amplitude. The process of updating the pheromones is done by Algorithm 8, which simply adds +1 to the current value of the pheromone. This addition aims to decrease the probability of the particle being selected in a tournament.

Data: S, J
1 **foreach** *paired* $(s_i, j_i) \in (S, J)$ **do**
2 $A_s \leftarrow Measure(s_i)$
3 $A_j \leftarrow Measure(j_i)$
4 **if** $A_j > A_s$ **then**
5 $s_i \leftarrow j_i$
6 **end**
7 **end**

Algorithm 7: Synchronize

Data: Q
1 **foreach** *particle* $p \in Q$ **do**
2 $Pheromone(p) \leftarrow Pheromone(p) + 1$
3 **end**

Algorithm 8: Update Pheromones

Algorithm 9 shows how the probabilistic amplitude of a particle is measured. Before measuring the amplitude, the Hadamard Matrix is applied to project the vector in a state where the probability of the particle being in state $|1\rangle$ can be measured (lines 1 and 2), that is, the probability of the particle being accepted as part of of a solution. After applying the matrix, the amplitude is measured according to the square of the state coefficient value $|1\rangle$ (line 3).

Data: ψ

Result: A_ψ

1 $H \leftarrow \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

2 $|\phi\rangle \leftarrow H|\psi\rangle$

3 $A_\psi \leftarrow |\phi_1|^2$

Algorithm 9: Measure

4

Problem Formulation

This chapter presents the formulation of the service placement problem in Fog Computing infrastructure in the context of IoT applications. The mathematical model of the network topology, devices and IoT applications, the optimization functions and the constraints that describe the functioning of the system are introduced in Section 4.1. Finally, the representation of the mathematical solution into computational data structure is explained in Section 4.2.

4.1 Fog Service Placement Problem

The service placement problem in Fog computing, also known as the Fog Service Placement Problem, is defined in the formulation of the physical topology of devices, IoT applications and optimization model.

4.1.1 Physical topology of devices

In this work, a Fog Computing infrastructure hierarchy divided into three layers is considered. The first layer is formed by a set C of Cloud nodes. The second layer is composed of a set F of Fog nodes. The last layer is formed by a set T of IoT devices or end devices with sensors and actuators. Let $N = \{C, F, T\}$ be the set of all nodes of the layered infrastructure and L be the set of all links connecting the nodes of N, the physical topology of the fog infrastructure is represented in the form of an undirected graph $G_{topo} = (N, L)$, where N represents the set of vertices and L represents the set of edges.

Let $n_i \in N$, such that $i \in [0, |N| - 1]$, be a Fog infrastructure device, it can be modeled with the following characteristics:

Given a Fog infrastructure device $n_i \in N$, such that $i \in [0, |N| - 1]$, we can model its physical resources as follows:

- $cpu_{n_i}^{total}$: total CPU capacity in MIPS.
- $cpu_{n_i}^{usage}$: CPU usage in MIPS.
- $cpu_{n_i}^{available}$: $cpu_{n_i}^{total} - cpu_{n_i}^{usage}$ in MIPS.
- $mem_{n_i}^{total}$: total RAM capacity in MB.
- $mem_{n_i}^{usage}$: RAM usage in MB.
- $mem_{n_i}^{available}$: $mem_{n_i}^{total} - mem_{n_i}^{usage}$ in MB.
- $bw_{n_i}^{total}$: total bandwidth capacity in Mbps.
- $bw_{n_i}^{usage}$: bandwidth usage in Mbps.
- $bw_{n_i}^{available}$: $bw_{n_i}^{total} - bw_{n_i}^{usage}$ in Mbps.
- $P_{n_i}^{idle}$: power consumption of the device when not in use.
- $P_{n_i}^{max}$: power consumption of the device when it is being used at maximum capacity.

Given an infrastructure communication link Fog $l_{ij} \in L$, where $i, j \in [0, |N| - 1]$, such that $i \neq j$, represent any adjacent nodes n_i and n_j of N, we can model its physical resources this way:

- $bw_{l_{ij}}^{total}$: total bandwidth capacity in Mbps.
- $bw_{l_{ij}}^{usage}$: bandwidth usage in Mbps.
- $pt_{l_{ij}}$: propagation time in ms.

Given a link $l_k \in path_{ij}$, where $path_{ij} \subset L$ is the set of links representing the shortest path connecting any two adjacent or non-adjacent nodes n_i and n_j, such that $n_i, n_j \in N$, the available bandwidth and the propagation time between these two nodes are given by:

- $bw_{path_{ij}}^{available} = min(bw_{l_k}^{available})$
- $pt_{path_{ij}}^{total} = \sum pt_{l_k}$

4.1.2 IoT applications

An IoT application is composed of a set M of modules that can be executed in a distributed way in different computing nodes. Also, it is composed by a set D of dataflows that represent the flow of data between the modules. Given the restriction that there cannot be data flow cycles between the modules of M, an IoT application is modeled in the form $G_{app} = (M, D)$, such that G_{app} is a Directed Acyclic Graph (DAG), where M represents the set of vertices and D the set of directed edges of the graph. A DAG can be classified as sequential (Figure 14), parallel (Figure 15) or hybrid (Figure 16).

Figure 14 – Services in sequence.

Figure 15 – Services in parallel.

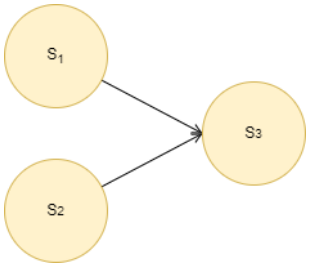
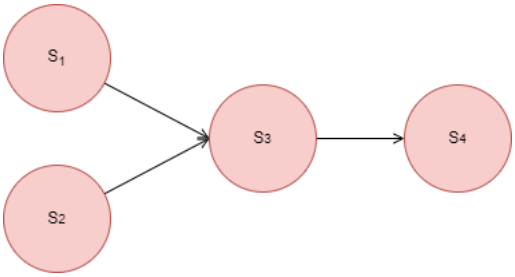


Figure 16 – Services in parallel and sequence.



Source: Adapted from (PHAM-NGUYEN; TRAN-MINH, 2019)

A module is divided into three types: Sensor: module responsible for capturing data from the environment. This module runs on data sources such as IoT devices or end devices. Actuator: module responsible for executing an action on the environment. This module runs on actuator devices such as IoT or end devices. Services: module responsible for processing data. This module runs on Fog or Cloud nodes. Thus, a service $s_i \in S$, such that $S \subset M$, has resource requirements that are modeled as follows:

- $cpu_{s_i}^{req}$: required CPU capacity in Million Instructions (MI).
- $mem_{s_i}^{req}$: required RAM capacity in MB.
- $bw_{s_i}^{req}$: required bandwidth capacity in Mbps.
- $delay_{s_i}^{exe}$: maximum service execution delay in ms.

Regarding the data flow, let $d_{ij} \in D$ be a data flow between any two adjacent modules m_i and m_j, where $i, j \in [0, |M|-1]$, such that $i \neq j$, we can model the stream d_{ij} with the following information:

- $data_{d_{ij}}^{size}$: size of data sent from m_i to m_j.
- $delay_{d_{ij}}^{com}$: maximum communication delay between m_i and m_j.

Finally, for any application a_i, where $i \in [0, |A|-1]$, such that A is a set of IoT applications, the maximum application execution delay is given by $delay_{a_i}^{max}$. This definition is given due to the delay in processing and communication that are related to the application request, waiting time for the request in the Fog Controller queue, time to calculate the placement plan and time to deploy the modules. Figure 17 shows the life-cycle of an application request.

4.1.3 Optimization model

This work models the Fog Service Placement Problem in the form of a constraint satisfaction problem. The model considers five objective functions: makespan, energy consumption gap, CPU load-balancing, memory load-balancing and bandwidth load-balancing. Furthermore, functions are subject to CPU, memory, bandwidth, service deadline and application deadline violations restrictions. Given the decision function $X(i, j)$ in Eq. 4.1, the functions and constraints are formulated below.

$$X(i,j) = \begin{cases} 1, & if\ service\ s_i\ is\ on\ node\ n_j, \\ 0, & if\ not \end{cases} \tag{4.1}$$

Makespan

The makespan (MS) (Eq. 4.4) is described as the total processing time of the services (t_S^{proc}) (Eq. 4.2) plus the total communication time ($delay_D^{com}$) between the services (Eq. 4.3).

$$t_S^{proc} = \sum_{i=0}^{|S|-1} \sum_{j=0}^{|F|-1} X(i,j) \frac{cpu_{s_i}^{req}}{(cpu_{n_j}^{capacity} - cpu_{n_j}^{usage})} \tag{4.2}$$

$$delay_D^{com} = \sum_{i=0}^{|S|-1} \sum_{j=0}^{|F|-1} \sum_{k=0}^{|F|-1} X(i,j)X(i+1,k)\left(\frac{data_{s_i}^{size}}{bw_{path_{jk}}^{available}} + pt_{path_{jk}}^{total}\right) \tag{4.3}$$

$$MS = t_S^{proc} + delay_D^{com} \tag{4.4}$$

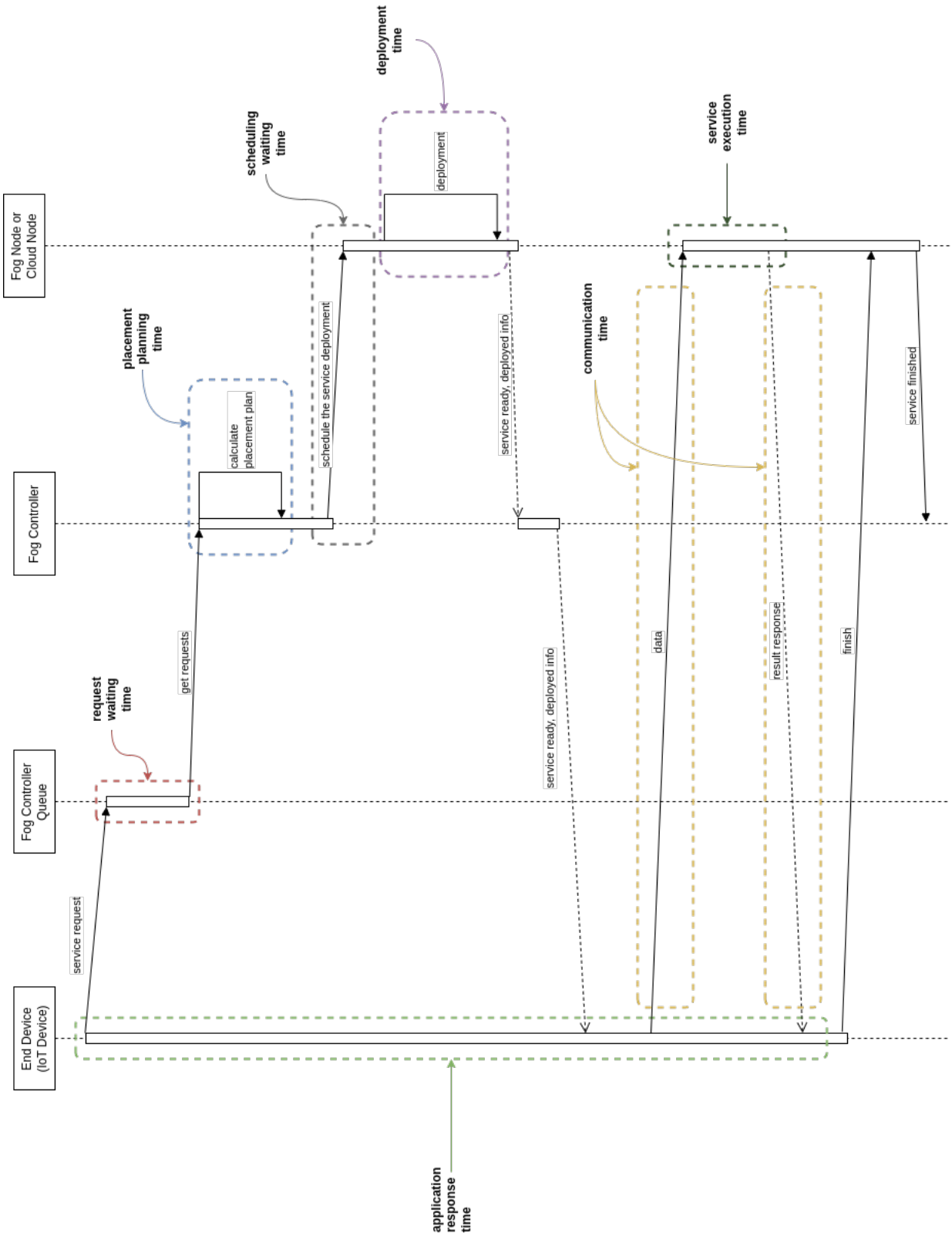


Figure 17 – Application request life-cycle.

Source: Author.

Energy consumption gap

The energy consumption gap of the Fog system (ΔE) (Eq. 4.7) is defined as sum of the power consumption gap (ΔP_{ij}) (Eq. 4.6) during the service processing time (t_{ij}^{proc}) (Eq. 4.5) of each device in the Fog. The λ_{ij} represents the CPU processing load of a node n_j with a service s_i, in million instructions, and C_j represents the CPU processing capacity of a node n_j in one second, in million instructions.

$$t_{ij}^{proc} = \frac{cpu_{s_i}^{req}}{(cpu_{n_j}^{capacity} - cpu_{n_j}^{usage})} \tag{4.5}$$

$$\Delta P_{ij} = (P_{n_j}^{max} - P_{n_j}^{idle})\frac{\lambda_{ij}}{C_j} \tag{4.6}$$

$$\Delta E = \sum_{i=0}^{|S|-1}\sum_{j=0}^{|F|-1} X(i,j)\Delta P_{ij} t_{ij}^{proc} \tag{4.7}$$

CPU load-balancing

CPU load-balancing ($cpu_F^{balancing}$) (Eq. 4.10) is defined as the variance of CPU utilization over CPU capacity at each Fog node.

$$cpu_j^{load} = \frac{cpu_{n_j}^{usage} + (\sum_{i=0}^{|S|-1} X(i,j) cpu_{s_i}^{req})}{cpu_{n_j}^{capacity}} \tag{4.8}$$

$$cpu_\mu^{load} = \frac{1}{|F|}\sum_{j=0}^{|F|-1} cpu_j^{load} \tag{4.9}$$

$$cpu_F^{balancing} = \frac{1}{|F|}\sum_{j=0}^{|F|-1} (cpu_j^{load} - cpu_\mu^{load})^2 \tag{4.10}$$

Memory load-balancing

Memory load-balancing ($mem_F^{balancing}$) (Eq. 4.13) is defined as the variance of Memory utilization over Memory capacity at each Fog node.

$$mem_j^{load} = \frac{mem_{n_j}^{usage} + (\sum_{i=0}^{|S|-1} X(i,j) mem_{s_i}^{req})}{mem_{n_j}^{capacity}} \tag{4.11}$$

$$mem_\mu^{load} = \frac{1}{|F|}\sum_{j=0}^{|F|-1} mem_j^{load} \tag{4.12}$$

$$mem_F^{balancing} = \frac{1}{|F|} \sum_{j=0}^{|F|-1} (mem_j^{load} - mem_\mu^{load})^2 \tag{4.13}$$

Bandwidth load-balancing

Bandwidth load-balancing ($bw_F^{balancing}$) (Eq. 4.16) is defined as the variance of bandwidth utilization over bandwidth capacity at each Fog node.

$$bw_j^{load} = \frac{bw_{n_j}^{usage} + (\sum_{i=0}^{|S|-1} X(i,j) bw_{s_i}^{req})}{bw_{n_j}^{capacity}} \tag{4.14}$$

$$bw_\mu^{load} = \frac{1}{|F|} \sum_{j=0}^{|F|-1} bw_j^{load} \tag{4.15}$$

$$bw_F^{balancing} = \frac{1}{|F|} \sum_{j=0}^{|F|-1} (bw_j^{load} - bw_\mu^{load})^2 \tag{4.16}$$

Constraints

The CPU (Eq. 4.17), memory (Eq. 4.18) and bandwidth (Eq. 4.19) constraints mean that the sum of resource utilization cannot exceed the maximum capacity of the resource in a Fog node. Likewise, the restriction service deadline violation (Eq. 4.20) means that the execution time of a service cannot exceed the specified delay.

$$\sum_{i=0}^{|S|-1} X(i,j) cpu_{s_i}^{req} \leq cpu_{n_j}^{available}, \forall j \in [0, |F|-1] \tag{4.17}$$

$$\sum_{i=0}^{|S|-1} X(i,j) mem_{s_i}^{req} \leq mem_{n_j}^{available}, \forall j \in [0, |F|-1] \tag{4.18}$$

$$\sum_{i=0}^{|S|-1} X(i,j) bw_{s_i}^{req} \leq bw_{n_j}^{available}, \forall j \in [0, |F|-1] \tag{4.19}$$

$$\frac{\sum_{i=0}^{|S|-1} X(i,j) cpu_{s_i}^{req}}{cpu_{n_j}^{available}} \leq delay_{s_i}^{exe}, \forall j \in [0, |F|-1] \tag{4.20}$$

Mathematical model

With the specified functions and constraints, the Fog Service Placement Problem is defined as the placement of application services from a set A in Fog nodes or Cloud, while minimizing the Euclidean distance of the objective functions (Eq. 4.22) subject to the constraint

conditions of the infrastructure. Mathematical description follows below. Given the objective functions:

$$f_1 : MS,\ f_2 : \Delta E,\ f_3 : cpu_F^{balancing},\ f_4 : mem_F^{balancing},\ f_5 : bw_F^{balancing} \tag{4.21}$$

The optimization model of the Fog Service Placement Problem is given by:

$$Minimize : \sqrt{f_1^2 + f_2^2 + f_3^2 + f_4^2 + f_5^2} \tag{4.22}$$

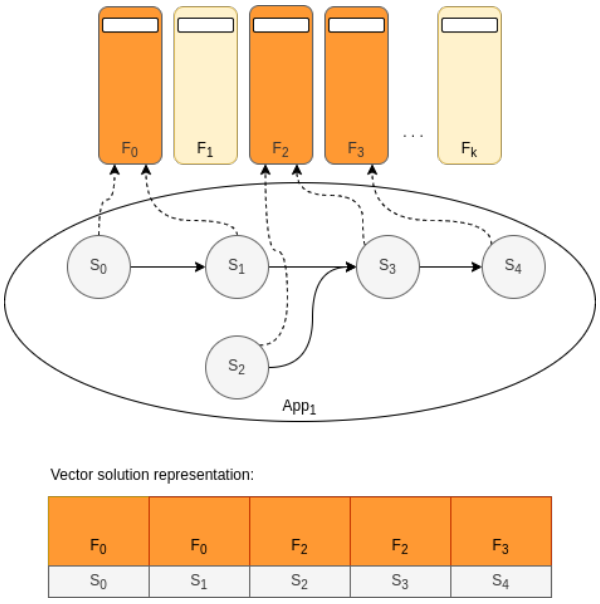
subject to the constraints 4.17, 4.18, 4.19, 4.20.

The range of output values of each objective function is $[0, +\infty)$, therefore, the range of output values of the minimization function is also $[0, +\infty)$.

4.2 Computational representation of a solution

In order to enable heuristic and metaheuristic algorithms solving the optimization problem of Eq. 4.22, it is necessary to model the representation of the solution. In this work, based on (NATESHA; GUDDETI, 2021), the solution is modeled in the form of a vector of components. Each array position represents a single service module of the requested application. The array size is equal to the number of services. Each component represents a pair (service, node) = (S_i, F_j), where the node is a fog device and can be repeated for more than one service. Figure 18 presents an example of a S_i service placement planning solution for an App_1 application in an infrastructure of Fog Computing Nodes F_k.

Figure 18 – Solution representation.



Source: Author.

Part II

Experimental Evaluations

5

Kintoun Simulator Validation

The purpose of this chapter is to explain the method used to validate the Kintoun simulator. Section 5.1 presents an overview of the methodology for conducting the validation experiment. Section 5.2 shows the experimental planning with a detailed description of context selection, dependent and independent variables, hypotheses formulation, step-by-step of the experimental design and the devices and tools used as instruments for the execution of the experiment. Section 5.3 presents the details of how the experiment was carried out. Results and statistical analyzes of the data are presented in Section 5.4. Finally, the conclusion about the validity of the simulator based on queuing theory is presented in Section 5.5.

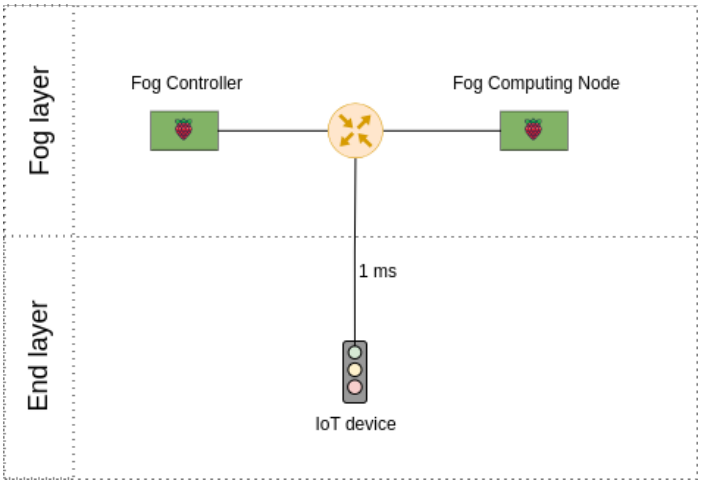
5.1 Methodology

The methodology used to validate the Kintoun simulator, created in this work, is based on an explanatory research, which addresses the modeling of the Fog Controller as an M/M/1 queue server, with infinite buffer, which queues IoT application request messages, as described in (HUSSAIN et al., 2020; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020). In this study, an *in silico* experiment was conducted to evaluate the performance of the system related to the Fog Controller queue. The system was stressed using requests from a synthetic application. Results were collected and analyzed using statistical methods.

The experiment simulated a star topology, as shown in Figure 19. The infrastructure is composed of an IoT device that performs the application requests, a Fog Controller that receives the request messages and performs the calculation of the application services placement plan, a Fog Computing Node that executes the application services and a Fog Gateway that forwards network packet traffic. Application requests were made following an exponential distribution (SHAIK; BASKIYAR, 2022; BARANWAL; VIDYARTHI, 2022; LLORENS-CARRODEGUAS et al., 2021; RAHBARI; NICKRAY, 2020; HUSSAIN et al., 2020; KHARE et al., 2018). As a

performance metric, the number of messages waiting in the Fog Controller buffer was collected. The metric collection interval was 1 second, measuring throughout 1 hour, that is, a total of 3600 samples were collected for each utility value of the M/M/1 queue flow rate.

Figure 19 – Simulator validation experiment topology.



Source: Author.

Before running the experiments, it was necessary to prepare and provision the components of the Fog architecture, the synthetic application service, the coding of the Fog Service Placement Problem and the placement algorithm. The topology of the Fog infrastructure prepared in the simulator was based on a star network. The script for executing the experimental steps was implemented according to the experimental design organized according to the Goal Question Metric (GQM) approach proposed by (BASILI; CALDIERA; ROMBACH, 1994) and described in subsection 6.2.2.

Finally, after running the experiments, using the one sample Student's t-test, the results obtained from running the simulator were compared with the theoretical utility values of queuing theory for M/M/1 queue with infinite buffer (JAIN, 1991).

5.2 Experimental Planning

5.2.1 Objective Definition

The objective of the experiment was formally defined using the GQM method, proposed by (BASILI; CALDIERA; ROMBACH, 1994), as follows: **analyze** the utility distribution; **for the purpose of** evaluate the Fog simulator tool in contrast to the M/M/1 queue theory; **concerning** the queue size metric; **from the point of view of** research scientists, network architects, network engineers, IoT developers, and smart city companies; **in the context of** the Fog Service Placement Problem.

5.2.2 Planning

Context Selection

Based on (NATESHA; GUDDETI, 2021; AYOUBI; RAMEZANPOUR; KHORSAND, 2021; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020), this study consider the experiment context as a simulation scenario. The experiments were carried out using a synthetic IoT application requests for placement of the services in the Fog Computing Node.

Independent Variables

The independent variables of the experiment were the IoT application requests rate and the Fog Controller placement plan calculation rate.

Dependent Variables

The dependent variables of the experiment was the Fog Controller buffer size.

Hypothesis Formulation

Adopting the GQM method, the following research question was designed to fully cover the objectives of the work:

1. Does the fog simulator tool obey the M/M/1 queue theory?

In order to evaluate the question, the Fog Controller buffer size was used to compare with the theorical M/M/1 queue value. Thus, with the objective and metric defined, the following hypothesis were formulated:

H_0: the simulator system queue mean size confidence interval of the utility contains the theorical M/M/1 queue distribution value.

H_1: the simulator system queue mean size confidence interval of the utility does not contains the theorical M/M/1 queue distribution value.

Formally, the hypothesis can be described as:

$$H_0 : \mu_{simulator}(queue\ size) = \mu_{theory}(queue\ size)$$

$$H_1 : \mu_{simulator}(queue\ size) \neq \mu_{theory}(queue\ size)$$

Object Selection

Since it is impossible to collect the entire population of the data metric monitored in the experiment, for the hypotheses evaluation were collected samples that represent it. Thus, to reach this, in this work, for each utility value the fog buffer size metric was collected $n_{samples}$

times, where $n_{samples}$ = 3600. The first 600 samples were discarded, because is the interval of the transient of the system, i. e., the time the system is reaching the stead state. This was adopted for the possibility of a normal distribution assumption of the data according to the Central Limit theorem (KWAK; KIM, 2017).

Experimental Design

The following high-level steps for experimenting were performed for each queue utility value:

1. The IoT device sends an IoT application request to the Fog Controller queue every $request_{interval}$ time with interval following a random exponential distribution.
2. For each random exponential $server_{interval}$ distribution, Fog Controller reads the queue;
3. The Fog Controller sends the requested IoT application service to the Fog Computing Node for placement;
4. Collect the Fog Controller queue size metric every 1 second for 3600 seconds;
5. Discard the first 600 seconds (the transient state);
6. Apply appropriate statistical tests to analyze all hypotheses.

Instrumentation

For experimental setup, the following software were used: Python 3.10 with pandas 1.5.0, numpy 1.24.0, networkx 3.0, and matplotlib 3.7.0 libraries, and Ubuntu 22.04 LTS operating system. Furthermore, the following hardware setup was employed: Intel dual core x86_64 processor with 3.1 GHz clock, 4 GB of RAM, and 1 TB of HDD.

5.3 Experimental Operation

5.3.1 Preparation

In this step the fog computing components and the experimental script was coded using the software specified in Instrumentation topic.

5.3.2 Execution

The experiment was carried out as planned in 5.2.2. The Ubuntu was used in the experiments without the Graphical User Interface (GUI) started, only via teletypewriter

(TTY). The infrastructure was simulated according to Figure 19. The application request followed an exponential distribution with an interval of X requests per second, where $X \in \{1, 2, 3, 4, 5, 6, 7, 8, 8.5, 9, 9.3, 9.5, 9.7, 9.9\}$. The placement algorithm used was a static placement strategy, always calculating the placement of the service in the single processing Fog node, however, with the calculation simulating a placement planning time following an exponential distribution of 10 requests served per second. Thus, the queue utility values tested were 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9, 0.93, 0.95, 0.97, and 0.99.

5.4 Results and Discussion

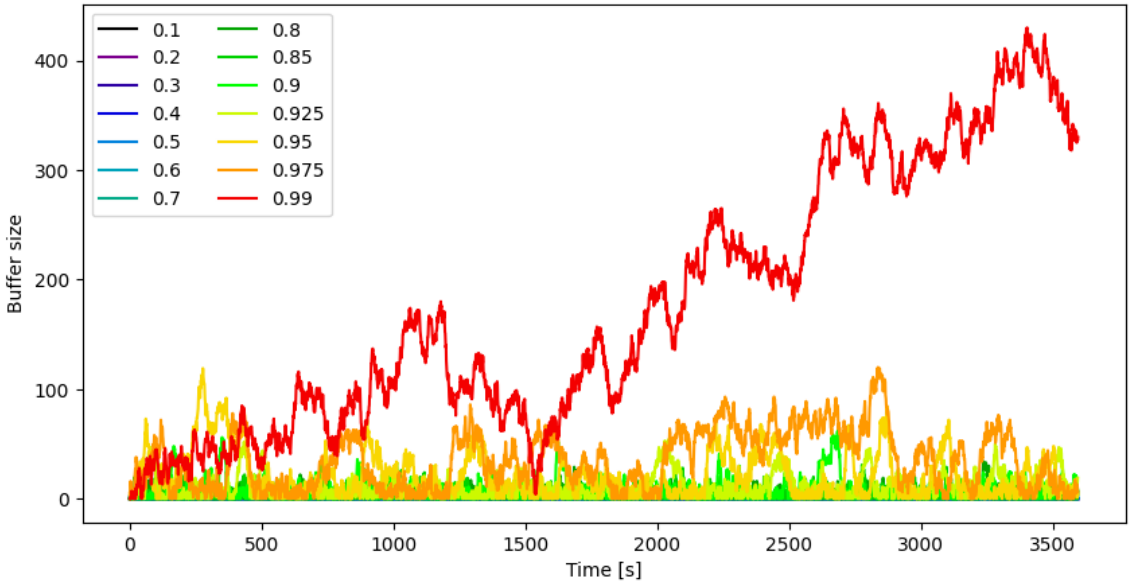
In order to validate the simulator, given that the data must be compared with the theoretical value of queuing theory, the Student's t-test for one sample was applied to validate the hypotheses formulated in 6.2.2 for a two-tailed normal distribution with 95% reliable. Thus, two analyzes were performed. The first was an individual analysis of queue utility values. In this analysis, given the sample size (3000 samples for each utility value) and invoking the Central Limit theorem, it was assumed that the data follow a normal distribution for the buffer size metric. The second analysis was performed to determine whether, overall, the simulation data follow the theoretical exponential distribution of M/M/1 queues. In this analysis, for each utility value, the average of the samples was subtracted from the theoretical value of the queue size (Eq. 5.1) and then the t-test was applied.

$$diff(utility) = \mu_{sample}(utility) - theory(utility) \tag{5.1}$$

In the individual analysis of the utilities, Figures 21 and 22 show that, visually, the simulator produced data similar to the theoretical exponential distribution, except for the utility value equal to 0.99. This exception means that the simulator has exceeded the stability limit of the exponential curve and has become unstable, as shown in Figure 20. Statistically, the p-values of Table 8 show that for the utility value 0.99 the null hypothesis is completely rejected, that is, the simulator cannot produce representative data for the theoretical value equal to 0.99. The table also shows rejection for the utility values 0.8 and 0.925, however, as seen in Figures 21 and 22, the values are very close to the theoretical value. The mean experimental value of utility 0.8 was approximately 3.46, while the theoretical value is 3.42, i.e., a relative difference of +8% from the theoretical value. As for the average experimental value of utility 0.925, it was approximately 10.34, while the theoretical value is approximately 11.41, that is, a relative difference of -9% from the theoretical value. Considering a relative error of ±10%, the values are acceptable.

In the general analysis of the utilities, the application of the Student's t-test in the subtraction of the average of the utilities by the theoretical value resulted in the p-value ≈ 0.340 with all the utility values and in the p-value ≈ 0.49 without the utility value of 0.99. Student's t-test was robust enough to validate that there is no rejection of the null hypothesis, which indicates that the simulator can, statistically, produce data following the theoretical exponential

Figure 20 – Fog Controller buffer size.



Source: Author.

Table 8 – Utility p-value.

Utility	**p-value**
0.1	0.706
0.2	0.101
0.3	0.345
0.4	0.647
0.5	0.030
0.6	0.810
0.7	0.541
0.8	4.859×10^{-5}
0.85	0.219
0.9	0.589
0.925	3.396×10^{-10}
0.95	0.745
0.975	0.622
0.99	0.000

Source: Author.

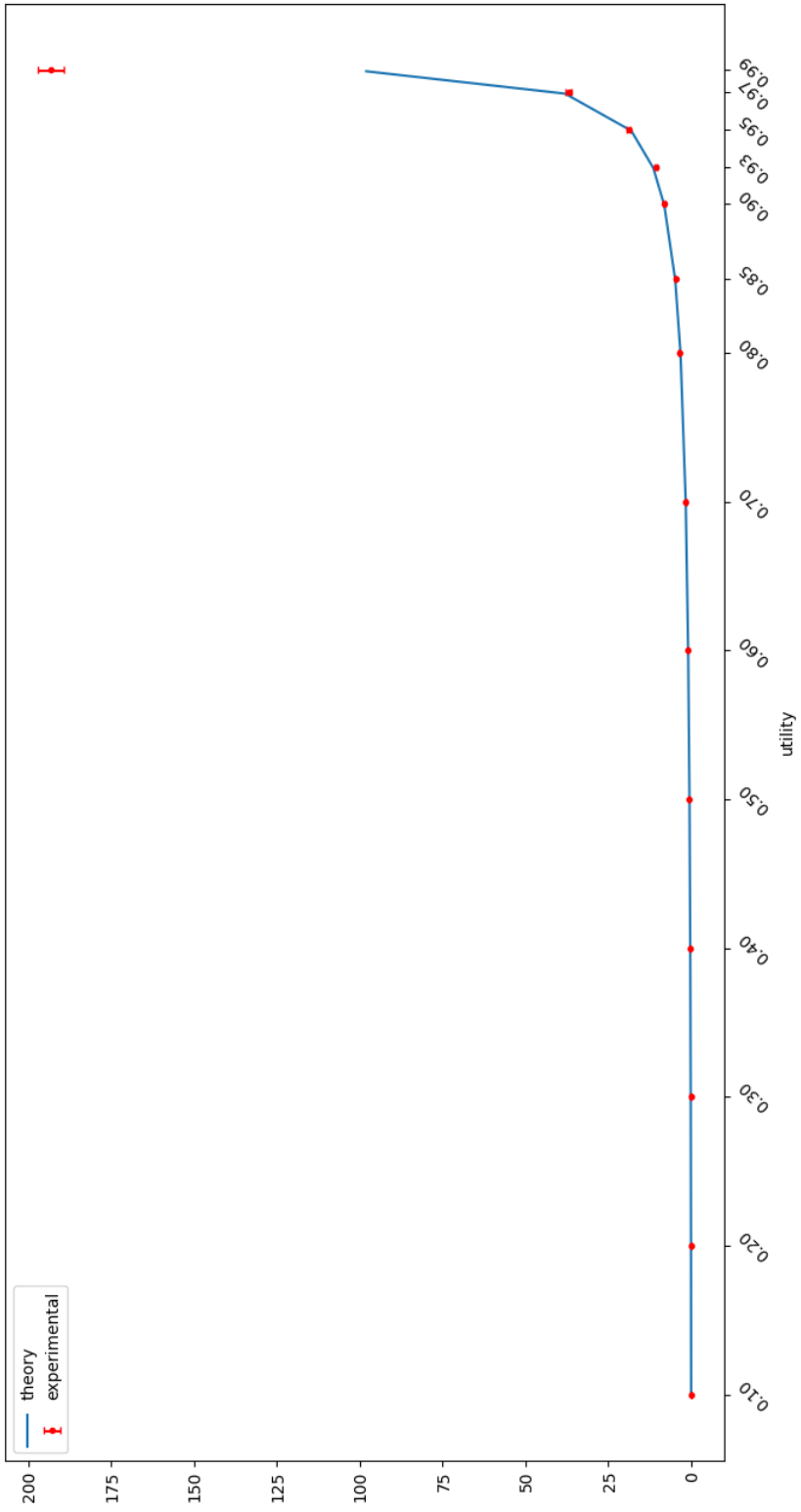


Figure 21 – Simulator M/M/1 utility.

Source: Author.

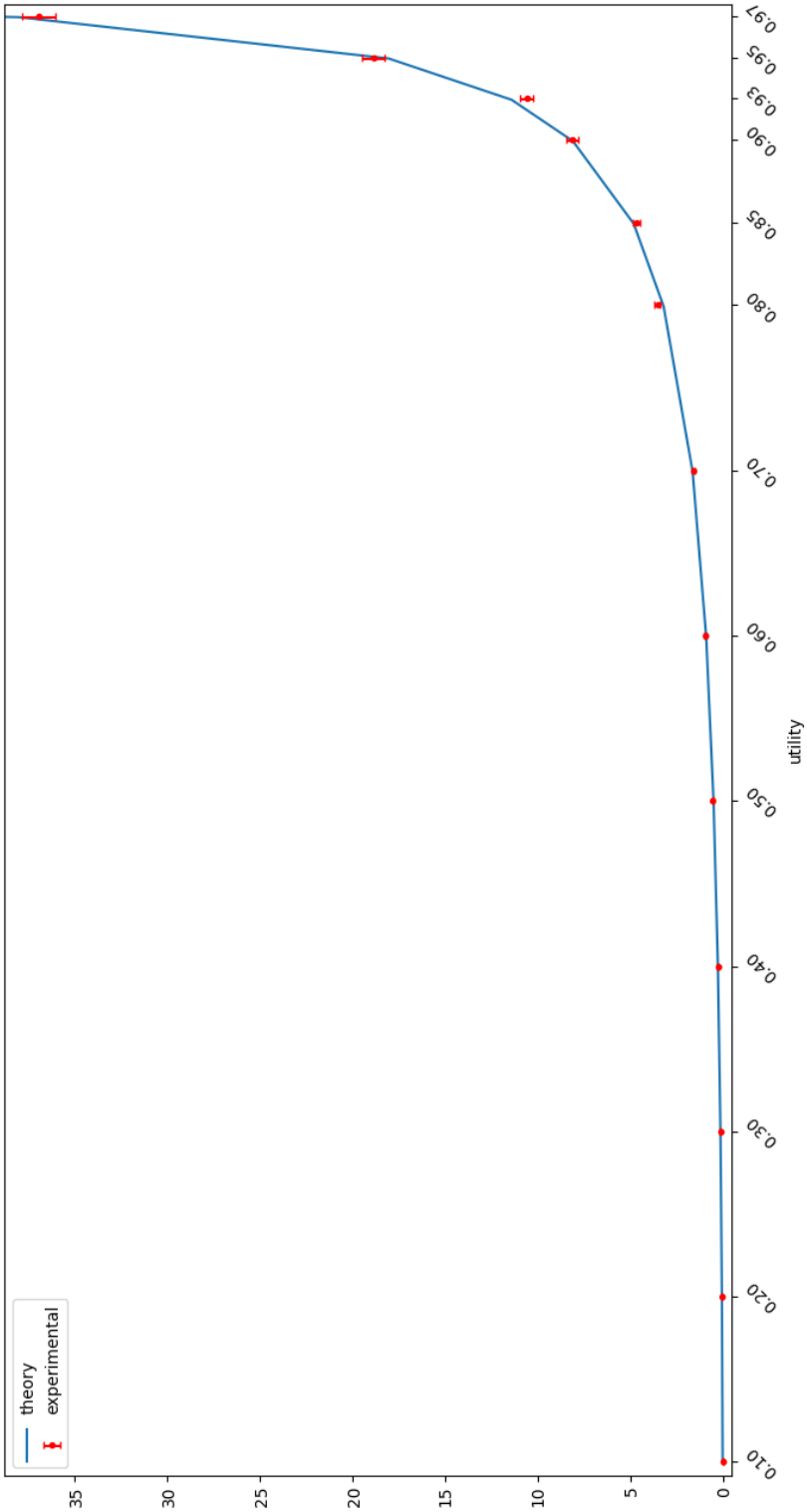


Figure 22 – Simulator M/M/1 utility zoom.

Source: Author.

distribution of M/M/1 queues. However, it is considered that the simulator follows the theoretical distribution up to the utility value of 0.97.

5.5 Conclusion

According to the analysis of the results of the experiment, the following research question can be answered. Does the fog simulation tool obey the M/M/1 queuing theory? Yes, according to the one sample student's t-tests, it was validated that the simulator can produce data following the theoretical exponential distribution of M/M/1 queue for utility values up to 0.97. Thus, studies that conduct experiments in the Kintoun simulator must guarantee utility values less than 0.97, however, this characteristic does not prohibit or limit its use.

6

R3GP Validation

In this chapter, the method used to evaluate the placement algorithms in a case study of Intelligent Transportation Systems is described. Section 6.1 presents an overview of the methodology for conducting the validation experiment. Section 6.2 shows the experimental planning with a detailed description of context selection, dependent and independent variables, hypotheses formulation, step-by-step of the experimental design and the devices and tools used as instruments for the execution of the experiment. Section 6.3 presents the details of how the experiment was carried out. Results and statistical analyzes of the data are presented in Section 6.4. Finally, the conclusion about the performance of the algorithms is presented in Section 6.5.

6.1 Methodology

The methodology of this work is based on an explanatory research, in which the optimization of a Fog Service Placement Problem (SKARLAT et al., 2017a; SKARLAT et al., 2017b) modeled as a Constraint Satisfaction Problem is addressed. Experimental evaluations of the performance of heuristic and meta-heuristic algorithms were performed in solving the placement problem to run IoT applications over a three-layered Fog Computing infrastructure, similar to what was done in (NATESHA; GUDDETI, 2021; AYOUBI; RAMEZANPOUR; KHORSAND, 2021; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020). In this study, an *in silico* experiment was conducted using a mission-critical IoT application for use in the field of Intelligent Transportation Systems.

Before defining the optimization algorithms used in the experiments described in this work, a pre-analysis of the algorithms used in the literature was performed. The criteria for choosing the algorithms were their performance in preliminary tests for solving the placement problem and also their representativeness in the literature. Modifications of the algorithms for solving the multiobjective problem with Pareto frontier were also tested, however, due to the

time criticality of the case study, the optimization algorithms with resolution via Pareto frontier were disregarded. These were some of the preliminary tested optimization algorithms: Simulated Annealing, Tabu-search, Bayesian Optimization Algorithm, Elitism-based Genetic Algorithm, NSGA-II, Greedy and Random. Finally, regarding the Genetic Algorithm crossover, the One-point Crossover strategy was tested, however, it did not achieve better performance than a Multi-point Crossover using the Greedy strategy.

Before running the validation experiment with the selected optimization algorithms, it was necessary to prepare and provision the components of the Fog architecture, the IoT services and application, the FSPP coding and the optimization algorithms. The topology of the prepared Fog infrastructure was based on a Barabasi network, as in (NEZAMI et al., 2021; BROGI et al., 2019; LERA; GUERRERO; JUIZ, 2018). The scripts for the execution of the experimental steps were implemented as described in the topic *Experimental Design* in 6.2. Thus, the experiments were organized according to the Goal Question Metric (GQM) approach proposed by (BASILI; CALDIERA; ROMBACH, 1994).

Finally, after carrying out the experiments, statistical tests were applied to the data of the results obtained for the analysis of each metric in order to compare the treatments and then evaluate the formulated hypotheses. The *Data Validation* topic in 6.4.1 explains what types of tests were used.

6.2 Experimental Planning

6.2.1 Objective Definition

The objective of the experiment was formally defined using the GQM method, proposed by (BASILI; CALDIERA; ROMBACH, 1994), as follows: **analyze** the proposed optimization algorithm for the Fog Service Placement Problem; **for the purpose of** evaluate the performance against the ϵ-Greedy, Guided Local Search, Genetic Algorithm, Hill Climbing, and First-Fit Decreasing (NATESHA; GUDDETI, 2021; NEZAMI et al., 2021; EYCKERMAN et al., 2020; LUKE, 2013); **concerning** euclidean distance of the CPU load-balancing, memory load-balancing, bandwidth load-balancing, makespan, and energy consumption gap (NEZAMI et al., 2021; YOSUF et al., 2020; DJEMAI et al., 2019); **from the point of view of** network architects, network engineers, IoT developers, ITS companies, and smart city companies; **in the context of** Fog IoT applications.

6.2.2 Planning

Context Selection

Based on (NATESHA; GUDDETI, 2021; AYOUBI; RAMEZANPOUR; KHORSAND, 2021; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020), in this chapter, this study

proposes a simulation experiment scenario. The experiment carried out using a synthetic IoT application request for placement of services in the fog computing layer. The application is a collision avoidance detection in Intelligent Transportation Systems as in (EYCKERMAN et al., 2020; MSEDDI et al., 2019; DONASSOLO et al., 2019b)

Independent Variables

The independent variables of the experiment were the number of IoT application services, the number of nodes, the resource configuration of the nodes, the network topology.

Dependent Variables

The dependent variables of the experiment were the placement planning time, the speed convergence, the objective function values makespan, energy consumption, CPU utilization load-balancing, memory utilization load-balancing, and bandwidth utilization load-balancing, and the euclidean distance of this objective functions.

Hypothesis Formulation

Adopting the GQM method, the following research questions were designed to fully cover the objectives of the work:

1. Does the proposed optimization algorithm calculate the best placement plan than those found in the literature?
2. Does the proposed optimization algorithm calculate the best placement plan faster than those found in the literature?
3. Does the proposed optimization algorithm calculate the best placement plan in less steps than those found in the literature?

In order to evaluate these questions were used the metrics described in the *Dependent Variables* topic. The question 1 was evaluated using the objective values and the euclidean distance. The question 2 was evaluated using the placement planning time. The question 3 was evaluated using the speed convergence. Thus, with the objectives and metrics defined, the following hypotheses were formulated:

H_0: all treatments have the same average value for the evaluated metric.

H_1: any treatment has a different average value for the evaluated metric.

Formally, the hypotheses can be described as:

$$H_0 : \ M_i(metric) = M_j(metric),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : \ M_i(metric) \neq M_j(metric), \ i \neq j \mid i, j \in Treatments$$

Object Selection

Since it is impossible to collect the entire population of the data monitored in the experiments, in this work, for $n_{rounds} = 500$, it was collected $n_{samples} = 50$ per round, i. e., a total of 25000 samples for each treatment. This was adopted in order to get statistical significance.

Experimental Design

The following high-level steps were performed 25000 times for each treatment running in a Fog Controller:

1. Synthesize IoT application to the Fog Controller;
2. Create synthetic IoT application for the Fog Controller;
3. Execute the treatment (algorithm) in the Fog Controller to place the IoT application services requested;
    a) The Fog Controller automatically places the services in the chosen fog compute nodes or in the cloud;
4. Collect performance metrics of the treatment execution for analysis;
    a) The performance metrics are specified in *Dependent Variables* topic.
5. After collect performance metrics 25000 times, apply appropriate statistical tests to analyze all hypotheses.

Instrumentation

For experimental setup, the following software were used: Python 3.10 with pandas 1.5.0, numpy 1.23.5, and matplotlib 3.6.2 libraries, and Ubuntu 22.04 LTS operating system. Furthermore, the following hardware setup was employed: Intel dual core x86_64 processor with 3.1 GHz clock, 8 GB of RAM, and 1 TB of HDD.

6.3 Experimental Operation

6.3.1 Preparation

In this step the simulation environment was configured. The host machine was equiped with the simulator tool implemented in Python 3.10. Before execution and evaluation of the multi-objective optimization algorithms, 50 rounds were executed to discover the best parameter

for each algorithm, for 25, 50, and 100 fog computing nodes. The search space of the parameters explored is presented in Table 9. The First-Fit Decreasing and the proposed algorithm do not need parameters.

Table 9 – Hyper-parameter search space.

Algorithm	**Parameter**	**Set of values**
ϵ-Greedy	ϵ	{0.02, 0.04, 0.06, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5}
Genetic Algorithm	population size	{6, 10, 20, 30, 40, 50, 100}
	crossover tweaker	{Greedy}
	mutation tweaker	{Bounded Uniform Convolution}
	mutation probability	{0.02, 0.04, 0.06, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5}
Guided Local Search	tweak probability	{0.02, 0.04, 0.06, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5}
	restart time	{6, 10, 20, 30, 40, 50, 100}
	beta	{0.1}
	tweaker	{Bounded Uniform Convolution}
Hill Climbing	tweak probability	{0.02, 0.04, 0.06, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5}
	tweaker	{Bounded Uniform Convolution}

Source: Author.

In order to find the hyper-parameter of each algorithm, the follow steps were applied 50 times:

1. Create the combination of the sets of all parameters for each algorithm;
2. Synthesize IoT application services to the Placement Algorithm;
3. Execute the algorithm to calculate the placement plan;
4. Execute the First-Fit Decreasing to calculate the placement plan to compare the results as a reference point;
5. Collect execution time and euclidean distance performance metrics of the algorithm execution for analysis;
6. Choose the hyper-parameter using the following sequence:
    a) Choose the parameters that calculate the placement plan in less than 300 milliseconds. This value is based on the case study scenario presented in section 7.1;
    b) Choose the best parameter that, in average, compared with First-Fit Decreasing, calculates the minimum euclidean distance.

In order to perform the hyper-parameter search and the experiment, it was used the configurations of the fog computing nodes and application service presented in Table 10. The configurations are based on the real hardware specifications and (NIKOUI et al., 2020; MAHMUD et al., 2019b).

Table 10 – Services and nodes configurations.

Fog computing nodes configuration	
Hardware model based	Raspberry Pi A+
CPU capacity	225.9 MIPS
Memory capacity	512 MB
Power idle	1.2 W
Power maximum	5.4 W
CPU usage	Random between [0%, 80%]
Memory usage	Random between [0%, 80%]

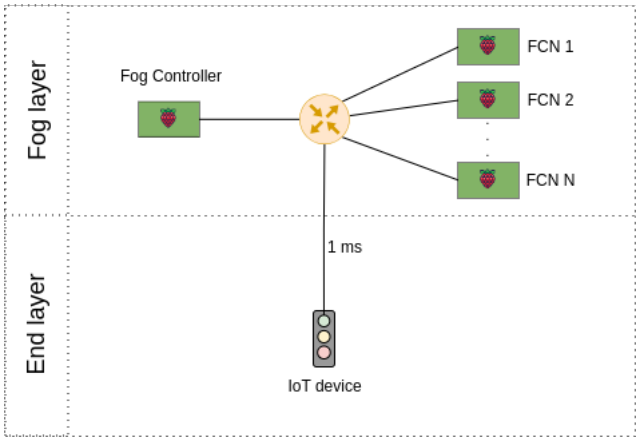
Network links configuration	
Communication latency	Random between [5 ms, 10 ms]
Bandwidth capacity	58.8 Mbps
Bandwidth usage	Random between [0%, 80%]

Application configuration	
Number of services	10
Total deadline time	600 ms

Service configuration	
CPU requirement	1 MI
Memory requirement	256 KB
Bandwidth requirement	1 Mbps
Input data size	10 KB
Output data size	10 KB
Deadline time	30 ms

Source: Author.

Figure 23 – Hyper-parameter and validation experiment topology.



Source: Author.

The topology of the hyper-parameter and the validation experiment is presented in Figure 23.

In order to execute and evaluate the multi-objective optimization algorithms, the hyper-parameters found are presented in Table 11. The B. U. C. abbreviation means Bounded Uniform Convolution.

Table 11 – Hyper-parameters.

Algorithm	**Parameter**	**25 nodes**	**50 nodes**	**100 nodes**
ϵ-Greedy	ϵ	0.06	0.04	0.06
Genetic Algorithm	population size	6	6	6
	crossover tweaker	Greedy	Greedy	Greedy
	mutation tweaker	B.U.C.	B.U.C.	B.U.C.
	mutation probability	0.2	0.2	0.1
Guided Local Search	tweak probability	0.2	0.2	0.2
	restart time	30	50	40
	beta	0.1	0.1	0.1
	tweaker	B.U.C.	B.U.C.	B.U.C.
Hill Climbing	tweaker probability	0.3	0.1	0.3
	tweaker	B.U.C.	B.U.C.	B.U.C.

Source: Author.

6.3.2 Execution

Using the hyper-parameters of the Table 11, the steps specified in Experimental Design topic in 6.2.2 were performed as described. The Ubuntu was used in the experiments without the Graphical User Interface (GUI) started, only via teletypewriter (TTY). The metrics collected were speed convergence, placement planning time, objective values (makespan, energy consumption, CPU consumption load-balancing, memory consumption load-balancing, and bandwidth consumption load-balancing) and euclidean distance of the objective values. The next subsection describes how these data were analyzed.

6.4 Results and Discussion

This section presents results of the validation of the algorithms shown in this chapter. It presents analysis of the results in order to validate the hypothesis and answer the research question formulated in subsection 6.2.2. The subsection 6.4.2 shows the values of each objective function defined to calculate the placement plan and also the euclidean distance of this results in order to evaluate the best placement plan. Subsection 6.4.3 reveals the time taken by the algorithms to calculate the placement plan. And, the subsection 6.4.4 shows how many steps the algorithms try to calculate its best placement plan. In order to compare the treatments, all the results are normalized with respect to the First-Fit Decreasing executions.

6.4.1 Data Validation

In order to validate the data and evaluate the hypotheses, the following statistical tests were used: Kolmogorov-Smirnov (KS) test, to check the normality of the data; if data is normal, apply the Bartllet test, otherwise, the Levene to test the homoscedasticity of the data; if homoscedastic data, it was used ANOVA test, to compare the mean values of the metrics obtained of all treatments. If heteroskedastic data, it was used Kruskal-Wallis test, to compare the median values of the metrics obtained of all treatments. At last, was used the Tukey's post hoc test to ranking the treatments after ANOVA test, and the Dunn's post hoc test to ranking the treatments after Kruskal-Wallis test.

6.4.2 Objective values results

Makespan

Figure 24 presents the mean distribution of the samples collected from the makespan metric. Outliers were removed and replaced by the mean of the data without outliers. Then, the Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data follow a normal distribution. Thus, Bartlett's test was applied to verify the homoscedasticity of the samples. The homoscedasticity test indicate that the variance of the treatment data is heterogeneous. Therefore, the hypotheses related to the makespan of the objective functions can be formulated in the following mathematical way:

$$H_0 : \ M_i(makespan) = M_j(makespan), \ i \neq j \mid i, j \in Treatments$$

$$H_1 : \ M_i(makespan) \neq M_j(makespan), \ i \neq j \mid i, j \in Treatments$$

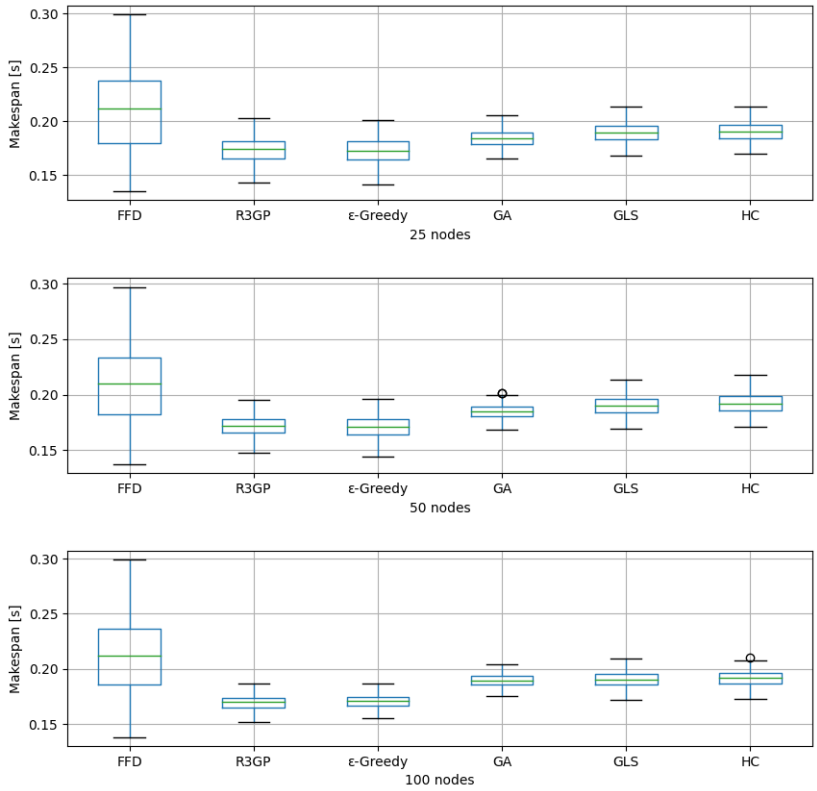
In order to evaluate these hypotheses, the Kruskal-Wallis test was applied. As a result, the calculated p-values for all scenarios (25, 50 and 100 nodes) were less than 0.025, that is, the null hypothesis H0 was rejected. Thus, there is a difference between the medians of the treatment samples.

Given the difference between medians, Dunn's test was applied to rank treatments. Table 12 shows the ranking of algorithms according to the median of the makespan metric. In addition, Figure 25 shows the graph of the means with a 95% confidence interval for the treatment data.

According to the p-value of Table 12, there is no difference between the medians of e-Greedy and the proposed algorithm (R3GP) in all scenarios. There is no difference between Hill Climbing (HC) and Guided Local Search (GLS) medians in all scenarios. In the 100-node scenario there is no difference between Hill Climbing (HC), Genetic Algorithm (GA) and Guided Local Search.

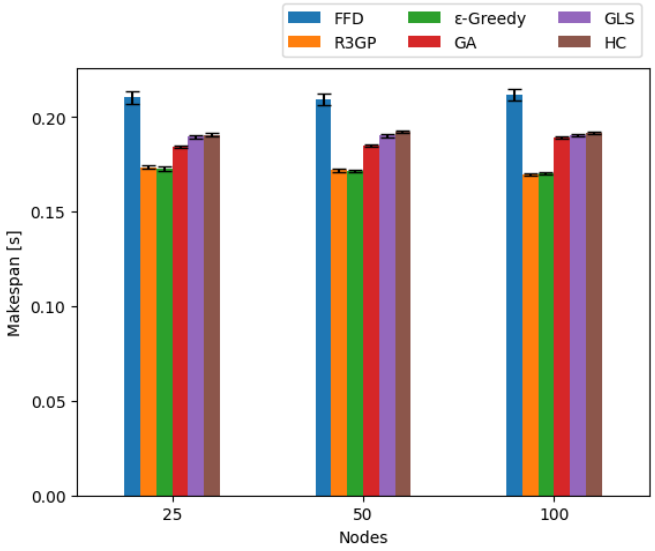
While the GA, GLS, HC and FFD tend to worsen the median minimization of the makespan value of the objective functions, the R3GP and e-Greedy algorithms tend to improve

Figure 24 – Average distribution of the makespan.



Source: Author.

Figure 25 – Average mean of the makespan.



Source: Author.

Table 12 – Makespan ranking.

25 nodes			
Rank	Algorithm	Median	p-value
1^{st}	ϵ-Greedy	0.194810	0.5169
	R3GP	0.195688	
3^{rd}	GA	0.205550	< 0.0250
4^{th}	GLS	0.211501	0.4095
	HC	0.212301	
6^{th}	FFD	0.232309	< 0.0250

50 nodes				
Rank	Algorithm	Median	p-value	
1^{st}	ϵ-Greedy	0.194558	0.7862	
	R3GP	0.195166		
3^{rd}	GA	0.207755	< 0.0250	
4^{th}	GLS	0.213122	0.055	
5^{th}	HC	0.215173		0.0580
6^{th}	FFD	0.232343		

100 nodes				
Rank	Algorithm	Median	p-value	
1^{st}	R3GP	0.193406	0.5328	
	ϵ-Greedy	0.194181		
3^{rd}	GA	0.211830	0.5328	0.3961
	GLS	0.213379		
	HC	0.214221		0.3961
6^{th}	FFD	0.233494	< 0.0250	

Source: Author.

with the increase in the number of nodes. This improvement is a relevant behavior due to the lower latency feature of Fog Computing service

Energy consumption gap

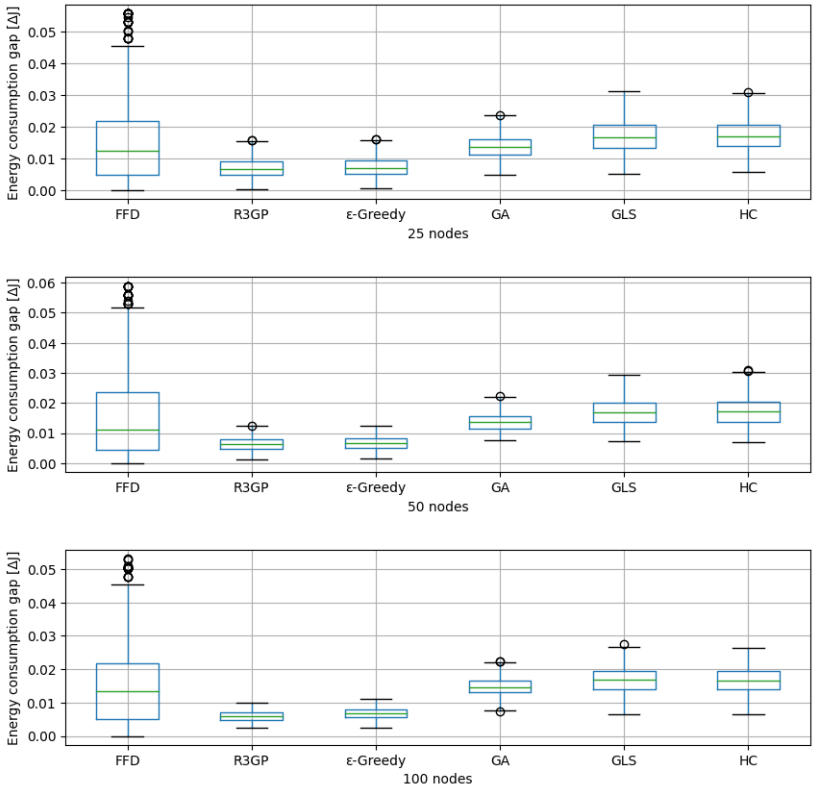
Figure 26 shows the mean distribution of the samples collected from the energy consumption metric. Outliers were removed and replaced by the mean of the data without outliers. Then, the Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data follow a normal distribution, except for the FFD. Thus, Levene's test was applied to verify the homoscedasticity of the samples. The p-values indicate that the variance of the treatment data is heterogeneous. Therefore, the hypotheses related to energy consumption can be formulated in the following mathematical way:

$$H_0 : M_i(energy\ consumption) = M_j(energy\ consumption),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : M_i(energy\ consumption) \neq M_j(energy\ consumption),\ i \neq j \mid i, j \in Treatments$$

In order to evaluate this hypothesis, the Kruskal-Wallis test was applied. As a result, the calculated p-values for all scenarios (25, 50 and 100 nodes) were less than 0.025, that is, the null hypothesis H0 was rejected. Thus, there is a difference between the medians of the treatment samples.

Figure 26 – Average distribution of the energy consumption.



Source: Author.

Given the difference between medians, Dunn's test was applied to rank treatments. Table 13 shows the ranking of algorithms according to the median of the energy consumption metric. In addition, Figure 27 shows the graph of the means with a 95% confidence interval for the treatment data.

According to the p-value of Table 13, there is no difference between the R3GP and e-Greedy medians in the 25 and 50 node scenarios. In the scenario of 100 nodes, there is a difference, making R3GP the best algorithm in minimizing energy consumption. The R3GP median differences for second and third place are 12.79% and 127.46%, respectively. Furthermore, R3GP is the only one to show a downward trend as the number of nodes increases. This feature makes the proposed algorithm potentially more suitable for Green Computing.

CPU utilization load-balancing

Figure 28 presents the mean distribution of the samples collected from the CPU load-balancing metric. Outliers were removed and replaced by the mean of the data without outliers. Then, the Kolmogorov-Smirnov normality test was applied and the p-values shown the data

Table 13 – Energy consumption ranking.

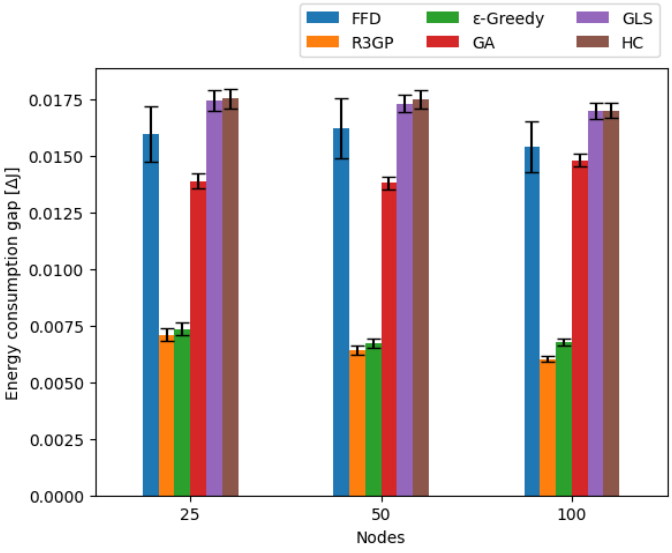
25 nodes			
Rank	Algorithm	Median	p-value
1^{st}	R3GP	0.006788	1.0000
	ϵ-Greedy	0.007049	
3^{rd}	FFD	0.012528	< 0.0250
4^{th}	GA	0.013579	
5^{th}	GLS	0.016849	1.0000
	HC	0.017030	

50 nodes			
Rank	Algorithm	Median	p-value
1^{st}	R3GP	0.006250	0.7291
	ϵ-Greedy	0.006527	
3^{rd}	FFD	0.011245	< 0.0250
4^{th}	GA	0.013602	
5^{th}	GLS	0.016895	0.8128
	HC	0.017171	

100 nodes			
Rank	Algorithm	Median	p-value
1^{st}	R3GP	0.005895	< 0.0250
2^{nd}	ϵ-Greedy	0.006649	
3^{rd}	FFD	0.013409	
4^{th}	GA	0.014680	
5^{th}	HC	0.016720	0.8652
	GLS	0.016870	

Source: Author.

Figure 27 – Average mean of the energy consumption.



Source: Author.

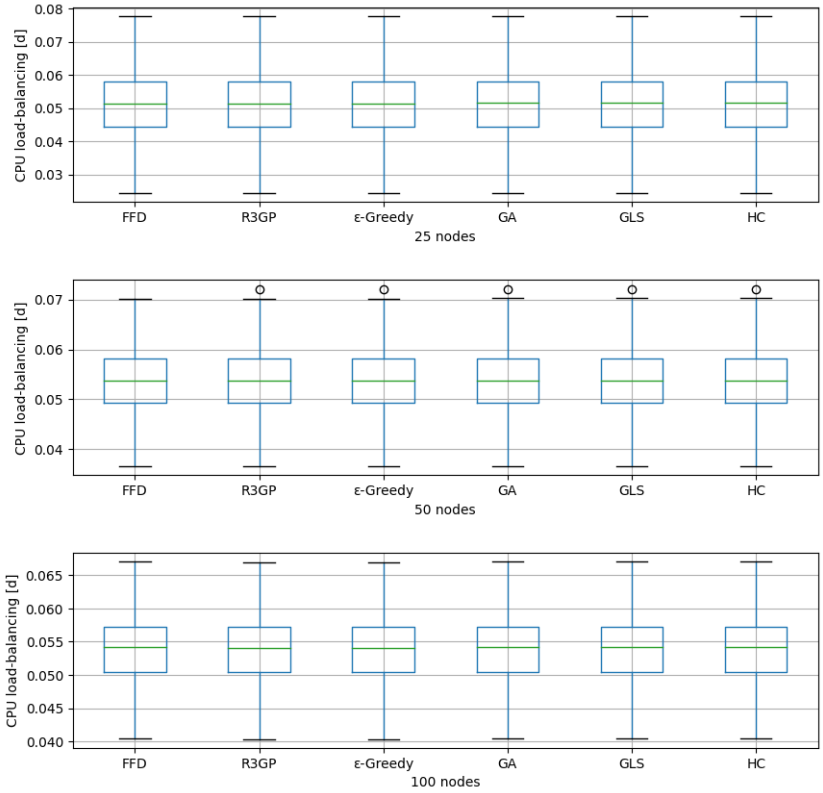
follow a normal distribution. Thus, Bartlett's test was applied to verify the homoscedasticity of the samples. The p-values of the Barlett's test indicate that the variance of the treatment data is homogeneous. Therefore, the assumptions related to CPU load-balancing can be formulated in the following mathematical way:

$$H_0 : \mu_i(CPU_{load-balancing}) = \mu_j(CPU_{load-balancing}),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : \mu_i(CPU_{load-balancing}) \neq \mu_j(CPU_{load-balancing}),\ i \neq j \mid i, j \in Treatments$$

In order to evaluate these hypotheses, the ANOVA test was applied. As a result, the calculated p-values for all scenarios (25, 50 and 100 nodes) were equal to 0.9, that is, the null hypothesis H0 was not rejected. Thus, we have that there is no difference between the means of the samples of the treatments.

Figure 28 – Average distribution of the CPU utilization load-balancing.



Source: Author.

Given the equality of means, only an ordering of the mean values was made. Table 14 shows the ordering of algorithms according to the average CPU load-balancing metric. In addition, Figure 29 shows the graph of the means with a 95% confidence interval for the treatment data.

Memory utilization load-balancing

Figure 30 shows the mean distribution of the samples collected from the memory load-balancing metric. Outliers were removed and replaced by the mean of the data without outliers.

Table 14 – CPU load-balancing mean.

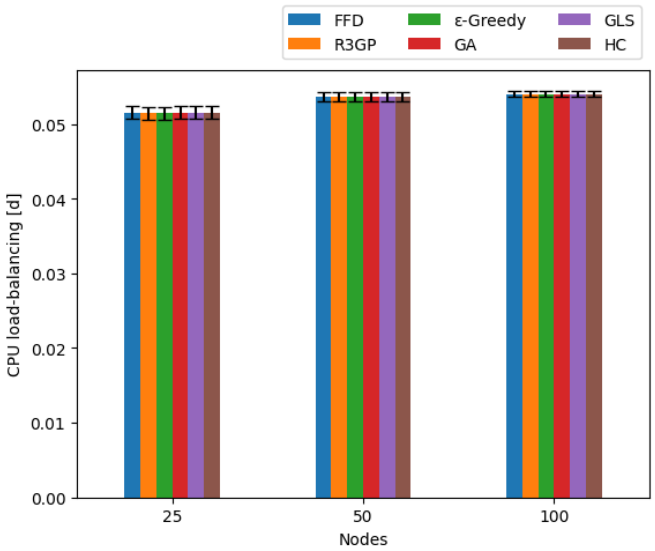
25 nodes	
Algorithm	Mean
R3GP	0.051478
ϵ-Greedy	0.051480
GA	0.051514
GLS	0.051530
HC	0.051531
FFD	0.051535

50 nodes	
Algorithm	Mean
FFD	0.053659
R3GP	0.053667
ϵ-Greedy	0.053668
GA	0.053687
GLS	0.053695
HC	0.053696

100 nodes	
Algorithm	Mean
R3GP	0.054024
ϵ-Greedy	0.054025
GA	0.054036
GLS	0.054039
HC	0.054039
FFD	0.054040

Source: Author.

Figure 29 – Average mean of the CPU utilization load-balancing.



Source: Author.

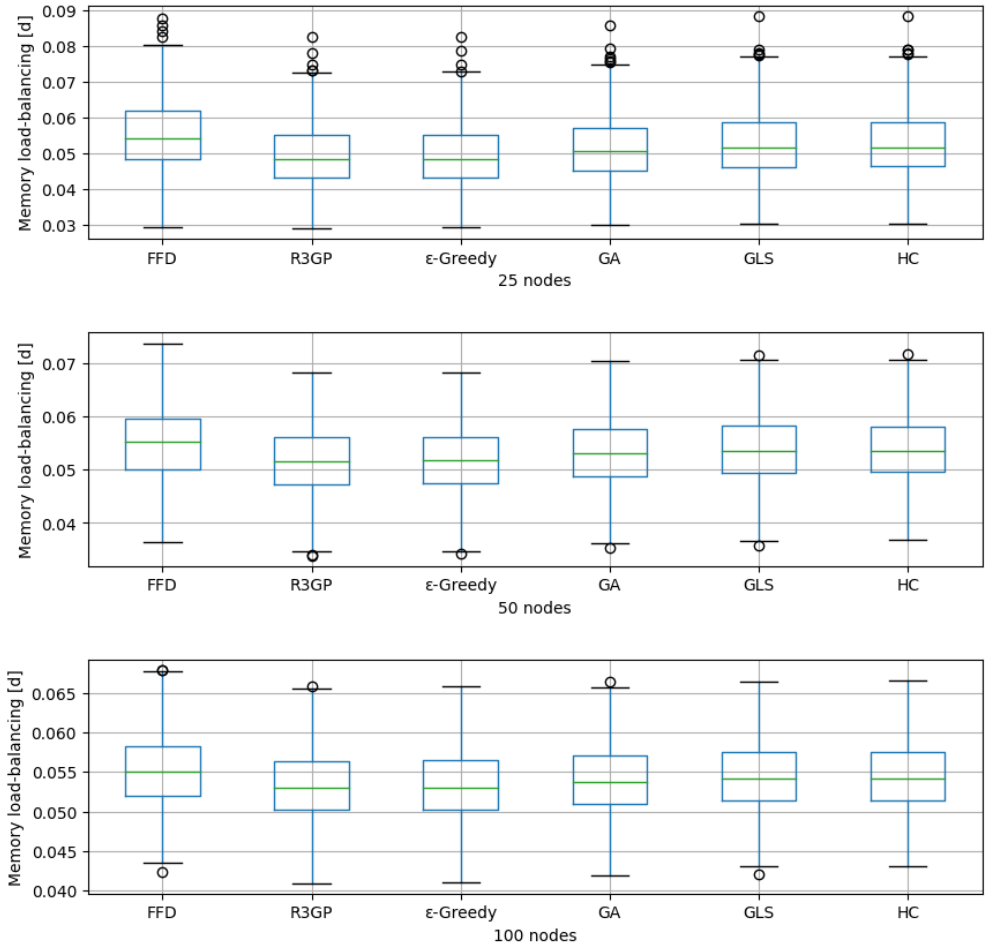
Then, the Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data follow a normal distribution. Thus, Bartlett's test was applied to verify the homoscedasticity of the samples. The homoscedasticity test indicate that the variance of the treatment data is homogeneous. Therefore, the hypotheses related to memory load-balancing can be formulated in the following mathematical way:

$$H_0 : \mu_i(MEM_{load-balancing}) = \mu_j(MEM_{load-balancing}),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : \mu_i(MEM_{load-balancing}) \neq \mu_j(MEM_{load-balancing}),\ i \neq j \mid i, j \in Treatments$$

In order to evaluate this hypothesis, the ANOVA test was applied. As a result, the calculated p-values for all scenarios (25, 50 and 100 nodes) were less than 0.025, that is, the null hypothesis H0 was rejected. Thus, we have that there is a difference between the means of the samples of the treatments.

Figure 30 – Average distribution of the memory utilization load-balancing.



Source: Author.

Given the difference between means, Tukey's test was applied to rank the treatments. Table 15 shows the ranking of algorithms according to the average memory load-balancing metric. In addition, Figure 31 shows the graph of the means with a 95% confidence interval for the treatment data.

According to Tukey's test, R3GP and e-Greedy outperform GLS, HC and FFD, as the p-values reject the null hypothesis in the pairwise evaluation of treatments between these two groups. Overall, GA, GLS, and HC have average performance, while FFD has the worst performance in memory load-balancing.

Bandwidth utilization load-balancing

Figure 32 presents the mean distribution of the samples collected from the bandwidth load-balancing metric. Outliers were removed and replaced by the mean of the data without outliers. Then, the Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data follow a normal distribution. Thus, Bartlett's test was applied to verify the

Table 15 – Memory load-balancing ranking.

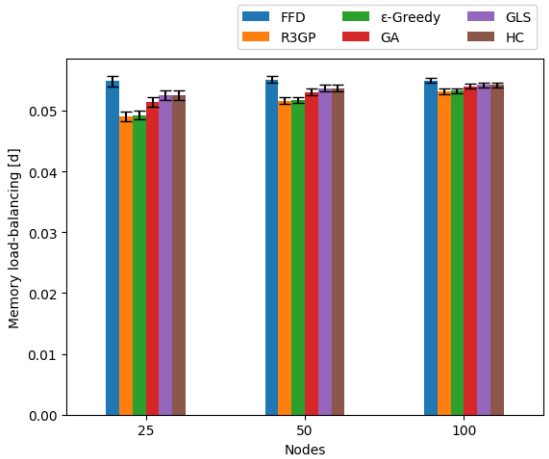
25 nodes			
Rank	Algorithm	Mean	p-value
1^{st}	R3GP	0.049129	0.9000
	ϵ-Greedy	0.049302	
3^{rd}	GA	0.051395	0.3847,
	GLS	0.052530	0.3812,
	HC	0.052534	0.9000
6^{th}	FFD	0.054879	< 0.0250

50 nodes			
Rank	Algorithm	Mean	p-value
1^{st}	R3GP	0.051646	0.9000
	ϵ-Greedy	0.051761	
3^{rd}	GA	0.053084	0.6287,
	GLS	0.053701	0.5857,
	HC	0.053731	0.9000
6^{th}	FFD	0.055161	< 0.0250

100 nodes				
Rank	Algorithm	Mean	p-value	
1^{st}	R3GP	0.053189	0.9000,	
	ϵ-Greedy	0.053282	0.040394,	
3^{rd}	GA	0.054050	0.095247	
4^{th}	GLS	0.054258	0.1294,	0.9000
	HC	0.054278	0.1523	
6^{th}	FFD	0.054988		

Source: Author.

Figure 31 – Average mean of the memory utilization load-balancing.



Source: Author.

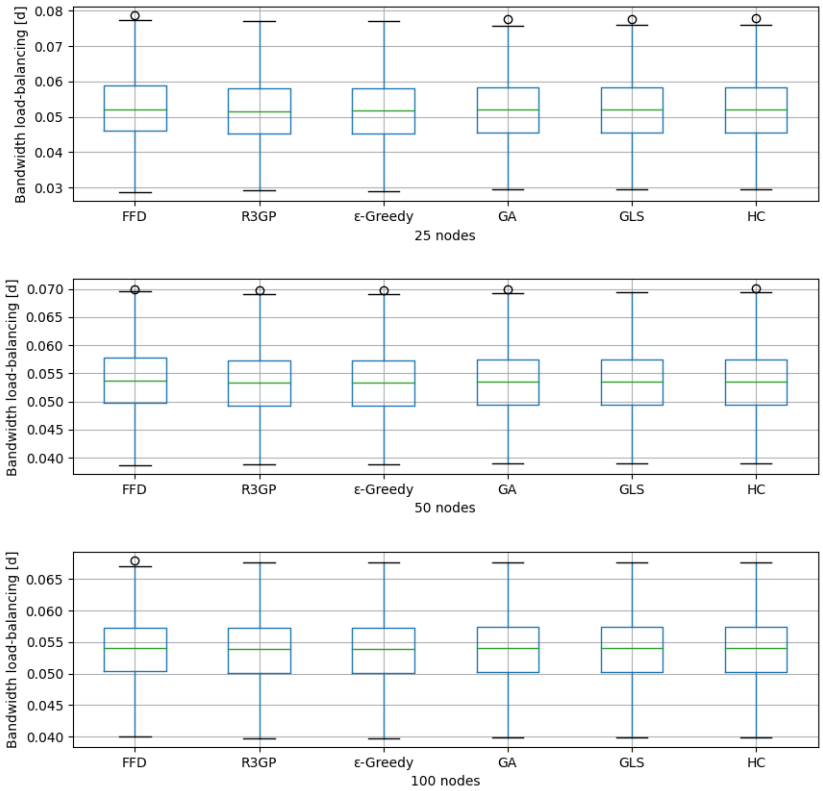
homoscedasticity of the samples. The homoscedasticity test indicate that the variance of the treatment data is homogeneous. Therefore, the assumptions related to bandwidth load-balancing can be formulated in the following mathematical way:

$$H_0 : \mu_i(BW_{load-balancing}) = \mu_j(BW_{load-balancing}),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : \mu_i(BW_{load-balancing}) \neq \mu_j(BW_{load-balancing}),\ i \neq j \mid i, j \in Treatments$$

In order to evaluate these hypotheses, the ANOVA test was applied. As a result, the calculated p-values for all scenarios (25, 50 and 100 nodes) were equal to 0.9, that is, the null hypothesis H0 was not rejected. Thus, we have that there is no difference between the means of the samples of the treatments.

Figure 32 – Average distribution of the bandwidth utilization load-balancing.



Source: Author.

Given the equality of means, only an ordering of the mean values was made. Table 16 shows the ordering of the algorithms according to the average of the bandwidth load-balancing metric. In addition, Figure 33 shows the graph of the means with a 95% confidence interval for the treatment data.

Euclidean distance of the objective values

Figure 34 presents the mean distribution of the samples collected from the Euclidean distance metric of the objective functions. Outliers were removed and replaced by the mean of the

Table 16 – Bandwidth load-balancing mean.

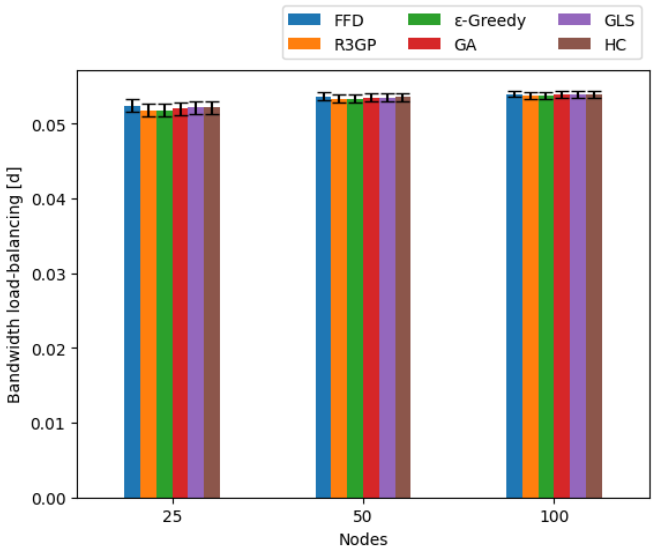
25 nodes	
Algorithm	Mean
R3GP	0.051773
ϵ-Greedy	0.051776
GA	0.052021
GLS	0.052149
HC	0.052160
FFD	0.052398

50 nodes	
Algorithm	Mean
R3GP	0.053316
ϵ-Greedy	0.053320
GA	0.053472
GLS	0.053499
HC	0.053545
FFD	0.053671

100 nodes	
Algorithm	Mean
R3GP	0.053797
ϵ-Greedy	0.053805
GA	0.053900
GLS	0.053917
HC	0.053920
FFD	0.053994

Source: Author.

Figure 33 – Average mean of the bandwidth utilization load-balancing.



Source: Author.

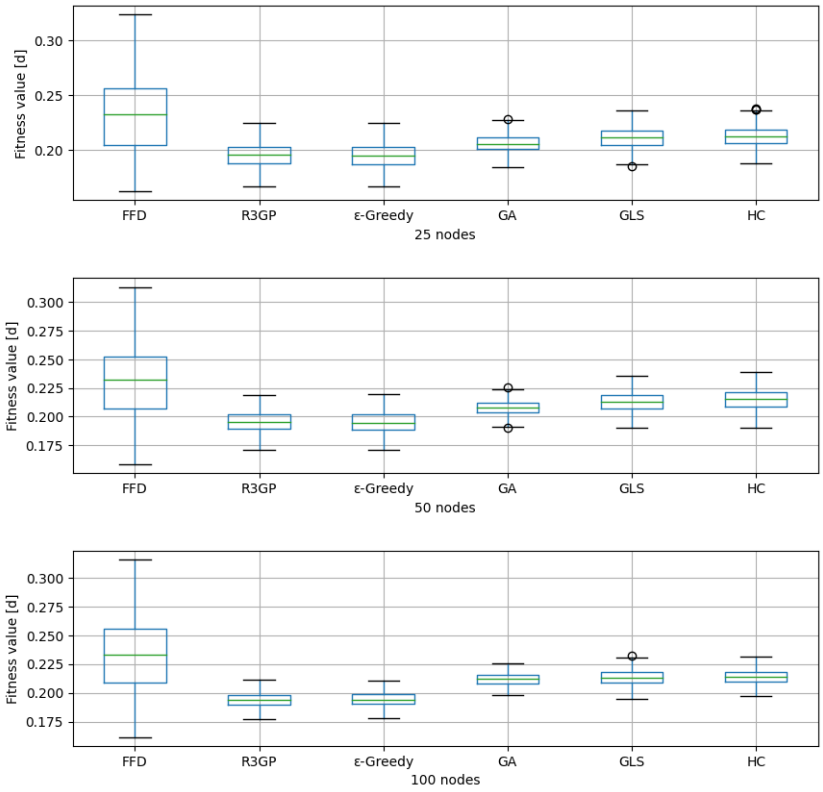
data without outliers. Then, the Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data follow a normal distribution. Thus, Bartlett's test was applied to verify the homoscedasticity of the samples. The p-values of the Barlett's test shown the variance of the treatment data is heterogeneous. Therefore, the hypotheses related to the Euclidean distance of the objective functions can be formulated in the following mathematical way:

$$H_0 : M_i(distance_{euclidean}) = M_j(distance_{euclidean}),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : M_i(distance_{euclidean}) \neq M_j(distance_{euclidean}),\ i \neq j \mid i, j \in Treatments$$

In order to evaluate these hypotheses, the Kruskal-Wallis test was applied. As a result, the calculated p-values for all scenarios (25, 50 and 100 nodes) were less than 0.025, that is, the null hypothesis H0 was rejected. Thus, there is a difference between the medians of the treatment samples.

Figure 34 – Average distribution of the euclidean distance of the objective values.



Source: Author.

Given the difference between medians, Dunn's test was applied to rank treatments. Table 17 shows the ranking of algorithms according to the median of the Euclidean distance metric of the objective functions. In addition, Figure 35 shows the graph of the means with a 95% confidence interval for the treatment data.

According to the p-value of Table 17, there is no difference between the medians of e-Greedy and the proposed algorithm (R3GP) in all scenarios. There is no difference between Hill Climbing (HC) and Guided Local Search (GLS) medians in all scenarios. Also, there is no difference between Genetic Algorithm (GA) and Guided Local Search in the 100 node scenario, however, there is difference between GA and HC.

While the GA, GLS, HC and FFD tend to worsen the median minimization of the Euclidean distance value of the objective functions, the R3GP and e-Greedy algorithms tend to improve with the increase in the number of nodes. This improvement in FSPP minimization with the increase in the number of nodes is a relevant behavior due to the scalability characteristic of Fog Computing, which can reach tens of thousands of nodes.

Table 17 – Euclidean distance ranking.

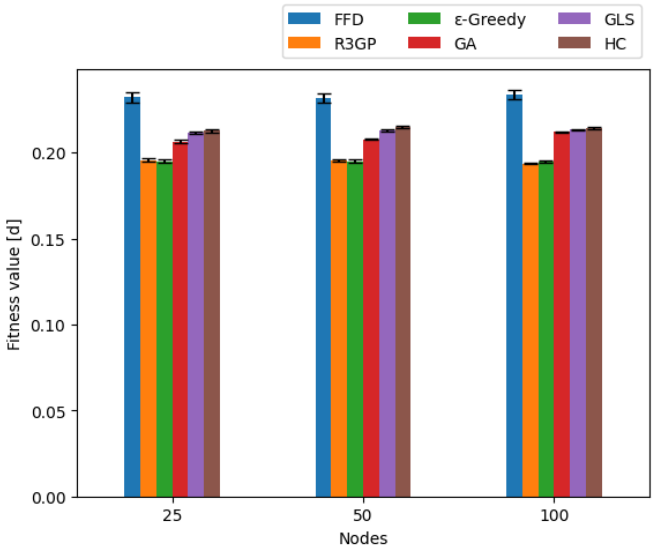
25 nodes			
Rank	Algorithm	Median	p-value
1^{st}	ϵ-Greedy	0.194810	0.5769
	R3GP	0.195688	
3^{rd}	GA	0.205550	< 0.0250
4^{th}	GLS	0.211501	0.4440
	HC	0.212301	
6^{th}	FFD	0.232309	< 0.0250

50 nodes			
Rank	Algorithm	Median	p-value
1^{st}	ϵ-Greedy	0.194558	0.7965
	R3GP	0.195166	
3^{rd}	GA	0.207755	< 0.0250
4^{th}	GLS	0.213122	0.04670
5^{th}	HC	0.215173	
6^{th}	FFD	0.232343	< 0.0250

100 nodes			
Rank	Algorithm	Median	p-value
1^{st}	R3GP	0.193406	0.3961
	ϵ-Greedy	0.194181	
3^{rd}	GA	0.211830	0.3198
4^{th}	GLS	0.213379	0.3961
5^{th}	HC	0.214221	
6^{th}	FFD	0.233494	< 0.0250

Source: Author.

Figure 35 – Average mean of the euclidean distance of the objective values.



Source: Author.

6.4.3 Placement planning time results

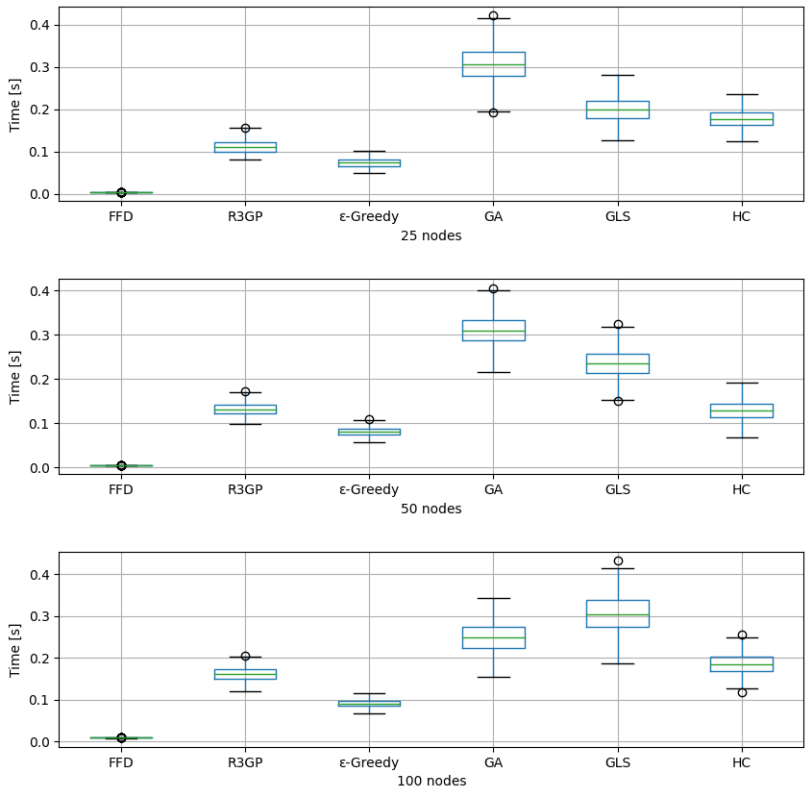
Figure 36 shows the distribution of the average of the samples collected from the placement planning time metric. Outliers were removed and replaced by the mean of the data without outliers. Then, the Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data follow a normal distribution, except for the FFD. Thus, Levene's test was applied to verify the homoscedasticity of the samples. The homoscedasticity test indicate that the variance of the treatment data is heterogeneous. Therefore, the hypotheses related to the placement planning time can be formulated in the following mathematical way:

$$H_0 : \ M_i(t_{placement}) = M_j(t_{placement}), \ i \neq j \mid i, j \in Treatments$$

$$H_1 : \ M_i(t_{placement}) \neq M_j(t_{placement}), \ i \neq j \mid i, j \in Treatments$$

In order to evaluate this hypothesis, the Kruskal-Wallis test was applied. As a result, the calculated p-values for all scenarios (25, 50 and 100 nodes) were less than 0.025, that is, the null hypothesis H0 was rejected. Thus, there is a difference between the medians of the treatment samples.

Figure 36 – Average distribution of the placement planning time.



Source: Author.

Given the difference between medians, Dunn's test was applied to rank treatments. Table 18 shows the ranking of algorithms according to the median of the placement planning time metric. In addition, Figure 37 shows the graph of the means with a 95% confidence interval for the treatment data.

Table 18 – Placement planning time ranking.

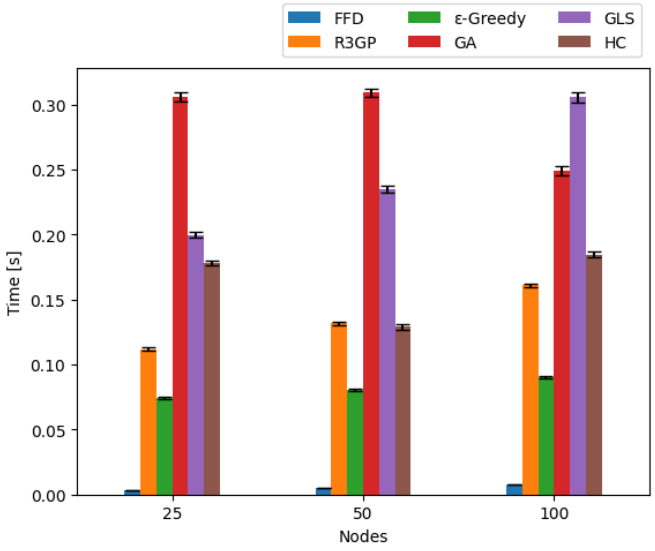
25 nodes			
Rank	Algorithm	Median	p-value
1^{st}	FFD	0.003194	< 0.0250
2^{nd}	ϵ-Greedy	0.073300	
3^{rd}	R3GP	0.109914	
4^{th}	HC	0.177054	
5^{th}	GLS	0.200069	
6^{th}	GA	0.305593	

50 nodes			
Rank	Algorithm	Median	p-value
1^{st}	FFD	0.004790	< 0.0250
2^{nd}	ϵ-Greedy	0.080668	
3^{rd}	HC	0.128197	0.3240
	R3GP	0.131053	
5^{th}	GLS	0.234835	< 0.0250
6^{th}	GA	0.309359	

100 nodes			
Rank	Algorithm	Median	p-value
1^{st}	FFD	0.008011	< 0.0250
2^{nd}	ϵ-Greedy	0.090295	
3^{rd}	R3GP	0.160279	
4^{th}	HC	0.184194	
5^{th}	GA	0.249473	
6^{th}	GLS	0.303650	

Source: Author.

Figure 37 – Average mean of the placement planning time.



Source: Author.

According to the p-value of Table 18, there is no difference between the medians of Hill Climbing (HC) and the proposed algorithm (R3GP) in the 50-node scenario. The drop in the Hill Climbing placement plan time can be explained by the objective function values that indicate that the algorithm may have fallen into local minima in the 50-nodes scenario. Also, the First-Fit Decreasing (FFD) has the best placement plan time performance. The FFD is at least 11 times faster than the others, however, the difference decreases with the increase in the number of nodes. Considering the confidence interval, the algorithms calculate the placement plan in up to 300 milliseconds in the Python language.

6.4.4 Speed convergence results

Figure 38 presents the mean distribution of the samples collected from the speed convergence metric, except for the First-Fit Decreasing, as it is not applicable. Outliers were removed and replaced by the mean of the data without outliers. Then, the Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data follow a normal distribution. Thus, Bartlett's test was applied to verify the homoscedasticity of the samples. The homoscedasticity test indicate that the variance of the treatment data is heterogeneous. Therefore, the hypotheses related to the speed convergence can be formulated in the following mathematical way:

$$H_0 : M_i(convergence) = M_j(convergence),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : M_i(convergence) \neq M_j(convergence),\ i \neq j \mid i, j \in Treatments$$

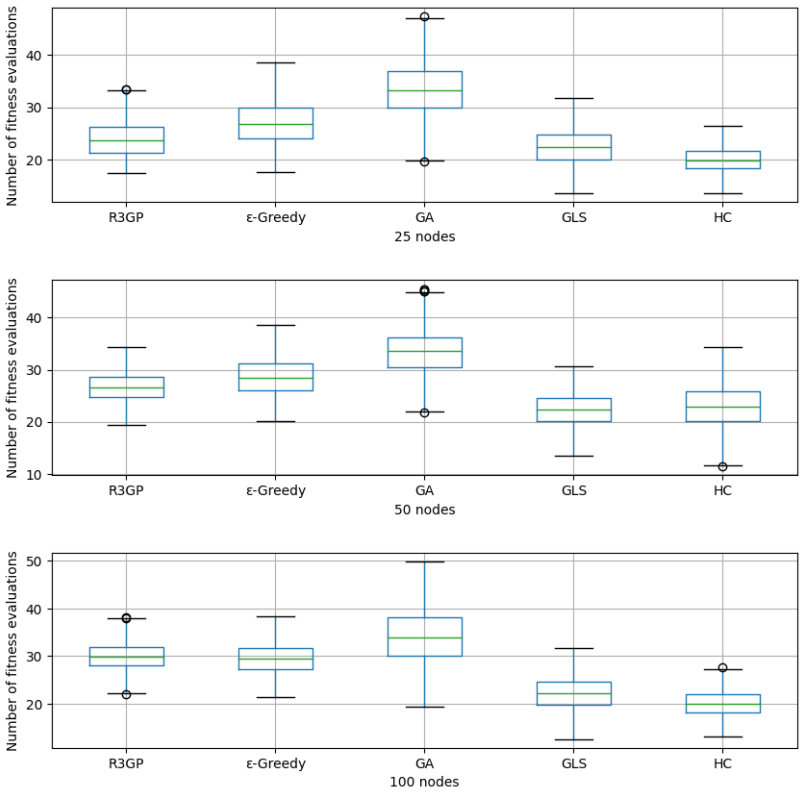
In order to evaluate this hypothesis, the Kruskal-Wallis test was applied. As a result, the calculated p-values for all scenarios (25, 50 and 100 nodes) were less than 0.025, that is, the null hypothesis H0 was rejected. Thus, there is a difference between the medians of the treatment samples.

Given the difference between medians, Dunn's test was applied to rank treatments. Table 19 shows the ranking of algorithms according to the median of the speed convergence metric. In addition, Figure 39 shows the graph of the means with a 95% confidence interval for the treatment data.

According to the p-value of Table 19, there is no difference between the e-Greedy medians and the proposed algorithm (R3GP) in the 100-nodes scenario. Unlike the others, the increase in R3GP speed convergence may indicate that the greater the number of nodes, the greater the number of optimal solutions that the algorithm can explore in relation to the others.

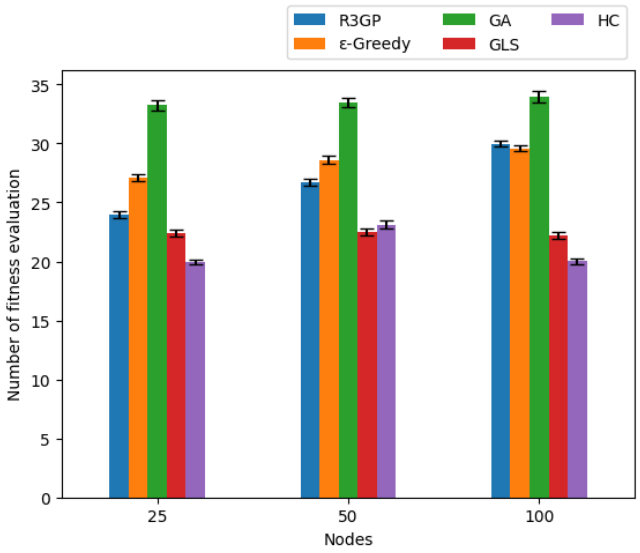
Regarding Hill Climbing, the increase in speed convergence contrasts with the decrease in placement plan time in the 50-nodes scenario. This behavior can be explained by the use of the tweaker that randomly selects the components of the solution that are adjusted. The same tweaker is used in the GLS which has an oscillatory behavior as shown in the graph.

Figure 38 – Average distribution of the speed convergence.



Source: Author.

Figure 39 – Average mean of the speed convergence.



Source: Author.

Table 19 – Speed convergence ranking.

25 nodes			
Rank	Algorithm	Median	p-value
1^{st}	HC	19.820000	
2^{nd}	GLS	22.403878	
3^{rd}	R3GP	23.639243	< 0.0250
4^{th}	ϵ-Greedy	26.870000	
5^{th}	GA	33.210642	

50 nodes			
Rank	Algorithm	Median	p-value
1^{st}	GLS	22.416413	
2^{nd}	HC	22.960000	
3^{rd}	R3GP	26.620000	< 0.0250
4^{th}	ϵ-Greedy	28.510204	
5^{th}	GA	33.600000	

100 nodes			
Rank	Algorithm	Median	p-value
1^{st}	HC	19.969796	< 0.0250
2^{nd}	GLS	22.200000	
3^{rd}	ϵ-Greedy	29.418367	0.2389
	R3GP	29.797755	
5^{th}	GA	33.840000	< 0.0250

Source: Author.

6.5 Conclusion

In general, the Rotation-Guided Greedy Genetic Particle (R3GP) algorithm, proposed in this work, performed similarly to e-Greedy, outperforming the other algorithms in most metrics, except for placement plan time and speed convergence in which it obtained median values, placing third in both. However, in the placement time metric, R3GP performed around two to three times faster than enough for the execution of the case study. Regarding speed convergence, although Hill Climbing (HC) and Guided Local Search (GLS) were in the first two places, they had the worst performances in the optimization of the Fog Service Placement Problem (FSPP), that is, R3GP despite running a few more steps, was due to the better way of discovering new optimal solutions. Finally, the highlight of the proposed algorithm was the optimization of the energy consumption metric, in which it surpassed all compared algorithms. Its robustness to the increase in the number of nodes is a strong indication of its effectiveness for Green Computing in highly scalable distributed environments, as in the case of Fog Computing.

Regarding the threats to the validity of this experimental validation, the implementation of algorithms based on pseudo-algorithms is considered a threat to the validity of construction, as the interpretation or the pseudo-algorithm itself can favor some aspects of the treatments. As a

threat to internal validity, the implementation of the First-Fit Decreasing service ordering strategy may not have been adequate. The strategy adopted was the ordering of services considering the resource requirements lexicographically and not based on the Euclidean distance of the requirements. Another threat to internal validity was the coding of ϵ-Greedy. In the validation experiment, an algorithm with selection of random solutions and component choice strategy was executed through ϵ-Greedy. As a threat to external validity we have the number of algorithms compared with the proposed algorithm. In order to mitigate this bias, the most used algorithms in the literature were chosen. Finally, in order to mitigate the threat to the validity of the conclusion, samples were collected that were large enough to apply statistical tests to validate the formulated hypotheses.

Finally, given the evidence, the following research questions are answered as follows:

1. Does the proposed optimization algorithm calculate the best placement plan than those found in the literature? Yes, despite being tied with e-Greedy in the Euclidean distance, makespan, CPU load-balancing, memory load-balancing and bandwidth load-balancing metrics, R3GP surpasses it in the energy consumption metric.

2. Does the proposed optimization algorithm calculate the best placement plan faster than those found in the literature? No, the fastest algorithm was First-Fit Decreasing. R3GP had an average performance, taking third place.

3. Does the proposed optimization algorithm calculate the best placement plan in less steps than those found in the literature? No, the fastest algorithms were Hill Climbing and GLS, however, this metric was not a determining factor for the success of the algorithms.

7 Case Study

This chapter presents the method used to evaluate the placement algorithms in the case study scenario described in section 7.1. Section 7.2 presents an overview of the methodology for conducting the validation experiment. Section 7.3 shows the experimental planning with a detailed description of context selection, dependent and independent variables, hypotheses formulation, step-by-step of the experimental design and the devices and tools used as instruments for the execution of the experiment. Section 7.4 presents the details of how the experiment was carried out. Results and statistical analyzes of the data are presented in Section 7.5. Finally, the conclusion about the performance of the algorithms is presented in Section 7.6.

7.1 Description

In order to evaluate and demonstrate the applicability of the optimization solutions of the FSPP, this section presents the motivation scenario that guided this work. The case study is inspired in (EYCKERMAN et al., 2020) and (MSEDDI et al., 2019) that compare optimization algorithms to solve the FSPP to avoid pedestrian accidents in a smart road system.

According to World Health Organization (WHO) in (ORGANIZATION et al., 2023), around 310 500 pedestrians in the world where killed in road traffic crashes in 2016. This represents 23% of the global road traffic fatalities. The agency also report that 29% are car occupants, 28% are motorized 2- or 3-wheels, and 3% are cyclists. The remaining 17% were categorized as other types of accidents or unspecified. Among the factors that cause the deaths there are alcohol impairment, driver distraction, and driver fatigue.

Depending of the status, the driver's reaction time to the security risk scenario varies (ČULÍK; KALAŠOVÁ; ŠTEFANCOVÁ, 2022) as shown in Table 20. For example, supposing the driver is indisposed, driving at 60 km/h of speed, and receives a security risk stimulus, the reaction time can be up to 2.4 seconds, consequently, the distance traveled by the car can be up to

40 meters. The higher is the reaction time, the higher is the distance traveled.

Table 20 – Driver's reaction time for different conditions.

		Traveled distance [m]	
Driver's status	**Reaction time [s]**	**30 km/h**	**60 km/h**
Attentive, focused, awaiting stimulus, and ready to brake	0.6 - 0.7	5.0 - 5.8	10.0 - 11.7
Attentive, but does not expect a stimulus	0.7 - 0.9	5.8 - 7.5	11.7 - 15.0
Attention focused on other activities related to driving (driving, preventing, sidewalk observation)	1.0 - 1.2	8.3 - 10.0	16.7 - 20.0
Inattentive (having fun with the passenger)	1.4 - 1.8	11.7 - 15.0	23.3 - 30.0
Indisposed (alcohol, illness, fatigue)	1.6 - 2.4	13.3 - 20.0	26.7 - 40.0

Source: Adapted from (ČULÍK; KALAŠOVÁ; ŠTEFANCOVÁ, 2022).

Beyond the human condition, another factor that influences the distance traveled by the car is the road condition. Table 21 shows the stopping distance, in meters, for a car in different speeds and road types. For example, a car traveling at 60 km/h on a paved road, after braking, only stops after 34 meters. Furthermore, if the paved road is wet, the vehicle travels another 6 meters, *i.e.* 40 meters in total. (MOHAMED et al., 2022).

Table 21 – Stopping distance for different road types and car speed.

Speed / **Road type**	**30 km/h**	**40 km/h**	**50 km/h**	**60 km/h**	**70 km/h**	**80 km/h**
Dry asphalt	12.2 m	18 m	25 m	32 m	40 m	50 m
Wet asphalt	13.4 m	20 m	28 m	37 m	47 m	58 m
Dry pavement	12.8 m	19 m	26 m	34 m	44 m	54 m
Wet pavement	14.2 m	22 m	30 m	40 m	52 m	64 m

Source: Adapted from (MOHAMED et al., 2022).

According to the Brazilian Traffic Code[1], the maximum speed that the vehicle can reach on an arterial road in an urban area is 60 km/h. Given this law, the IoT application of the use case was modeled to perform the placement of services in order to avoid vehicle collisions at a maximum of 60 km/h against other vehicles, pedestrians, animals, or objects. In addition, the worst driver status scenario was considered, that is, indisposed, as specified in Table 20. Therefore, assuming that the road condition is paved and wet, the application deadline time is calculated as follows.

Given that:

$$car\ speed = 60\ km/h = 16.67\ m/s$$

$$driver\ reaction = 2.4\ s$$

[1] Available in: <https://www.planalto.gov.br/ccivil_03/leis/l9503compilado.htm>

$$distance_{before}^{reaction} = car\ speed \times driver\ reaction$$
$$distance_{before}^{reaction} = 16.67\ [m/s] \times 2.4\ [s] = 40\ m$$
$$distance_{after}^{reaction} = 40\ m$$

Thus, assuming that the application request happens with the vehicle 100 meters from the target, the application response time must be:

$$deadline_{time}^{app} = \frac{distance_{app}^{request} - (distance_{before}^{reaction} + distance_{after}^{reaction})}{car\ speed}$$
$$deadline_{time}^{app} = \frac{100 - (40 + 40)}{16.67} \frac{[m]}{[m/s]} = 1.2\ s$$
$$deadline_{time}^{app} = 1.2\ s$$

Given this scenario, for the traffic collision avoidance application case study, it was assumed in the experiment that the application deadline time is 1.2 seconds.

7.2 Methodology

The methodology of this work is based on an explanatory research, in which the optimization of a Fog Service Placement Problem modeled as a Constraint Satisfaction Problem in (SKARLAT et al., 2017a; SKARLAT et al., 2017b) is addressed. Experimental evaluations of the performance of heuristic and meta-heuristic algorithms were performed in solving the placement problem to run IoT applications over a Fog Computing infrastructure, similar to what was done in (NATESHA; GUDDETI, 2021; AYOUBI; RAMEZANPOUR; KHORSAND, 2021; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020). In this study, an *in silico* experiment was conducted, simulated in the Kintoun simulator (Chapter 5), using two real-time and mission-critical IoT applications in the field of Intelligent Transportation Systems.

Before running the experiments, it was necessary to prepare and provision the Fog architecture components, IoT services and applications, FSPP coding and optimization algorithms. The topology of the prepared Fog infrastructure was based on a Barabasi network, as well as (NEZAMI et al., 2021; BROGI et al., 2019; LERA; GUERRERO; JUIZ, 2018). The experiment scripts necessary for the execution of the experimental steps were implemented, as described in the topic Experimental Design in 7.3.2. Thus, the experiments were organized according to the Goal Question Metric (GQM) approach proposed by (BASILI; CALDIERA; ROMBACH, 1994).

As performance metrics, application response time, makespan, placement plan time, fog utilization, application deadline timeout and energy consumption were collected. The frequency of collection for this last metric was the average of 10 samples in 1 second. The other metrics were collected as needed. As treatments of the experiment, the proposed algorithm (R3GP),

First-Fit Decreasing, ϵ-Greedy and the direct deployment strategy in the cloud called Cloud-only (NATESHA; GUDDETI, 2021; NEZAMI et al., 2021; YOUSEFPOUR et al., 2019) were used. For each of the algorithms, a 1-hour experiment was performed.

In order to mitigate the threats to internal validity exposed in section 6.5, the ordering strategy used by First-Fit Decreasing was modified so that, instead of carrying out a lexicographical ordering by the resources requested by the services, it performs the ordering according to the Euclidean distance of the resources requested by the services. Thus, the strategy is paired with the strategy adopted in the evaluation of the objective function and cost function of the metaheuristic algorithms. In addition, the random ϵ-Greedy was disregarded and replaced by the pure ϵ-Greedy, that is, only the ϵ-Greedy acting in the selection of components at once, without iteration with random solution selection.

Finally, after carrying out the experiments, statistical tests were applied to the data of the results obtained for the analysis of each metric in order to compare the treatments and then evaluate the formulated hypotheses. The *Data Validation* in 7.5.1 explains what types of tests were used.

7.3 Experimental Planning

7.3.1 Objective Definition

The objective of the experiment was formally defined using the GQM method, proposed by (BASILI; CALDIERA; ROMBACH, 1994), as follows: **analyze** the proposed optimization algorithm for the Fog Service Placement Problem; **for the purpose of** evaluate the performance against the First-Fit Decreasing, ϵ-Greedy, and Cloud-only heuristics (NATESHA; GUDDETI, 2021; NEZAMI et al., 2021); **concerning** application response time, makespan, placement plan time, fog utilization, application deadline time out, and energy consumption (YOSUF et al., 2020; DJEMAI et al., 2019); **from the point of view of** network architects, network engineers, IoT developers, Intelligent Transportation Systems companies, and Smart City companies; **in the context of** vehicle collision avoidance and heavy vehicle detection applications.

7.3.2 Planning

Context Selection

Based on (NATESHA; GUDDETI, 2021; AYOUBI; RAMEZANPOUR; KHORSAND, 2021; MARTIN; KANDASAMY; CHANDRASEKARAN, 2020), in this chapter, this study proposes an *in silico*. The experiment carried out using two synthetic IoT application request for placement of services whether in the fog computing layer or in the cloud computing layer. The first application is a vehicle collision avoidance detection for Intelligent Transportation Systems

as in (EYCKERMAN et al., 2020; MSEDDI et al., 2019; DONASSOLO et al., 2019b). The second is a heavy vehicle detection, such as bus and fire truck, in order to control semaphore preference.

Independent Variables

The independent variables of the experiment are, the IoT applications resource, the number of IoT application services, the applications requests distribution time, the number of nodes, the resource configuration of the nodes, and the network topology.

Dependent Variables

The dependent variables of the experiment are the application response time, makespan, placement plan time, fog utilization, application deadline time out, and energy consumption.

Hypothesis Formulation

Adopting the GQM method, the following research questions were designed to fully cover the objectives of the work:

1. Does R3GP plan the placement of applications with the shortest application response time?
2. Does R3GP have the lowest number of application deadline timeouts?
3. Does R3GP make the best use of Fog layer resources?
4. Does R3GP enable the lowest system energy consumption?

In order to evaluate these questions, the metrics described in the topic *Dependent Variables* were used. Question 1 was evaluated using the application response time variable as the main metric and the makespan and placement plan time variables as a complementary analysis. To answer question 2, the application deadline timeout metric was used. Question 3 is answered using the fog utilization variable. Finally, question 4 is answered by analyzing the energy consumption metric. Thus, with the objectives and metrics defined, the following hypotheses were formulated:

H_0: all treatments have the same average value for the evaluated metric.

H_1: any treatment has a different average value for the evaluated metric.

Formally, the hypotheses can be described as:

$$H_0 : \ M_i(metric) = M_j(metric), \ i \neq j \mid i, j \in Treatments$$

$$H_1 : \ M_i(metric) \neq M_j(metric), \ i \neq j \mid i, j \in Treatments$$

Object Selection

In order to evaluate the formulated hypotheses, representative samples of the population were collected. Samples were collected during 3600 seconds of execution for each algorithm. The frequency of collection of the energy consumption metric was the average of 10 samples in 1 second during 1 hour, totaling 3600 samples. The other metrics were collected as they were used. For analysis, the first 600 seconds were discarded due to the system stabilization time. Thus, 3000 seconds of samples collected for each treatment were statistically analyzed.

Experimental Design

The following high-level steps for experimenting were performed for each algorithm:

1. Run the experimental script that contains the following steps:
   a) The IoT devices send IoT application requests to the Fog Controller queue buffer each random exponential $request_{interval}$ distribution;
   b) The Fog Controller read the queue buffer and calculate the placement plan of the last application request using the algorithm (treatment);
   c) The Fog Controller send the requested IoT application service to the Fog Computing Node for placement.
2. Collect the metrics during 3600 seconds;
3. Discard the first 600 seconds (the transient state);
4. Apply appropriate statistical tests to analyze all hypotheses.

Instrumentation

For experimental setup, the following software were used: Python 3.10 with pandas 1.5.0, numpy 1.24.0, networkx 3.0, and matplotlib 3.7.0 libraries, and Ubuntu 22.04 LTS operating system. Furthermore, the following hardware setup was employed: Intel dual core x86_64 processor with 3.1 GHz clock, 8 GB of RAM, and 1 TB of HDD.

7.4 Experimental Operation

7.4.1 Preparation

In this step the simulation environment was configured. The host machine was equiped with the Kintoun simulator implemented in Python 3.10. Before execution and evaluation of the optimization algorithms

In order to perform the experiment, it was used the configurations of the fog computing nodes, links, and applications services presented in Tables 22 and 23. The configurations are based on the real hardware specifications and in (NIKOUI et al., 2020; MAHMUD et al., 2019b).

Table 22 – Nodes and links configurations of the case study.

Fog computing nodes configuration		
Raspberry Pi A+	CPU capacity	225.9 MIPS
	Memory capacity	512 MB
	Bandwidth capacity	58.8 Mbps
	Power idle	1.2 W
	Power maximum	5.4 W
Raspberry Pi B+	CPU capacity	224.89 MIPS
	Memory capacity	1024 MB
	Bandwidth capacity	59 Mbps
	Power idle	2.9 W
	Power maximum	6.4 W

Fog gateway nodes configuration		
Core Gateway	CPU capacity	13500 MIPS
	Memory capacity	2048 MB
	Bandwidth capacity	10 Gbps
	Power idle	20 W
	Power maximum	70 W
Edge/Aggregate Gateway	CPU capacity	6750 MIPS
	Memory capacity	1024 MB
	Bandwidth capacity	10 Gbps
	Power idle	10 W
	Power maximum	45 W

Network links configuration		
IoT to Fog	Bandwidth capacity	{58.8, 59} Mbps
	communication latency	1 ms per hop
Internal Fog	Bandwidth capacity	10 Gbps
	communication latency	1 ms per hop
Fog to Cloud	Bandwidth capacity	10 Gbps
	communication latency	100 ms per hop

Source: Author.

In order to execute and evaluate the optimization algorithms, the parameter used in the ϵ-Greedy was $\epsilon = 0.06$, the same as in Table 11 for 25 nodes. The topology of the case study experiment is presented in Figure 40. The FCN cluster region 1 contains 4 Fog Computing Nodes of the model Raspberry Pi A+ and 4 of the model B+. The FCN cluster region 2 contains 2 Fog Computing Nodes of the model Raspberry Pi A+ and 2 of the model B+. There are 20 IoT devices in region 1 and 20 IoT devices in region 2. Each IoT device has an exponential probability of 1 Heavy Vehicle Detection application request every 120 second, and 1 Vehicle Collision Avoidance

Table 23 – Applications configurations of the case study.

Application configuration		
Vehicle Collision Avoidance	Number of services	5
	Total deadline time	1200 ms
Heavy Vehicle Detection	Number of services	10
	Total deadline time	5000 ms

Service configuration		
Vehicle Collision Avoidance Services	CPU requirement	{2, 4, 8} MI
	Memory requirement	16 MB
	Bandwidth requirement	256 Kbps
	Input data size	{1, 10} KB
	Output data size	{1, 10} KB
	Deadline time	30 ms
Heavy Vehicle Detection Services	CPU requirement	{3, 6, 9, 12, 15} MI
	Memory requirement	32 MB
	Bandwidth requirement	512 Kbps
	Input data size	{1, 10} KB
	Output data size	{1, 10} KB
	Deadline time	300 ms

Source: Author.

application request every 600 seconds.

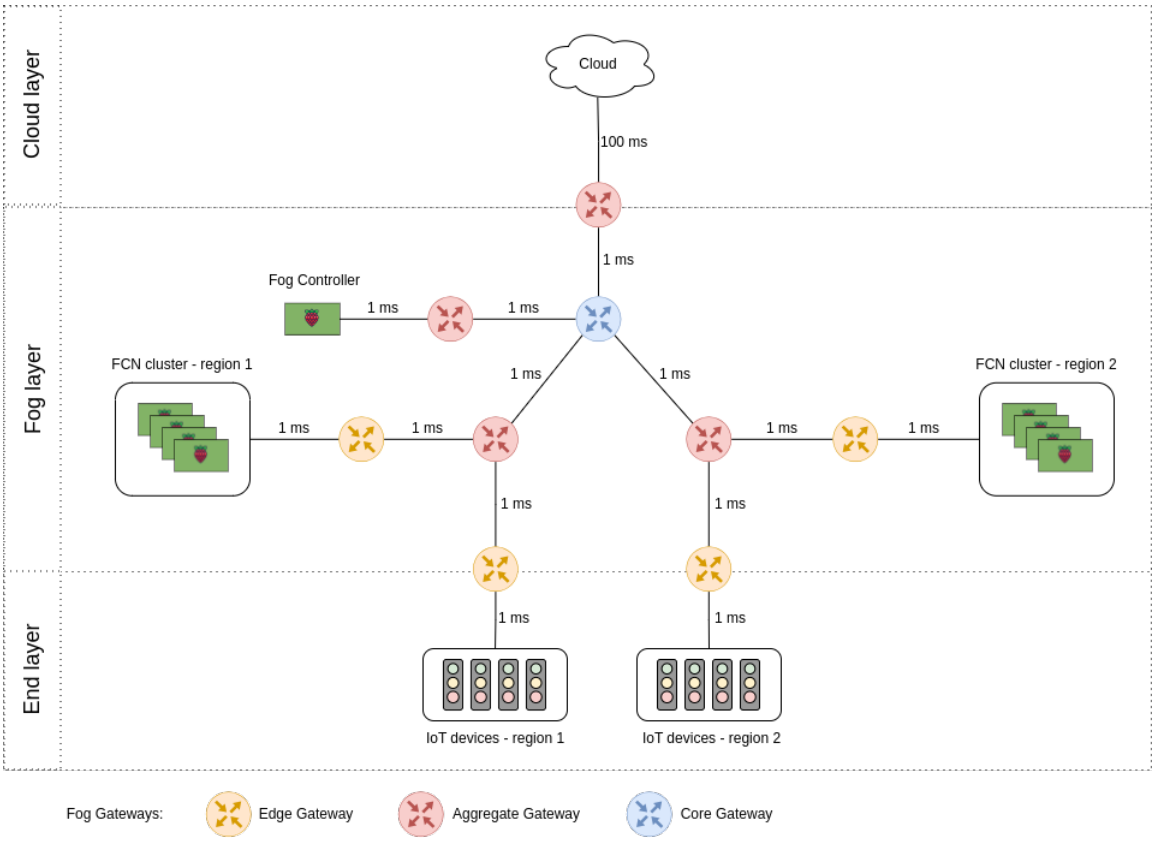
7.4.2 Execution

The steps specified in Experimental Design topic in 7.3.2 was performed as described. The Ubuntu was used in the experiments without the Graphical User Interface (GUI) started, only via teletypewriter (TTY). The treatments used was the R3GP, First-Fit Decreasing, ϵ-Greedy, and Cloud-only. It was ran 3600 seconds of experiment for each treatment. The metrics collected were application response time, makespan, placement plan time, number of placements in fog, number of placements in cloud, number of applications with timeout, and energy consumption. The number of placements in fog and in cloud composes the fog utilization metric.

7.5 Results and Discussion

This section presents the results of the performance of the R3GP, First-Fit Decreasing, ϵ-Greedy, and Cloud-only algorithms in regards to the case study of Fog Service Placement Problem for Vehicle Collision Avoidance and Heavy Vehicle Detection IoT applications. It presents the analysis of the results of the case study experiment in order to validate the hypothesis and answer the research question formulated in subsection 7.3.2. Section 7.5.1 presents a summary of how the data were statistically analyzed. Subsection 7.5.2 presents the results of the application

Figure 40 – Fog topology of the case study.



Source: Author.

response time metric and a secondary analysis using the makespan and placement plan time metrics are in subsection 7.5.3 and 7.5.4, respectively. Subsection 7.5.5 presents the use of fog resources by services in relation to the cloud. Finally, the analysis of energy consumption in the system is done in subsection 7.5.6.

7.5.1 Data Validation

In order to validate the data and evaluate the hypotheses, the following statistical analysis were used: Kolmogorov-Smirnov (KS) test, to check the normality of the data; Most of the data were not normal, then, it was used Kruskal-Wallis test, to compare the median values of the metrics obtained of all treatments. At last, was used the Dunn's post hoc test to ranking the treatments.

7.5.2 Application response time results

Figure 41 presents the distribution of the samples collected from the application response time metric. The Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data do not follow a normal distribution. Therefore, the hypotheses related to the

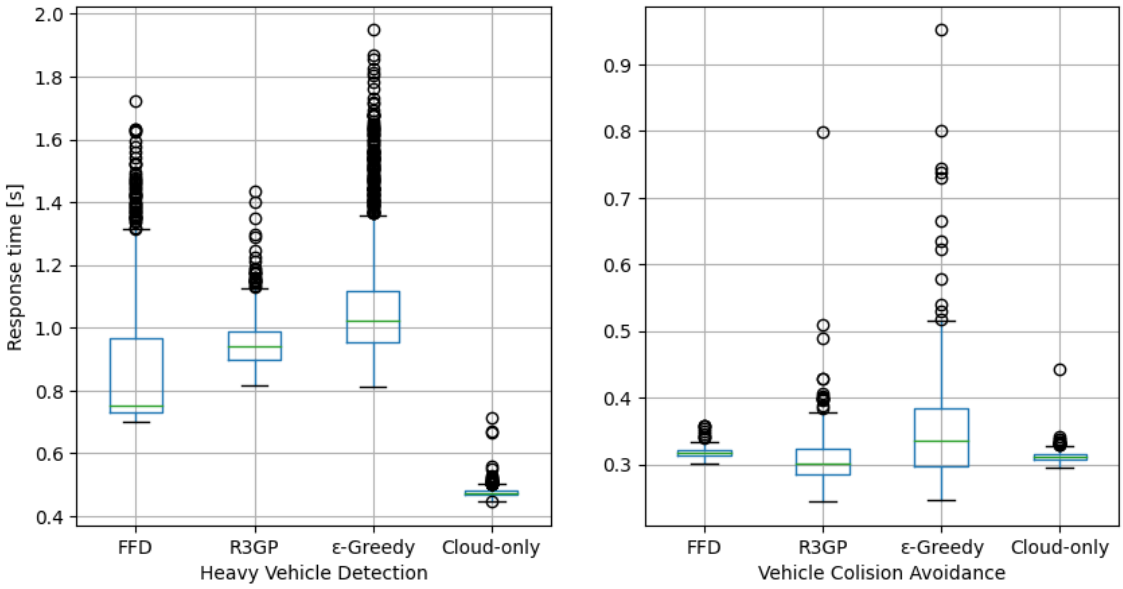
application response time can be formulated in the following mathematical way:

$$H_0 : M_i(response\ time) = M_j(response\ time),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : M_i(response\ time) \neq M_j(response\ time),\ i \neq j \mid i, j \in Treatments$$

In order to evaluate these hypotheses, the Kruskal-Wallis test was applied. As a result, the calculated p-value was less than 0.025, that is, the null hypothesis H_0 was rejected. Thus, there is a difference between the medians of the treatment samples.

Figure 41 – Distribution of the application response time of the case study.



Source: Author.

Given the difference between medians, Dunn's test was applied to rank treatments. Table 24 shows the ranking of algorithms according to the median of the application response time metric. In addition, Figure 42 shows the graph of the means with a 95% confidence interval for the treatment data.

Two factors that the application response time takes into account are the placement planning time of the algorithm used and the makespan. The explanation for the Cloud-only having lower values than the other algorithms, for the Heavy Vehicle Detection application, is that the Cloud-only placement strategy only transfers the application to the cloud resource directly, therefore, the placement planning time is very small. Another point is that the cloud is modeled as a single resource with large capacity. In this way, there is no network communication time between one service and another in the cloud, as it is considered as if it were a single node.

It is important to consider that fog server clusters are formed by Raspberry Pi model devices, that is, devices with extremely limited resources when compared to the resource modeled for the cloud. However, as shown in Table 24, R3GP was able to obtain one of the best application

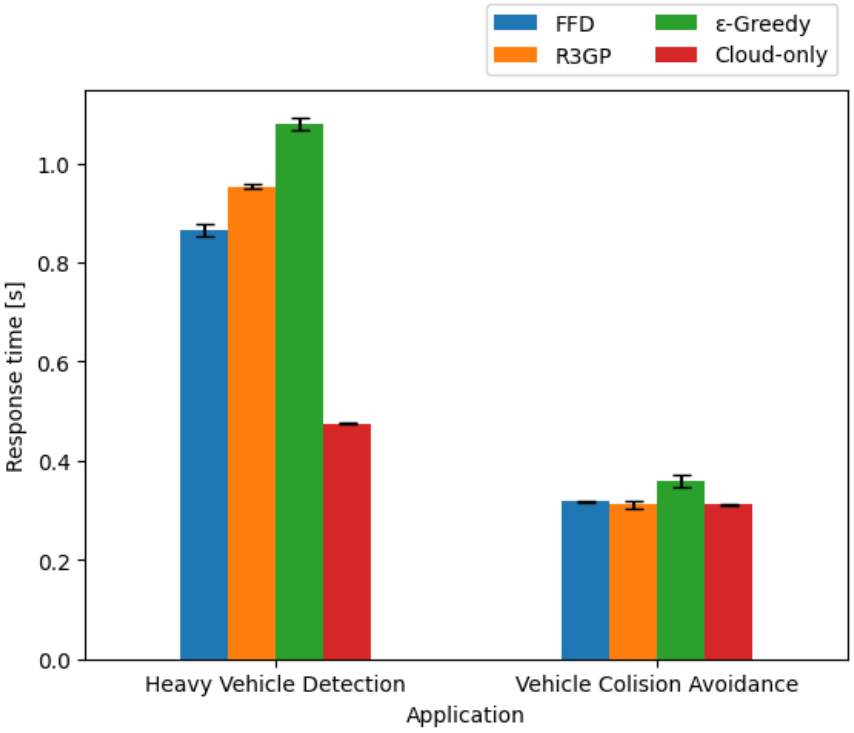
Table 24 – Application response time ranking.

Heavy Vehicle Detection			
Rank	Algorithm	Median	p-value
1^{st}	Cloud-only	0.474468	< 0.0250
2^{nd}	FFD	0.751412	
3^{rd}	R3GP	0.942878	
4^{th}	ϵ-Greedy	1.023218	

Collision Avoidance			
Rank	Algorithm	Median	p-value
1^{st}	R3GP	0.300601	0.1257
	Cloud-only	0.310330	
3^{rd}	FFD	0.316739	0.6690
	ϵ-Greedy	0.335465	

Source: Author.

Figure 42 – Mean of the application response time of the case study.



Source: Author.

response times for the Collision Avoidance application by placing services only on fog nodes, as shown in the analysis of the fog utilization metric.

7.5.3 Application makespan results

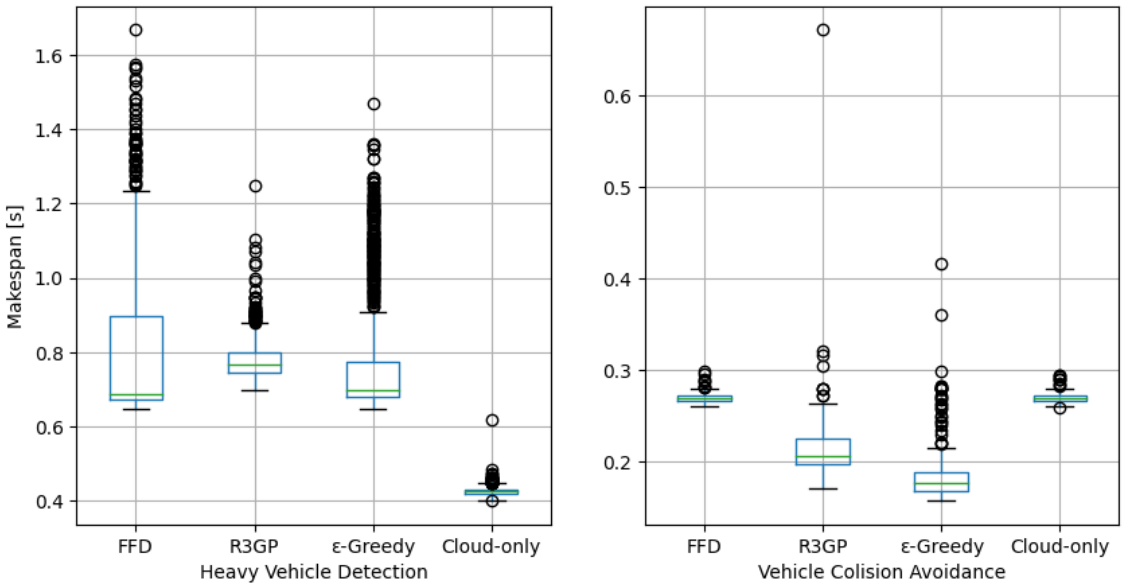
Figure 43 presents the mean distribution of the samples collected from the makespan metric. The Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data do not follow a normal distribution. Therefore, the hypotheses related to the makespan can be formulated in the following mathematical way:

$$H_0 : M_i(makespan) = M_j(makespan),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : M_i(makespan) \neq M_j(makespan),\ i \neq j \mid i, j \in Treatments$$

In order to evaluate these hypotheses, the Kruskal-Wallis test was applied. As a result, the calculated p-value was less than 0.025, that is, the null hypothesis H_0 was rejected. Thus, there is a difference between the medians of the treatment samples.

Figure 43 – Distribution of the application makespan of the case study.



Source: Author.

Given the difference between medians, Dunn's test was applied to rank treatments. Table 25 shows the ranking of algorithms according to the median of the makespan metric. In addition, Figure 44 shows the graph of the means with a 95% confidence interval for the treatment data.

As discussed in the analysis of the application response time, there is no network communication time in the cloud resource, in addition to the resource being extremely superior to those of fog. This combination makes the makespan of the Cloud-only strategy low. However, for the Collision Avoidance application, R3GP and ϵ-Greedy manage to outperform the other two strategies, with a difference of at least 30% faster.

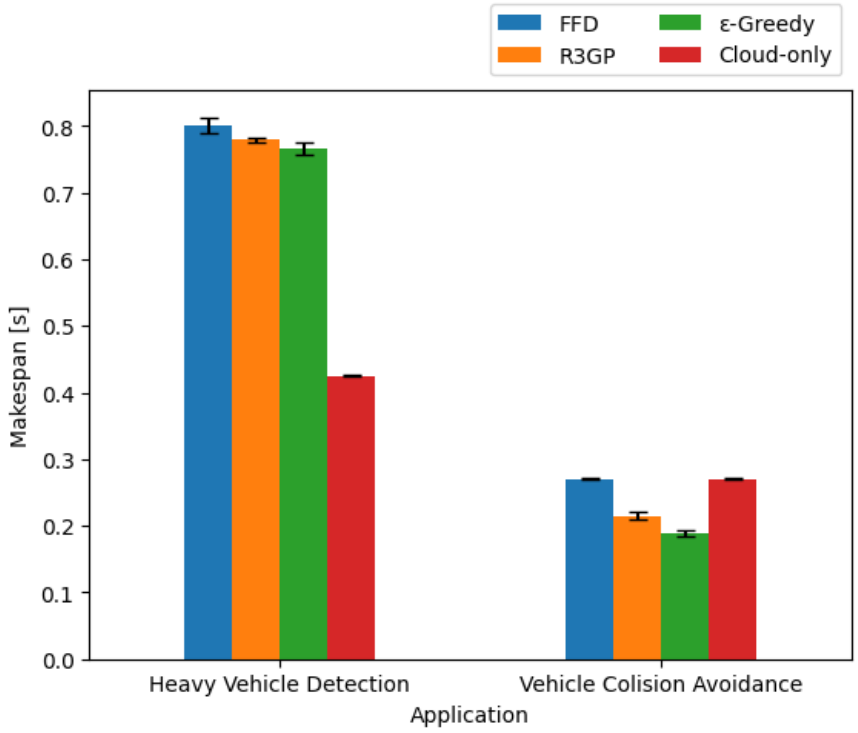
Table 25 – Makespan ranking of the case study.

Heavy Vehicle Detection			
Rank	Algorithm	Median	p-value
1^{st}	Cloud-only	0.424688	< 0.0250
2^{nd}	FFD	0.686760	0.6090
	ϵ-Greedy	0.697554	
4^{th}	R3GP	0.765748	< 0.0250

Collision Avoidance			
Rank	Algorithm	Median	p-value
1^{st}	ϵ-Greedy	0.177166	< 0.0250
2^{nd}	R3GP	0.206462	
3^{rd}	Cloud-only	0.269236	0.7550
	FFD	0.269516	

Source: Author.

Figure 44 – Average mean of the makespan of the case study.



Source: Author.

7.5.4 Placement planning time results

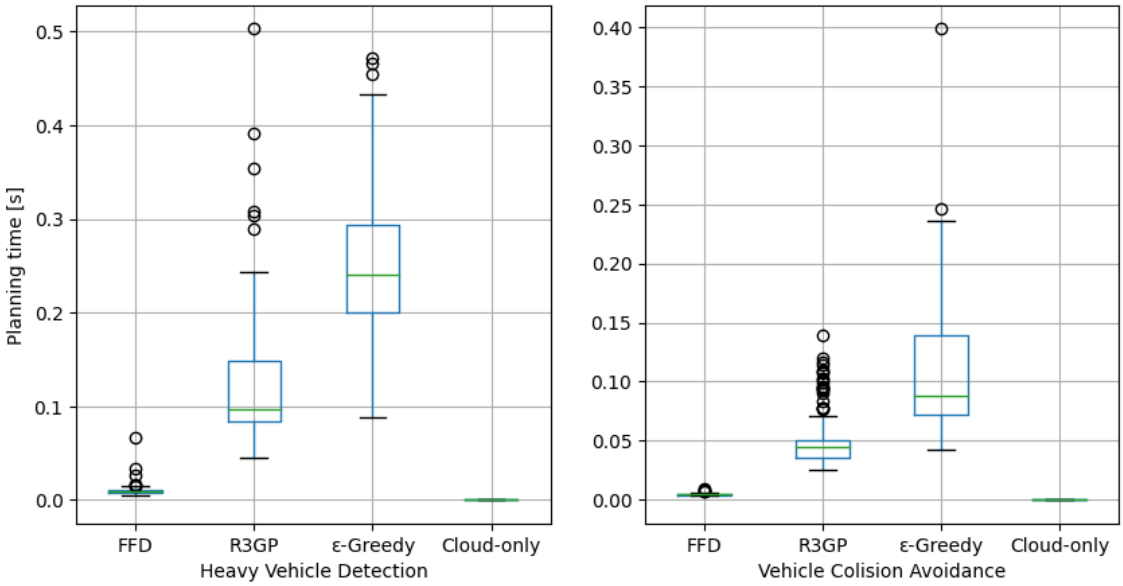
Figure 45 presents the mean distribution of the samples collected from the placement planning time metric.The Kolmogorov-Smirnov normality test was applied. The p-values of the test show that the data do not follow a normal distribution. Therefore, the hypotheses related to the placement planning time can be formulated in the following mathematical way:

$$H_0 : M_i(response\ time) = M_j(response\ time),\ i \neq j \mid i, j \in Treatments$$

$$H_1 : M_i(response\ time) \neq M_j(response\ time),\ i \neq j \mid i, j \in Treatments$$

In order to evaluate these hypotheses, the Kruskal-Wallis test was applied. As a result, the calculated p-value was less than 0.025, that is, the null hypothesis H_0 was rejected. Thus, there is a difference between the medians of the treatment samples.

Figure 45 – Distribution of the placement planning time of the case study.



Source: Author.

Given the difference between medians, Dunn's test was applied to rank treatments. Table 26 shows the ranking of algorithms according to the median of the placement planning time metric. In addition, Figure 46 shows the graph of the means with a 95% confidence interval for the treatment data.

As discussed in the application response time analysis, the explanation for the extremely low placement planning time of Cloud-only is that this strategy only instantly offloads the application to the cloud resource. It is important to remember that, unlike the others, despite R3GP being a metaheuristic, the proposed algorithm manages to calculate the placement plan approximately twice as fast as ϵ-Greedy for both applications, however, maintaining the solution quality as good as ϵ-Greedy, as analyzed by makespan and response time.

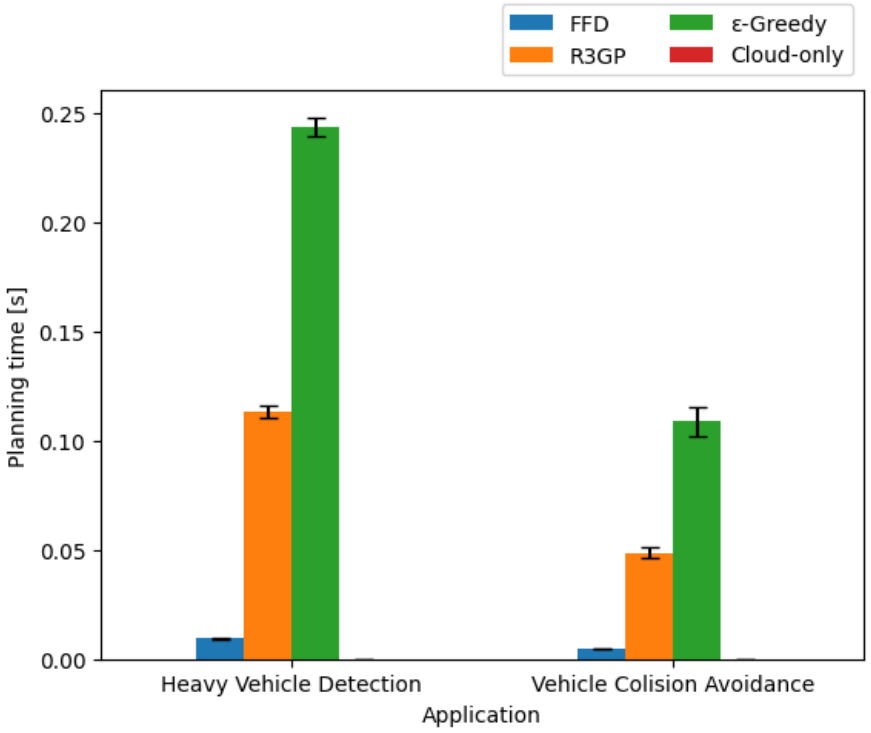
Table 26 – Placement planning time ranking of the case study.

Heavy Vehicle Detection			
Rank	Algorithm	Median	p-value
1^{st}	Cloud-only	0.000119	< 0.0250
2^{nd}	FFD	0.009029	
3^{rd}	R3GP	0.096663	
4^{th}	ϵ-Greedy	0.239939	

Collision Avoidance			
Rank	Algorithm	Median	p-value
1^{st}	Cloud-only	0.000108	< 0.0250
2^{nd}	FFD	0.004639	
3^{rd}	R3GP	0.044923	
4^{th}	ϵ-Greedy	0.087599	

Source: Author.

Figure 46 – Mean of the placement planning time of the case study.

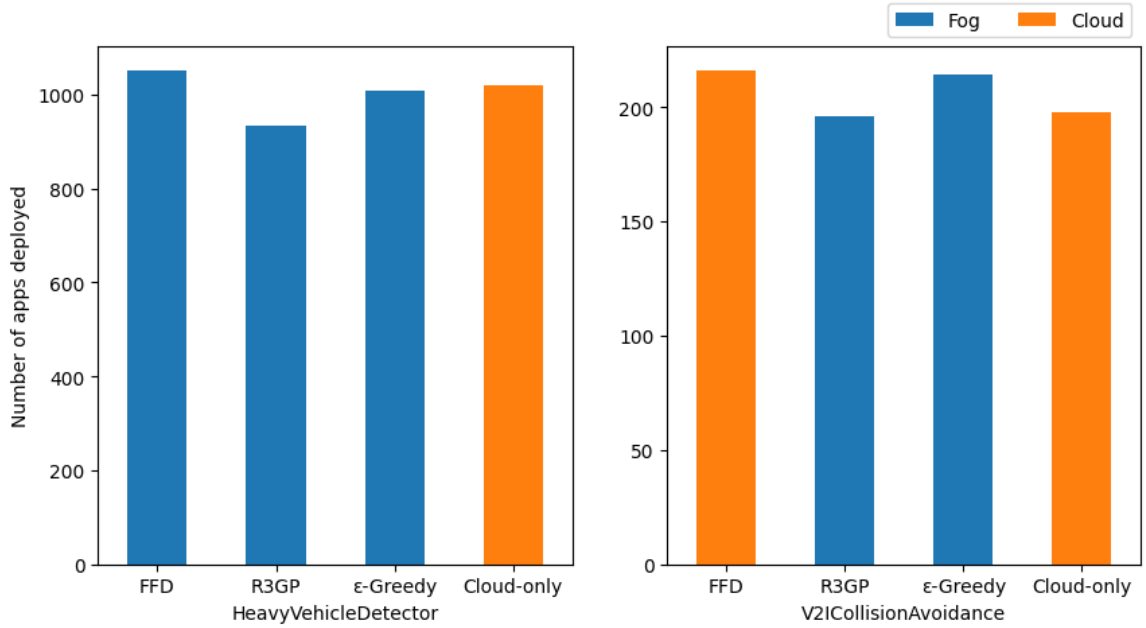


Source: Author.

7.5.5 Fog utilization results

Figure 47 shows that the R3GP, ϵ-Greedy and FFD maintain the services of the Heavy Vehicle Detection in the fog infrastructure. However, related to the Collision Avoidance application, for 100% of the requests, the FFD offloaded the services to the cloud infrastructure.

Figure 47 – Number of deployments in fog and cloud.



Source: Author.

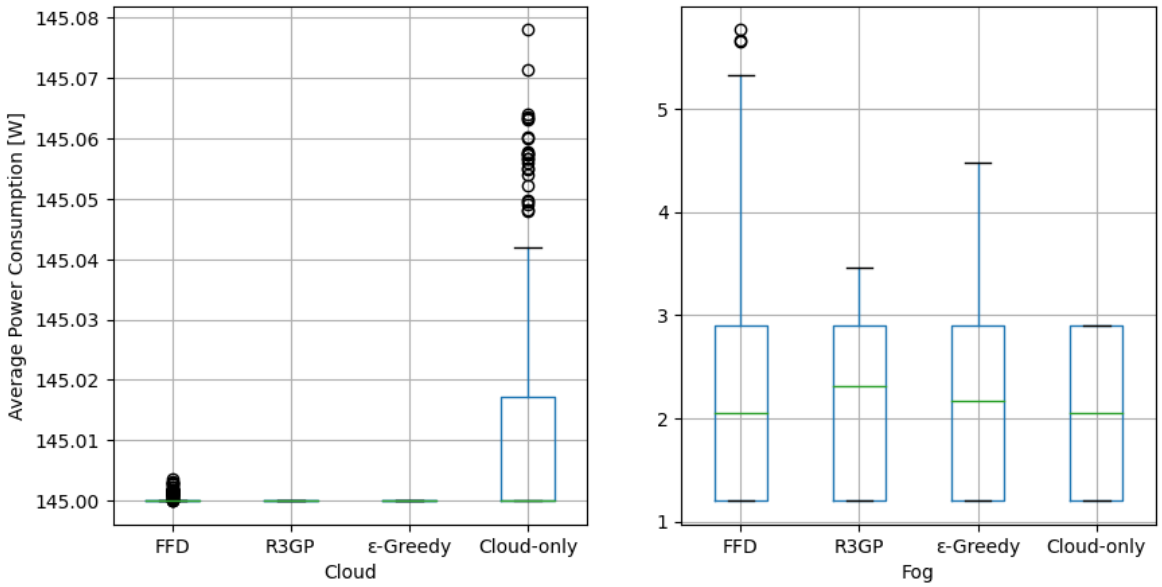
7.5.6 Energy consumption results

Figure 48 shows the average power consumption of the system over 3600 seconds of simulation. Application requests were not enough to have a relevant difference between power idle and average consumption. It is necessary to have a greater load of application requests in order to be able to better analyze the behavior of the placement of algorithms in relation to energy consumption.

7.6 Conclusion

In general, in the Kintoun simulator, the Rotation-Guided Greedy Genetic Particle algorithm, proposed in this work, performed better than ϵ-Greedy, and better than First-Fit Decreasing and Cloud-only in the placement of Vehicle Collision Avoidance applications. The highlight of R3GP in the simulator was the application response time metric for applying collision avoidance. None of the algorithms generated application timeout, but R3GP and ϵ-Greedy had better use of Fog nodes than FFD, as seen in the fog utilization metric.

Figure 48 – Average power consumption.



Source: Author.

Finally, given the evidence, the following research questions are answered as follows: Does R3GP plan the placement of applications with the shortest application response time? Yes, for the Vehicle Collision Avoidance application. Does R3GP have the lowest number of application deadline timeouts? Yes, however, none of the treatments generated application time out. Does R3GP make the best use of Fog layer resources? Yes, just like ϵ-Greedy, R3GP manages to take advantage of all the resources available in fog. Does R3GP enable the lowest system power consumption? No, however it is necessary to have a more detailed investigation, with a greater load of different types of applications.

8

Conclusion

Fog computing is a paradigm that emerged as an extension of the Cloud, to enable the processing of services on edge devices, close to end devices. One of the main features of fog is the short response time. In order to manage the infrastructure resources in a fog, it is necessary to run an algorithm to calculate the application placement plan. This plan contains the mapping of which node each service will run on. The calculation of this plane is an NP-complete problem called Fog Service Placement Problem. In the literature, several authors try to create the plan using heuristic and metaheuristic algorithms.

The main objective of this work was to evaluate the performance of the proposed metaheuristic algorithm in solving the Fog Service Placement Problem for a case study of Intelligent Transportation Systems applications. The proposed algorithm, called R3GP (Rotation-Guided Greedy Genetic Particle), was evaluated against the algorithms found in the literature in relation to the metrics of application response time, makespan, placement planning time, speed convergence, energy consumption CPU load-balancing, Memory load-balancing, Bandwidth load-balancing, application deadline timeout and fog utilization. The results of the *in silico* experiments showed that R3GP can outperform the literature algorithms in all metrics, except for the placement planning time. However, for these metrics there is the bias of using the Python language. As it is an interpreted programming language, the execution time of the algorithm increases considerably in relation to compiled languages. Furthermore, R3GP showed good performance in energy consumption optimization, which makes it favorable for Green Computing.

As complements, to achieve the main objective, two other works were carried out. The first was a systematic mapping of the literature, which allowed identifying the main related works and the algorithms, metrics and tools used in solving the FSPP. The second was the creation of the Fog Computing infrastructure simulator, called Kintoun. Inspired by the iFogSim and YAFS simulators, however, with details of implementations that are more faithful to reality, such as, for example, the execution of services in parts, using Round Robin. Unlike other simulators, Kintoun

has been statistically tested and validated based on queuing theory.

As future work, a better investigation is suggested regarding the energy consumption of the algorithms in the Kintoun simulator. It is also suggested a performance improvement in the placement planning time of the proposed algorithm (R3GP). Tests are needed with a larger number of fog nodes, application services and number of application requests. Finally, it is suggested that the algorithm be used in other case studies, compared to other algorithms such as Ant Colony Optimization and Particle Swarm Optimization.

Bibliography

AL-TARAWNEH, M. A. Bi-objective optimization of application placement in fog computing environments. *Journal of Ambient Intelligence and Humanized Computing*, Springer, v. 13, n. 1, p. 445–468, 2022. Citado na página 29.

ALGHAMDI, A.; ALZAHRANI, A.; THAYANANTHAN, V. Execution time and power consumption optimization in fog computing environment. *International Journal of Computer Science & Network Security*, v. 21, n. 1, p. 137–142, 2021. Citado na página 35.

ALI, H. M. et al. Planning a secure and reliable iot-enabled fog-assisted computing infrastructure for healthcare. *Cluster Computing*, Springer, v. 25, n. 3, p. 2143–2161, 2022. Citado 2 vezes nas páginas 31 and 34.

ALMURSHED, O.; RANA, O.; CHARD, K. Greedy nominator heuristic: Virtual function placement on fog resources. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 34, n. 6, p. e6765, 2022. Citado 2 vezes nas páginas 29 and 41.

ALQAHTANI, A. M. et al. Energy efficient resource allocation in federated fog computing networks. In: IEEE. *2021 IEEE Conference on Standards for Communications and Networking (CSCN)*. [S.l.], 2021. p. 199–204. Citado na página 31.

ALQAHTANI, A. M. et al. Energy minimized federated fog computing over passive optical networks. In: IEEE. *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. [S.l.], 2021. p. 1–6. Citado 2 vezes nas páginas 29 and 37.

AMARASINGHE, G. et al. A data stream processing optimisation framework for edge computing applications. In: IEEE. *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*. [S.l.], 2018. p. 91–98. Citado na página 32.

APAT, H. K. et al. Energy efficient resource management in fog computing supported medical cyber-physical system. In: IEEE. *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*. [S.l.], 2020. p. 1–6. Citado 3 vezes nas páginas 34, 35, and 46.

ARKIAN, H. R.; DIYANAT, A.; POURKHALILI, A. Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications. *Journal of Network and Computer Applications*, Elsevier, v. 82, p. 152–165, 2017. Citado na página 31.

AYOUBI, M.; RAMEZANPOUR, M.; KHORSAND, R. An autonomous iot service placement methodology in fog computing. *Software: Practice and Experience*, Wiley Online Library, v. 51, n. 5, p. 1097–1120, 2021. Citado 10 vezes nas páginas 20, 32, 37, 38, 40, 68, 75, 76, 103, and 104.

BARANWAL, G.; VIDYARTHI, D. P. Trappy: a truthfulness and reliability aware application placement policy in fog computing. *The Journal of Supercomputing*, Springer, p. 1–27, 2022. Citado 2 vezes nas páginas 31 and 66.

BARANWAL, G.; YADAV, R.; VIDYARTHI, D. P. Qoe aware iot application placement in fog computing using modified-topsis. *Mobile Networks and Applications*, Springer, v. 25, n. 5, p. 1816–1832, 2020. Citado 2 vezes nas páginas 35 and 46.

BASILI, G.; CALDIERA, V. R.; ROMBACH, H. D. The goal question metric approach. *Encyclopedia of software engineering*, p. 528–532, 1994. Citado 5 vezes nas páginas 21, 67, 76, 103, and 104.

BONOMI, F. et al. Fog computing and its role in the internet of things. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. [S.l.: s.n.], 2012. p. 13–16. Citado 2 vezes nas páginas 19 and 44.

BOURHIM, E. H.; ELBIAZE, H.; DIEYE, M. Inter-container communication aware container placement in fog computing. In: IEEE. *2019 15th International Conference on Network and Service Management (CNSM)*. [S.l.], 2019. p. 1–6. Citado 2 vezes nas páginas 29 and 37.

BROGI, A. et al. Meet genetic algorithms in monte carlo: optimised placement of multi-service applications in the fog. In: IEEE. *2019 IEEE International Conference on Edge Computing (EDGE)*. [S.l.], 2019. p. 13–17. Citado 4 vezes nas páginas 31, 37, 76, and 103.

CHATTERJEE, T. et al. A survey of vanet/v2x routing from the perspective of non-learning-and learning-based approaches. *IEEE Access*, IEEE, v. 10, p. 23022–23050, 2022. Citado na página 47.

CHEKIRED, D. A.; KHOUKHI, L. Multi-tier fog architecture: A new delay-tolerant network for iot data processing. In: IEEE. *2018 IEEE International Conference on Communications (ICC)*. [S.l.], 2018. p. 1–6. Citado na página 32.

CHOI, J.; AHN, S. Optimal service provisioning for the scalable fog/edge computing environment. *Sensors*, MDPI, v. 21, n. 4, p. 1506, 2021. Citado na página 19.

ČULÍK, K.; KALAŠOVÁ, A.; ŠTEFANCOVÁ, V. Evaluation of driver's reaction time measured in driving simulator. *Sensors*, MDPI, v. 22, n. 9, p. 3542, 2022. Citado 2 vezes nas páginas 101 and 102.

DASH, S.; AHMAD, M.; IQBAL, T. Mobile cloud computing: a green perspective. In: SPRINGER. *Intelligent Systems: Proceedings of ICMIB 2020*. [S.l.], 2021. p. 523–533. Citado na página 31.

DESIKAN, K. S.; KOTAGI, V. J.; MURTHY, C. S. R. Topology control in fog computing enabled iot networks for smart cities. *Computer networks*, Elsevier, v. 176, p. 107270, 2020. Citado na página 44.

DJEMAI, T. et al. A discrete particle swarm optimization approach for energy-efficient iot services placement over fog infrastructures. In: IEEE. *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. [S.l.], 2019. p. 32–40. Citado 5 vezes nas páginas 35, 40, 41, 76, and 104.

DJEMAI, T. et al. Mobility support for energy and qos aware iot services placement in the fog. In: IEEE. *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. [S.l.], 2020. p. 1–7. Citado 2 vezes nas páginas 38 and 39.

DJEMAI, T. et al. Investigating mobility-aware strategies for iot services placement in the fog under energy and qos constraints. *Journal of Communications Software and Systems*, Udruga za komunikacijske i informacijske tehnologije, Fakultet . . . , v. 17, n. 2, p. 73–86, 2021. Citado 3 vezes nas páginas 31, 34, and 38.

DONASSOLO, B. et al. Fog based framework for iot service provisioning. In: IEEE. *2019 16th IEEE annual consumer communications & networking conference (CCNC)*. [S.l.], 2019. p. 1–6. Citado 2 vezes nas páginas 35 and 36.

DONASSOLO, B. et al. Load aware provisioning of iot services on fog computing platform. In: IEEE. *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. [S.l.], 2019. p. 1–7. Citado 7 vezes nas páginas 20, 29, 34, 41, 42, 77, and 105.

EYCKERMAN, R. et al. Requirements for distributed task placement in the fog. *Internet of Things*, Elsevier, v. 12, p. 100237, 2020. Citado 9 vezes nas páginas 20, 34, 35, 41, 42, 76, 77, 101, and 105.

FARZIN, P. et al. Flex: a platform for scalable service placement in multi-fog and multi-cloud environments. In: *Australasian Computer Science Week 2022*. [S.l.: s.n.], 2022. p. 106–114. Citado 3 vezes nas páginas 29, 31, and 37.

FATICANTI, F. et al. Cutting throughput with the edge: App-aware placement in fog computing. In: IEEE. *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. [S.l.], 2019. p. 196–203. Citado na página 36.

FATICANTI, F. et al. Throughput-aware partitioning and placement of applications in fog computing. *IEEE Transactions on Network and Service Management*, IEEE, v. 17, n. 4, p. 2436–2450, 2020. Citado na página 29.

GHOBAEI-ARANI, M.; SHAHIDINEJAD, A. A cost-efficient iot service placement approach using whale optimization algorithm in fog computing environment. *Expert Systems with Applications*, Elsevier, v. 200, p. 117012, 2022. Citado 4 vezes nas páginas 34, 38, 39, and 41.

GILL, M.; SINGH, D. Aco based container placement for caas in fog computing. *Procedia Computer Science*, Elsevier, v. 167, p. 760–768, 2020. Citado 2 vezes nas páginas 29 and 32.

GODINHO, N.; CURADO, M.; PAQUETE, L. Optimization of service placement with fairness. In: IEEE. *2019 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.], 2019. p. 1–6. Citado 2 vezes nas páginas 32 and 34.

GOUDARZI, M.; PALANISWAMI, M. S.; BUYYA, R. A distributed deep reinforcement learning technique for application placement in edge and fog computing environments. *IEEE Transactions on Mobile Computing*, IEEE, 2021. Citado 2 vezes nas páginas 30 and 39.

GUPTA, H. et al. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, Wiley Online Library, v. 47, n. 9, p. 1275–1296, 2017. Citado na página 48.

HAPP, D.; BAYHAN, S.; HANDZISKI, V. Joi: Joint placement of iot analytics operators and pub/sub message brokers in fog-centric iot platforms. *Future generation computer systems*, Elsevier, v. 119, p. 7–19, 2021. Citado 2 vezes nas páginas 31 and 34.

HARVEY, J.; KUMAR, S. A survey of intelligent transportation systems security: challenges and solutions. In: IEEE. *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. [S.l.], 2020. p. 263–268. Citado na página 46.

HERRERA, J. L. et al. Optimizing the response time in sdn-fog environments for time-strict iot applications. *IEEE Internet of Things Journal*, IEEE, v. 8, n. 23, p. 17172–17185, 2021. Citado na página 32.

HIESSL, T. et al. Optimal placement of stream processing operators in the fog. In: IEEE. *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*. [S.l.], 2019. p. 1–10. Citado na página 37.

HOSSEINPOUR, F. et al. A resource management model for distributed multi-task applications in fog computing networks. *IEEE Access*, IEEE, v. 9, p. 152792–152802, 2021. Citado na página 32.

HUANG, T. et al. An ant colony optimization-based multiobjective service replicas placement strategy for fog computing. *IEEE Transactions on Cybernetics*, IEEE, v. 51, n. 11, p. 5595–5608, 2020. Citado 3 vezes nas páginas 29, 37, and 38.

HUSSAIN, M. et al. Fog computing for big data analytics in iot aided smart grid networks. *Wireless Personal Communications*, Springer, v. 114, n. 4, p. 3395–3418, 2020. Citado 4 vezes nas páginas 29, 37, 41, and 66.

HUSSAIN, M.; BEG, M. et al. Fog computing for internet of things (iot)-aided smart grid architectures. *Big Data and cognitive computing*, Multidisciplinary Digital Publishing Institute, v. 3, n. 1, p. 8, 2019. Citado na página 32.

HUSSEIN, N. H. et al. A comprehensive survey on vehicular networking: Communications, applications, challenges, and upcoming research directions. *IEEE Access*, IEEE, v. 10, p. 86127–86180, 2022. Citado na página 48.

JACOB, B. et al. A practical guide to the ibm autonomic computing toolkit. *IBM Redbooks*, IBM Corp. International Technical Support Organization North Castle, NY, USA, v. 4, n. 10, p. 1–268, 2004. Citado na página 39.

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: Wiley New York, 1991. v. 1. Citado na página 67.

KARAMOOZIAN, A.; HAFID, A.; ABOULHAMID, E. M. On the fog-cloud cooperation: How fog computing can address latency concerns of iot applications. In: IEEE. *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.], 2019. p. 166–172. Citado na página 32.

KAYAL, P.; LIEBEHERR, J. Distributed service placement in fog computing: An iterative combinatorial auction approach. In: IEEE. *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.], 2019. p. 2145–2156. Citado na página 38.

KHARE, S. et al. Scalable edge computing for low latency data dissemination in topic-based publish/subscribe. In: IEEE. *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. [S.l.], 2018. p. 214–227. Citado 2 vezes nas páginas 34 and 66.

KHOSROABADI, F.; FOTOUHI-GHAZVINI, F.; FOTOUHI, H. Scatter: Service placement in real-time fog-assisted iot networks. *Journal of Sensor and Actuator Networks*, MDPI, v. 10, n. 2, p. 26, 2021. Citado 2 vezes nas páginas 35 and 39.

KIM, W.-S.; CHUNG, S.-H. User incentive model and its optimization scheme in user-participatory fog computing environment. *Computer Networks*, Elsevier, v. 145, p. 76–88, 2018. Citado na página 31.

KIM, W.-S.; CHUNG, S.-H. User-participatory fog computing architecture and its management schemes for improving feasibility. *IEEE Access*, IEEE, v. 6, p. 20262–20278, 2018. Citado 2 vezes nas páginas 19 and 29.

KOCHOVSKI, P. et al. Pareto-optimised fog storage services with novel service-level agreement specification. *Applied Sciences*, MDPI, v. 12, n. 7, p. 3308, 2022. Citado 2 vezes nas páginas 37 and 38.

KWAK, S. G.; KIM, J. H. Central limit theorem: the cornerstone of modern statistics. *Korean journal of anesthesiology*, Korean Society of Anesthesiologists, v. 70, n. 2, p. 144, 2017. Citado na página 69.

LERA, I.; GUERRERO, C.; JUIZ, C. Availability-aware service placement policy in fog computing based on graph partitions. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 2, p. 3641–3651, 2018. Citado 3 vezes nas páginas 35, 76, and 103.

Lera, I.; Guerrero, C.; Juiz, C. Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, v. 7, p. 91745–91758, 2019. ISSN 2169-3536. Citado na página 48.

LIANG, M. et al. A comparative survey of connectivity models in vehicular networks. In: IEEE. *2020 International Conference on Internet of Things and Intelligent Applications (ITIA)*. [S.l.], 2020. p. 1–5. Citado na página 47.

LIU, C. et al. Solving the multi-objective problem of iot service placement in fog computing using cuckoo search algorithm. *Neural Processing Letters*, Springer, v. 54, n. 3, p. 1823–1854, 2022. Citado 6 vezes nas páginas 20, 29, 32, 37, 38, and 39.

LLORENS-CARRODEGUAS, A. et al. An energy-friendly scheduler for edge computing systems. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 21, p. 7151, 2021. Citado 3 vezes nas páginas 35, 36, and 66.

LUKE, S. *Essentials of Metaheuristics*. second. [S.l.]: Lulu, 2013. Available for free at http://cs.gmu.edu/~sean/book/metaheuristics/. Citado na página 76.

MAHMUD, R.; RAMAMOHANARAO, K.; BUYYA, R. Latency-aware application module management for fog computing environments. *ACM Transactions on Internet Technology (TOIT)*, ACM New York, NY, USA, v. 19, n. 1, p. 1–21, 2018. Citado 2 vezes nas páginas 32 and 34.

MAHMUD, R.; RAMAMOHANARAO, K.; BUYYA, R. Edge affinity-based management of applications in fog computing environments. In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*. [S.l.: s.n.], 2019. p. 61–70. Citado 2 vezes nas páginas 35 and 36.

MAHMUD, R. et al. Quality of experience (qoe)-aware placement of applications in fog computing environments. *Journal of Parallel and Distributed Computing*, Elsevier, v. 132, p. 190–203, 2019. Citado 2 vezes nas páginas 35 and 46.

MAHMUD, R. et al. Context-aware placement of industry 4.0 applications in fog computing environments. *IEEE Transactions on Industrial Informatics*, IEEE, v. 16, n. 11, p. 7004–7013, 2019. Citado 4 vezes nas páginas 34, 36, 79, and 107.

MAITI, P. et al. Internet of things applications placement to minimize latency in multi-tier fog computing framework. *ICT Express*, Elsevier, v. 8, n. 2, p. 166–173, 2022. Citado na página 35.

MANIHAR, S.; PATEL, R.; AGRAWAL, S. Learning based task placement algorithm in the iot fog-cloud environment. *International Journal of Computer Networks and Applications (IJCNA)*, v. 8, 2018. Citado na página 30.

MANOGARAN, G.; RAWAL, B. S. An efficient resource allocation scheme with optimal node placement in iot-fog-cloud architecture. *IEEE Sensors Journal*, IEEE, v. 21, n. 22, p. 25106–25113, 2021. Citado na página 44.

MARCOS, H. et al. Intelligent traffic management system using internet of things: A systematic literature review. In: IEEE. *2022 IEEE 8th International Conference on Computing, Engineering and Design (ICCED)*. [S.l.], 2022. p. 1–6. Citado na página 46.

MARTIN, J. P.; KANDASAMY, A.; CHANDRASEKARAN, K. Crew: Cost and reliability aware eagle-whale optimiser for service placement in fog. *Software: Practice and Experience*, Wiley Online Library, v. 50, n. 12, p. 2337–2360, 2020. Citado 12 vezes nas páginas 19, 29, 34, 37, 38, 39, 66, 68, 75, 76, 103, and 104.

MEHRAN, N.; KIMOVSKI, D.; PRODAN, R. Mapo: a multi-objective model for iot application placement in a fog environment. In: *Proceedings of the 9th International Conference on the Internet of Things*. [S.l.: s.n.], 2019. p. 1–8. Citado 8 vezes nas páginas 19, 20, 34, 35, 36, 37, 38, and 41.

MINH, Q. T. et al. Toward service placement on fog computing landscape. In: IEEE. *2017 4th NAFOSTED conference on information and computer science*. [S.l.], 2017. p. 291–296. Citado 2 vezes nas páginas 32 and 45.

MOALLEMI, R.; BOZORGCHENANI, A.; TARCHI, D. An evolutionary-based algorithm for smart-living applications placement in fog networks. In: IEEE. *2019 IEEE Globecom Workshops (GC Wkshps)*. [S.l.], 2019. p. 1–6. Citado na página 35.

MOHAMED, S. A. E. et al. Safe driving distance and speed for collision avoidance in connected vehicles. *Sensors*, MDPI, v. 22, n. 18, p. 7051, 2022. Citado na página 102.

MORKEVICIUS, N. et al. Method for dynamic service orchestration in fog computing. *Electronics*, MDPI, v. 10, n. 15, p. 1796, 2021. Citado 2 vezes nas páginas 29 and 32.

MOURADIAN, C. et al. Application component placement in nfv-based hybrid cloud/fog systems with mobile fog nodes. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 37, n. 5, p. 1130–1143, 2019. Citado 3 vezes nas páginas 34, 37, and 41.

MOURADIAN, C.; KIANPISHEH, S.; GLITHO, R. H. Application component placement in nfv-based hybrid cloud/fog systems. In: IEEE. *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. [S.l.], 2018. p. 25–30. Citado na página 37.

MSEDDI, A. et al. Joint container placement and task provisioning in dynamic fog computing. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 6, p. 10028–10040, 2019. Citado 9 vezes nas páginas 20, 29, 34, 37, 41, 42, 77, 101, and 105.

NATESHA, B.; GUDDETI, R. Adopting elitism-based genetic algorithm for minimizing multi-objective problems of iot service placement in fog computing environment. *Journal of Network and Computer Applications*, Elsevier, v. 178, p. 102972, 2021. Citado 10 vezes nas páginas 29, 34, 38, 39, 64, 68, 75, 76, 103, and 104.

NATESHA, B.; GUDDETI, R. M. R. Heuristic-based iot application modules placement in the fog-cloud computing environment. In: IEEE. *2018 IEEE/ACM international conference on utility and cloud computing companion (UCC Companion)*. [S.l.], 2018. p. 24–25. Citado 2 vezes nas páginas 32 and 35.

NATESHA, B.; GUDDETI, R. M. R. Meta-heuristic based hybrid service placement strategies for two-level fog computing architecture. *Journal of Network and Systems Management*, Springer, v. 30, n. 3, p. 47, 2022. Citado 2 vezes nas páginas 29 and 38.

NATH, S. B. et al. Ptc: Pick-test-choose to place containerized micro-services in iot. In: IEEE. *2019 IEEE Global Communications Conference (GLOBECOM)*. [S.l.], 2019. p. 1–6. Citado 3 vezes nas páginas 19, 28, and 40.

NEZAMI, Z. et al. Decentralized edge-to-cloud load balancing: Service placement for the internet of things. *IEEE Access*, IEEE, v. 9, p. 64983–65000, 2021. Citado 7 vezes nas páginas 29, 37, 39, 41, 76, 103, and 104.

NIKOUI, T. S. et al. Cost-aware task scheduling in fog-cloud environment. In: IEEE. *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*. [S.l.], 2020. p. 1–8. Citado 5 vezes nas páginas 34, 37, 38, 79, and 107.

NTUMBA, P.; GEORGANTAS, N.; CHRISTOPHIDES, V. Efficient scheduling of streaming operators for iot edge analytics. In: IEEE. *2021 Sixth International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.], 2021. p. 1–8. Citado na página 34.

OMONIWA, B. et al. Fog/edge computing-based iot (feciot): Architecture, applications, and research issues. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 3, p. 4118–4149, 2018. Citado na página 46.

ORGANIZATION, W. H. et al. *Pedestrian safety: a road safety manual for decision-makers and practitioners*. [S.l.]: World Health Organization, 2023. Citado na página 101.

PALLEWATTA, S.; KOSTAKOS, V.; BUYYA, R. Qos-aware placement of microservices-based iot applications in fog computing environments. *Future Generation Computer Systems*, Elsevier, v. 131, p. 121–136, 2022. Citado 5 vezes nas páginas 32, 34, 39, 40, and 41.

PATRO, R. et al. Module placement scheme using mpc4. 5 with markov chain process for mobile fog computing environment. In: IEEE. *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. [S.l.], 2021. p. 304–309. Citado 2 vezes nas páginas 29 and 30.

PETTICREW, M.; ROBERTS, H. Systematic reviews in the social sciences: A practical guide. Blackwell (an imprint of Wiley), Malden, Massachusetts, year, 2006. Citado na página 25.

PHAM-NGUYEN, H.-N.; TRAN-MINH, Q. Dynamic resource provisioning on fog landscapes. *Security and Communication Networks*, Hindawi, v. 2019, 2019. Citado 2 vezes nas páginas 45 and 59.

PLATEL, M. D.; SCHLIEBS, S.; KASABOV, N. Quantum-inspired evolutionary algorithm: A multimodel eda. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 13, n. 6, p. 1218–1232, 2008. Citado 2 vezes nas páginas 51 and 54.

QIAO, J. Smart city and intelligent upgrading of urban transportation system: based on sustainable investment strategy. In: IEEE. *2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)*. [S.l.], 2022. p. 1–6. Citado na página 46.

RABBY, M. K. M.; ISLAM, M. M.; IMON, S. M. A review of iot application in a smart traffic management system. In: IEEE. *2019 5th International Conference on Advances in Electrical Engineering (ICAEE)*. [S.l.], 2019. p. 280–285. Citado 2 vezes nas páginas 46 and 47.

RAGHAVENDRA, M. S.; CHAWLA, P.; NARASIMHULU, Y. A probability based joint-clustering algorithm for application placement in fog-to-cloud computing. In: IEEE. *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. [S.l.], 2021. p. 1–5. Citado na página 30.

RAHBARI, D.; NICKRAY, M. Task offloading in mobile fog computing by classification and regression tree. *Peer-to-Peer Networking and Applications*, Springer, v. 13, p. 104–122, 2020. Citado 3 vezes nas páginas 29, 30, and 66.

REDDY, K. H. K. et al. A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities. *Sustainable Cities and Society*, Elsevier, v. 63, p. 102428, 2020. Citado 2 vezes nas páginas 32 and 35.

REZAZADEH, Z.; RAHBARI, D.; NICKRAY, M. Optimized module placement in iot applications based on fog computing. In: IEEE. *Electrical Engineering (ICEE), Iranian Conference on*. [S.l.], 2018. p. 1553–1558. Citado 2 vezes nas páginas 31 and 34.

REZAZADEH, Z.; REZAEI, M.; NICKRAY, M. Lamp: A hybrid fog-cloud latency-aware module placement algorithm for iot applications. In: IEEE. *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*. [S.l.], 2019. p. 845–850. Citado na página 34.

ROY, B.; PATNAIK, S.; DUTTA, P. Congestion detection techniques in road network. In: IEEE. *2021 Smart City Challenges & Outcomes for Urban Transformation (SCOUT)*. [S.l.], 2021. p. 252–255. Citado na página 47.

SAHOO, J. Optimal secure placement of iot applications for smart farming. In: IEEE. *2021 8th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. [S.l.], 2021. p. 1–6. Citado 2 vezes nas páginas 28 and 37.

SALIMIAN, M.; GHOBAEI-ARANI, M.; SHAHIDINEJAD, A. Toward an autonomic approach for internet of things service placement using gray wolf optimization in the fog computing environment. *Software: Practice and Experience*, Wiley Online Library, v. 51, n. 8, p. 1745–1772, 2021. Citado 2 vezes nas páginas 29 and 37.

SALIMIAN, M.; GHOBAEI-ARANI, M.; SHAHIDINEJAD, A. An evolutionary multi-objective optimization technique to deploy the iot services in fog-enabled networks: an autonomous approach. *Applied Artificial Intelligence*, Taylor & Francis, v. 36, n. 1, p. 2008149, 2022. Citado 4 vezes nas páginas 29, 32, 37, and 39.

SAMANI, Z. N.; SAURABH, N.; PRODAN, R. Multilayer resource-aware partitioning for fog application placement. In: IEEE. *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*. [S.l.], 2021. p. 9–18. Citado na página 35.

SAMI, H.; MOURAD, A. Dynamic on-demand fog formation offering on-the-fly iot service deployment. *IEEE Transactions on Network and Service Management*, IEEE, v. 17, n. 2, p. 1026–1039, 2020. Citado 2 vezes nas páginas 35 and 36.

SANTOS, J. et al. Resource provisioning for iot application services in smart cities. In: IEEE. *2017 13th International Conference on Network and Service Management (CNSM)*. [S.l.], 2017. p. 1–9. Citado 4 vezes nas páginas 29, 34, 37, and 44.

SANTOS, J. et al. Live demonstration of service function chaining allocation in fog computing. In: IEEE. *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. [S.l.], 2020. p. 362–364. Citado 3 vezes nas páginas 35, 36, and 41.

SANTOS, J. et al. Towards delay-aware container-based service function chaining in fog computing. In: IEEE. *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. [S.l.], 2020. p. 1–9. Citado 3 vezes nas páginas 34, 36, and 44.

SHAIK, S.; BASKIYAR, S. Distributed service placement in hierarchical fog environments. *Sustainable Computing: Informatics and Systems*, Elsevier, v. 34, p. 100744, 2022. Citado 2 vezes nas páginas 34 and 66.

SHARMA, K.; BUTLER, B.; JENNINGS, B. Graph-based heuristic solution for placing distributed video processing applications on moving vehicle clusters. *IEEE Transactions on Network and Service Management*, IEEE, v. 19, n. 3, p. 3076–3089, 2022. Citado 2 vezes nas páginas 34 and 35.

SKARLAT, O. et al. A framework for optimization, service placement, and runtime operation in the fog. In: IEEE. *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*. [S.l.], 2018. p. 164–173. Citado 4 vezes nas páginas 35, 36, 45, and 46.

SKARLAT, O. et al. Optimized iot service placement in the fog. *Service Oriented Computing and Applications*, Springer, v. 11, n. 4, p. 427–443, 2017. Citado 3 vezes nas páginas 20, 75, and 103.

SKARLAT, O. et al. Towards qos-aware fog service placement. In: IEEE. *2017 IEEE 1st international conference on Fog and Edge Computing (ICFEC)*. [S.l.], 2017. p. 89–96. Citado 4 vezes nas páginas 32, 37, 75, and 103.

SKARLAT, O.; SCHULTE, S. Fogframe: a framework for iot application execution in the fog. *PeerJ Computer Science*, PeerJ Inc., v. 7, p. e588, 2021. Citado na página 35.

TAVOUSI, F.; AZIZI, S.; GHADERZADEH, A. A fuzzy approach for optimal placement of iot applications in fog-cloud computing. *Cluster Computing*, Springer, p. 1–18, 2022. Citado na página 29.

TRAN, M.-Q. et al. Task placement on fog computing made efficient for iot application provision. *Wireless Communications and Mobile Computing*, Hindawi, v. 2019, 2019. Citado 4 vezes nas páginas 32, 34, 44, and 45.

TULI, S. et al. Cosco: Container orchestration using co-simulation and gradient based optimization for fog computing environments. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 33, n. 1, p. 101–116, 2021. Citado 2 vezes nas páginas 38 and 39.

ULLAH, A. et al. Advances in position based routing towards its enabled fog-oriented vanet–a survey. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 21, n. 2, p. 828–840, 2019. Citado na página 47.

VENTICINQUE, S.; AMATO, A. A methodology for deployment of iot application in fog. *Journal of Ambient Intelligence and Humanized Computing*, Springer, v. 10, n. 5, p. 1955–1976, 2019. Citado 2 vezes nas páginas 35 and 45.

VIJOUYEH, L. N. et al. Efficient application deployment in fog-enabled infrastructures. In: IEEE. *2020 16th International Conference on Network and Service Management (CNSM)*. [S.l.], 2020. p. 1–9. Citado 2 vezes nas páginas 28 and 34.

XIA, Y. et al. Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed iot applications in the fog. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. [S.l.: s.n.], 2018. p. 751–760. Citado na página 35.

XIA, Y. et al. Combining heuristics to optimize and scale the placement of iot applications in the fog. In: IEEE. *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*. [S.l.], 2018. p. 153–163. Citado 2 vezes nas páginas 34 and 35.

YADAV, A. M.; TRIPATHI, K. N.; SHARMA, S. A bi-objective task scheduling approach in fog computing using hybrid fireworks algorithm. *The Journal of Supercomputing*, Springer, v. 78, n. 3, p. 4236–4260, 2022. Citado 2 vezes nas páginas 29 and 37.

YADAV, V.; NATESHA, B.; GUDDETI, R. M. R. Ga-pso: service allocation in fog computing environment using hybrid bio-inspired algorithm. In: IEEE. *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. [S.l.], 2019. p. 1280–1285. Citado 5 vezes nas páginas 20, 29, 32, 38, and 40.

YAO, J.; ANSARI, N. Fog resource provisioning in reliability-aware iot networks. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 5, p. 8262–8269, 2019. Citado 2 vezes nas páginas 29 and 37.

YOSUF, B. A. et al. Energy-efficient ai over a virtualized cloud fog network. In: *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*. [S.l.: s.n.], 2021. p. 328–334. Citado na página 29.

YOSUF, B. A. et al. Energy efficient distributed processing for iot. *IEEE Access*, IEEE, v. 8, p. 161080–161108, 2020. Citado 5 vezes nas páginas 31, 32, 41, 76, and 104.

YOUSEFPOUR, A. et al. Fogplan: A lightweight qos-aware dynamic fog service provisioning framework. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 3, p. 5080–5096, 2019. Citado 4 vezes nas páginas 31, 32, 46, and 104.

ZHAO, D.; ZOU, Q.; ZADEH, M. B. A qos-aware iot service placement mechanism in fog computing based on open-source development model. *Journal of Grid Computing*, Springer, v. 20, n. 2, p. 12, 2022. Citado 4 vezes nas páginas 32, 34, 38, and 39.