Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Department of Information Networks

AN IMPROVED BAT ALGORITHM FOR SOLVING PLACEMENT PROBLEM IN EDGE COMPUTING

A Dissertation

Submitted to the Council of the College of Information Technology, University of Babylon in Partial Fulfillment of the Requirements for the Degree of Doctorate of Philosophy in Information Technology / Information Networks

SAMRAA ADNAN ABID MUSLIM JASIM

Supervised by

Prof. Dr. Safaa Obayes Mahdi Madhlum

2023 A.D. 1445 A.H.

Supervisor Certification

I certify that the dissertation entitled (**An Improved BAT Algorithm for Solving Placement Problem in Edge Computing**) was prepared under my supervision at the department of Information Networks/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctorate of Philosophy in Information Technology-Information Networks.

Signature:

Supervisor Name: **Prof. Dr. Safaa Obayes Mahdi**

Date: / /2023

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "**An Improved BAT Algorithm for Solving Placement Problem in Edge Computing**" for debate by the examination committee.

Signature:

Assist. Prof. Dr. Alharith AbdulKareem Abdullah

Head of Information Networks Department

Date: / /2023

Certification of the Examination Committee

We hereby certify that we have studied the dissertation entitled (**An Improved BAT Algorithm for Solving Placement Problem in Edge Computing**) presented by the student (**Samraa Adnan Abid Muslim**) and examined him/her in its content and what is related to it, and that, in our opinion, it is adequate with (**Excellent**) standing as a dissertation for the degree of Doctorate of Philosophy in Information Technology-Information Networks.

Signature:
Name: **Dr. Ghaidaa A. Al-Sultany**
Title: **Professor**
Date: / / 2023
(Chairman)

Signature:
Name: **Dr. Nada A. Z. Abdullah**
Title: **Assist. Professor**
Date: / / 2023
(Member)

Signature:
Name: **Dr. Mehdi Ebady Manaa**
Title: **Assist. Professor**
Date: / / 2023
(Member)

Signature:
Name: **Dr. Ahmed M. Al-Salih**
Title: **Assist. Professor**
Date: / / 2023
(**Member**)

Signature:
Name: **Dr. Raaid N. Alubady**
Title: **Assist. Professor**
Date: / / 2023
(Member)

Signature:
Name: **Dr. Safaa O. Mahdi**
Title: **Professor**
Date: / / 2023
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: **Dr. Wesam Sameer Bhaya**
Title**: Professor**
Date: / / 2023
(Dean of Collage of Information Technology)

بسم الله الرحمن الرحيم

(وأن ليس للإنسان إلا ما سعى * وأن سعيه سوف يرى * ثم يجزاه الجزاء الأوفى)

صدق الله العظيم

النجم (39-41)

Dedications

To the generous hands that extended to me and raised me from behind the unseen and those with them who worked hard with me and supported me with sincere love, my family, loved ones and friends, I dedicate my thesis with my prayers to Allah for them with my affection and gratitude…

Samraa Adnan Al-Asadi

Acknowledgement

First and foremost, I would like to thank Allah, my God, without divine care I could not even have contemplated all the work involved in this study.

My heartfelt thanks to my supervisor Prof. Dr. Safaa O. Al-Mamory, for his help and support during the period of my study.

Finally, but most importantly, I would like to thank all the Researchers in my college who have imparted immense knowledge to me.

Abstract

In smart cities, there is a huge number of users devices that make enormous requests. Using Cloud servers pose a long latency time for sending, processing and receiving the response for these requests. Bringing Cloud services to the network edge (Edge Computing), will improve the network performance by reducing the overall network latency.

The problem of finding the optimal locations of edge nodes within large number of network devices like Base Stations or Access Points is NP-Hard problem. As a results, optimization methods are used instead.

The edge nodes' proposed placement problem is modeled mathematically as a multi-objective optimization problem. The problem is then solved using an improved BAT metaheuristic algorithm.

BAT algorithm is a nature-inspired metaheuristic algorithm that depends on the principle of the echolocation behavior of bats. However, due to its poor exploration, the algorithm suffers from being stuck in the local optima.

An improved BAT algorithm based on DBScan and Fractal Clustering techniques is proposed to enhance the algorithm's performance. The initial population is improved by generating two populations, randomly and depending on the clusters' center information, and getting the fittest individuals from these two populations to generate the initial improved one. The random walk function is improved by diversifying the solutions using chaotic maps instead of the fixed-size movement, so the local search is improved, as well as the global search abilities. Another improvement is dealing with stagnation by partitioning

the search space into two parts depending on the generated clusters' information to obtain a newly generated solution, comparing its quality with the previously generated one, and choosing the best.

The performance of the proposed improved BAT algorithm is evaluated by comparing it with the original BAT algorithm over ten IEEE CEC 09 benchmark optimization test functions. Depending on the results, the improved BAT outperforms the original BAT by obtaining the optimal global solutions for most of the benchmark test functions.

This improved BAT metaheuristic algorithm is used to find the optimal placement of edge nodes with the main objectives of minimizing the total number of edge servers to maximize the edge system utilization and balancing the workload, beside minimizing the total network latency.

The performance of the proposed edge nodes' placement approach is compared with two benchmark placement methods: Random-k and Top-k by using a real-world dataset form Shanghai Telecom. The simulation experiments show that the proposed approach leads to lower number of deployed edge nodes (70.5%-55%) less than Top-k approach, and (58.1%-35.5%) less than Random approach, and better system utilization (30.7%-50%) more than Top-k approach, and (30.1%-40.5%) more than the Random approach, and more balanced workload than these benchmark placement approaches. The network latency is slightly higher than the latency of the benchmark placement approaches.

Declaration Associated with this Thesis

I. Published Articles

1- Al‐Asadi, S.A., Al‐Mamory, S.O.: A survey on edge and fog nodes' placement methods, techniques, parameters, and constraints. IET Netw. 1–32 (2023). https://doi.org/10.1049/ntw2.12087

2- Al-Asadi, S., & Al-Mamory, S., (2023), "An Improved BAT Algorithm Using Density-Based Clustering". *Inteligencia Artificial journal*, *26*(72), pp.102-123. http://dx.doi.org/10.4114/intartif.vol26iss72pp102-123

Table of Contents

Chapter One: General Introduction

Chapter Two: Scientific Background

Chapter Three: The Proposed System

Chapter Four: Result, Analysis and Discussions

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

Abbreviation	Description
ACO	Ant Colony Optimization
AP	Access Point
BS	Base Station
COA	Chaos Optimization Algorithm
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
FD	Fractal Dimension
GA	Genetic Algorithm
HAF	Heaviest-AP First Placement
ILP	Integer Linear Programming
INLP	Integer Nonlinear Programming
IoV	Internet of Vehicle
IoT	Internet of Things
LP	Linear Programming
MEC	Mobile Edge Computing
MFC	Mobile Fog Computing
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Nonlinear Programming
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
RL	Reinforcement Learning
SA	Simulated Annealing
SLA	Service Level Agreements
STD	Standard Diviation
VM	Virtual Machine
WMAN	Wireless Metropolitan Area Network

CHAPTER ONE

GENERAL INTRODUCTION

1.1 Introduction

The rapid development of Internet of Things (IoT) based technologies take its role in different life aspects. The main goal of IoT technologies is to simplify the processes in many different fields, to ensure better efficiency of systems (technologies or specific processes), and finally to improve life quality. The generated data of the mobile and IoT devices has grown significantly. The limited energy and computational resources characterize IoT devices like sensors, smartphones, and wearable gadgets. These limitations are addressed by offloading processing and storage from resource constrained devices to the cloud. Because of the scalable and on-demand nature of the cloud, it is considered an ideal solution for computation offloading [1], However, the centralized, far away cloud servers location cause a too-long latency to respond to the requests from a vast number of IoT sensors distributed in a large working area, especially for applications with latency-critical needs, such as e-health and the Internet of Vehicles (IoV). This type of central data processing also causes colossal network traffic as the number of services and objects increases [2][3].

Mobile Edge Computing (MEC) and Mobile Fog Computing (MFC) platforms are both network architectures that complement the central cloud by bringing cloud functionalities to the network edge using fog and edge nodes. Edge/ fog nodes are small-scale servers (cloudlets) geographically distributed near end-user devices for more efficient service access [4][5]. The number of smart devices at the network edge is increasing rapidly, and many IoT applications have specific requirements and characteristics that make edge and fog computing a crucial solution

rather than the cloud. Low-latency communication is the main requirement of several IoT applications, including connected vehicles, mobile gaming, remote health monitoring, warehouse logistics, and industrial control systems. These scenarios are characterized by real-time actions or responses based on processing data generated by end devices.

Many IoT applications and devices generate vast amounts of bandwidth-intensive data like cameras and user devices. This local edge processing provides real-time information to the users. Using one-hop-away edge computing results in less need for the central far away cloud servers.

Mobility of some IoT devices imposes additional demands on low latency processing of their data. Edge computing supports the migration of virtualized resources based on the mobility of end devices, thus allowing the data generated by these devices to be processed locally and with a satisfactory quality of experience [1]. Many other requirements are location awareness, scalability, heterogeneity, temporary storage, high-speed data dissemination, decentralized computation, and security [2][6].

1.2 Dissertation Motivation

IoT smart devices are the main application of the edge and fog computing. Edge and fog computing are similar to the cloud computing, with a major difference being the distance between where the data is created, and where the data is processed. generally, the data is created at an end device, like a laptop or IoT device.

With cloud computing, the generated data will be processed in a cloud data center that are far away from the user devices. With edge and

fog computing, that data will be processed in a much closer location to the end devices. Edge and fog computing rely upon widely dispersed micro data centers than cloud main data centers.

Optimal placement of edge /fog nodes is obligatory to fully utilize the nodes' total capacity, reduce the average edge /fog nodes access delay of IoT smart devices and minimize the deployment cost of these nodes.

1.3 Dissertation Problem

Edge/ fog nodes' location is critical to the mobile users' latency requirement, especially in the case of large-scale networks like Wireless Metropolitan Area Network (WMAN) that consists of hundreds or thousands of Access Points (APs) or Base Stations (BSs) that mobile users access the edge/ fog nodes through. Another reason for the necessity of optimal nodes' placement is that these nodes are resource-limited compared to the cloud servers. So, edge/ fog nodes' resources must be fully utilized by distributing them perfectly.

An absolute result is that the poor placement of edge/fog nodes will result in a long latency between mobile users and their service nodes. This improper placement will also cause load imbalance among these edge/ fog nodes, where some are overloaded while others are underutilized or idle. Therefore, strategic placement of cloudlets and edge/ fog nodes will significantly improve the performance of various mobile applications, besides reducing the overall network latency, reducing the server's energy consumption, and promoting the coverage rate. Minimizing the total cost is also considered by reducing the number of deployed edge/ fog nodes.

1.4 Dissertation Scope and Limitations

The thesis is mainly taking consideration of edge and fog nodes' placement in large scale smart cities with the objective of minimizing both the overall network latency and workload balancing, and maximizing the edge/ fog system utilization; however, some topics are not addressed in this thesis, such as: Communication cost, and security issues.

1.5 Dissertation Aim and Objectives

The main aim and objectives of the dissertation is to propose a system for finding the optimal number and locations of edge nodes within a smart city. The main stages to implement this aim are as follows:

1. Using density-based clustering technique to define *k* dense regions.
2. Defining the main placement problem objectives: minimizing the network latency, balancing the workload, and maximizing the edge system utilization.
3. Defining the main constraints regarding the placement problem like: how many edge nodes that the base station can attach with, and how many clusters that the edge node can belong to.
4. Developing BAT metaheuristic algorithm for finding the optimal number and distribution of the edge nodes.
5. Comparing the performance of the system with the benchmark placement methods by using the Shanghai telecom dataset.

1.6 Dissertation Contributions

The main contributions of the study are as two parts, depending on which aspect in concern, the improvement of a BAT metaheuristic

algorithm and placing edge/ fog nodes optimally. So, regarding the BAT metaheuristic algorithm improvements, the main contributions are:

1- Dataset reduction for converting the used large number of records with some raw information dataset to smaller meaningful dataset, beside extracting a new feature that make the proposed edge nodes' placement more efficient.
2- Proactive Fractal clustering technique contribute to handle the outliers that are produced from DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Fractal clustering will attach the outliers' base stations to the suitable cluster from the k produced clusters.
3- Using of the Clusters' information, and chaotic maps to improve the BAT metaheuristic algorithm as the following:
    a. proposing a new approach for the initial population by generating two populations, one randomly and the other based on the clusters' center information, and selecting the fittest individuals from both populations to create an improved initial population. The goal is to kickstart the algorithm with more diverse and potentially promising solutions.
    b. Balancing the local and global search of the BAT algorithm using two chaotic maps instead of using a fixed-size movement for random walks. This change enhances the exploration capabilities of the algorithm, allowing it to escape local optima more efficiently and guide the search process more effectively.
    c. Overcoming the stagnation issue, by partitions the search space based on the generated clusters' information. Then generate new solutions and compare their quality with the previously generated

solutions, and selecting the best one. This approach helps the algorithm to avoid getting stuck in suboptimal regions of the search space and therefore it helps handling the stagnation problem.

4- Using of the improved BAT algorithm as an edge nodes' placement method with multi objectives regarding the overall network latency, workload balancing, and edge system utilization.

1.7 Related Works

This section will present the most recent related works regarding edge and fog nodes' placement problem and BAT metaheuristic algorithm improvements. The following Subsections 1.7.1, and 1.7.2 will show these works respectively.

1.7.1 Related Works Regarding Edge and Fog Nodes Placement

There are many works regarding the problem of edge and fog servers' placement. Generally, these works can be organized depending on the proposed method of finding the optimal placement as the following categories:

- Related publications that relay on ***Integer Programming:*** like Kun Cao et al. [7] who use Integer Linear Programming as a strategy for finding the optimal placement of the edge nodes without considering the mobility of the mobile users. The authors' main objective is to minimize the expected response time of multiple base stations and the expected response time of the system, beside maximizing the utility function of base stations. Olga Chukhno et al. [8] also depend on

mathematical modeling in finding the optimal placement of Social Digital Twins (SDT) in edge network, where they use Mixed Integer Nonlinear Programming (MINLP) and Mixed-integer linear programming (MILP) with the objectives of Minimizing the overall network latency. Song Yang et al. [9] propose a system for both placing the cloudlets (edge nodes) optimally on the network, and allocating the requested users' tasks to the cloudlets and the cloud. The main objective is to minimize the total network delay and minimize the energy consumption.

- Related publications that relay on ***Heuristic and Metaheuristic approaches***: like Yuanzhe Li et al. [10] who aim to both minimizing the access delay and maximizing the profit of the edge servers by using Modified Particle Swarm Optimization-based algorithm with parameter q (QPSO). Chun-Cheng Lin et al. [2] use metaheuristic algorithm (Bat- inspired Algorithm (BA)) that is integrated with three local search (LS) methods for dynamical deployment of fog nodes and edge servers, with the objective of maximizing both the weighted network connectivity of fog computing system and the total weighted edge device coverage. Xiaoyu Zhang et al. [11] use Coverage First Search (CFS) algorithm to find the optimal deployment of the edge nodes along with the resource allocation in a polynomial time. Their main objective was minimizing the deployment cost of the servers as well as minimizing the overall latency.

- Related publications that relay on ***Hybrid methods between Integer programming with Heuristic and/or Metaheuristic approaches***: like Xingbing Zhao et al. [12] who use two methods for the placement of k-edge server, where they use a multi-objective optimization problem

under some constraints for placing the edge nodes with optimal system delay and balanced workload, and they use a modified Multi-Objective Non-dominated Sorting Genetic Algorithm with elite policy (MNSGA-II) for optimized system delay and balanced workload. Irian Leyva-Pupo et al. [13] use Integer Linear Programming (ILP) and Heuristic Evolutionary algorithm and modified Hybrid Simulated Annealing (HAS) for finding the optimal number and placement of edge servers in the 5G system, with the aim of minimizing the general cost. Guangming Cui et al. [14] use Integer Programming for finding the optimal solutions for small-scale Robust k- Edge Server Placement (RkESP) problems, and use an approximate approach for solving large-scale RkESP problems efficiently. Robustness is the main objective considered by the authors.

- Related publications that relay on ***Clustering approaches***: like Tero Lähderanta et al. [15] who propose an approach for optimal placement of edge servers with clustering of APs, the authors use a k-mean clustering based approach to solve the problem with the aim of minimizing the weighted distances between edge servers and their associated APs. Bo Li et al. [16] use k-means based clustering algorithm for the deployment of edge server in edge computing environments with the aim of minimizing the system's average completion time.

- Related publications that relay on ***Reinforcement Learning approaches***: like Jiawei Lu et al. [17] and Fei Luo et al. [18] use Markov Decision Process (MDP) for modeling the edge servers' placement problem, and use DESP strategy which is a deep reinforcement learning (Deep Q-Network (DQN))-based multi-objective strategy to find the optimal edge servers' placement. Both aims to minimize the average delay

and balance the workload between edge servers. Mumraiz Khan Kasi et al. [19] also use multi-agent reinforcement learning approach to solve a multi-objective optimization problem for optimally placing edge servers with the same aim of minimizing the average latency and balancing the workload.

- Related publications that relay on *the* ***Hybrid methods between two or more optimization methods mentioned above***: like Xiaolong Xu et al. [20] who use a collaborative method for quantification and placement of edge servers in industrial Cognitive Internet of Vehicles (CIoV). Their collaborative method is implemented by using Canopy and K-medoid clustering methods for initializing the population, and Non-dominated Sorting Genetic Algorithm III (NSGA-III) for achieving solutions with higher QoS and proper (minimum) number of edge servers while minimizing the overall latency.

Zhihan Lv et al. [21] use two methods, one for getting the optimal placement of the edge server (which is computationally ineffective) and the other method for getting optimized (near optimal) placement of the edge servers. First one implemented using Enumeration-based Optimal Edge Server Placement Algorithm (EOESPA) to enumerates all the deployment plans of k edge servers, calculates the average access delay of each plan, and compares the results to obtain the optimal deployment plan. Second one is implemented using Ranking-based Near-optimal Edge Server Placement Algorithm (RNOESPA) to obtain an approximate optimal deployment with low computational complexity.

Yan Guo et al. [22] deal with the problem of edge nodes placement at a candidate locations by using an approximate approach that depends

on k-means and mixed-integer quadratic programming with the objective of balancing the workload between edge servers, and minimizing the service communication delay of mobile users.

Manasvi G. et al. [23] proposed a Social network Aware Dynamic Edge Server placement (SADES) strategy, which uses the information from the overlay social network and the underlay physical network topology to efficiently identify some influential base stations to place the edge servers, in 4G LTE network with the objective of minimizing the operational latency of the 4G LTE network.

Guangming Cui et al. [24] models the placement of k edge servers with consideration of both user coverage and network robustness, they consider two cases, first one for small scale problem where an integer programming-based Optimal approach was used, second one for large scale problem where an approximation sub-optimal approach was used.

Feng Zeng et al. [25] uses two approaches for the effective deployment of edge servers on WMAN. The authors use Simulated Annealing for deploying edge servers with on demand capacity configuration, and Greedy-based algorithm for deploying the same capacity edge servers. The main objectives considered by the authors are minimizing both the number of edge servers and network's latency. Table 1.1 summarizes all important aspects regarding the previously discussed articles.

The dissertation considers the hybrid method between clustering technique and metaheuristic algorithm. The choice of the hybrid method come from the truth that the NP-hard placement problem can't be solved using the traditional mathematical approaches, and then the use

metaheuristic is a better solution. Using the clustering technique will improve the obtained results regarding the number and location of edge nodes.

Table 1.1 A Summary of the Related Works Regarding Edge/ Fog Nodes' Placement Methods

	Solution's Optimization Method				Solution's Main Objectives						
Ref.	Integer Programming	Heuristic/ Metaheuristic	Clustering Algorithms	Reinforcement Learning	Minimize the Latency	Minimize the Cost	Minimize Energy Consumption	Balance the Workload	Maximize the Edge Node Utilization	Minimize the Number of Edge	Maximize Edge System Robustness
[2]		✓							✓		
[7]	✓				✓				✓		
[8]	✓				✓						
[9]	✓				✓		✓				
[10]		✓			✓				✓		
[11]		✓			✓	✓					
[12]	✓	✓			✓			✓			
[13]	✓	✓				✓				✓	
[14]	✓	✓									✓
[15]			✓		✓						
[16]			✓		✓						
[17]				✓	✓			✓			
[18]				✓	✓			✓			
[19]				✓	✓			✓			
[20]		✓	✓		✓					✓	
[21]	✓	✓									
[22]	✓		✓		✓			✓			
[23]		✓			✓						

	Solution's Optimization Method				Solution's Main Objectives						
Ref.	**Integer Programming**	**Heuristic/ Metaheuristic**	**Clustering Algorithms**	**Reinforcement Learning**	**Minimize the Latency**	**Minimize the Cost**	**Minimize Energy Consumption**	**Balance the Workload**	**Maximize the Edge Node Utilization**	**Minimize the Number of Edge**	**Maximize Edge System Robustness**
[24]	✓	✓							✓		✓
[25]	✓	✓			✓					✓	
Our work		✓	✓		✓	✓	✓	✓	✓	✓	

The main objectives of the proposed approach are minimizing the number of edge nodes, network latency and balancing the workload, and maximizing the edge system utilization. By achieving these objectives, the general cost will be already minimized (the smaller number of edge nodes, the lower the cost), also the consumed energy will me minimized (the better the edge system utilization, the lower the energy consumption).

1.7.2 Related Works Regarding BAT Algorithm Modifications

Intensification (exploitation) and diversification (exploration) are two major components of any metaheuristic algorithm. Diversification means exploring the search space globally to generate diverse solutions. In contrast, intensification means focusing on the search in a local region by exploiting the information of the current good solution that is found in this region. For each swarm intelligence algorithm, the exploration capability should be implemented first in order to search in the whole space globally, while the exploitation capability should be considered later by enhancing solution's quantity in the local search process [26].

Selecting the best solution ensures that the solutions will converge to optimality.

The diversification via randomization avoids the solutions being trapped at local optima and, at the same time, increases the diversity of the solutions. A good combination of diversification and intensification ensures global optimality [27].

Several modifications have been proposed to enhance the BAT algorithm's performance since it needs a better balance between exploration and exploitation, so it sometimes fails to find the global optimum and easily gets stuck into the local optima [28][29]. This subsection will briefly describe the modifications and improvements of the BAT algorithm. Table 1.2 summarizes the main improvements aspects regarding exploration, exploitation and improving the initial population.

Selim Yılmaz et al. [26] enhanced the bat algorithm's local search (exploitation) and global search (exploration) characteristics through two modifications using Inertia weight, and modifying the distribution of the population. The BAT algorithm is hybridized with the invasive weed optimization algorithm.

Zaharuddeen Haruna et al. [28] proposed modified BAT algorithm using Elite opposition–based learning to enhance the diversification of the solution search space and modify the inertia weight to improve its exploitation capability. The main objective of using EOBL is to utilize some elite individuals from the current population to generate a corresponding opposite population by using opposition-based learning and depending on the dynamic search boundary. By evaluating both generated populations (current population and the opposite population),

promising regions which may contain the global optimum will be better reached. This will increase the diversification of the algorithm and prevent the premature convergence. Using the number of individuals in the population, the number of iterations, and the current iteration, a weight is generated to utilized with the local search to improve the exploitation too.

Xian Shan et al. [29] proposed a modified search equation with more useful information from the search experiences to generate a candidate solution and incorporate Lévy Flight random walk with the algorithm to avoid being trapped into local optima by improving the exploitation. Furthermore, opposition-based learning is embedded in the algorithm to enhance the diversity and convergence capability. The search space is explored using best solution and its neighbors' solutions using echolocation to balance exploration and exploitation. An opposition-based learning population and the random population are generated, merged and the individuals will be ranked to generate the initial population.

Another modification is the improvement proposed by Min-Rong Chen et al. [30], who suggested an improvement using the Extremal Optimization (EO) algorithm to increase the BAT algorithm's exploitation. The improved update strategy is proposed to obtain the solutions generated from the randomly selected bats to enhance the global search capability (exploration ability) by reducing the dependence on the optimal solution. Beside these improvements, Boltzmann selection and a monitor mechanism are employed to make a balance between exploration and exploitation abilities.

S. Yılmaz et al. [31] improved the exploration mechanism of the BAT algorithm by modifying the equation of pulse emission rate and loudness of bats. With the original BAT algorithm, each bat has only one pulse emission rate and loudness. That is each solution satisfies the condition (rand > ri) will search around the best solution with all dimensions. In the proposed modified BAT, the loudness and pulse rate, which acts as a balance, are equalized to number of problem dimension, where each dimension j of solution i, which satisfies the condition ($rand_j > r_{ij}$), will searches around the dimension j of the best solution and the rest dimensions of solution i keep on seeking the search space.

Daranat Tansui et al. [32] enhanced BAT algorithm's exploitation power by introducing two new random walk processes that made its local search more thorough. The decision on what random walk to use depends on the generated random number. The authors also improved the exploration power by introducing inertia weight to intensify its global search near the end of the optimization process.

Xiaowei Wang et al. [33] proposed an improved BAT algorithm called an Adaptive Bat Algorithm (ABA). Each bat has the ability to adjust its flight speed and direction dynamically and adaptively while searching for food and uses the hunting approach of combining random search with shrinking search to provide better global convergence property and avoid the problem of premature convergence effectively.

Jianqiang Huang and Yan Ma [34] proposed a novel BAT algorithm based on an integration strategy with the aim of enhancing the Algorithm's global search ability. The proposed bat disturbs the local optimum through a linear combination of Gaussian functions with different variances to

avoid being stuck in the local optima. An adaptive weight is used with the velocity equation to balance the exploration and exploitation.

Sha-Sha Guo al. [35] proposed an improved BAT algorithm based on a chaotic map and the algorithm of Levy flight search strategy and contraction factor to improve the search performance and the BAT algorithm's convergence speed and optimization precision.

A. Rezaee Jordehi [36] proposed a chaotic-based BAT swarm optimization algorithm to alleviate the premature convergence problem of the original BAT algorithm. The authors use the chaotic map function in the loudness updating by multiplying a linearly decreasing function by the chaotic map function.

Table 1.2 A Summary of the Main Improvements of the Related Works

Authors	Initial Population	Exploration	Exploitation
Selim Yılmaz et al. [26]		✓	✓
Zaharuddeen Haruna et al. [28]	✓	✓	✓
Xian Shan et al. [29]	✓	✓	✓
Min-Rong Chen et al. [30]		✓	✓
S. Yılmaz et al. [31]		✓	
Daranat Tansui et al. [32]		✓	✓
Xiaowei Wang et al. [33]			✓
Jianqiang Huang and Yan Ma [34]		✓	✓
Sha-Sha Guo et al. [35]	✓	✓	✓
A. Rezaee Jordehi [36]		✓	✓
Our Work	✓	✓	✓

1.8 Dissertation Organization

The rest of this dissertation is organized as follows:

- **Chapter 2 – Scientific Background**

This chapter presents the scientific background of edge and fog computing. A comprehensive study involving the architecture, benefits, applications, characteristics and the challenges of the edge and fog computing paradigms. In addition to edge and fog computing, metaheuristic algorithms are discussed generally and Bat metaheuristic algorithm is discussed more specifically. Besides that, this chapter discuss clustering techniques, and chaotic map functions. Moreover, the performance evaluation technique and metrics regarding both edge/ fog nodes placement and Bat metaheuristic algorithm improvement have been viewed.

- **Chapter 3 – The Proposed System**

This chapter presents the framework of the proposed approaches. It comes into two parts, the proposed model of the edge/ fog nodes placement, and the proposed improved BAT metaheuristic algorithm.

- **Chapter 4 – Result, Analysis, and Discussions**

This chapter presents the evaluation of the performance and the results of experiments as graphs and tables besides a discussion of the implementation of the proposed approaches.

- **Chapter 5 – Conclusions and Future Works**

This chapter presents the conclusion for the dissertation as well as some suggestions for future works.

CHAPTER TWO

SCIENTIFIC BACKGROUND

2.1 Overview

Edge and fog computing regarded a complementary paradigm to the cloud computing, by solving the latency issue caused by remote cloud servers. This chapter discusses the scientific background of Fog and Edge computing by presenting its architecture, challenges, benefits, and their requirements. Following, discussing the most important characteristics of metaheuristic algorithm and their role in solving many optimization problems including edge and fog nodes placement. In addition to that, machine learning techniques like data preprocessing and clustering techniques are discussed. Finally, the performance metrics used with metaheuristic algorithm's improvements has been covered, beside the metrics of evaluating the edge and fog nodes' placement system.

2.2 Edge and Fog Computing

In the era of increasingly connected devices and the proliferation of data, traditional centralized cloud computing architectures face latency, bandwidth constraints, security and scalability challenges [37] . As a result, edge computing and fog computing concepts have emerged as promising solutions to address these limitations and enable efficient processing and analysis of data closer to its source, where edge and fog computing will help in processing the network data that has stringent latency and throughput requirements, while the rest of data with less requirements may be processed by cloud computing [38].

Edge computing refers to the processing and analyzing of the data at or near the network edge, typically on devices such as sensors, gateways, and edge servers. By bringing computation and storage capabilities closer to the

data source, edge computing minimizes the latency and bandwidth requirements of transmitting data to centralized cloud servers. This proximity enables real-time and near-real-time processing, making it suitable for applications requiring low latency, high responsiveness, and efficient bandwidth utilization [39]. Figure 2.1 shows edge computing paradigm.

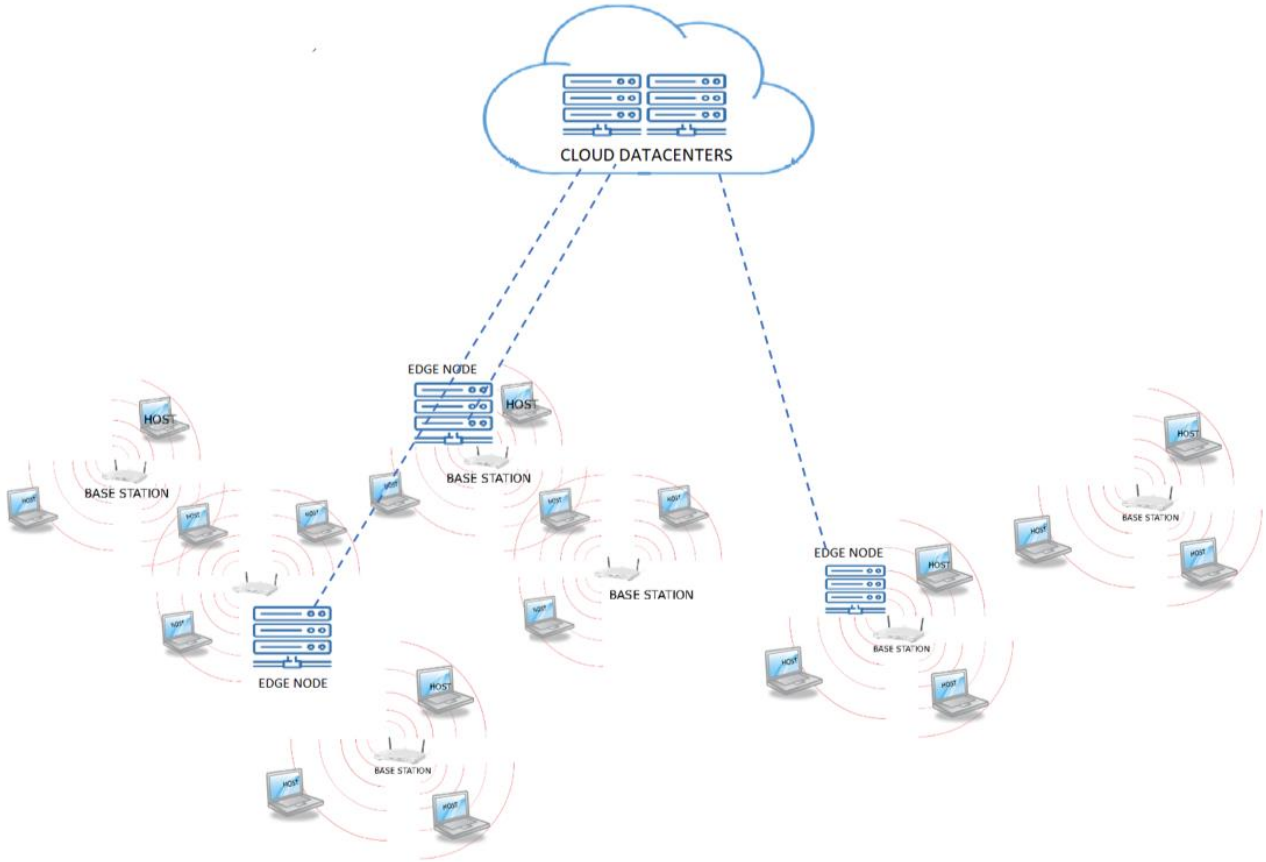


Fig. 2.1: Edge Computing Paradigm

On the other hand, ***fog computing*** extends the edge computing paradigm by introducing an intermediate layer of computing resources between edge devices and centralized cloud servers. Fog nodes, deployed at various points in the network infrastructure, provide additional computational

power and storage capacity, enabling more sophisticated data processing and analysis. Fog computing leverages the distributed nature of resources to distribute the workload across multiple nodes, reducing the burden on individual devices and enhancing overall system performance [40].

The standard fog system has 3-tier architecture, where , the fog nodes are located in tier-2 between IoT devices in tier-1 and the cloud in tier-3, as shown in Figure 2.2 [41]. But, the fog network has an n-tier architecture, offering more flexibility to the system [41]. Within four tier fog system, the edge nodes (mini clouds or Fog instances) are located in tier-3. Another four-tier fog system present two types of fog nodes, where tier-2 and tier-3 are dedicated to low-capacity and high-capacity fog nodes respectively. These low-capacity nodes, like Tablets and Raspberry Pi, are located at tier-2 to be as close as possible to the end devices to process lightweight tasks with minimum latency, transmission, and computation costs. The high-capacity nodes with higher computing power, like servers and workstations, are at tier-3 to handle heavier tasks. Another example for a four-tier fog architecture is by adding a fog orchestrator at tier-3 between the cloud (tier-4) and fog nodes (tier-2) to manage to add and remove fog nodes and maintain services [42]. There is also five-tier fog system [43], and six-tier fog system [44].

Furthermore, edge and fog computing architectures have gained prominence due to their potential to address privacy and security concerns. By keeping data closer to the source, sensitive information can be processed and analyzed locally, reducing the risk of unauthorized access. This aspect is particularly relevant in sectors such as healthcare and finance, where data privacy and regulatory compliance are paramount.

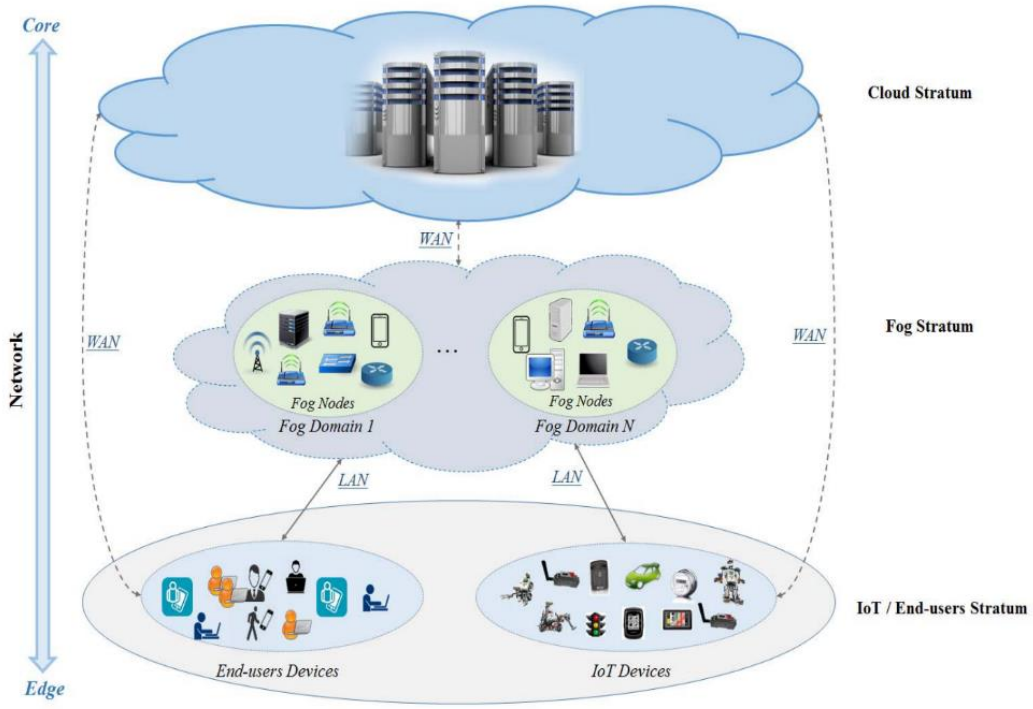


Fig. 2.2: Three-Tier Fog Computing [41]

In conclusion, edge and fog computing offer compelling solutions to the challenges posed by traditional centralized cloud architectures. By enabling localized data processing, reduced latency, and enhanced privacy, these paradigms empower various industries to leverage the vast potential of connected devices and real-time analytics. With continued advancements and adoption, edge, and fog computing are poised to reshape the distributed computing landscape and usher in a new era of intelligent, decentralized systems.

2.2.1 Applications of Edge Computing and Fog Computing

Edge and fog computing find applications in a wide range of domains, including the IoT applications, such as, smart transportation systems & autonomous vehicular Networks, smart city and smart home systems, smart

healthcare systems, smart grid system, industrial automation, and more [45] as Figure 2.3 shows.

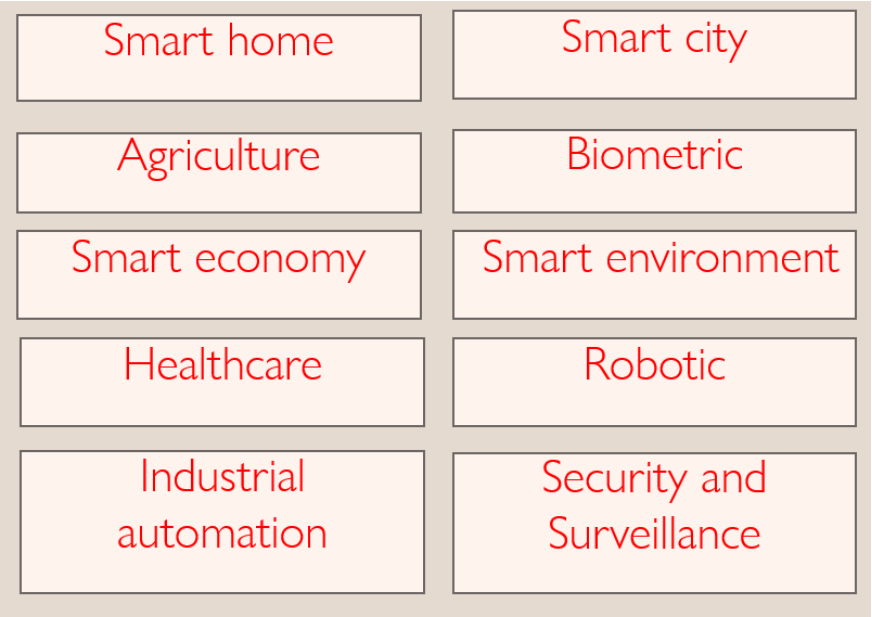


Fig. 2.3: Applications of Edge/ Fog Computing [45]

Edge and Fog computing facilitate real-time analytics, predictive maintenance, intelligent decision-making, and enhanced user experiences by enabling localized processing and minimizing reliance on the cloud for every computational task. Some examples of edge and fog computing applications are:

1- *In smart city:* Edge and fog computing contribute to the development of smart cities by processing data from various sensors, cameras, and devices in real time. This leads to efficient traffic management, waste disposal, energy consumption, and improved public services [41][46]. Smart city is the application takes in consideration throughout this dissertation.

2- *In industrial automation:* Edge and fog computing enhance industrial processes by enabling real-time monitoring, control, and predictive maintenance. Manufacturing plants, factories, and energy production facilities benefit from reduced latency and improved efficiency. [45].

3- *In healthcare:* Edge and fog computing enable real-time monitoring of patient data, remote diagnostics, and timely delivery of critical information to healthcare providers. This is vital for telemedicine, wearable health devices, and patient monitoring [47].

4- *In transportation and autonomous vehicles*: Edge and fog computing facilitate data processing for real-time traffic management, navigation, and vehicle-to-vehicle communication. They are essential for the development of autonomous vehicles [48] [49].

2.2.2 Characteristics of Edge and Fog Computing

Fog and edge computing are innovative paradigms that bring computing resources closer to the data source, enabling real-time processing, reducing latency, and enhancing the efficiency of various applications. The characteristics of fog and edge computing reflect their ability to address the challenges of data-intensive and latency-sensitive environments. Some characteristics are as follows and as Figure 2.4 shows:

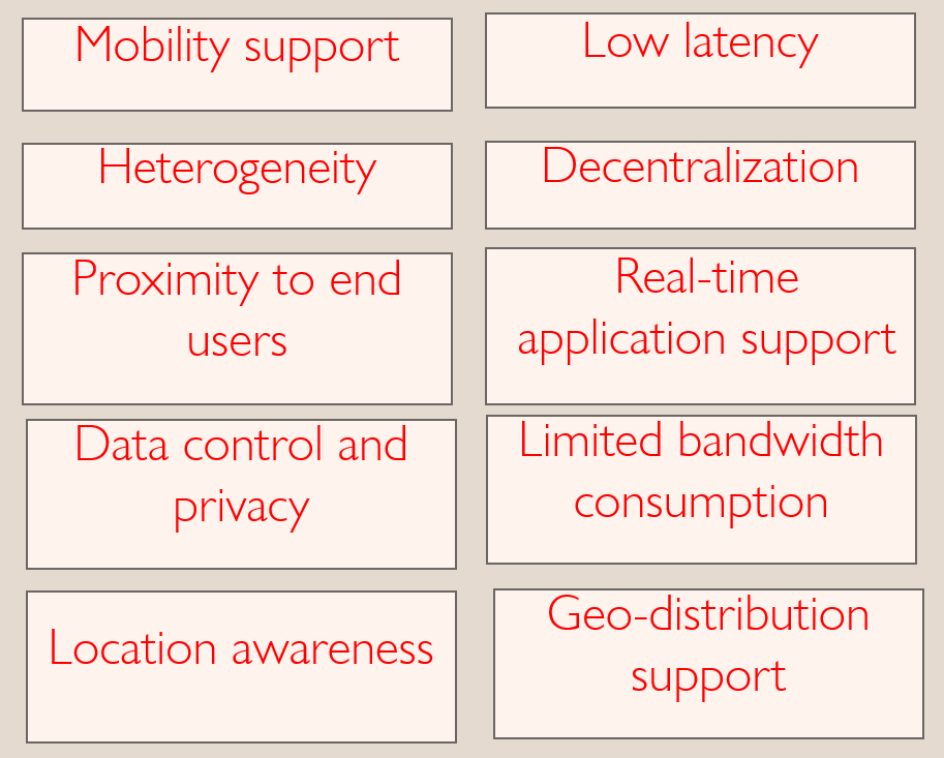


Fig. 2.4: Characteristics of Edge/ Fog Computing [50]

1- *Proximity to Data Sources and low latency:*

Edge and fog computing involve deploying computational resources closer to the data sources, so reduce the need for the data to travel long distances to the centralized cloud servers, so the edge and fog computing are a good solution for delay sensitive applications like live gaming, traffic management and monitoring, smart health, etc. [50].

2- *Real time processing:*

Edge and fog computing enable real-time data processing and analytics, making them suitable for applications that require instant decision-making and response like virtual reality, augmented reality, traffic monitoring, tele-surgery, etc. [50].

3- *Bandwidth efficiency:*

By processing data locally, fog and edge computing reduce the need to transmit large volumes of data to remote cloud servers located thousands of miles away from the end users, and so optimizing bandwidth usage because of the distributed nature of edge and fog computing [50].

4- *Mobility:*

Edge and fog applications need to directly communicate with mobile devices, so it support mobility techniques[49]

5- *Resources heterogeneity:*

Edge and fog environments encompass a variety of devices with different capabilities, ranging from resource-constrained sensors to more powerful gateways [49].

6- *Privacy and data localization:*

Edge and fog computing enable data processing to occur closer to where data is generated, reducing concerns about data privacy and security during transmission [50].

2.2.3 Challenges of Edge Computing and Fog Computing

Beside the benefits that edge and fog computing offer, they also come with several challenges that need to be considered for their successful implementation. These challenges including architecture, resource management, security, scalability, and application-specific requirements. Following are some of the challenges of edge and fog computing:

1- *Latency and Quality of Service (QoS) management:* Appropriate Service Level Agreements (SLA) management techniques are critical to maintaining acceptable QoS in highly dynamic fog system environments [41], so meeting low-latency requirements and ensuring an acceptable QoS across various applications, particularly in real-time systems, is a challenge. .
2- *Privacy and security:* Edge and fog computing are vulnerable to security threats about data security, privacy, and the potential for unauthorized access due to the distributed nature of processing and the implementation of the edge and fog nodes on the network's edge makes it [51][52].
3- *Scalability and management:* As the number of edge and fog devices grows, managing, monitoring, and scaling these devices need more interest [47].
4- *Reliability:* Ensuring fault tolerance and reliable operation in edge and fog environments, where devices may be prone to failures, is a key challenge [53].
5- *Energy consumption:* Energy consumption of edge and fog devices are need to be managed and optimized to minimize the energy waste [10][39].

2.3 Optimization Problems in Edge and Fog Computing

Optimization problems in edge and fog computing encompass a wide range of challenges related to resource allocation, task scheduling, node placement, energy efficiency, and data management, etc. [50]. These problems arise due to the distributed and heterogeneous nature of edge and fog computing environment.

Depending on the involving layers (cloud, fog/edge, and user), the optimization problem can be classified as follows [54]:

1- Optimization problems within all the three layers (Cloud, Edge/ Fog and user devices): offloading the users' tasks to the fog and cloud

servers, that is whether the task are offloaded from the users to the fog nodes, or from the fog nodes to the cloud servers.

2- Optimization problems within two layers (Cloud and Edge/ Fog): determine the problem of distributing the data and applications on the fog servers or on the cloud servers.
3- Optimization problems within two layers (Edge/ Fog and user devices): are the problem of offloading users' tasks to the edge/ fog nodes, and to determine which edge/ fog node to offload the data to.
4- Optimization problems within Edge/ Fog layer only: within this architecture layer, there are many optimization problems as:
   - Data migration: deciding to migrate the data between edge/ fog nodes (from overloaded node to the underutilized one) or not.
   - Task scheduling: giving the priority for executing the task within the edge/ fog node itself.
   - Distributing the physical resources: this problem need to deal with before the operation of the edge/ fog paradigm, it deal with the proper number and placement of edge/ fog nodes.

2.4 Resource Allocation in Edge and Fog Computing

Resource management in edge and fog computing need a high priority for provisioning services and resources. Since edge and fog devices are energy constrained, resource allocation and management affect the network's lifetime and performance directly [48].

The problem of resource allocation in edge and fog computing have three branches [55]:

1- Task offloading problem: where, and how much data need to be offloaded to the edge/ fog servers.

2- Resource provisioning problem: determining the appropriate number of resources (edge and fog nodes) that are need to be deployed in order to serve and execute the tasks.
3- Task allocation problem: determining the order of task and the best mapping of these task to the available resources (edge/ fog servers).
4- Task scheduling problem: is the problem that merge both resource provisioning and task allocation problems.

2.5 Edge and Fog Nodes' Placement Problem

Beside determining the appropriate number of edge/ fog nodes, the proper placement of them are crucial because of the different various demands of users and their mobility.

Depending on the users' service demands, the placement of these nodes must be determined, where busy geographical areas like business centers and crowded areas require more edge devices with high resource abilities than areas with low or moderate services demands. The proper placement of edge servers will cause increasing the efficiency of edge servers' resources and improves the quality of service received by users.

Optimal placement of edge/fog nodes is obligatory to fully utilize the nodes' total capacity, reduce the average cloudlet/fog nodes access delay of IoT smart devices [56] and minimize the deployment cost of these nodes [57].

As described before, Edge servers' location is critical to the mobile users' latency requirement, especially in the case of large-scale networks like WMAN that consists of hundreds or thousands of APs or BSs that mobile users access the cloudlets through [48][56]. Another reason for the necessity of optimal nodes' placement is that these nodes are resource-limited compared

to the cloud. So, nodes' resources must be fully utilized by distributing the nodes perfectly [57].

An absolute result is that the poor placement of cloudlet/fog nodes will result in long latency between mobile users and their service nodes. This improper placement will also cause load imbalance among these cloudlets/fog nodes, where some are overloaded while others are underutilized or idle. Therefore, strategic placement of cloudlets and edge/fog nodes will significantly improve the performance of various mobile applications [56], besides reducing the overall network latency, reducing the server's energy consumption [14], and promoting the coverage rate [17]. Minimizing the total cost is also considered by reducing the number of deployed edge/ fog nodes [58].

Besides many similarities between Edge nodes and cloudlets placement, the most significant difference is that the cloudlet is placed on the AP, while the edge node is placed on the base station [12].

Edge server placement in mobile edge computing is done in different approaches with many objectives like reducing latency, better load balance, increasing the edge servers' efficiency, decreasing the consumed energy, and user cost.

Despite the different methods used for nodes' placement, they almost share the same main common aspects: *Where to place the edge/ fog nodes and servers within a particular area? How many nodes to deploy? How to alternate between shutting down and waking up these nodes to balance the load between them and ensure that no one becomes idle?*

Many techniques and methods are used to address all the above questions. As Figure 2.5 shows, the following points of view that are considered by every method are:

- **Optimization Problem:** that shows the main considered problem, such as placement of fog or edge node, the paradigm that has been viewed (edge or fog computing), if there is another problem considered (like task offloading, migration, or load balancing) or there is only node's placement problem.

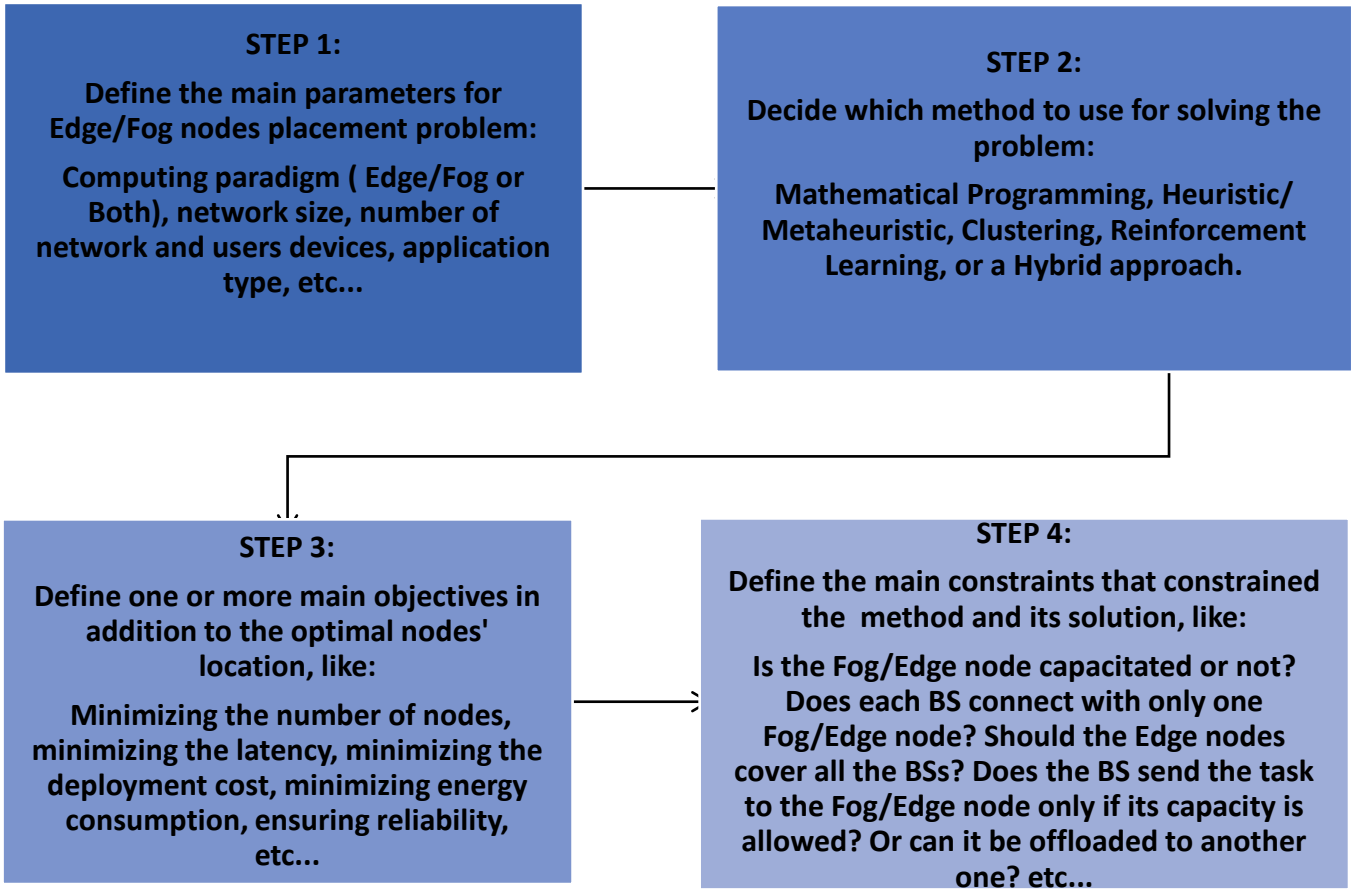


Fig. 2.5: The Methodology of Nodes' Placement Problem

- **Solution/ Technique:** the technique used to address the optimization problem, like (mathematical modeling -integer programming-, heuristic or

metaheuristic methods, clustering methods, reinforcement learning, or even hybrid techniques that mix more than one technique.

- **Objective Function/ Aim:** the main objective (single or multi-objectives) of the optimization problem, which includes minimization or maximization of one or more metrics like minimizing the cost of edge/ fog nodes' deployment, minimizing the network latency, and maximizing the utilization of the edge/ fog nodes.
- **Constraints:** the main restrictions must be considered when finding the available solutions. Many variables and thresholds are defined, like determining the network's upper permitted delay or latency and the allowable upper cost

Figure 2.6 shows the placement of edge servers.

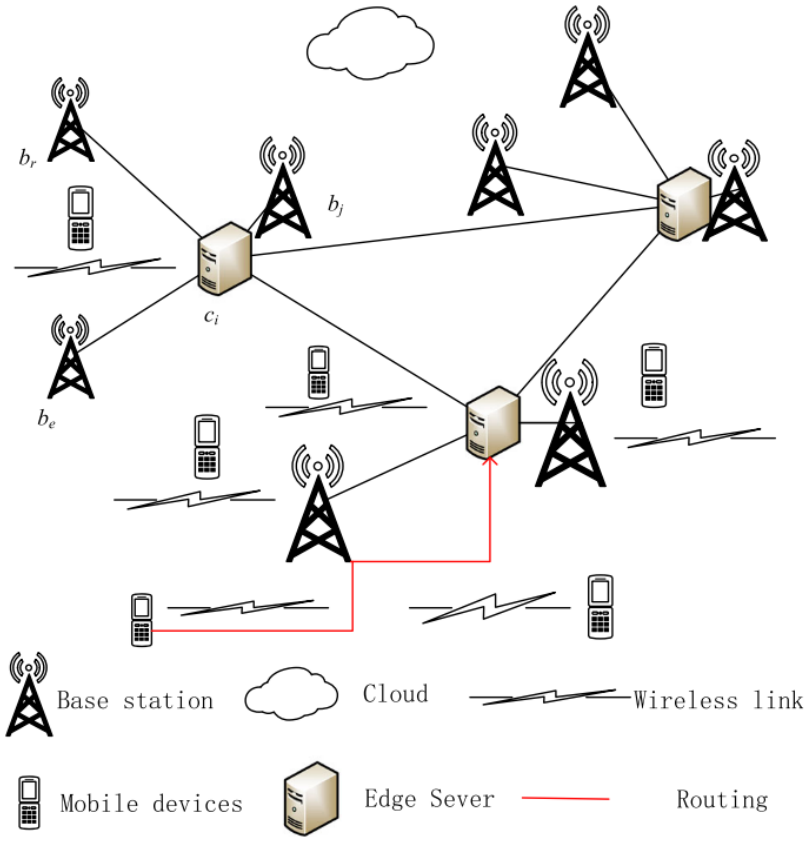


Fig. 2.6: The Placement of Edge Servers [12]

2.5.1 Placement Problem's Methods

There are some different methods used with the problem of edge and fog nodes' placement, each has its strength and limitations. Table 2.1 shows a brief description of these methods used with the problem of edge and fog nodes' placement, while a detailed description is given in the following subsections.

Table 2.1: Brief Description of Placement Problem Technique/ Method

Algorithm	Description
Mathematical Modeling	- Provide optimal solution - It takes a long time especially for large-scale problem cases, so it is used for small-scale problem cases. - Examples: ILP, Integer Nonlinear Programming (INLP), MILP, and MINLP.
Heuristic/ Metaheuristic	- Is an optimization method or a strategy that does not explore all possible states of the problem, but still provide good solutions (near-optimal solutions). - Find the solution in an acceptable time - Used especially for hard problem (NP-Hard problem). - Examples for Heuristic: Local Search algorithms, Tabu Search and Simulated Annealing. - Examples For Metaheuristic: Evolutionary- based algorithms like Genetic Algorithm, and Nature-inspired algorithm like Particle Swarm Optimization (Swarm-based), BAT algorithm, Ant Colony Optimization (Bio-inspired), and the Imperialistic Competitive Algorithm (Human- based).

Algorithm	Description
Clustering	- It is a type of unsupervised learning method of machine learning. - The inferences are drawn from dataset with no class label. - It is the process of dividing the dataset into a certain number of clusters. - Clustering process depends on the similarity between data points, so the data points of each cluster have the same characteristics. - Examples: Hierarchical Clustering, Partitioning Clustering like K-means algorithm and Density Based Clustering like DBSCAN.
Reinforcement Learning (RL)	- It is an area of Machine Learning. - Using RL, the environment (state space) is represented and modeled. - It is used to get a solution for a problem by training the model using a sequence of decisions. - Markov Decision Processes is a method used for formulating the RL problem. - In RL problems, a state space, action space, reward, state transition probability, and discount factor should be defined through the MDP tuple.
Hybrid	- Combining two or more methods from the previously mentioned methods.

2.5.1.1 Mathematical Modeling

Mathematical modeling is used to accurately understand and solve the problem using the objective function, which finds the minimum or maximum solution when some or all variables are integers [4]. When using mathematical modeling, the problem can be formulated as ILP model, INLP model, MILP model, or MINLP model.

classification depends mainly on the structure of the equations (objective function and constraints functions) used in the optimization problem. The optimization problem will be classified as nonlinear programming when any of these functions are nonlinear [59].

Over integer variables, to solve optimization problems with linear objective functions and constraints, use ILP. At the same time, INLP addresses problems with linear objective functions but is subject to nonlinear constraints.

MILP optimization techniques are designed for problems when variables are discrete and continuous, with linear objective functions and/or linear constraints. If both the objective functions and constraints are nonlinear with continuous and discrete variables, MINLP optimization techniques are used [50].

MILP problems are generally solved, and the optimal global solution is found using a linear-programming based Branch-and-Bound algorithm by first ignoring the integer constraints to formulate a relaxed continuous linear programming problem that comprises continuous variables, then check if this relaxed continuous problem has a solution with only integers, then this is the optimal solution, else select one non-integer variable and construct two subproblems with constraints to the non-integer chosen variable. The branch with no feasible solution must be stopped. If

the branch has a solution with only integers, it will be a candidate for the final optimal solution. The branch with a solution with non-integer variables must be branched again. During this process, a search tree is constructed, the root is the initial MILP, and the leaves are the feasible solutions. After that, and when the branching process is completed, all generated integer solutions will be compared to get the final optimal solution [60].

Every candidate solution (a leaf) must be examined with the current feasible solution according to the objective function. If the objective function is a minimization, then the best bound (the best solution among all the leaves) or (solution with the lower bound among all the leaves) will be examined with the current upper bound solution. The difference between the best and upper bound is called the gap, and when the gap is zero, the optimality is demonstrated [60].

To achieve optimality with nonlinear algorithms, they require analytical computation of the derivatives or give an approximate differentiation. Direct search methods are a category of deterministic optimization techniques designed for "black-box" optimization problems. In contrast, Pattern search methods are completely derivative-free algorithms that neither employ or approximate any derivative information nor attempt to linearize any constraint [61].

In a deterministic algorithm, for a particular given input, the algorithm will always produce the same output going through the same sequence of states, but in the case of the non-deterministic algorithm, for the same input, the algorithm may have different outcomes in different runs [61]. The deterministic algorithms can solve the problem in polynomial time, while the non-deterministic algorithms take exponential time and

can't solve the problem in polynomial time as deterministic algorithms [62]. Non-deterministic (Stochastic) methods have a randomness feature. So they can't determine what is the next step, while deterministic methods do not have this feature since they take the same sequence of states to produce the output [61]. The effectiveness of the deterministic and mathematical algorithms depends on the algorithm itself and how many steps it performs from the initial state (input) until the final state (output) is reached.

2.5.1.2 Heuristic/ Metaheuristic

The ILP formulation models an NP-hard problem, so it scales poorly as the problem instance increases. It is the same case with all mathematical modeling INLP, MILP, and MINLP[63][64][65]. A sub-optimal solution must be found instead of an exact solution to tackle this issue. Heuristic approaches minimize the computation time without loss of accuracy and generality based on a set of rules [64] [66].

Heuristic algorithms are classified into three categories. The first one is *construction-based heuristics* which builds solutions step by step based on a set of predefined rules. These rules are: choosing a starting point (initialization), generating the next element to be added to the solutions (selection), selecting the position to add the following element to (insertion), and the stopping criteria. Construction heuristics can achieve feasible solutions concisely, though the solutions may not be as good as those obtained through other heuristics. As a result, they are useful for problems where solutions can be obtained very quickly or to seek initial solutions for improvement heuristics [50]. The second category is the *improvement-based heuristics* that begins with a candidate solution and

makes improvement on it by applying small changes successively through searching the neighbor. This improvement will enhance the solution and produce a better one, and the quality of the final solution will depend mainly on the quality of the initial solution [67]. The third one is the *hybrid method* which mixes the two previous categories [50].

Metaheuristic or nature-inspired search techniques are high-level problem-independent heuristic algorithms that provide a near-optimal solution to an optimization problem by iteratively improving the solution's quality within a reasonable time interval. The metaheuristic algorithm starts with the initial population (a set of individuals representing a candidate solutions). Swarm intelligence, evolutionary algorithm, ecological algorithm, and hybrid (combination of two or more) are the main classification of metaheuristic methods [50]. Metaheuristic methods balance between intensification and diversification phases of the search space. They can efficiently solve NP-hard problems even with a large number of variables and non-linear objective functions [68]. Metaheuristics optimization algorithms efficiently solve many challenging issues; simultaneously, these algorithms face many difficulties while optimizing several problems. For example, when the problem has a high dimension, the algorithm will fail to produce high-quality solutions and cannot achieve approximate global optimum solutions. So, the metaheuristic as a deployment strategy must be chosen depending on the application and the environment's nature [69].

2.5.1.3 Clustering Techniques

A MEC system's efficiency mainly depends on the distribution of workloads and communications in each particular time and geographical

space. The cost of a MEC system depends on the servers' density, capacity, and interconnection.

The situation when one cluster handles high traffic volume while the other handles low traffic or idle will lead to imbalanced MEC clusters. If the clusters are imbalanced, the result will be an inefficient use of resources and so insufficient Quality of Experience (QoE). Optimal placement of MEC servers within the efficient partition of MEC areas will improve the MES system [70]. Some researchers use clustering techniques to achieve this goal.

2.5.1.4 Reinforcement Learning

The previous methods (integer programming, heuristic, and metaheuristic) have many drawbacks. These drawbacks include poor scalability (NP-hard), sticking in the local optima, and difficulty tuning the metaheuristic algorithms' parameters [18]. To address these drawbacks, node placement methods that depend on reinforcement learning are proposed.

2.5.1.5 Hybrid Methods

Combining more than one different method to achieve the aim of optimal placement of fog/ edge nodes is used to get a better solution than the particular method. This mix enables achieving a better solution to overcome the drawbacks of some methods and get the benefit of others' advantages.

2.5.2 Placement Problem's Parameters

Each technique is supported with some parameters, and has its main objectives, and the solution is restricted with some constraints.

When dealing with the parameters, some concerns must take place. The ***computing paradigm*** must first be considered. It deals with whether the placement is for fog nodes in a fog computing system or edge servers in an edge computing system.

Heterogeneity is another feature; it deals with the main features of the deployed nodes, whether they have different costs, communication ranges, and resource capacities. Authors in [71][72][73] consider that the deployed edge servers and cloudlets are heterogeneous.

Another important thing that must be determined is the ***network deployment model***; authors in [74] consider urban, suburban, and rural scenarios as used network deployment models for cloudlet placement over the optical access network. The cloudlets can be placed in the field, remote node (RN) and optical line terminal (OLT), or central office (CO) locations depending upon computation task requirements and subject to the optimized connectivity with Optical Network Units (ONUs) as the authors in [75] who also consider the placement of edge nodes in the Time-Division Multiplexed Passive Optical Network (TDM-PON) based on optical and Fiber-Wireless (FiWi) access networks. Authors in [76][77][25] consider cloudlet placement in WMAN. While authors in [78] consider cloudlet placement over a 4G cellular network. Authors in [75] present a method for placing cloudlets in SDN-based IoT network architecture. Authors in [91] propose a collaborative approach for the quantification and placement of edge servers in industrial Cognitive Internet of Vehicles networks, as the authors in [13] who also propose a method for efficient deployment of edge nodes in 5G-enabled urban

vehicular networks.

In addition to the discussed parameters, a ***static and dynamic feature of the network environment*** determines whether the placement considers the users' movement and the mobility of the IoT devices. A static framework of edge/fog nodes' placement considers that the network is static in time and does not consider user mobility and virtual machine (VM) mobility into account.

In contrast, the dynamic placement framework considers the case when the user exits from the range of one edge node to another. So its tasks must be offloaded (migrated) to the newly entered edge node. This movement will cause a difference in the load on various edge nodes. Some will become overloaded, and others will become underloaded or idle because of the network delay fluctuation throughout the day [75][77]. Authors in [74] only consider the static deployment of cloudlets over an existing access network infrastructure, as well as authors in [75]. Authors in [78] propose static and dynamic planning with VM bulk and live migrations. Authors in [77] also present both static and dynamic network planning.

2.5.3 Placement Problem's Objectives

The nodes' placement problem is a multi-objective optimization problem that simultaneously optimizes more than one objective, like optimizing the system delay and workload balancing. The performance comparison is more complicated in the multi-objective optimization problem than in the single-objective optimization problem, so objective weighting is used for transforming the multi-objective optimization problem into a single-objective problem [12]. There are many objectives besides the optimal placement of edge/ fog nodes. These objectives must be assigned clearly after determining

the main parameters.

Minimizing the number of edge/ fog nodes; the number of edge/fog nodes must be well optimized; authors in [79] show that too large or too small a number of fog nodes decreases the average data rate. Authors in [25] propose a system for deploying edge servers in WMAN with the objective of reducing the number of edge nodes while satisfying the latency constraints.

The number of deployed edge servers and the cost of the deployment is highly related [12]. ***Minimizing the nodes' deployment cost*** objective is met by reducing the number of deployed edge/ fog nodes; few nodes imply less cost [58]; that is because the edge servers deployment cost is mainly related to two factors which are site rentals and computation demands. Site rentals mean the deployment cost will increase as the number of locations selected to deploy edge servers increases. Computation demands mean the number of edge servers must increase for greater computation demand, resulting in higher costs [25].

Network Robustness; as the cloud servers, edge/ fog nodes deployed in the dynamic and distributed edge computing and fog computing are subject to runtime failures because of many different events, e.g., software exceptions, hardware faults, cyberattacks, etc. Such failures result in the disconnection of the mobile users from the edge/ fog network if they are not covered by any other edge/ fog node, and so their quality of experience will decline immediately and significantly, especially for those of latency-sensitive applications. This issue is especially critical in areas with high user density. Deploying more edge servers within a specific area can increase the robustness of the network in that area. However, this increases the placement cost. Thus, the trade-off between user coverage and network robustness must be appropriately managed in edge server placement [24]. Authors in [14]

propose a method for placing k- edge servers to maximize the robustness.

Determining the optimal location of the edge/ fog nodes is another essential objective and challenge; it is determining the location of a limited number of edge/ fog nodes among a large number of candidate locations (ex., Available Base Stations, Access Points, or Road Side Units). These final locations directly relate to the final placement effect [12].

Minimizing the overall network latency; is one of the main objectives and benefits of edge and fog computing. When addressing the placement problem, minimizing the overall latency must be one of the main concerns. The delay of load from the base station and its edge node is *the communication delay*. The minimum communication delay of the users of a base station is achieved when the edge node is co-located with that base station [22]. *The access delay* to cloudlets or edge/fog nodes is an essential factor affecting the performance of the services provided by these nodes, mainly composed of *propagation delay* and *forwarding delay* [80]. The overall system delay is the sum of system *transmission latency* and system *queuing latency* [12].

Minimizing the edge/ fog nodes' total energy consumption; this objective is related to the edge/fog nodes' utilization, where the energy of an idle edge/fog node is no less than 66% of the energy of a fully utilized one, so to minimize the energy consumption, ***Maximizing the edge/ fog nodes utilization*** must be met, by reducing the number of nodes working in low utilization [10].

Balancing the workload between the edge/ fog nodes; the workload of the edge/fog node is the user requests offloaded through base stations that link with that edge node. Balancing the workload between these nodes is the objective that ensures that the edge nodes are neither overloaded nor underloaded, or even idle [22].

Satisfying the Quality of Service level; authors in [81] detail the main attributes that influence the QoS level as 1) Network throughput (Gb/s): the transferred data rate between two endpoints, without losses (successful transfer) in a unit of time. 2) Network latency (ms): the transferred time of a packet across the network. 3) Packet loss (%): the percentage of lost packets in the connection between the client and the running software service (component). 4) CPU utilization (%): the sum of work handled by the CPU varies according to the system workload of the deployed application. 5) Database throughput (requests/s): the rate at which the database processes the read/write requests. 6) Database Read/Write latency (ms): the time required to fulfill read/write operations begins when the database instance receives a client request and ends when it responds. 7) Cost ($/month): the monthly cost of using the Cloud infrastructure. QoS level is one of the objectives to consider in the deployment problem. Authors in [52] propose a solution to maximize the QoS level and suggest that the QoS is measured as the number of completed tasks before the deadline.

2.5.4 Placement Problem's Constraints

There are many constraints; each author considers some constraints when designing the solving method; ***edge server/ fog node capacity;*** the capacity of edge nodes is of two types. The first one is the on-demand edge servers (servers with customized capacity determined by the amount of computation tasks offloaded). In contrast to the first one, the second one is that all edge servers' capacities must be identical and limited [25]. Authors in [73] consider the cloudlet's resource constraints on the placement decision because the assignment of user-to-cloudlet workload should not overload the cloudlet. Authors in [40] restrict the workload hosted in each fog node (both

strict and flexible workload) by guaranteeing that this workload is never greater than the fog node's capacity (each fog node has a number of servers, so to get the capacity of the fog node, the number of servers is multiplied by the capacity of a single server). Authors in [10][12] proposed that the edge server's workload is less than the maximum workload that the edge server can handle (edge server capacity).

Another constraint is determining ***where and how to deploy the node.*** For example, authors in [10][12][80] [82][21] suggest that a cloudlet/ edge server can only correspond to one AP/BS, and each AP/ BS can only connect to one cloudlet/ edge server. Authors in [83] restrict the deployment of the cloudlets only to highly populated areas.

Edge/ Fog node deployment cost; some authors determine a maximum budget (cost) for node deployment, as authors in [84] who restrict the cost of edge nodes' deployment to a predefined budget.

Network delay threshold; is one of the main objectives of edge and fog computing; minimizing the network delay is the first aim of almost all the proposed solutions. The edge servers must be placed as close as possible to the end users to minimize the network delay. Authors in [25] propose an upper bound of access delay, which users can tolerate as a constraint for the proposed placement solution. This is achieved by ensuring one hop distance between users and the edge servers within APs.

Network QoS level; as shown earlier, many attributes influence the QoS level, authors in [75] determine the QoS latency time as a constraint with targets of 1 ms, 10 ms, and 100 ms to find optimal cloudlet placement locations to guarantee the good performance of the network, and the authors in [85] consider the QoS latency time for all users in the network as a constraint. Authors in [25] restrict their proposed work by limiting the cluster

size and the degree of each node within a cluster to ensure the QoS requirements because if the node has too many neighbors, and if the cluster has too many nodes, then the QoS will decrease.

2.6 Benchmark Edge and Fog Nodes' Placements Methods

Many well-known placement techniques exist, like Greedy First-Fit, random, Top-K, Greedy First-K, Heaviest-AP First Placement strategy, and the K-median, K-means algorithms [12][86][87][63]. These techniques are used as a comparative technique to evaluate the performance of the proposed placement solutions that are depend on the previously discussed methods.

2.6.1 Top-K Approach

This method selects the heaviest k BSs/ APs locations to deploy the edge/ fog nodes. AP/ BS is a top-K AP/ BS if the number of user requests at it is one of the top-K values. All remaining BSs or APs also assigned to the closest edge/ fog node [88][22]. All BSs are sorted in descending order according on some features (like, BS's Workload, BS's number of received requests, … etc.) then the first-K BSs are selected as the Top-K BSs.

2.6.2 Top-DoF Approach

DoF is the Degree of Freedom that shows the number of neighbors within an accepted distance. This algorithm has the same structure as Random algorithm, but the selecting criteria of the edge servers are different. In Top-DoF, the nodes are sorted based on the number of their neighbors. The node with the most neighbors is selected as the edge server [58].

2.6.3 Random Approach

This algorithm randomly selects K BSs/ APs locations to deploy edge/ fog nodes. All remaining BSs or APs will be assigned to the closest edge/ fog node [12] [89].

2.6.4 K-means Approach

K-means clustering approach is used to cluster the BSs to k clusters based on their locations, edge/ fog nodes will be placed at the closest BS to the cluster's center, and the BSs within each cluster will be assigned to the cluster's edge/ fog node [12]. One method for finding the optimal value for the parameter K is by implementing the K- means algorithm repetitively by increasing the parameter K until satisfying the load and distance constraints [58]. This approach depends mainly on the communication delay and does not consider the workload balance, so the produced k clusters will minimize the overall communication delay [22].

2.6.5 Optimal Approach

This method iterates through all the possible edge/fog node placements and users assignments to find the optimal placement with the optimal system response time [77].

2.7 Metaheuristic Algorithms

Metaheuristic algorithms are a class of optimization algorithms that are used to solve complex problems. These algorithms are based on heuristics and they do not guarantee finding the global optimal solution, but they are often able to find near-optimal solutions in a reasonable amount of time.

Metaheuristic algorithms have been applied to a wide range of problems in various fields, including engineering, finance, medicine, and logistics, among others. Metaheuristics are used to solve optimization problems by the process of searching optimal solutions to a particular problem of interest. The process of searching can be carried out using multiple agents which essentially form a system of evolving solutions using a set of rules or mathematical equations during multiple iterations. These iterations carry until the solution found meets some predefined criterion. This final solution (near optimal solution) is said to be an optimal solution and a system is deemed to have reached a converged state [90].

Metaheuristic algorithms are a class of optimization algorithms that are designed to find good solutions for difficult optimization problems.

Unlike traditional optimization algorithms, which typically work by systematically searching through the space of possible solutions, metaheuristic algorithms are often inspired by natural or social phenomena, and may use probabilistic or stochastic methods to search for solutions.

There are many different metaheuristic algorithms, each with its own strengths and weaknesses, and the choice of algorithm depends on the specific problem being solved. Some of the most commonly used metaheuristic algorithms include:

1. Genetic Algorithms (GA): GA is first proposed by J. H. Holland, in 1992 [91], and it is inspired by natural selection and genetics, and uses a population of candidate solutions to iteratively improve the quality of the solutions over time. It has been applied to a wide range of optimization problems, including scheduling, routing, and clustering.

2. Ant Colony Optimization (ACO): ACO is first introduced by Marco Dorigo, in 1992 [92], and is inspired by the behavior of ants, which use pheromone trails to communicate and coordinate their movements. It works by simulating the behavior of ants searching for food, and uses the pheromone trails to guide the search for the optimal solution. ACO has been applied to a variety of problems, including routing, scheduling, and graph problems.
3. Particle Swarm Optimization (PSO): PSO is first proposed by J. Kennedy and R. Eberhart, in 1995 [93], and is inspired by the collective behavior of social organisms such as flocks of birds or schools of fish. It works by simulating the behavior of particles moving through a high-dimensional search space to find the optimal solution. It has been successfully used for optimization problems in various fields, including engineering, finance, and biology.
4. Bat Algorithm: it is a swarm intelligence-based metaheuristic optimization algorithm introduced by Xin-She Yang in 2010 [94] as a global optimization algorithm that can solve complex optimization problems effectively and be applied to various real-world optimization problems in engineering, science, and economics. The Bat algorithm effectively solves many optimization problems, including continuous, discrete, and multi-objective problems. The algorithm is also known for its simplicity, fast convergence, and ability to handle noisy and dynamic environments.

2.8 General Types of Metaheuristics Algorithms

Due to technological development, many complicated optimization problems appear in many different fields, like engineering, manufacturing, information technology, and economic management. [95]. These problems are solved using optimization algorithms. Optimization algorithms are categorized as deterministic algorithms and stochastic algorithms. Deterministic algorithms, like the simplex method in linear programming, usually need gradient information. The problems with one global optimum are effectively solved by deterministic algorithms, but for problems with many local optima or problems with unavailable gradient information, they may be invalid. On the other hand, stochastic algorithms require only the information of the objective function. Heuristic and metaheuristic are generally the two types of the stochastic algorithms [96]. Heuristic approaches are used to find the optimum or at least near-optimum solutions. These approaches do not guarantee to reach that global optimum solution. A high-level heuristic is called a metaheuristic [97].

Metaheuristic algorithms can be classified in many ways. Population-based and trajectory-based are one way of classification. The population-based algorithm like PSO algorithm initializes the population randomly, and at each iteration, the population will be enhanced and substituted partially or totally by the newly generated population (solutions). On the other hand, simulated annealing uses a single agent or solution that moves through the design space or search space in a piecewise style.

A better move or solution is always accepted at each iteration, while a not-so-good move can be accepted with a certain probability. The steps

or movements trace a trajectory in the search space, with a non-zero probability that this trajectory can reach the global optimum [96][98]. Another classification is based on whether the algorithm is nature inspired or not. Nature-inspired algorithms are classified as Swarm Intelligence and Evolutionary Algorithms. Swarm Intelligence like PSO, BAT inspired Algorithm, Ant Colony Optimization. Genetic Algorithm is an example of Evolutionary Algorithms. Examples of non-nature-inspired algorithms are Simulated Annealing and Tabu Search [99]. Other classifications are used as whether the algorithm is deterministic or stochastic, memory or memoryless, and whether it is iterative or greedy [98].

2.9 Applications of Metaheuristic Algorithms

Metaheuristic algorithms, for example, Simulated Annealing (SA) [100] and PSO [93], are very powerful in solving hard optimization problems, so they have been applied in almost all significant areas of engineering and science and industrial applications [101]. For example, the optimum weights of the artificial neural network are found by using PSO instead of using the Backpropagation algorithm due to its better classification accuracy and faster processing time when compared with the Backpropagation algorithm [102], and BAT algorithm is used for attributes and features selection [103].

Generally, some prominent applications of metaheuristic algorithms [104][105]:

1. *Combinatorial Optimization:* Metaheuristic algorithms like (GA), (PSO), and (ACO) are widely used for solving combinatorial optimization

problems, such as the Traveling Salesman Problem (TSP), Knapsack Problem, Vehicle Routing Problem (VRP), and Graph Coloring Problem.

2. *Engineering Design and Manufacturing:* Metaheuristic algorithms are applied in engineering design optimization to find optimal or near-optimal solutions for complex design problems, such as structural optimization, aerodynamic shape optimization, and material selection.
3. *Machine Learning and Data Mining:* Metaheuristic algorithms are employed in feature selection, hyperparameter tuning, and training of machine learning models. Evolutionary Algorithms and Bayesian Optimization are used to optimize hyperparameters of machine learning algorithms.
4. *Scheduling and Resource Allocation:* Metaheuristic algorithms are applied to solve scheduling and resource allocation problems in various domains, including transportation, manufacturing, project management, and healthcare.
5. *Network Design and Routing:* Metaheuristic algorithms are used in network design to optimize the placement of network components and routing of data packets in communication networks.
6. *Financial and Economic Applications:* Metaheuristic algorithms are utilized for portfolio optimization, stock market prediction, and risk management in finance and economics.
7. *Robotics and Control:* Metaheuristic algorithms are employed to optimize control parameters for autonomous robots and multi-agent systems.
8. *Bioinformatics:* Metaheuristic algorithms are used in bioinformatics for tasks such as protein structure prediction, gene expression analysis, and sequence alignment.

2.10 BAT Metaheuristic Algorithm

BAT Algorithm uses sound waves and echoes to determine the location and nature of objects is termed echolocation [95]. The algorithm is inspired by the echolocation behavior of bats, a biological phenomenon bats use to navigate in the dark. The BA mimics the behavior of bats in searching for food. Bats use their echolocation system to detect prey and fly toward it. They emit ultrasonic waves and listen to the echoes reflected from their surrounding objects to locate their prey. The BAT algorithm employs the same strategy, where bats represent candidate solutions, and their frequency of emitting pulses corresponds to the quality of their fitness function.

2.11 Original BAT Algorithm Steps

Bats can forage and can accurately avoid the obstacles by its echolocation ability. Original BAT algorithm can be explained by doing three phases as shown in Algorithm 2.1 [94], these phases are: initialization of the bat's, movement of the bats, and the update of the loudness and pulse emission rate. In detail, these three phases are as the following steps:

Step 1: Initializing bat position, velocity, frequency, pulse rate, and loudness:

The algorithm starts by randomly initializing the population of bats within the search space, where each bat represents a potential solution to the optimization problem. The position and velocity of each bat are randomly generated within the search space. Then, the bats are sorted based on their fitness values, and the best bat in the population is identified as the global best solution.

The generation counter is initialized to 1; the positions of each D dimension of every bat in the population are initialized randomly to take a value within the available value range for each dimension as described in Equation (2.1), along with their velocity v_i; frequency f_i, loudness A_P, and pulse rate r_i.

$$x_{ij} = x_{min} + Rand(0,1) * (x_{max} - x_{min}) \quad \text{... (2.1)}$$

Where (i = 1, 2, …., n) (j = 1, 2,…., D) denotes n bats in the population, each with D dimensions, each dimension within the range of the available boundaries x_{min} and x_{max}.

The initial loudness A^0 is randomly generated within the range of [1,2], while the initial pulse rate is generated randomly within the range [0,1].

Step 2: Adjusting the frequency, updating velocities and positions of the candidate solutions to generate new solutions:

Each bat searches for the optimal solution by using echolocation. The bat's position is updated based on its current position, velocity, and random walk component. The frequency of emitting pulses, the loudness of the emitted pulse, and the pulse emission rate are the three primary parameters that control the bat's position update. The frequency of emitting pulses determines the step size of the bat's motion, while the loudness of the emitted pulse corresponds to the magnitude of the step. The pulse emission rate controls the exploration-exploitation balance of the algorithm, where a high rate leads to more exploration and a low rate leads to more exploitation.

Each bat is assigned a frequency and loudness value that controls its movement. The loudness decreases over time, and the frequency increases, allowing the bats to explore the search space effectively.

The frequency is adjusted, the velocity is updated, and a new solution is generated as described in Equations (2.2), (2.3), and (2.4), respectively.

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta \quad \text{... (2.2)}$$

$$V_i^t = V_i^{t-1} + (x_i^t - x_*)\,f_i \quad \text{... (2.3)}$$

$$x_i^t = x_i^{t-1} + v_i^t \quad \text{... (2.4)}$$

Where f_i is the pulse frequency that affects the i_{th} bat's velocity, f_{max} and f_{min} are the maximum and minimum available values of f_i. β represent a random number within the range [0,1]. x_* is the best position found within the population.

Step 3: Applying local search

Some bats perform a local search by randomly adjusting their positions to efficiently explore the search space using the random walk represented in Equation (2.5).

$$x_{new}^i = x_{old}^i + \varepsilon\, A^t \quad \text{... (2.5)}$$

After generating a new solution and if the ith bat's pulse emission rate is less than a randomly generated value, local search using random walk is used to create a solution using one solution x_{old}^i among the current population and generating new one x_{new}^i. A^t is the average loudness of all the bats, while ε is a random variable $\in$ [0, 1] showing the direction and intensity of the random walk.

Step 4: Accepting the solution and updating the loudness and the pulse rate:

The loudness and frequency of each bat's echolocation pulse are updated based on the quality of the solution found in the previous iteration. If the generated random variable is greater than the Loudness and the fitness value of the current solution is better than the previous one, then accept the solution, then increase the pulse rate and decrease the loudness as described in Equations (2.6) and (2.7), respectively.

$$A_i^{t+1} = \alpha A_i^t \quad \text{... (2.6)}$$

$$r_i^t = r_i^o\,(1 - e^{\gamma t}) \quad \text{... (2.7)}$$

Where α and γ are two constants, and the variation with respect to iterations in A_i^t and r_i^t is given in the following Figures 2.7 and 2.8 respectively [106].

Step 5: Update:

The best solution found by the bats is updated. The algorithm iteratively updates the position and velocity of each bat until the stopping criteria are met. The algorithm's stopping criteria can be defined based on the number of iterations or the convergence of the fitness values of the bats, so Steps 2-4 will be repeated until a termination criterion is met.

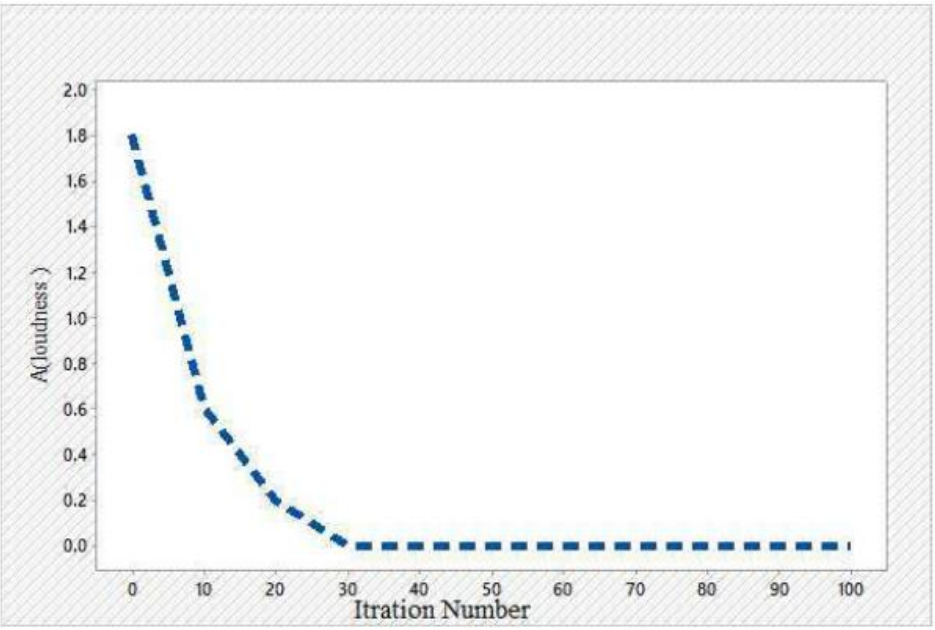


Fig. 2.7: A_i^t Value decreasing Through Iterations

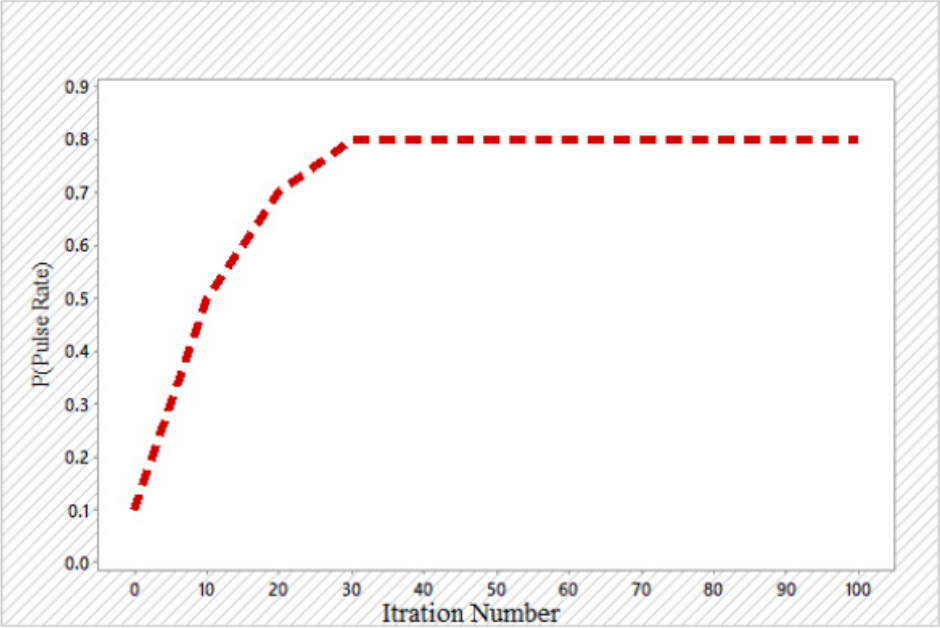


Fig. 2.8: r_i^t Value Increasing Through Iterations

The standard parameters' minimum and maximum values used in the BAT algorithm are shown in Table 2.2 below.

Table 2.2 Standard Values of BAT Algorithm's Parameters [94]

Parameter	Min value	Max value
Frequency	0	100 (depending on the domain size of the problem of interest)
Velocity	0 or random value between [0, v_max]	v_max depends on the problem being solved
Position	Min and Max values depend on the domain size of the problem of interest.	
Pulse rate	0	1
Loudness (2 examples)	1	100 (it is the initial value)
	0	1 (it is the initial value)

Algorithm 2.1 Original BAT Algorithm

Begin
Objective function f(x), x = (x1, ..., xd)^T
Step 1: Initialization.
- Set the generation counter t = 1;
- Initialize the positions of the population x_i (i = 1, 2, ..., n) Equation (2.1), and their velocities v_i
- Initialize frequency f_i, loudness A_p, and pulse rate r_i

Step 2: While the termination criteria are not satisfied or t < Maximum Generation **do**
- Adjusting the frequency updating velocities and positions of the candidate solutions to generate new solutions Equations [(2.2) – (2.4)],
- **if** (random number > r_i), **then**
    - Select a solution among the better solutions;
    - Generate a local solution around the selected solution as in Equation (2.5),

  end if
- Generate a new solution by flying randomly
-
- i**f** (random number < A_i & $f(x_i) < f(x^*)$), **then**
    - Accept the new solution
    - Increase r_i and reduce A_i according to Equation (2.6) and Equation (2.7),

  end if
- Rank the bats and find the current best x^*
- $t = t+ 1$;

Step 3: end while
Step 4: Post-processing the results

2.12 Limitations of the BAT Metaheuristic Algorithm

There are two components which have crucial significance in evolutionary algorithms, exploration and exploitation. Exploitation is the local search ability and exploration is the global search ability. Exploitation can be viewed as the phenomena of optimization algorithm that updates the best solution with the solution it has found so far by searching a small area of the landscape. A right balance between both of these components is crucial to get highly successful results from the algorithm [106].

The original BAT algorithm has good exploitation but poor exploration [32], so it can easily get trapped at a local minimum of most multimodal test functions. This poor exploration ability of the algorithm causes another issue, which is the premature convergence under some conditions [107].

2.13 Improvements of the BAT Metaheuristic Algorithm

Improving any metaheuristic algorithms consider enhancing their performance, efficiency, and convergence speed. Regarding the BAT algorithm, there are many improvement Strategies:

2.13.1 Parameters Tuning:

Optimizing algorithm-specific parameters, such as the rate of pulse emission and loudness reduction, can significantly influence the algorithm's performance. Parameter tuning ensures the algorithm behaves effectively for different problem domains [31]. As example, in [31] the BAT algorithm is improved by updating the equations of pulse rate and loudness modifications. The update depends mainly on considering each dimension of the solution. The equation of generating candidate solutions around the best solution is also

updated to participate each dimension independently. The improved BAT has better exploration capability than the original one.

2.13.2 Population Initialization:

Proper initialization of the initial population can help the algorithm start the search in promising regions of the solution space, potentially leading to faster convergence and improved solution quality [35], The initial population's quality mainly affects the algorithm's convergence and has a role in the quality of the produced solution [108]. Authors in [35] divide the population into K subpopulations, and then calculate the fitness function value of each subpopulation, where each subpopulation performs a global search only on the island, and depending on the calculated fitness function value, the best individuals from each island will form the algorithm population.

2.13.3 Adaptive Strategies:

Adaptive mechanisms for parameters like pulse emission rate and loudness can enable the algorithm to dynamically adjust its behavior during optimization, enhancing its exploration-exploitation balance [36], where the authors in [36] incorporate chaotic-based strategies into BAT algorithm to mitigate its premature convergence problem, and so outperforms conventional BAT algorithm.

2.13.4 Local Search Techniques:

Integrating local search techniques within the metaheuristic can refine solutions and enhance the algorithm's ability to converge to optimal or near-optimal solutions [2] [106]. Authors in [2] apply BAT algorithm as a solution for fog nodes' placement problem, and increase the solution searching ability

of BAT algorithm by integrating it with three local search methods. Local search technique is applied to the half of bats with better fitness values than the other half. The idea of the proposed local search methods is to move the positions of one or multiple fog nodes locally while the other fog devices stay at the original positions.

2.13.5 Hybridization With Other Algorithms:

Combining the strengths of the BAT Algorithm with other metaheuristics or optimization techniques can lead to improved performance and robustness [109]. Authors in [109] propose a new BAT algorithm, called Hybrid BAT Algorithm that was obtained by hybridizing the original BAT using the Differential Evolution (DE) technique. DE technique is used in the hybrid BAT in modifying the produced solution on each iteration.

2.14 Clustering Techniques

Generally, clustering is an unsupervised learning task that aims to find and identify distinct groups (clusters) within multidimensional data based on some similarity measure (e.g., Euclidean distance). These clusters are more similar than others [110][111][112]. The main clustering algorithms are hierarchical, partitioning, and density-based clustering [113] as the following subsections describe.

2.14.1 Hierarchical Clustering

Splitting or merging techniques are used by the algorithms of this category to generate a cluster tree or dendrogram. The decision tree shows the sequence of clustering where each clustering is a partition of the dataset. There are two types of algorithms, divisive (Top-Down clustering) and

agglomerative (Bottom-Up clustering) [113][114]. Divisive algorithm uses splitting to generate the decision tree while agglomerative algorithm uses merging to generate the decision tree [112]. With the divisive hierarchical algorithms, all the patterns assigned to a single cluster initially. Then, splitting is applied to a cluster in each stage until each cluster consists of one pattern. while agglomerative hierarchical algorithms are initially start with each pattern assigned to one cluster. Then, merge the two most similar clusters together. The merging step is repeated until all the patterns are assigned to a single cluster [115] [112].

CHEMELEON [116], and CURE (Clustering Using Representatives) [117] are agglomerative hierarchical clustering algorithms. Bisecting k-Means (BKMS) [118] is a divisive hierarchical clustering algorithm.

2.14.2 Partitioning Clustering

Partitioning algorithms are based on specifying an initial number of groups, and iteratively reallocating objects among groups to convergence. This algorithm typically determines all clusters at once [119].

One popular algorithm is the K-means algorithm which starts with K centroids (initial values for the centroids are randomly selected or derived from a priori information). Then, assigns each point to the cluster of the nearest distance to its centroid. The centroid is the average of all the points in the cluster or the arithmetic mean for each dimension separately over all the points in the cluster. Finally, the centroids are recalculated according to the associated patterns. This process is repeated until convergence is achieved [112][113]. The algorithm aims to minimize the sum of squared distances between data points and their assigned cluster centroids.

Another popular algorithm is k-medoids, which uses representative data points (medoids) instead of centroids to define the cluster centers, the centroids are taken from the data itself, and that is the main difference between it and the K-means algorithm [120].

2.14.3 Density-based Clustering

Density-based clustering is one of the prominent paradigms for clustering large data sets [111]. Density-based clustering algorithms group data points based on the density of points in their neighborhood.

Ester et.al [121] first introduced the DBSCAN algorithm, which depends on a density-based notion of clusters. Clusters are identified by looking at the density of points. Regions with a high density of points depict clusters, whereas regions with a low density indicate clusters of noise or outliers. This algorithm is particularly suited to deal with large, noisy datasets and can identify clusters of different sizes and shapes [113].

DBSCAN clustering approach produces k-dense clusters. Clustering approach need an input, the attributes that the points will be grouped according to, and a global constant parameter settings [122].

2.14.3.1 DBSCAN Clustering Constant Parameters

For DBSCAN clustering, the constant parameters are, the neighborhood radius (Eps) and the minimum number of points in the neighborhood (MinPts). These parameters are used for classifying the points into one of three types: core points in clusters, border points in clusters and outliers [122] [123].

The point is a core point if it has at least MinPts points in its Eps neighborhood. Border point is the non-core point that is reachable from a core point. The outlier is both the non-core and non-border point [123].

2.14.3.2 DBSCAN Clusters Construction

DBSCAN begins with an arbitrary point *p,* retrieve all points within *p* neighborhood radius (Eps), if the number of the retrieved points is equal or more than the (MinPts) parameter, then the cluster is constructed and take a clusterID to differentiate it from other clusters [121]. The distance between the point *p* and each other point is computes using Euclidean distance represented in Equation 2.8 [115]. where every points *p* and *q* have *n* dimension. DBSCAN algorithm basic version is shown in Algorithm 2.2 where each point in the data set is examined if it is a core point and construct a cluster or not, and Algorithm 2.3 shows the process of this examination [121].

Algorithm 2.2: DBSCAN Algorithm

Input: Set_of_unclassified_points, Eps, MinPts

Output: Set of Clusters

Begin

```
ClusterId= Set_NextID(NOISE)   // initial id is -1 (Noise)
For I = 1 to Set_of_unclassified_points. Size Do
P= Set_of_unclassified_points.get(i) // get the i^th point
If P.ClusterID= Unclassified Then
        If ExpandCluster (Set_of_unclassified_points, P, ClusterID, Eps, MinPts) Then  //
        Call Algorithm 2.3
                ClusterID=Set_NextID (ClusterID)
    Endif
Endif
End For
```

End.

$$diS(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \quad \text{... (2.8)}$$

Algorithm 2.3: Expand Cluster

Input: Set_of_unclassified_points, P, ClusterID, Eps, MinPts

Output: Boolean value, True or False

Begin

```
Seeds = Set_of_unclassified_points. RegionQuery (P, Eps)
If Seeds.Size < MinPts Then       // there is no core points
        Set_of_unclassified_points. ChangeClusterID (P, Noise)
        Return False
Else
        Set_of_unclassified_points. ChangeClusterID (Seeds, ClusterID)
        Seeds. Delete (P)
        While (Seeds <> Empty ) Do
                CurrentP = Seeds. First
                Result = Set_of_unclassified_points. RegionQuery (CurrentP, Eps)
                If Result. Size >= MinPts Then
                    For i=1 to Result.Size Do
                        ResultP = Result. Get(i)
                        If ResultP.ClusterID In {Unclassified, Noise} Then
                            If ResultP.ClusterID = Unclassified Then
                                Seeds. Append (ResultP)
                            Endif
                            Set_of_unclassified_points. ChangeClusterID (ResultP,
        ClusterID)
                        Endif  // noise or unclassified
                    End For
                Endif
                Seeds. Delete (CurrentP)
        End While
        Return True
Endif
```

2.15 Fractal Clustering

Fractal clustering is a clustering algorithm based on self-similarity properties of the data sets. It is a clustering algorithm that places points incrementally in the cluster for which the change in the fractal dimension after adding the point is the least. This is a very natural way of clustering points since points in the same cluster have a great degree of self-similarity among them (and much less self-similarity with respect to points in other clusters) [124]. Incremental clustering using the fractal dimension is a form of grid-based clustering where the space is divided in cells by a grid. The main idea behind Fractal clustering is to group points in a cluster in such a way that none of the points in the cluster changes the cluster's fractal dimension radically, and this reflect the meaning of the concept "Fractal" which is defined as a fragmented geometric shape that can be divided into several parts; each one is approximately a smaller copy of the whole shape [125].

Fractal clustering algorithm takes a first step of initializing a set of clusters, and then, incrementally adds points to that set [124].

As an example of the initialization algorithm is the use of a traditional distance-based procedure in order to cluster the initial sample of points by taking points at random, and recursively finding the points that are close to them. When no more close points can be found, a new cluster is initialized, choosing another point at random out of the set of points that have not been clustered yet [126].

Fractal clustering approach is based on the notion of fractal dimension [127] [128], where the incremental step proceeds to take each point and add it to each cluster, computing its new fractal dimension. This step is done by computing the fractal dimension for each modified cluster (after adding the

point to it). Finding the proper cluster to place the point by computing the minimal fractal impact, i.e., the minimal change in fractal dimension experienced by any of the current clusters when the new point is added. The point added to the cluster with extra condition, it must never violate a threshold, otherwise, it will be regarded as a noise [124] [126].

2.16 Chaotic Systems

Chaos is defined mathematically as generating randomness by simple deterministic system [129], that can be applied in communication, automation, pattern recognition, etc. [130] . It is generally characterized by three dynamic properties, ergodicity, stochastic, and sensitivity to its initial conditions.

The main property of chaos is “ergodicity,” which means uniformly and randomly visiting all the parts of the space that the system moves in [36][131].

Chaotic search patterns are derived from chaotic dynamics deterministic systems characterized by sensitive dependence on initial conditions. In the context of optimization and search algorithms, chaotic search patterns are used to explore solution spaces efficiently, often exhibiting fast convergence and better exploration of potential solutions compared to traditional search methods. This gives a good ability to the optimization algorithm to search about the optimal solution and avoiding the stagnation into the local optima. It also increases the convergence towards the global solution if it tuned properly. In general, chaotic maps are applicable for use in computational applications such as optimization problems, especially for initialization the population [35] [132].

Chaotic maps play a key role in generating these chaotic search patterns. A chaotic map is a mathematical function that exhibits chaotic behavior, typically characterized by aperiodicity, sensitivity to initial conditions, and a random-like appearance. Some common chaotic maps used in optimization and search algorithms include:

2.16.1 One Dimensional Chaotic Maps

In one-dimensional chaotic maps, the system is described by a single equation that involves only one variable. The behavior of the system is often observed along a single axis, and the dynamics are relatively simpler compared to higher-dimensional systems.

Some one-dimensional maps or chaotic variable generators are Circle map, Sine map, Gaussian map, Bernoulli shift map, Chebyshev map, Logistic and Tent maps [131][132].

Logistic and Tent maps are the two chaotic maps used in the improvement of BAT algorithm.

- ***Logistic map:*** a one-dimensional polynomial map that leads to chaotic dynamics. The following Equation 2.9 defines the logistic map [131].

$$X_{n+1} = \lambda X_n (1 - X_n) \quad \text{... (2.9)}$$

Where $0 < \lambda \leq 4$

- ***Tent map:*** it is also a one-dimensional polynomial map; it is like the logistic map and displays some particular chaotic effects. It is expressed as the following Equation 2.10 [131]

$$X_{n+1} = \begin{cases} \mu X_n & X_n < 0.5 \\ \mu (1 - X_n) & X_n \geq 0.5 \end{cases} \quad \text{... (2.10)}$$

Where $\mu = 2$

1.17 The Performance of the Improved Metaheuristic Algorithms

Performance evaluation of metaheuristic algorithms is crucial to understand their effectiveness, efficiency, and suitability for solving optimization problems. The key aspects and methodologies for evaluating the performance of metaheuristic algorithms are:

1.17.1 Benchmark Test Functions

To evaluate the performance of the new or improved metaheuristic algorithms, benchmark numerical test functions are used. Metaheuristic algorithms with well performance on these functions can solve real-life hard optimization problems. Benchmark test functions are fundamentally optimization problems presented as mathematical numerical functions [133][134][135]. These functions are optimized with a set of best suitable parameter values that help achieve the best solution. There are a large amount of sub-optimal solutions and the best solution is hidden within. These solutions are spread all over the problem landscape with various numbers and types of hills and valleys. Metaheuristic algorithms, tends to find the best solution as quickly as possible, but it has no guarantee obtain it. The performance efficiency of any metaheuristic algorithm is measured by its global and local search, and its convergence ability. The algorithms with good global search are hard to be stuck in local minima or maxima locations. At the same time, it is hard for any metaheuristics with efficient convergence ability to miss any best solution within the neighborhoods [135].

The benchmark test functions have different properties. Modality, dimensionality, valleys, Basins, and separability are the main properties [136]. Multimodal functions are the functions with more than one local optimum. They are used to test the algorithm's ability to escape from any local minimum. These functions are among the most difficult problems for many algorithms. In order to search the function landscape effectively, the algorithm's exploration process needs to be better designed.

Since the function's flatness feature does not provide any information to the algorithm for directing the search process toward the minima, functions with flat surfaces are difficult [136]. The multimodal test function tests the algorithm's local and global search ability and convergence speed stability. The unimodal test function tests the algorithm's exploitation capability and convergence.

Multimodal test functions are used for the evaluate the diversification process capability of any algorithm. Some multimodal benchmark test functions are as follows:

Ackley [135]: is one of the most commonly used test functions to evaluate the metaheuristic algorithm. It has one global optimal solution within a numerous local minimum. this global optimal solution is found in the middle within a deep narrow basin. The value of the best solution is 0 and it is found at f(x*)=[0,0,…,0] within [-32, 32] domain. The function is mathematically written as Equation 2.11.

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x^2}\right) - \exp\left(\frac{1}{D}\sum_{i-1}^{D}\cos 2\pi\, x_i\right) \quad \text{... (2.11)}$$

Rastrigin Function [135]**:** A function that presents numerous local minima locations. The function has only one global best solution 0 that is

found at f(x*)=[0,0,…,0] within the domain of [-5.12,5.12]. The function is mathematically written as Equation 2.12.

$$f(x) = \left|\sum_{i=1}^{D}(x_i^2 - 10\ \cos(2\pi x_i) + 10)\right| \qquad \textbf{... (2.12)}$$

Griewank Function [135]**:** A function that has widespread suboptimal solutions spread throughout the search environment, and has only one global optimum solution. The value of the global best solution is 0 that is found at f(x*)=[0,0,…,0], within the domain [-600,600]. The function is mathematically written as Equation 2.13.

$$f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad \textbf{... (2.13)}$$

Sphere Function [136]**:** A continuous, differentiable, separable, scalable, multimodal function, and it is subject to $0 \leq xi \leq 10$. The global minimum solution is located at $f(x*) = f(0, \cdots, 0)$, $f(x*) = 0$. It is mathematically written as Equation 2.14.

$$f(x) = \sum_{i=1}^{D} x_i^2 \qquad \textbf{... (2.14)}$$

Alpine Function [135]: A multimodal function with global minima value 0 that is found at f(x*)=[0,0,…,0] within the domain [-10,10]. The function is mathematically expressed as Equation 2.15.

$$f(x) = \sum_{i=1}^{D-1}|x_i\ \sin(x_i) + 0.1\ x_i| \qquad \textbf{... (2.15)}$$

Himmelblau [135]**:** A function that solved with continuous values in the domain [-6 ,6]. The value of the best solution is 0, and it can be found at four locations: f(x*) = [3.2, 2.0], f(x*) = [-2.805118, 3.131312], f(x*) = [-3.779310, -3.283186], and f(x*)=[3.584428,-1.848126] in 2-dimensional space. The function is defined as Equation 2.16.

$$f(x) = \frac{1}{D} \sum_{i=1}^{D} \left(x_i^4 - 16x_i^2 + 5x_i\right) \quad \text{... (2.16)}$$

Schwefel [135]: It is a difficult to solve function, as it contains several local minima locations. The value of the global minima is 0, and it is located at f(x*) = [1, 1, …, 1] in the domain of [-500, 500], This function is expressed Mathematically as Equation 2.17.

$$f(x) = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{|x_i|}\right) \quad \text{... (2.17)}$$

Besides evaluating the capability of diversification (finding a global solution), the algorithm must be checked to determine whether it has good exploitation (local search ability and good convergence). Unimodal test functions are used for this purpose. The following unimodal test functions are used:

SumSquare [135]**:** A function that is also known as Axis Parallel Hyper-Ellipsoid function. It maintains no local optima but one global optima f(x*) = [0, 0, …, 0] within the range of [-10, 10] of continuous values. The function is written as Equation 2.18.

$$f(x) = \sum_{i=1}^{D} i x_i^2 \quad \text{... (2.18)}$$

Sphere Function [135]**:** An easy to solve unimodal and continuous function. it is evaluated using a domain [-5.12,5.12], and its minimum solution value is 0, and it is located at f(x*)=[0,0,…,0]. The function is mathematically written as Equation 2.19.

$$f(x) = \sum_{i=1}^{D} x_i^2 \qquad \textbf{… (2.19)}$$

Step Function [135]**:** A flat surface unimodal function that is often considered difficult to solve as no proper direction towards the globally optimum location is easily found. The value of the global minimum solution is 0, and t located at f(x*)=[0,0,…,0] within [-100,100] range. It is mathematically represented as Equation 2.20.

$$f(x) = \sum_{i=1}^{D} (x_i + 0.5)^2 \qquad \textbf{… (2.20)}$$

Table 2.3 shows a summary of the above-mentioned benchmark test functions.

Table 2.3: A Summarized Benchmark Test Functions

Function	Type	Mathematical equation	Domain	f*	x*
Ackely	Multimodal	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x^2}\right) - \exp\left(\frac{1}{D}\sum_{i-1}^{D}\cos 2\pi\, x_i\right)$	[-32 – 32]	0	(0, 0, …. ,0)
Rastrigin	Multimodal	$f(x) = \left\lvert\sum_{i=1}^{D}(x_i^2 - 10\,\cos(2\pi x_i) + 10)\right\rvert$	[-5.12 – 5.12]	0	(0, 0, …. ,0)
Griewank	Multimodal	$f(x) = \frac{1}{4000}\sum_{i=1}^{D}x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600 – 600]	0	(0, 0, …. ,0)
Sphere	Multimodal	$f(x) = \sum_{i=1}^{D}x_i^2$	[0 – 10]	0	(0, 0, … ,0)
Alpine	Multimodal	$f(x) = \sum_{i=1}^{D-1}\lvert x_i\ sin(x_i) + 0.1\,x_i\rvert$	[-10 – 10]	0	(0, 0, … ,0)
Himmelblau	Multimodal	$f(x) = \frac{1}{4000}\sum_{i=1}^{D}x_i^2 - \prod_{i=1}^{D}cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-6 – 6]	0	Example x*=[3.2,2.0], in 2D space
Schwefel	Multimodal	$f(x) = \sum_{i=1}^{D} -x_i\, sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500 – 500]	0	(1, 1, … ,1)
SumSquare	Unimodal	$f(x) = \sum_{i=1}^{D} ix_i^2$	[-10 – 10]	0	(0, 0, …. ,0)
Sphere	Unimodal	$f(x) = \sum_{i=1}^{D}x_i^2$	[-5.12 – 5.12]	0	(0, 0, …. ,0)
Step	Unimodal	$f(x) = \sum_{i=1}^{D}(x_i + 0.5)^2$	[-100 – 100]	0	(0, 0, …. ,0)

1.17.2 Testing Parameters and Evaluation Metrics

Many testing parameters must be specified as follows:

1. *Population size:* the total number of individuals in the population; for example, a population of 100, 200, 300, 1000, and less or more individuals can be used in the experiments.
2. *Dimensions:* the total number of design variables used in the experiments; for example, 10, 20, or 30 dimensions. 30 is the average dimension with almost all benchmark test functions.
3. *Total runs:* represent the number of times the algorithm runs, for example, 6, 8, 10, and less or more independent runs for each algorithm.
4. *Maximum iterations:* the total number of iterations the algorithm executed for each run, for example, 100.

After executing the algorithms, the following information need to be documented, for performance analysis:

a. Convergence speed: the computation time that is taken by each algorithm.
b. Statistical results: some information is calculated to evaluate the performance of both original and modified algorithms like:
    - *Best value:* the minimum value among all obtained values from all runs.
    - *Worst value:* the maximum value among all obtained values from all runs.
    - *Mean value:* the average value obtained from all runs.
    - *Standard deviation (STD):* to measure how the results spread out from its mean.

2.18 The Performance of the Edge and Fog Nodes' Placements Methods

The performance evaluation of edge and fog nodes' placement is crucial to assess the effectiveness of different strategies and algorithms. Evaluating the placement methods helps in optimizing resource utilization, reducing latency, improving application performance, and achieving overall efficiency in edge and fog computing environments. Evaluation techniques and metrics are discussed in the following sub-sections.

2.18.1 Performance Evaluation Techniques

Many applications, simulators, and programming languages (analytical tools) are used to build the systems and their scenarios to implement and evaluate them. There are three main approaches:

- ***Simulation experiments*** using simulators like iFogSim, NS2, a discrete event network simulator, and NS3, an extension of NS2 written in C++ and Python.
- ***Analytical tools*** such as Matlab software [137], C#, Python, C++, Java, and optimization solvers.

  Simulation tools and mathematical analytical tools are used to create scenarios that mimic real-world edge and fog computing environments. These simulations help researchers evaluate the placement methods under controlled conditions.
- ***Hybrid method*** uses both methods (simulation with some tools). In nodes' placement problems, simulators and hybrid approaches (which also use simulators) are more commonly used than analytical tools and other evaluation environments [50].

2.18.2 Datasets

Datasets play an important role in evaluating and optimizing the performance of benchmark and proposed improved edge and fog nodes' placement algorithms. These datasets represent real-world workloads, application requirements, and system characteristics, enabling researchers to test and validate their strategies. In edge and fog computing resource allocation scope, the use of appropriate datasets helps ensure that the suggested algorithms are effective, efficient, and well-suited to the dynamic and heterogeneous edge and fog computing environments. There are many datasets that are used by many researchers as Table 2.4 shows.

2.18.3 Performance Evaluation Metrics

Various performance metrics, including latency, resource utilization, energy efficiency, and QoS are used to measure the outcomes of placement methods.

2.18.3.1 Network Latency

Latency measures the delay in data transmission between the source and destination nodes. Lower latency is crucial for real-time applications. In order to minimize end- to-end delay, the nearest edge node's resources should be provided to the users. As the number of the deployed edge nodes increases, the end-to-end delay of user request processing will decrease correspondingly, but the deployment cost of the edge nodes will increase [141].

Table 2.4: Used Datasets

References Use the Dataset	The Dataset
[84]	Luxembourg city dataset
[70]	Open Data by Telecom Italia
[138][83]	New York City open data NYC Open Data: https://nycopendata.socrata.com
[40]	Two datasets: • A multi-source dataset of urban life in the city of Milan and the Province of Trentino • OpenCellID dataset. Available online: http://www.opencellid.org
[14] [24]	EUA dataset https://github.com/swinedge/eua-dataset
[58][87][11][10][22][39] [18][19]	Shanghai Telecom's base station dataset (http: //sguangwang.com/TelecomDataset.html)
[78]	Ile-de-France Orange network dataset
[139]	real 4G cellular network datasets from the Île-de-France Orange network
[15]	The real-world dataset was collected across a city-wide open Wi-Fi network deployment in Oulu, Finland.
[73]	City of Darmstadt, Germany dataset
[140]	Telecom Italia dataset
[20]	Real-world ITS social media dataset from China
[23]	OpenCelliD real-world 4G LTE network dataset

2.18.3.2 Edge System Utilization

Edge system resource utilization metrics assess how efficiently the edge node's resources are utilized. Higher edge/ fog node's resources utilization indicates better efficiency, and vice versa. Edge system utilization is improved by reducing the number of idle or underutilized edge nodes [39].

2.18.3.3 Balanced Workload

Ensuring a balanced distribution of workloads among edge and fog nodes helps in efficient resource utilization, reduced congestion, and improved overall system performance. Balanced the load among edge nodes' resources increases their efficiency and provides better services to the users. This avoids overloading resources while some other resources are underloaded [142].

CHAPTER THREE

THE PROPOSED SYSTEM

3.1 Overview

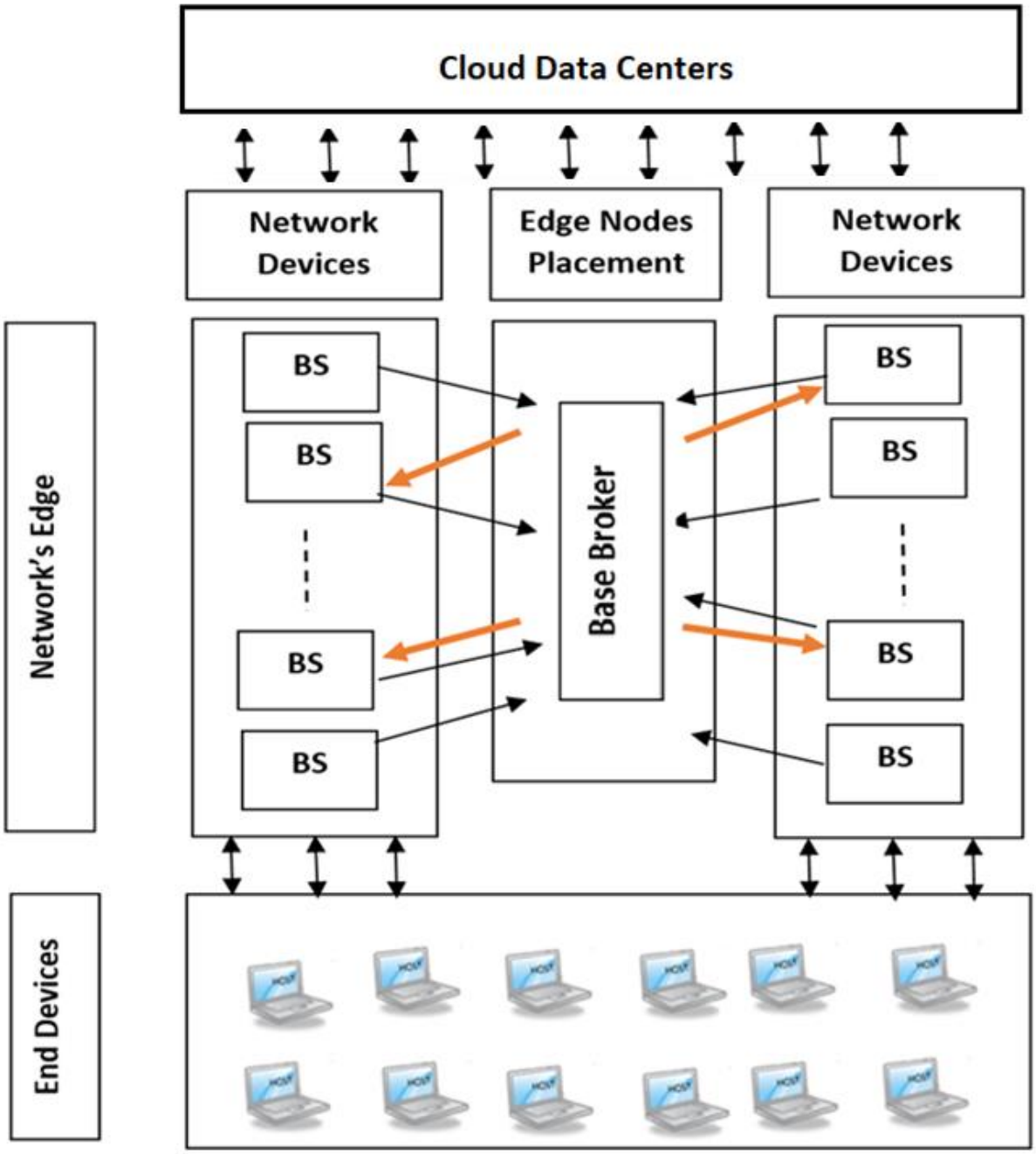
BAT metaheuristic algorithm with the clustering techniques is used a s the Edge nodes' placement approach. This chapter discusses the proposed placement approach generally with all its phases, base stations clustering phase, using of metaheuristic algorithm phase, and network statistics computation phase. All the required algorithms for this chapter are explained thoroughly. Figures and tables that paved the way to support full understanding for the purpose of the work are also represented.

3.2 The Proposed Edge Nodes' Placement Approach

The proposed approach depends mainly on the BAT metaheuristic algorithm which depends on the clustering technique.

The proposed edge nodes' placement method is a centralized approach which is implemented within the base broker, an intermediary network device receives the information of all available BSs, analyze this information and depending on the proposed approach with the considered objectives and constraints, the decision of how many edge nodes to deploy and where to deploy them is taken. Figure 3.1 shows the Network architectural layers.

The proposed approach considers the deployment of edge nodes in a smart city, which contains a thousand of BSs, each edge node serves many users within its coverage area. Algorithm 3.1 shows the proposed placement system generally with all its phases, base stations clustering phase, using of metaheuristic algorithm phase, and network statistics computation phase. Figure 3.2 shows the main phases of the proposed approach.



BS → Base Broker: BSs info goes to the Base Broker for Edge nodes' placement decision

Base Broker → BS: a placement decision of making this BS as an Edge node

Fig. 3.1: Network Architectural Layers

Algorithm 3.1: The Proposed Edge Nodes Placement System

Input: Information for 180 day includes (BS locations and ids, number of users connected to each BS, number of requests made to each BS, Workload of each BS)

Output: Optimal location and number of edge nodes with the best network statistics

Begin

Step 1: Get the information of one day

Step 2: For each time period, do the following:

2.1: Apply clustering to the BSs of that day and get K dense clusters depending mainly on the locations and Workload of the BSs. The outliers are handled using Fractal clustering, Algorithm 3.2, and Algorithm 3.3.

2.2: Depending on the clusters' info, use the improved BAT algorithm to find the optimal placement of edge node within the BSs, Algorithm 3.4.

2.3: Associate the rest BSs (non-edge nodes) to the edge nodes while satisfying the capability of the BS since it is capacitated edge node.

2.4: Turn off idle edge nodes.

2.5: Check the status of the edge nodes system (balanced workload, and overall delay, edge servers' utilization), and:

a) If there is any overloaded edge node do:
- Offload the over workload to the nearest available edge node

b) If there is any underutilized edge node do:
- Offload the remaining workload to the nearest available edge node, to make this edge node "idle"
- Shutdown the idle edge node.

Step 3: If it is not the end of the day, return to Step 2, for the next time period.

Step 4: Return to Step 1, to deal with the next day.

End.

3.2.1 Using of Improved FRactal DBscan Clustering (FRDBClustering)

Generally, the first step in the proposed system is the use of clustering algorithms to generate k-dense clusters.

DBSCAN is the Density based clustering approach that is used to define the k-dense clusters (each cluster share the same properties even cluster's members are geographically sparce) with a set of outliers (set of Base stations that are not belong to any cluster, but they may represent an important base station that cannot be neglected).

Fractal-based clustering approach is used to deal with these outliers' base stations in order to attach each one to the closest cluster (regarding to the features). Algorithm 3.2 shows the clustering phase of the proposed placement system using the FRDBClustering.

Fractal- based clustering approach depends mainly on the Fractal Dimension (FD) value. This value is used mainly for deciding if the outlier point will attach any of the *k* clusters, or not. In case of the ability of the point to join some cluster, FD value will decide which cluster from the *k* produced clusters, the point will be added to.

As mentioned previously in Chapter 2, fractal Clustering algorithm takes a first step of initializing a set of clusters, and then, incrementally adds points to that set. Here in the proposed placement system, using of DBSCAN produces k dense clusters, these clusters are regarded as the initialization step, then the incremental step is start by computing the FD value, and ended by computing fuzzy membership of each point to every cluster as explained below.

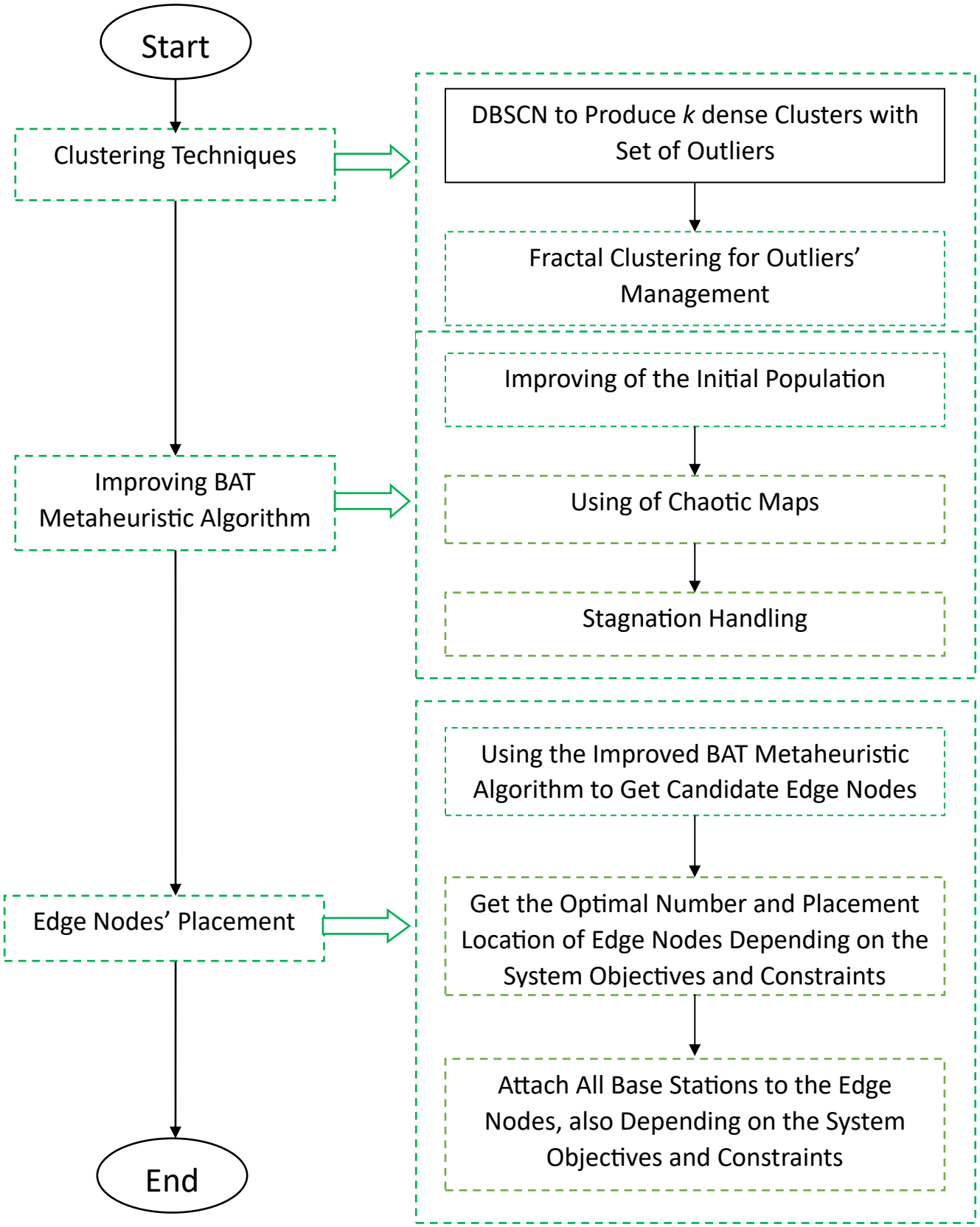


Fig. 3.2: The Phases of the Proposed Approach

Algorithm 3.2: Base Stations Clustering using FRDBClustering.

Input: Set of n Base Stations $B = \{b_1, b_2..., b_n\}$ in a one time period within a day, either 1st period (07:00 AM - 03:00 PM), 2nd period (03:00 PM -10:00 PM) or the 3rd period (10:00 PM – 07:00 AM)

Output: Set of K dense clusters $C = \{c_1, c_2..., c_k\}$ *with outlier nodes*

Begin

Step 1: Apply DBSCAN clustering to get k dense clusters along with a set of m non-clustered BSs (that are regarded as outlier).

Step 2: Deal with the m outlier points set $S= \{P_1, P_2, ..., P_i, ..., P_m\}$, using Fractal clustering technique:

2.1: For each point P_i in the outlier set S, compute its Fractal Dimension $FD(P_i)$ using Equation 3.1 and Equation 3.2

2.2: For each point P_i in the outlier set S, compute its fuzzy membership to each cluster c_j (FMM_{ij}) using Equation 3.3 and Equation 3.4

2.3: Get the largest fuzzy membership among all.

2.4: Compare this fuzzy membership with a predefined attachment threshold, and:

a) If it is less than or equal that threshold, attach the outlier point to that cluster.

$$FMM_{ij} <= Threshold \rightarrow add\ P_i\ to\ c_j$$

b) Otherwise, put the outlier point to a specific cluster (Outliers cluster).

$$FMM_{ij} > Threshold \rightarrow regard\ P_i\ as\ an\ outlier$$

End.

Fractal- based clustering approach depends mainly on the Fractal Dimension (FD) value. This value is used mainly for deciding if the outlier

First step in fractal clustering is the computing of fractal dimension for each outlier point as the following:

For each outlier point x_i,

a. The attributes average (av_i) is computed as Equation 3.1 represents.

$$av_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij} \quad \text{... (3.1)}$$

Where m is the number of point's attributes and x_{ij} is the j^{th} attribute of the point x_i.

b. The fractal dimension of the outlier point x_i (FD_i) is computed after computing its attributes' average value, as Equation 3.2.

$$FD_i = \frac{\sum_{j=1}^{m}(x_{ij}-av_i)^2}{\sum_{j=1}^{m}(x_{ij}-av_i)^4} \quad \text{... (3.2)}$$

Where m is the number of point's attributes, x_{ij} is the j^{th} attribute of the point x_i, and av_i is the attribute average of the point x_i.

After computing the fractal dimension FD for each outlier point, now the fuzzy membership of this outlier point to each cluster is computed as below:

a) For each cluster c_j, compute the fractal dimension for each point member p_k (FDc_{jk}).
b) Compute the fractal dimension difference from each outlier point x_i to every cluster's member and get the smallest fractal dimension difference between this outlier point with each cluster c_j (dFD_{ij}) as Equation 3.3.

For each outlier point x_i
For each cluster c_j
For each cluster point p_k
dFD_{ij} = Min (FDx_i - FDc_{jk}) **... (3.3)**

where FDx_i is the fractal dimension of the outlier point x_i, and FDc_{jk} is the fractal dimension of point p_k in cluster c_j.

c) Compute the fuzzy membership (FMM_{ij}) of the outlier point x_i with every cluster c_j as Equation 3.4

For each outlier point x_i
For each cluster c_j

$$FMM_{ij} = \frac{1}{\sum_{l=1}^{n} \left(dFD_{ij}/dFD_{il}\right)^2} \quad \text{... (3.4)}$$

Where n is the number of clusters, and dFD_{ij} is the smallest fractal dimension difference between the outlier point x_i and every point in cluster c_j.

After that, each outlier point x_i, will attach the cluster with the highest fuzzy membership.

3.3 The Proposed Improved BAT Metaheuristic Algorithm

This section describes the proposed improved BAT Algorithm based on the clustering approach.

As mentioned in Chapter 2, the original BAT algorithm has good exploitation but poor exploration, so it can easily get trapped at a local minimum of most multimodal test functions. A modification of the original BAT algorithm is applied to overcome this problem. The proposed improved BAT algorithm is shown in the flowchart of Figure 3.3. The modification is about the following phases:

- Improving population initialization to improve the global search ability.
- Balancing the algorithm's local and global search abilities.
- Overcome stagnation.

The following subsections describe these phases in detail.

3.3.1 Improving the Population's Initialization

Metaheuristic algorithms work by initializing the population of candidate solutions, then iteratively updating these candidate solutions until convergence occurs or the stopping criteria are met. The initial population's quality mainly affects the algorithm's convergence and has a role in the quality of the produced solution.

A method to improve the population depends on the information obtained from density-based clustering is proposed. Density-based clustering is used to produce several clusters. These clusters are almost the most important features.

When the clustering approach is applied to a dataset of base stations for a large smart city, then, depending on the workload and users' requests to these base stations and the number of users connected to the base station, the base stations will be grouped depending on the workload heaviness grades. So, each cluster will represent a degree of importance. For each cluster, the center will be computed.

Four regions are calculated for each cluster depending on the closeness to the centers; the two closest areas to the center will get a higher probability of producing individuals to the population.

A uniformly random population is initialized and called the 1st population, and a cluster-center-based population is initialized too and called the 2nd population. The final improved initial population is generated by taking the fittest individuals from the two populations. Algorithm 3.3 shows a pseudo-code for the population initialization.

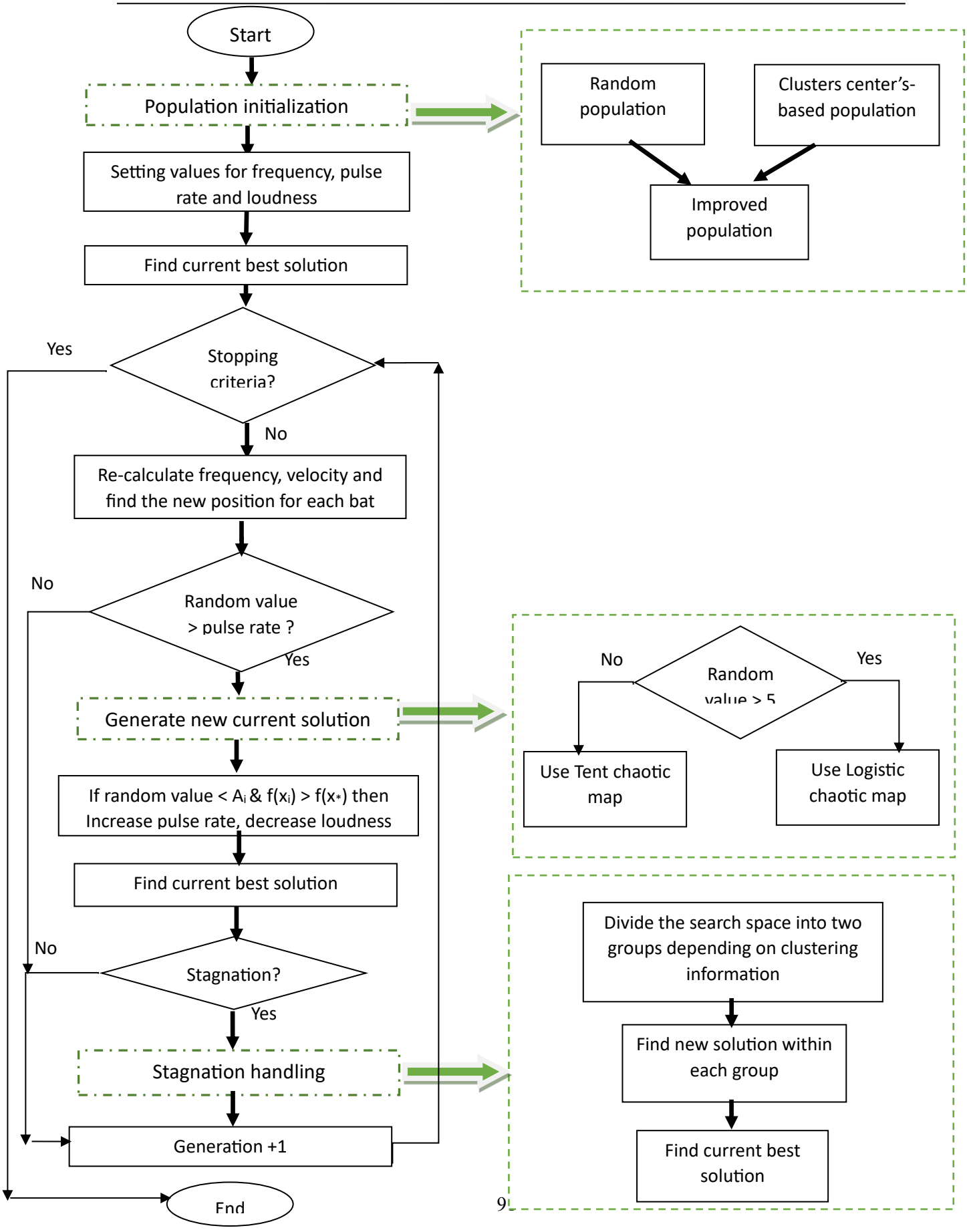


Fig. 3.3: An Improved BAT Algorithm

After clustering the search space, the center and upper and lower boundaries are extracted for each cluster. Equation (3.5) represents the calculation of the cluster's center.

center = lower bound + (upper bound – lower bound) / 2 **… (3.5)**

From each cluster, the range of the four regions will be computed as the following Equations (3.6), (3.7), (3.8), and (3.9), where 'a' is the cluster's lower bound, 'b' is the cluster's upper bound:

Region1 (R1) domain: [a, a+(center-a)/2] **… (3.6)**

Region2 (R2) domain: [a+(center-a)/2, center] **… (3.7)**

Region3 (R3) domain: [center, center+(b-center)/2] **… (3.8)**

Region4 (R4) domain: [center+(b-center)/2, b] **… (3.9)**

For each cluster, and depending on the number of points within, the ratio of the population's individuals that are produced from that cluster is computed. The larger the cluster, the higher the participation ratio in the population initialization. As shown in Figure 3.4, four regions are computed for each cluster.

Algorithm 3.3 The Proposed Cluster-based Method for BAT Algorithm Population Initialization

Input: the total number of individuals in the population *N_P*

Output: initial population "*Population"*

Begin

Step 1: Generate a uniformly random population "*1st Population*"

Step 2: Generate density-based k clusters, then get the (k centers) $(c_1, c_2, \ldots, c_k)$, (k lower bounds) $(a_1, a_2, \ldots, a_k)$, and (k upper bounds) $(b_1, b_2, \ldots, b_k)$

Step 3: Weight each cluster depending on the number of points within each.

$$w_i = p_i / sum$$

w_i is the weight of cluster i, p_i is the no. of points within cluster i, and the *sum* is the total number of points within all clusters

Step 4: Depending on the cluster's weight, get the number of population's individuals within each cluster N_P_i

$$N\_P_i = w_i * N\_P$$

Step 5: For each cluster where cluster center $c_i \in (c_1, c_2, \ldots, c_k)$, lower bound $a_i \in (a_1, a_2, \ldots, a_k)$, and upper bound $b_i \in (b_1, b_2, \ldots, b_k)$ do:

- generate four regions' boundaries (R1, R2, R3, R4) from the available variables range

  $R1:(a_i , a_i + (c_i - a_i) / 2)$ $R2: (a_i + (c_i - a_i) / 2 , c_i)$

  $R3: (c_i , c_i + (b_i - c_i) / 2)$ $R4: (c_i + (b_i - c_i) / 2 , b_i)$

- generate a $2*N\_P_i/3$ individual for the two closest regions to the center, and $N\_P_i/3$ individuals for the two other regions and insert all generated individuals into the "*2nd Population"*

end for

Step 6: Evaluate both populations (*1st Population and 2nd Population*) to get the final initial population (*Population*) by selecting the best individuals from both populations.

End.

The two closest regions to the cluster's center will produce a higher ratio of individuals than the other two regions since the cluster's center will have the most important and valuable concentrated points than the two farthest isolated regions. So, the regions R2 and R3 are more important than R1 and R4.

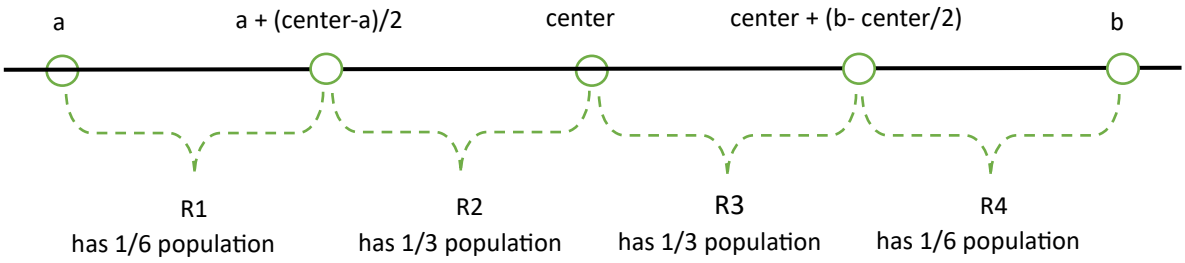


Fig. 3.4: The Four Regions of Each Cluster

Generating the cluster's center-based population will improve the initial population used by the algorithm. As a result, the convergence will be enhanced, and the final solution will be improved too.

When using only the randomly generated population, the individuals may be far from the best optimal solution, so that the convergence will worsen.

3.3.2 Balancing the Algorithm's Local and Global Search

As mentioned previously, the BAT algorithm needs better exploration. To find other good solutions around the current global solution and explore the search space more thoroughly, the BAT algorithm uses a random walk, which is a random process consisting of taking a series of consecutive random steps. The following mathematical Equation (3.10) represents the random walk.

$$X_{t+1} = X_t + \omega_t \quad \text{... (3.10)}$$

X_t is the current location or state at t, and ω_t is a step or random variable with a known distribution. The original BAT algorithm uses ω_t within step size +1 or -1.

In the proposed improved BAT algorithm, a Logistic chaotic map is used instead of the random walk, and the algorithm's behavior is improved.

Chaotic functions are used to balance the exploration and the exploitation of the algorithm due to its non-repetitive nature and, as a result, will tackle premature convergence problems.

The random-based optimization algorithms can use a chaotic dynamic instead of randomness. Optimization algorithms that use chaotic (chaos optimization algorithms (COAs)) can easily escape from local minima than other stochastic optimization algorithms. Stochastic optimization algorithms have to accept some bad solution to escape from the local minima, while COA searches on the regularity of chaotic motion to escape from local minima.

Logistic and Tent maps are used in the improved BAT algorithm, which are both one-dimensional polynomial map that leads to chaotic dynamics. Both are defined as Equations (3.1) and (3.12) respectively.

$$X_{n+1} = \lambda X_n (1 - X_n) \quad \text{... (3.11)}$$

Where $0 < \lambda \leq 4$

$$X_{n+1} = \begin{cases} \mu X_n & X_n < 0.5 \\ \mu (1 - X_n) & X_n \geq 0.5 \end{cases} \quad \text{... (3.12)}$$

Where $\mu = 2$

3.3.3 Stagnation Handling

Like many other metaheuristic algorithms (PSO, for example), the BAT algorithm suffers from stagnation during the search process. To enhance the search process of the BAT algorithm when stagnation occurs, the search space will be divided into two parts depending on the produced clusters. The two closest clusters (depending on the similarity measurement) will be the first part; the other clusters will be the other.

A temporal new population is initialized for each part following the same improved population initialization process. After that, the best part's solution will be derived by picking the fittest one. By evaluating the produced best solutions from both parts, the best one will be set as the current best solution, and the process will be continued.

In this case, the search process will become diversified, so stagnation will likely occur less.

3.4 Edge Nodes' Placement Mathematical Model

The proposed edge nodes' placement approach is modeled mathematically. The main parameters used in the proposed approach's mathematical model is shown in Table 3.1. The main objective of the proposed system is to maximize the network system efficiency ($Sys_{Efficiency}$), where the best $Sys_{Efficiency}$ achieves with the maximum utilization of the edge servers ($Sys_{Utilization}$), and minimum overall system latency ($Sys_{Latency}$) and balanced workload ($Sys_{Workload}$) as computed in Equation 3.13.

Table 3.1: Used Parameters Within the Proposed Placemen Mathematical Model

Notation,	Its Description
n	Total number of Base Stations, (n = 1, 2, 3, . . .)
m	Total number of Edge Servers, (m = 1, 2, 3, . . .)
k	Total number of Clusters, (k = 1, 2, 3, . . .)
B	B = {b_1, ..., b_n}, is the set of the *n* Base Stations
S	S = {s_1, . . ., s_m}, is the set of *m* Edge Servers
C	C = {c_1, …, c_k}, is the set of *k* Clusters
b_i	Base Station i (i = 1, 2, 3, . . ., n)
s_j	Edge Server j (j = 1, 2, 3, . . ., m)
c_l	Cluster l (l = 1, . . ., k)
L_x(b_i)	Latitude location of Base Station b_i
L_y(b_i)	Longitude location of Base Station b_i
WL(b_i)	Workload of Base Station b_i
L_x(s_j)	Latitude location of Edge Server s_j
L_y(s_j)	Longitude location of Edge Server s_j
WL(s_j)	Workload of Edge Server s_j
U(s_j)	Utilization of the edge server s_j
T	Time periods within the day. Here there is 3 time periods, T=3

Since the system latency has a greater impact on the system efficiency then the balanced workload especially in the user point of view, so the system latency gets 60% importance ratio, and 40% importance ratio is given to the balanced workload.

$$Sys_{Efficiency} = MAX\ (Sys_{Utilization}\ /(60\%\ Sys_{Latency} + 40\%\ Sys_{Workload})) \quad \textbf{… (3.13)}$$

- $Sys_{Utilization}$ *is the* total utilization of edge servers for all time periods within the day, as Equation (3.14) shows.

$$Sys_{Utilization} = 1/m \sum_{t=1}^{T} Edge_{Utilization} \quad \textbf{… (3.14)}$$

$Edge_{Utilization}$ is computed for all edge servers s_j ($1 \leq j \leq m$) which belong to every cluster c_i ($1 \leq i \leq k$), as Equation 3.15

$$Edge_{Utilization} = \sum_{i=1}^{k} \sum_{j=1}^{m} x_{ij} * U\left(s_{ij}^{t}\right) \text{ ... (3.15)}$$

Where $U\left(s_{ij}^{t}\right)$ is the utilization of the edge server s_j within the cluster c_i at time period (t), and is given as Equation 3.16

$$U\left(s_{ij}^{t}\right) = \frac{WL(s_{ij})}{WL_{max}} \text{ ... (3.16)}$$

$WLmax$ is the maximum workload that any edge server can handle, and the workload of each edge server s_j is the total workload of all base stations allocated with this edge server s_j as Equation 3.17 shows.

$$WL(s_j) = \sum_{i=1}^{n} x_{ij} * WL(b_i) \text{ ... (3.17)}$$

Where x_{ij} *is* a Boolean indicator to indicate whether the base station b_i *is allocated with edge server* s_j.

- The overall network access delay needs to be minimized, and it is mainly computed depends on the communication delay between mobile users and the edge node they are connected with. Equation 3.18 shows the computation of the system delay.

$$Sys_{Latency} = 1/n \sum_{t=1}^{T} Edge_{Latency} \text{ ... (3.18)}$$

Where n is the total number of base stations, and $Edge_{Delay}$ is computed as Equation 3.19

$$Edge_{Latency} = \sum_{j=1}^{k} \sum_{i=1}^{m} \sum_{l=1}^{n} x_{ij} * x_{li} * D\left(s_{ij}^{t}, b_{li}^{t}\right) \quad \textbf{... (3.19)}$$

Where x_{ij} *is* a Boolean indicator to indicate whether the *edge server* s_i *is within cluster* c_j. x_{li} *is* a Boolean indicator to indicate whether the *base station* b_l *is allocated to the edge server* s_i.

$D\left(s_{ij}^{t}, b_{ki}^{t}\right)$ is the communication delay between edge server s_i within cluster c_j, and every base station b_l allocated to the edge server s_i, and it is computed as Euclidian distance between every pair (geometrically depending on their latitude and longitude) as Equation 3.20

$$D\left(s_{ij}^{t}, b_{ki}^{t}\right) = \sqrt{\left(L\_x\left(s_{ij}\right) - L\_x\left(b_{li}\right)\right)^2 + \left(L\_y\left(s_{ij}\right) - L\_y\left(b_{li}\right)\right)^2} \textbf{... (3.20)}$$

- The workload balance among all edge servers also needs to be minimized. Equation 3.21 shows the computation of the balanced workload.

$$Sys_{Wokload} = 1/m \sum_{t=1}^{3} Edge_{WorkloadBalance} \textbf{... (3.21)}$$

Where: $Edge_{WorkloadBalance}$ is computed for all edge servers within all clusters as Equation 3.22 represents.

$$Edge_{WorkloadBalance} = \sum_{i=1}^{k} \sum_{j=1}^{m} WL\left(s_j\right) - WL_{average} \textbf{... (3.22)}$$

$WL_{average}$ is the average workload among all edge servers, computed as Equation 3.23.

$$WL_{average} = 1/m \sum_{j=1}^{m} WL\left(s_j\right) \quad \textbf{... (3.23)}$$

Subject to the following constraints:

$$WL\left(s_{ij}^{t}\right) \leq WLmax \quad \text{... (3.24)}$$

$$\sum_{j=1}^{k} x_{ij} = 1 \quad \text{... (3.25)}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \text{... (3.26)}$$

$$D\left(s_j, b_i\right) \leq Dist\_threshold \text{ ... (3.27)}$$

Equation 3.24 indicates that at time period t the workload of the edge server s_j which belong to cluster c_i never violate its maximum available workload.

Equation 3.25 indicates that each edge server s_i is belong only to one cluster c_j, where if the edge server s_i is belong to the cluster c_j then x_{ij} is 1, otherwise, it is 0;

$$x_{ij} = \begin{cases} 1 & edge\ server\ s_i\ within\ cluster\ c_j \\ 0 & otherwise \end{cases}$$

Equation 3.26 indicate that each base station b_i is allocated only to one edge server s_j, where if the base station b_i is allocated with the edge server s_j then x_{ij} is 1, otherwise, it is 0;

$$x_{ij} = \begin{cases} 1 & BS\ b_i\ allocated\ with\ edge\ server\ s_j \\ 0 & otherwise \end{cases}$$

Equation 3.27 indicate that every base station b_i can be allocated with the edge server s_j only if the distance between them is less than or equal the distance threshold which represent the coverage area of the edge server.

3.5 Summary

This chapter discusses the main phases of the proposed edge nodes' placement system. The mathematical model of proposed system shows a multi objective optimization problem which is NP-Hard problem to be solved. Using of BAT metaheuristic algorithm will provide near optimal solution to the problem, with excellent results. An improvement to the BAT algorithm is proposed which make the algorithm more efficient. The improvement depends mainly on the use of clustering technique. Fractal clustering technique is used to handle the outliers that DBSCAN clustering algorithm produces with the clusters.

CHAPTER FOUR

RESULTS, ANALYSIS, AND DISCUSSIONS

4.1 Overview

This chapter will present the obtained results from implementing the proposed methods regarding the clustering phase, improving BAT metaheuristic algorithm, and Fog/ Edge nodes placement method.

The following subsections will explain the experiment results for the proposed approaches employed in this thesis. A comparison with the benchmark approaches is made for evaluating the overall performance of the suggested techniques.

4.2 Experimental Analysis

The goal of the experiments is to evaluate the performance of the proposed edge nodes placement system and compare it with the benchmark placement methods.

All the benchmarks and the proposed algorithms are implemented using C#, and the experiments are conducted on a computer with 11th Gen Intel® Core™ i9-11900H CPU @2.50GHz and 16.0 GB RAM in Windows 10 Pro environment. The general execution of the proposed system phases is as Figure 4.1 shows.

4.2.1 Dataset Description

The performance of the proposed algorithms is evaluated using many experiments depending on a real-world dataset which is Shanghai Telecom's base station dataset. The dataset, provided by Shanghai Telecom, contains more than 7.2 million records of accessing the Internet through 3,233 base stations from 9,481 mobile phones for six successive months. Figure 4.2 shows the distribution of base stations. Each node denotes a base station in Shanghai, China.

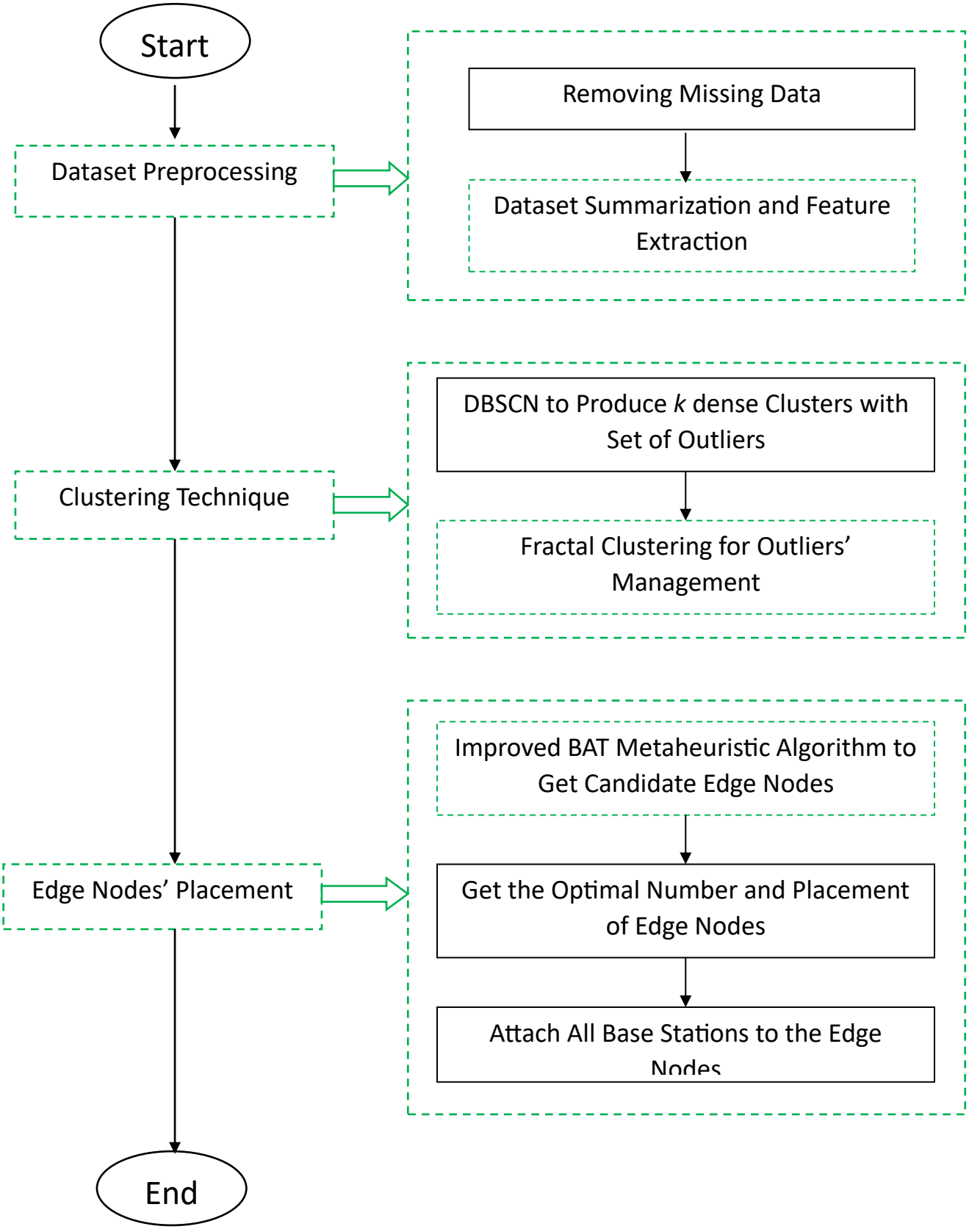


Fig. 4.1: The Sequence of the Proposed System

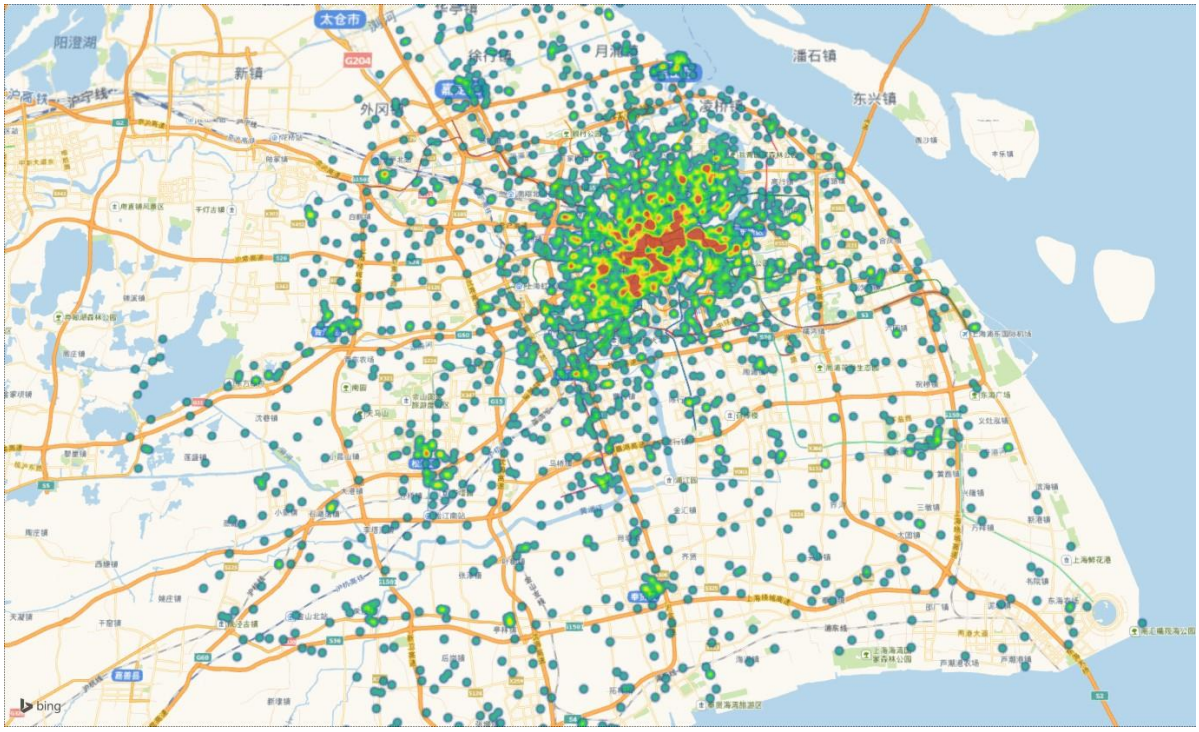


Fig. 4.2: Base Stations' Distribution in Shanghai, China [143]

The dataset contains 12 Excel files, each contains information of 15 days, with approximately more than 600000 records. Table 4.1 shows a sample from the dataset, for the first 9 records from the 1st day. Each record contains the following raw parameters:

- **Month:** year and month of internet access;
- **Date:** day of the access;
- **Start time:** start time of the call in the form "*mm/dd/yyyy hh:mm:ss"* 24-Hour format.
- **End time:** start time of the call in the form "*mm/dd/yyyy hh:mm:ss"* 24-Hour format.
- **BS Latitude, and BS Longitude:** represented with decimal degrees, using degrees, minutes, and seconds.
- **User id:** the id of the user who accesses the base station.

Table 4.1: A Sample from the Used Dataset

Month	date	start time	end time	latitude	longitude	user id
201406	1	6/1/2014 10:22	6/1/2014 11:09	31.237872	121.470259	edbc54bddf16494a49f39ac057b4185d
201406	1	6/1/2014 7:00	6/1/2014 8:49	31.237872	121.470259	f8206ab58b9bdb070673f7050242e9ee
201406	1	6/1/2014 2:49	6/1/2014 5:49	31.237872	121.470259	f8206ab58b9bdb070673f7050242e9ee
201406	1	6/1/2014 8:51	6/1/2014 9:20	31.237872	121.470259	f8206ab58b9bdb070673f7050242e9ee
201406	1	5/31/2014 23:49	6/1/2014 2:49	31.237872	121.470259	f8206ab58b9bdb070673f7050242e9ee
201406	1	5/31/2014 23:39	5/31/2014 23:49	31.237872	121.470259	f8206ab58b9bdb070673f7050242e9ee
201406	1	6/1/2014 17:17	6/1/2014 17:18	31.237872	121.470259	eb7f2c8f7030376959889a2fae9c3a1d
201406	1	6/1/2014 17:17	6/1/2014 18:32	31.237872	121.470259	eb7f2c8f7030376959889a2fae9c3a1d
201406	1	6/1/2014 11:17	6/1/2014 11:17	31.237872	121.470259	edbc54bddf16494a49f39ac057b4185d

Shanghai Telecom's base station dataset is large dataset, with the following extra features:

- The BSs alternate between On and Off among some days and others. For example, BS at the location 31.251986, 121.456773 is 'ON' in the days 1, 2 and 4, but 'OFF' in day 3 (day 3 is not mentioned in the table "OFF", and days 1,2, and 4 are mentioned "ON"), as Table 4.2 shows.

Table 4.2: Base Station's ON, OFF Status

Record No.	Month	date	start time	end time	latitude	longitude
24559	201406	1	6/1/2014 11:54	6/1/2014 11:56	31.251986	121.456773
24560	201406	1	6/1/2014 11:52	6/1/2014 11:54	31.251986	121.456773
61394	201406	2	6/2/2014 10:47	6/2/2014 13:44	31.251986	121.456773
61395	201406	2	6/2/2014 15:03	6/2/2014 15:03	31.251986	121.456773
143194	201406	4	6/4/2014 14:37	6/4/2014 17:35	31.251986	121.456773

- It considers the mobility of users. For example, within the 1st day, for example there are users enter the range of many different BSs, as Table 4.3 shows below.

Table 4.3: User Mobility in the Shanghai Telecom Dataset

Record No.	Month	date	start time	end time	latitude	longitude	user id
12	201406	1	6/1/2014 9:37	6/1/2014 10:07	31.237872	121.470259	edbc54bddf16494a49f39ac057b4185d
197	201406	1	6/1/2014 11:48	6/1/2014 11:48	31.242893	121.465382	edbc54bddf16494a49f39ac057b4185d
24559	201406	1	6/1/2014 11:54	6/1/2014 11:56	31.251986	121.456773	edbc54bddf16494a49f39ac057b4185d

4.2.2 Dataset Reduction

From these raw data, summarized informatic data is constructed. The summarized data are about 1800-2100 records for each day (6% from the total number of records), with the following information:

- **BS id:** the id of the base station.
- **BS Latitude.**
- **BS Longitude.**
- **BS number of Users:** number of users who are connected with this BS.
- **BS number of Requests:** number of requests the BS receives from the connected users.
- **BS Time Span:** the time interval that the BS is busy with.
- **BS Bandwidth:** the bandwidth of the Base station.
- **BS work's start time:** first time the BS receives a request.
- **BS work's end time:** last time the BS receives a request and process it.
- **BS total Workload:** total workload of the BS through the day.
- **BS 1st period Workload:** the workload of the BS from 07:00 AM until 03:00 PM.
- **BS 2nd period Workload:** the workload of the BS from 03:00 PM until 11:00 PM.
- **BS 3rd period Workload:** the workload of the BS from 11:00 PM until 07:00 AM.

Table 4.4 shows a sample of the summarized data, with all the described fields.

Table 4.4: A Sample from the Summarized Dataset

BS id	Day	BS Latitude	BS Longitude	BS no. of Request	BS no. of Users	BS Workload	BS start work time	BS end work time	1st Workload	2nd Workload	3rd Workload
1	1	31.237872	121.470259	26	6	83312	5/31/2014 23:39	6/2/2014 23:15	26488	43076	13748
2	1	31.246946	121.513919	2	2	3825	6/1/2014 13:02	6/2/2014 11:08	2913	0	912
3	1	31.232877	121.48753	18	8	204764	5/31/2014 23:00	6/2/2014 12:56	25962	11786	167016
4	1	31.227933	121.45361	34	16	132925	6/1/2014 0:19	6/2/2014 21:20	65533	65940	1452
5	1	31.235682	121.487831	13	6	43553	6/1/2014 8:27	6/2/2014 17:56	20119	23434	0
6	1	31.251358	121.485526	46	7	115650	6/1/2014 5:58	6/2/2014 23:11	42963	66520	6167
7	1	31.244229	121.475231	15	8	28755	6/1/2014 16:38	6/2/2014 23:20	773	27741	241
8	1	31.23936	121.488752	11	6	94781	6/1/2014 13:27	6/2/2014 20:18	26658	10855	57268
9	1	31.244859	121.489305	20	7	80645	6/1/2014 7:38	6/2/2014 16:45	28991	44916	6738

As shown clearly, the second time period (from 03:00 PM until 11:00 PM) has the heaviest workload through the day (busiest period) and it is highlighted with green color. Third time period (from 11:00 PM until 07:00 AM) has the lowest workload (less busy period) highlighted with red color. First time period has a moderate workload.

Depending on the workload heaviness, the ratio of edge nodes to the existed base station is vary from one time period to another as follows:

- First time period will get edge nodes equal to 20% base stations.
- Second time period will get edge nodes equal to 25% base stations.
- Third time period will get edge nodes equal to 15% base stations.

4.2.3 Using of Density-based Clustering and Fractal Clustering

Upon using the previously described dataset, and for each time period within each day, the base stations will be grouped in some distinct clusters. The following Subsections 4.2.3.1 and 4.2.3.2 discusses the used constant parameters and attributes and their effects on the DBSCAN clustering process.

4.2.3.1 DBSCAN Clustering Constant Parameters

The used parameters (the neighborhood radius (Eps) and the minimum number of points in the neighborhood (MinPts)) are represented in the Table 4.5, where the clustering approach is applied for the first day in the data set which contain 1895 Base stations, with different workload, number of users and number of requests. Depending on the constant parameters setting, the quality and characteristics of the produced clusters will be differed.

Table 4.5: The Produced Cluster upon Different Input Parameters

Input Constant Parameters	**Its Effect**
Eps. = 0.9, MinPts = 45	There are 4 clusters collectively contain 1028 BSs, with 867 non-clustered base stations
Eps. = 0.9, MinPts = 25	**There are 6 clusters collectively contain 1212 BSs, with 683 non-clustered base station**
Eps. = 0.6, MinPts = 45	There are 2 clusters collectively contain 559 BSs, with 1336 non-clustered base stations
Eps. = 0.6, MinPts = 25	There are 3 clusters collectively contain 886 BSs, with 1009 non-clustered base station
Eps. = 0.4, MinPts = 45	There are 2 clusters collectively contain 438 BSs, with 1457 non-clustered base stations
Eps. = 0.4, MinPts = 25	There are 8 clusters collectively contain 730 BSs, with 1165 non-clustered base station

As shown from Table 4.5, as the value of the minimum points (MinPts) increases, the number of clusters decreases for the same Eps, and as the value of the Eps increases, the number of clusters decreases for the same MinPts.

The bolded field's values are the used parameters' values, since they give a moderate number of clusters with good number of points within each one.

4.2.3.2 DBSCAN Clustering Attributes

The attributes that are used are the features of the BSs, *the workload of the BS, number of connected users to the BS, and the number of requests the BS receives*. Clustering approach is mainly influenced by the workload, and the two other BS's attributes (number of requests and number of users) have a lower impact in the clustering results.

As the clusters are mainly formed based on the received workload of each base station, so the clusters will be of different workload, indeed with many non-clustered base stations. Figure 4.3 shows the number of generated clusters, each with the number of its members, along with the number of the non-clustered base stations (outliers). These generated clusters are with the constant parameters Eps. is 0.9 and MinPts is 25.

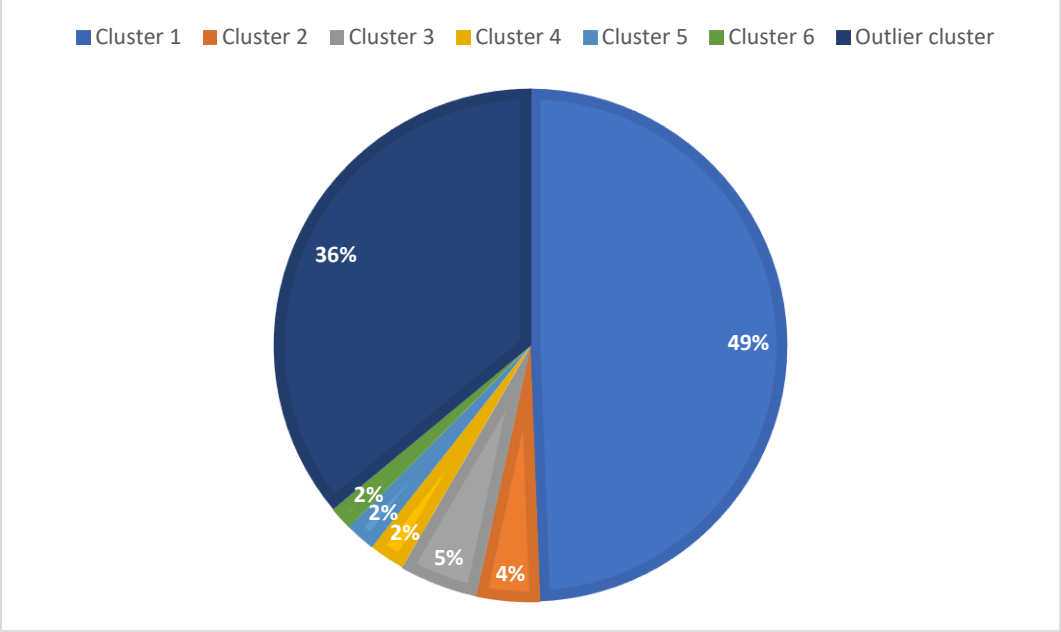


Fig. 4.3: BSs Ratio for Each Produced Clusters

The workload of all generated clusters' members is shown in Figure 4.4 where it's shown that the 1st cluster is the largest cluster in size (number of member) but it has the BSs with the lower workload demand. The rest clusters have a highest workload demand with a smaller number of members. The workload of the outlier set is shown in Figure 4.5 where the outlie cluster is the second largest cluster in number of members with the highest workload demand (far away from all produced six clusters).

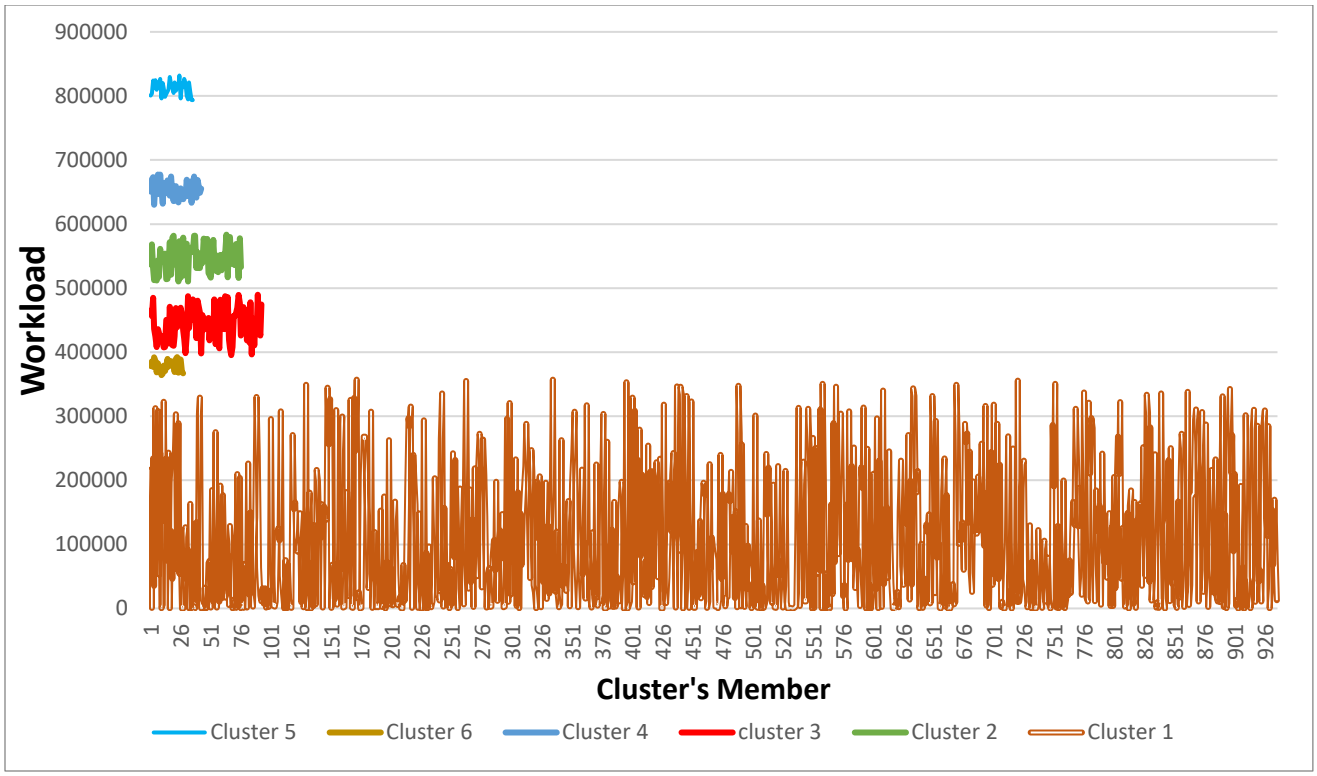


Fig. 4.4: The Workload of the Produced Six Clusters

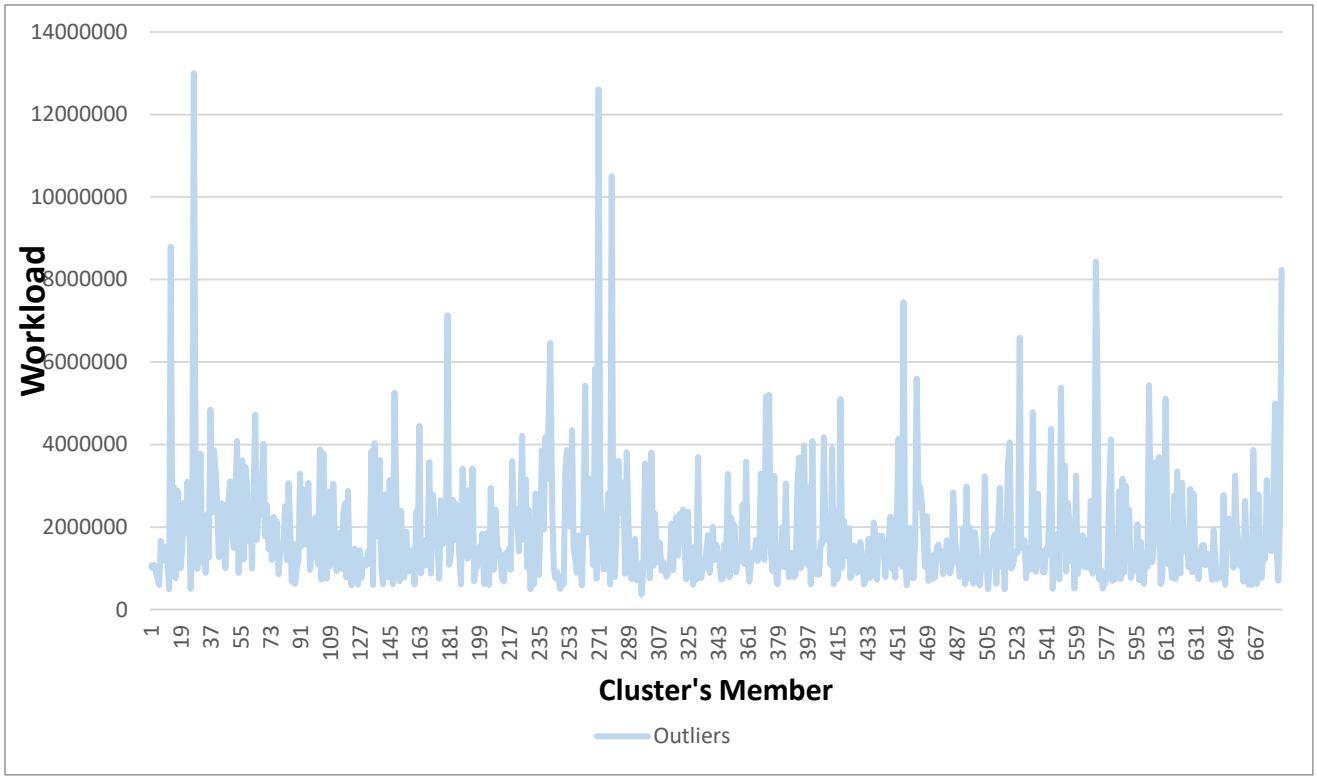


Fig. 4.5: The Workload of the Outlier Set

The resulted clusters members' workload is as the following Tables shows, where Table 4.6 shows a sample from the 1st cluster, and it clearly show that it contains the lowest loaded base stations with the minimum number of users and request.

Table 4.6: Details of a Sample of Members from Cluster 1

Cluster ID	BS Latitude	BS Longitude	BS Workload	BS number of Requests	BS number of Users
1	30.974035	121.554233	41725	5	2
1	31.070388	121.520258	282607	11	4
1	30.802978	121.397313	12425	2	1
1	31.054051	121.741771	45752	3	2
1	30.91796	121.543057	154466	19	4
1	30.977355	121.74808	240636	6	2
1	30.931335	121.472783	1459	1	1
1	31.039851	121.603151	9918	1	1
1	30.815977	121.436091	929	1	1
1	30.846647	121.431405	3761	3	1

Tables 4.7, 4.8 and 4.9 shows a sample from the 2nd, 3rd and 4th clusters respectively, these clusters contain a higher loaded base stations with different workload range with a higher number of users and request than the 1st cluster.

Table 4.7: Details of a Sample of Members from Cluster 2

Cluster ID	BS Latitude	BS Longitude	BS Workload	BS number of Requests	BS number of Users
2	30.723654	121.330778	513753	16	6
2	30.766975	121.362906	514444	12	4
2	31.308964	121.403345	539083	9	6
2	31.355885	121.375057	571736	15	11
2	31.320829	121.389518	520716	13	4
2	31.488248	121.349833	579048	11	1
2	31.293561	121.362372	581932	12	5
2	31.406426	121.476433	537221	14	6
2	31.312069	121.172955	537038	10	2
2	31.209904	121.234203	548321	11	6

Table 4.8: Details of a Sample of Members from Cluster 3

Cluster ID	BS Latitude	BS Longitude	BS Workload	BS number of Requests	BS number of Users
3	31.240686	121.480699	408233	16	3
3	31.118723	121.248449	435723	12	6
3	31.012926	121.423939	430291	12	6
3	30.962059	121.244644	420585	9	1
3	30.847557	121.226776	424799	22	5
3	31.107362	121.020176	414268	10	3
3	30.934254	121.279662	407416	11	1
3	31.073061	120.970637	408195	8	2
3	30.983162	121.054057	450636	10	1
3	30.838141	121.189035	449867	10	3

Table 4.9: Details of a Sample of Members from Cluster 4

Cluster ID	BS Latitude	BS Longitude	BS Workload	BS number of Requests	BS number of Users
4	30.999472	121.378674	661602	15	4
4	31.01613	121.203009	658943	14	5
4	31.099085	121.386207	677298	16	8
4	31.08532	121.237002	646955	12	8
4	31.065206	121.803371	677127	20	7
4	31.353347	121.441038	652730	18	2
4	31.420016	121.44113	631839	12	4
4	31.273767	121.222048	663525	30	7
4	31.346801	121.577398	647730	37	6
4	31.252495	121.390194	658253	33	4

5th cluster has the highest workload base stations among all the six clusters, with the highest number of users and requests. Table 4.10 shows a sample from the 5th cluster.

Table 4.10: Details of a Sample of Members from Cluster 5

Cluster ID	BS Latitude	BS Longitude	BS Workload	BS number of Requests	BS number of Users
5	31.241014	121.48529	823897	25	11
5	31.217372	121.460724	812139	14	6
5	31.143754	121.434357	824429	35	10
5	31.042415	121.482986	809634	12	2
5	31.111396	121.379353	818856	23	6
5	31.015452	121.154814	819365	24	5

5	31.121152	121.141999	826466	18	7
5	31.04822	121.231928	796045	19	7
5	31.113159	121.06421	819911	26	10
5	31.058407	121.330724	811985	19	11

6th cluster a different workload range than 2nd, 3rd, and 4th clusters as shown in Table 4.11.

Table 4.11: Details of a Sample of Members from Cluster 6

Cluster ID	BS Latitude	BS Longitude	BS Workload	BS number of Requests	BS number of Users
6	31.145654	121.375598	368440	7	5
6	31.038911	121.226883	382510	7	3
6	31.261132	121.093167	373047	8	3
6	31.387389	121.431922	373817	9	4
6	31.425786	121.422506	364182	11	5
6	31.469563	121.384552	366339	14	2
6	31.25007	121.546845	379088	12	4
6	31.200179	121.551597	370294	15	4
6	31.22135	121.391319	377423	11	2
6	31.197573	121.37641	389642	11	6

As shown in Table 4.12 the non-clustered base stations have higher workload through the day (for all base stations during the day) that not belong to any cluster. These base stations cannot be neglected due to its higher load and so its important role since it receive too much requests from higher number of users. These important base stations need to be considered in any

resource allocation problem, for example, for the edge nodes placement problem, and so they need to be attached to the produced clusters. Where in the edge nodes placement problem, a method for deciding the optimal number and location of edge nodes will consider each cluster independently, where each cluster will gain its need of resources, like storage, processing and networking need. Outlier base stations need to be attached to these clusters after deciding the number and location of edge nodes. These outliers will affect the allocation problem of the physical edge nodes resources.

Table 4.12: Details of a Sample of Members from Outlier Cluster

Cluster ID	BS Latitude	BS Longitude	BS Workload	BS number of Requests	BS number of Users
-1	31.163432	121.354451	1640947	113	20
-1	31.157241	121.418919	5593230	77	21
-1	31.197909	121.426542	2489282	37	13
-1	31.182415	121.381272	2966422	43	7
-1	31.17023	121.337413	2599551	63	16
-1	31.134548	121.39095	1906992	66	9
-1	31.194567	121.453929	1035583	14	1
-1	31.179835	121.429502	2263627	32	6
-1	31.193409	121.393885	697892	40	10
-1	31.201965	121.452457	1179127	32	13

4.2.4 Using of Fractal-based Clustering to Deal with the Outlier

Fractal clustering is used to attach each outlier (non-clustered) base station to its closest cluster from the already exist produced clusters. The closest cluster is chosen based on the computed fuzzy membership that

depends on the computation of fractal dimension of the outlier base station and all clusters computed using Equation 3.17 mentioned in Chapter 3, then the base station will belong to the cluster of the highest fuzzy membership computed using Equation 3.19 mentioned in Chapter 3. Table 4.13 shows the computation of the fractal dimension of a sample of 6 outlier points, while Table 4.14 shows the attachment of these 6 points to the already existed clusters.

Table 4.13: Fractal Dimension for a Sample from the Outlier Set

Cluster Id	Day	BS Latitude	BS Longitude	BS no. of Request	BS no. of Users	BS Workload	Fractal Dimension
-1	1	31.130891	121.472384	8	23	680802	4.14967E-12
-1	1	31.100465	121.401931	12	26	1596787	7.54273E-13
-1	1	31.10114	121.181649	7	13	622805	4.95853E-12
-1	1	30.97533	121.271449	4	23	1000756	1.92034E-12
-1	1	31.054594	121.339133	6	27	1205142	1.3242E-12
-1	1	31.149628	121.43924	10	31	1545381	8.05291E-13

Table 4.14: The Fuzzy Membership for a Sample of Outlier Points to Every Cluster

Fractal Point	The Nearest Distance from the Non-Clustered Point to Every Cluster						The Fuzzy Membership of the Non-Clustered Point to Every Cluster						New Cluster ID
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	
1.7027E-12	1.3382E-11	3.95E-12	6.32E-12	2.49E-12	1.08E-12	1.08E-11	0.001531	0.017605	0.00687	0.044226	0.236672	0.00235	5
1.9024E-12	1.3182E-11	3.75E-12	6.12E-12	2.29E-12	8.77E-13	1.06E-11	0.001213	0.015017	0.005632	0.040191	0.274328	0.00188	5
1.6428E-12	1.34E-11	4.01E-12	6.38E-12	2.55E-12	1.14E-12	1.09E-11	0.001626	0.018298	0.007221	0.045174	0.227492	0.00249	5
2.2328E-12	1.2852E-11	3.42E-12	5.79E-12	1.96E-12	5.46E-13	1.03E-11	0.000681	0.009638	0.003358	0.029286	0.376958	0.00107	5
3.4493E-12	1.1635E-11	2.2E-12	4.57E-12	7.43E-13	3.95E-13	9.05E-12	0.000327	0.00916	0.002122	0.080216	0.28477	0.00054	5
5.5273E-12	9.5571E-12	1.22E-13	2.49E-12	6.83E-13	2.47E-12	6.97E-12	9.53E-05	0.585244	0.001401	0.018681	0.001424	0.00018	2

The features of the produced clusters using the DBSCAN clustering algorithm is shown in the Table 4.15, both before and after using of fractal clustering algorithm, where the outlier cluster is empty after using the fractal clustering, that is because every outlier BS attaches the closest cluster to it (with the highest fuzzy membership).

Table 4.15: Produced Cluster Features (Number of Members and Workload) Before and After Use of Fractal Clustering Algorithm

Workload Statistics for All 6 Clusters and the Outlier Cluster					
	Before or After Fractal Clustering	**No. of Members**	**Min Workload Value**	**Max Workload Value**	**Mean Workload Value**
Cluster 1	**Before**	936	0	357096	107200
	After	**937**	0	360468	107471
Cluster 2	**Before**	76	510337	583491	546471
	After	**95**	500100	605250	552400
Cluster 3	**Before**	93	395378	489709	442940
	After	**98**	395378	498820	445579
Cluster 4	**Before**	43	630079	677298	653893
	After	**79**	606245	726877	656921
Cluster 5	**Before**	36	793484	831903	811884
	After	**658**	730594	13001511	1903613
Cluster 6	**Before**	28	364182	392311	378745
	After	28	364182	392311	378745
Outliers Cluster	**Before**	683	360468	13001511	1846134
	After	**0**	-	-	-

4.3 Improved BAT Metaheuristic Algorithm – Experimental Analysis

The goal of the experiments is to evaluate the performance of the improved BAT algorithm and compare it with the original one.

Both original and improved BAT algorithms are implemented using C#, and the experiments are conducted on a computer with 11th Gen Intel® Core™ i9-11900H CPU @2.50GHz and 16.0 GB RAM in Windows 10 Pro environment.

Many testing parameters must be specified as follows:

1. *Population size:* populations of 100, 200, 300, and 1000 individuals are used in the experiments.
2. *Dimensions:* 30 is the average dimension with almost all benchmark test functions and it is used within the experiments.
3. *Total runs:* Within the experiments, the results are documented within 30 runs.
4. *Maximum iterations:* the total number of iterations for each individual run is 100.
5. Convergence speed: the computation time that is taken by the algorithm.
6. Statistical results: some information is calculated to evaluate the performance of both original and modified BAT algorithms like the best value, worst value, mean value, and the STD.

The experiments are run over ten benchmark test functions represented in Chapter 2, Section 2.17, where each function is implemented 30 times for both original and the improved BAT algorithms. All the results are documented; the statistical results (best, worst, mean, and STD values) are computed and shown in Tables 4.16, 4.17, 4.18 and 4.19 for different population sizes, such as 100, 200, 300, and 400 individuals, respectively.

Table 4.16: Results of the Comparison Between the Original and Improved BAT Algorithms with (Population Size = 100, Dimension = 30)

	Original BAT Algorithm			Modified BAT Algorithm		
Test Function	**Best Value**	**Worst Value**	**Mean**	**Best Value**	**Worst Value**	**Mean**
Ackley	0.966	1.241	1.071± 0.060	0.780	0.988	0.882 ± 0.049
Rastrigin	367.871	460.830	403.134 ± 21.025	0	0	0
Sphere Multimodal	4E-121	0.0246	0.001 ± 0.006	0	0	0
Schwefel	4.684E-14	11.831	3.1959297 ± 5.008	0	5.68E-14	1.14E-14 ± 2.94E-14
Alpine	49.445	67.874	57.462 ± 4.349	2.7E-194	1.1E-176	6.4E-178
Himmelblau	6.155E-195	3.361E-104	1.120E-105 ± .032E-105	0.012	9.253	1.700 ± 2.636
Griewank	440.483	655.143	529.789 ± 56.617	108.101	158.8376	133.8413± 14.204
SumSquare	5217.17	9560.65	7423.77 ± 1071.4	0	0	0
Sphere Unimodal	125.835	190.252	154.885 ± 15.782	0	0	0
Step	48749.03	72786.59	57759.02 ± 5772.236	5884.260	z17904.083	13147.230 ± 2999.498

Table 4.17: Results of the Comparison Between the Original and Improved BAT Algorithms with (Population Size = 200, Dimension = 30)

	Original BAT Algorithm			Modified BAT Algorithm		
Test Function	**Best Value**	**Worst Value**	**Mean**	**Best Value**	**Worst Value**	**Mean**
Ackley	0.923	1.183	1.074 ± 0.060	0.79482	0.976	0.892 ± 0.042
Rastrigin	338.404	437.766	404.402 ± 23.512	0	0	0
Sphere Multimodal	0	0	0	0	0	0
Schwefel	0	9.929	0.594 ± 2.315	0	5.684E-14	3.790E-15 ± 1.468E-14
Alpine	39.426	63.022	54.140 ± 4.935	8.9E-193	7.2E-179	4.9E-180
Himmelblau	0.030	6.266	1.473 ± 2.052	5.5E-193	3.21E-97	1.07E-98 ± 5.77E-98
Griewank	411.545	578.960	514.568 ± 46.258	94.1358	162.9374	131.514 ± 16.331
SumSquare	6042.286	8681.658	7536.947 ± 687.924	0	0	0
Sphere Unimodal	117.970	173.404	152.717 ± 15.185	0	0	0
Step	40642.83	69701.07	59886.71 ± 6889.308	5736.577	16852.91	13474.76 ± 2433.201

Table 4.18: Results of the Comparison Between the Original and Improved BAT algorithms with (Population Size = 300, Dimension = 30)

	Original BAT Algorithm			Modified BAT Algorithm		
Test Function	**Best Value**	**Worst Value**	**Mean**	**Best Value**	**Worst Value**	**Mean**
Ackley	0.9439	1.126	1.059 ± 0.049	0.814	0.967	0.888 ± 0.042
Rastrigin	324.708	433.280	398.981 ± 21.847	0	0	0
Sphere Multimodal	0	0	0	0	0	0
Schwefel	0	0	0	0	0	0
Alpine	40.131	58.485	51.873 ± 4.530	2.1E-193	5.5E-178	1.9E-179
Himmelblau	0.005	3.482	0.896 ± 1.205	3.3099E-193	7.6512E-83	2.550E-84 ± 1.373E-83
Griewank	367.932	592.541	506.638 ± 43.759	104.0246	151.7316	128.4485 ± 12.01705
SumSquare	5351.231	7926.316	6962.806 ± 744.079	0	0	0
Sphere Unimodal	114.690	174.366	147.325 ± 15.717	0	0	0
Step	43421.28	64068.7	55745.03 ± 5739.469	6897.456	16386.44	12465.72 ± 2044.637

Table 4.19: Results of the Comparison Between the Original and Improved BAT Algorithms with (Population Size = 1000, Dimension = 30)

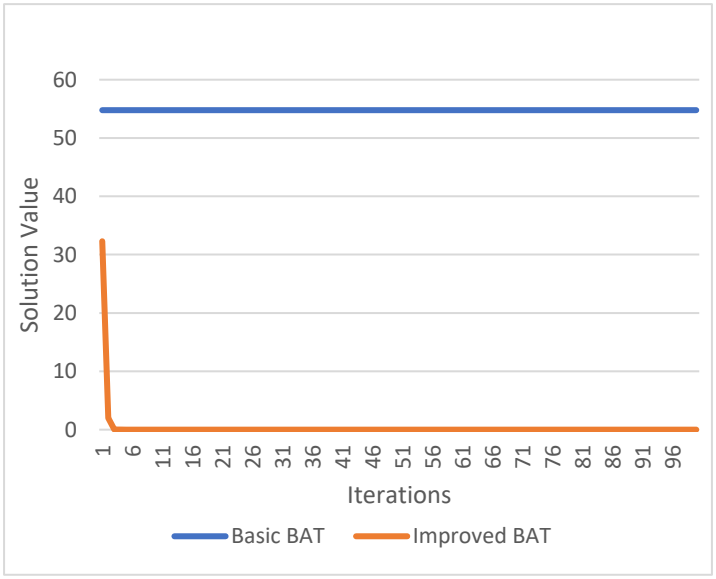
	Original BAT Algorithm			Modified BAT Algorithm		
Test Function	**Best Value**	**Worst Value**	**Mean**	**Best Value**	**Worst Value**	**Mean**
Ackley	0.934	1.090	1.018 ± 0.039	0.772946	0.885841	0.837585 ± 0.031687
Rastrigin	318.810	413.542	373.44 ± 23.477	0	0	0
Sphere Multimodal	0	0	0	0	0	0
Schwefel	0	0	0	0	0	0
Alpine	37.275	55.146	48.340 ± 4.624	0	2.65E-174	8.85E-176
Himmelblau	0.034	1.088	0.381 ± 0.480	0	7.525E-76	2.5.8E-77 ± 1.351E-76
Griewank	342.572	533.830	458.124 ± 38.226	88.541	139.992	116.454 ± 11.946
SumSquare	4721.069	8134.941	6357.135 ± 841.033	0	0	0
Sphere Unimodal	96.855	154.901	131.470 ± 12.208	0	0	0
Step	41465.39	58794.17	50709.8 ± 4928.865	5193.702	15241.46	11655.64 ± 2293.18

As shown from the previous results in Tables 4.16, 4.17, 4.18, and 4.19, the improved BAT algorithm outperforms the original BAT algorithm, especially as the population grows, and reaches much better values on all the ten benchmarks, especially for Rastrigin, Sphere (both multimodal and unimodal), Schwefel, Alpine, Himmelblau, and SumSquare functions where it reaches the optimal value (which is 0) compared to the original BAT algorithm. The original BAT gets the optimal value for Schwefel and the multimodal sphere functions only (0 value), and it is close to the optimal value in the Himmelblau function, while with the other functions, it produces worse values.

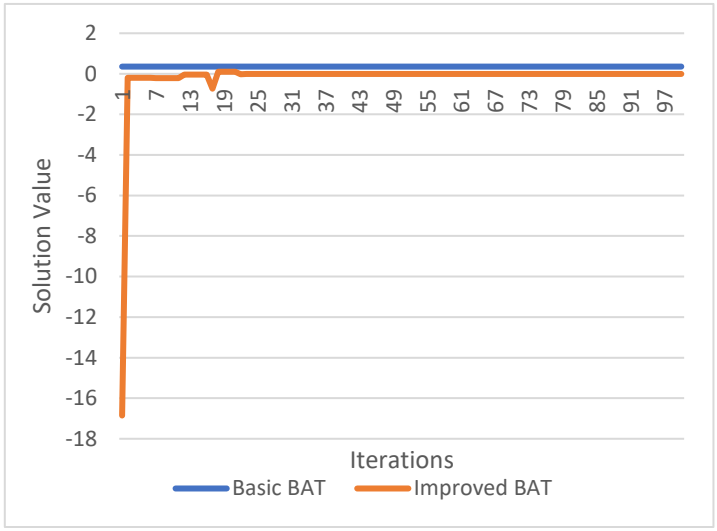
The improved BAT algorithm has a great mean value near or equal to the optimal value, except for the Griewank and Step test functions. In contrast, the original BAT algorithm mean values are far from the optimal value, except for the Himmelblau and Ackley test functions, where the mean value is near the optimal one, and it reaches the optimal value for both Schwefel and the multimodal sphere functions.

Regarding the convergence speed, the convergence to the optimal value of the improved BAT algorithm is much faster and better than the original BAT algorithm as shown in Figures 4.6 and 4.7. As a general result, the performance of the enhanced BAT algorithm is significantly better than the original BAT algorithm.

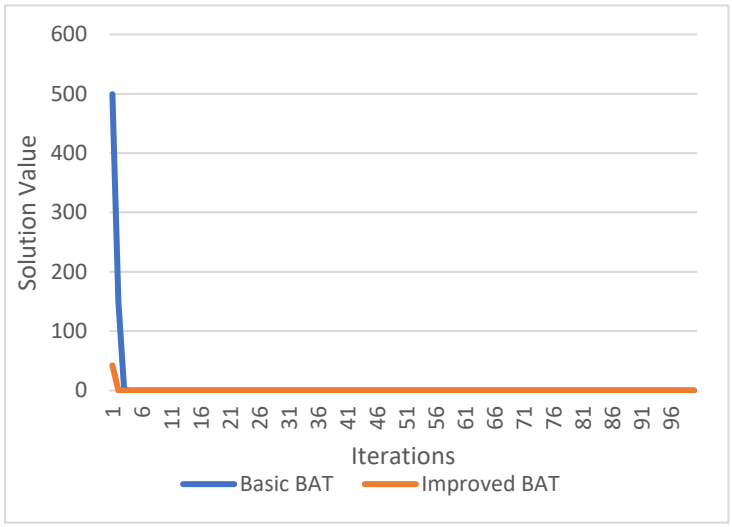
(a) Using the Rastrigin Test Function



(b) Using the Alpine Test Function

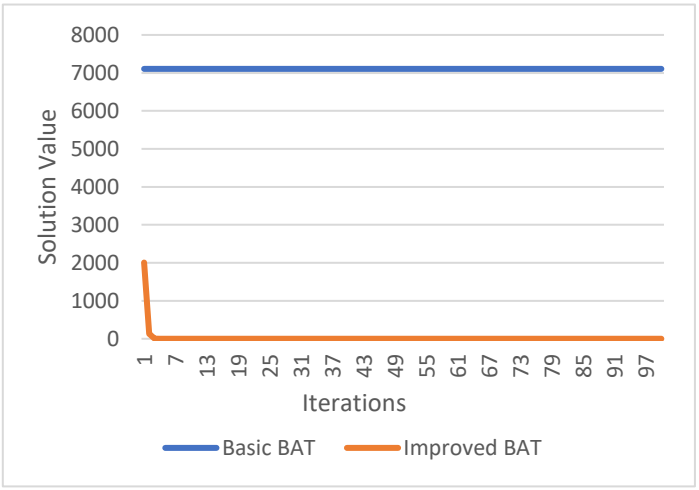


(c) Using the Himmelblau Test Function

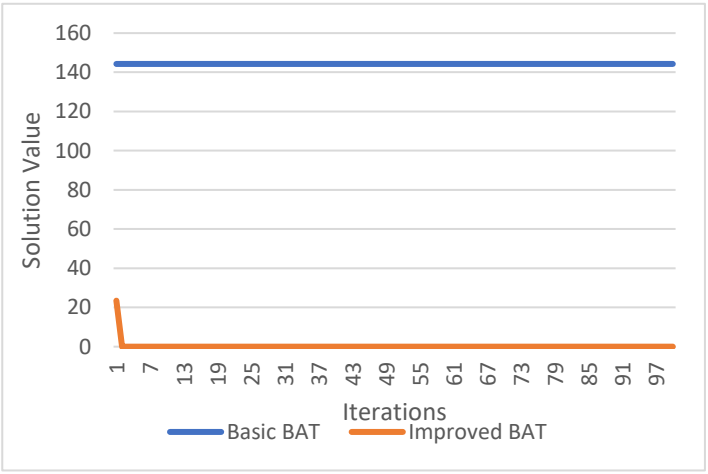


(d) Using the Multimodal Sphere Test Function

Fig. 4.6: Convergence Curves Using Four Multimodal Benchmark Test Functions (a) Rastrigin, (b) Alpine, (c) Himmelblau, (d) Multimodal Sphere



(a) Using SumSquare Test Function



(b) Using Unimodal Sphere Test Function

Fig. 4.7: Convergence Curves Using Two Unimodal Benchmark Test Functions (a) SumSquare, (b) Unimodal Sphere

All the previous results demonstrate the performance behavior of the improved BAT algorithm with all three improvements together (improved initial population, improved local search, and stagnation handling) against the original one.

4.4 Improved BAT Metaheuristic Algorithm – Incremental Analysis

An incremental analysis approach is made to diagnose which improvement has the bulk effect. This analysis is based on evaluating the performance of the algorithm over six cases, three cases of them are with only one improvement as follows:

Case 1: Partially Improved BAT algorithm WITH ONLY improved initial population.

Case 2: Partially Improved BAT algorithm WITH ONLY improved local search.

Case 3: Partially Improved BAT algorithm WITH ONLY stagnation handling.

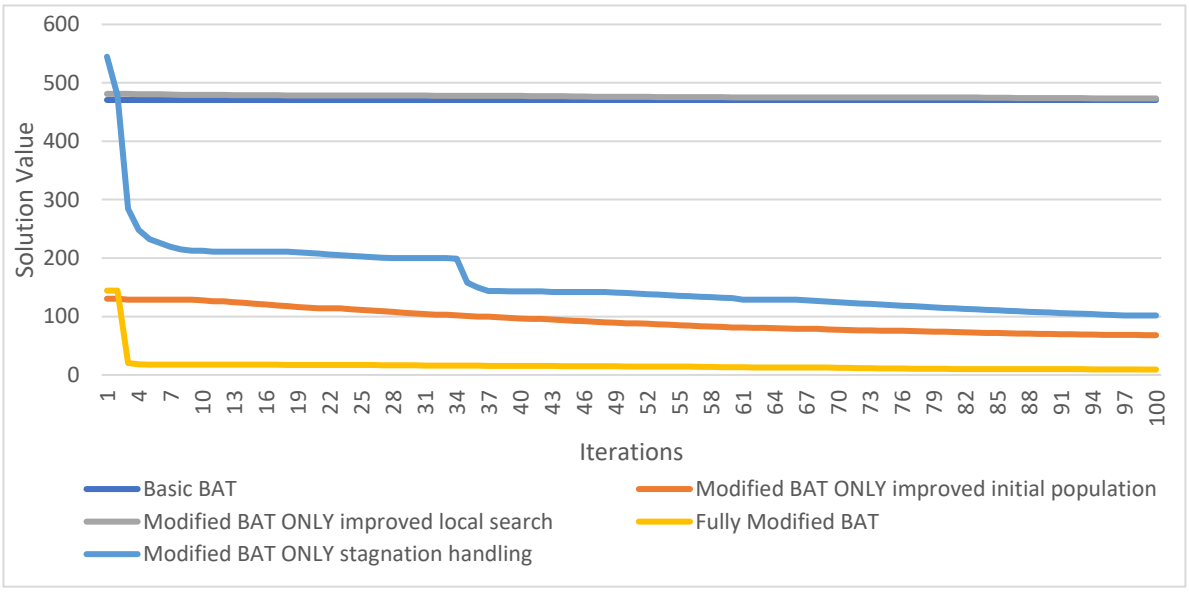
The other three cases are with two improvements as follows:

Case 4: Partially Improved BAT algorithm but WITHOUT improved initial population.

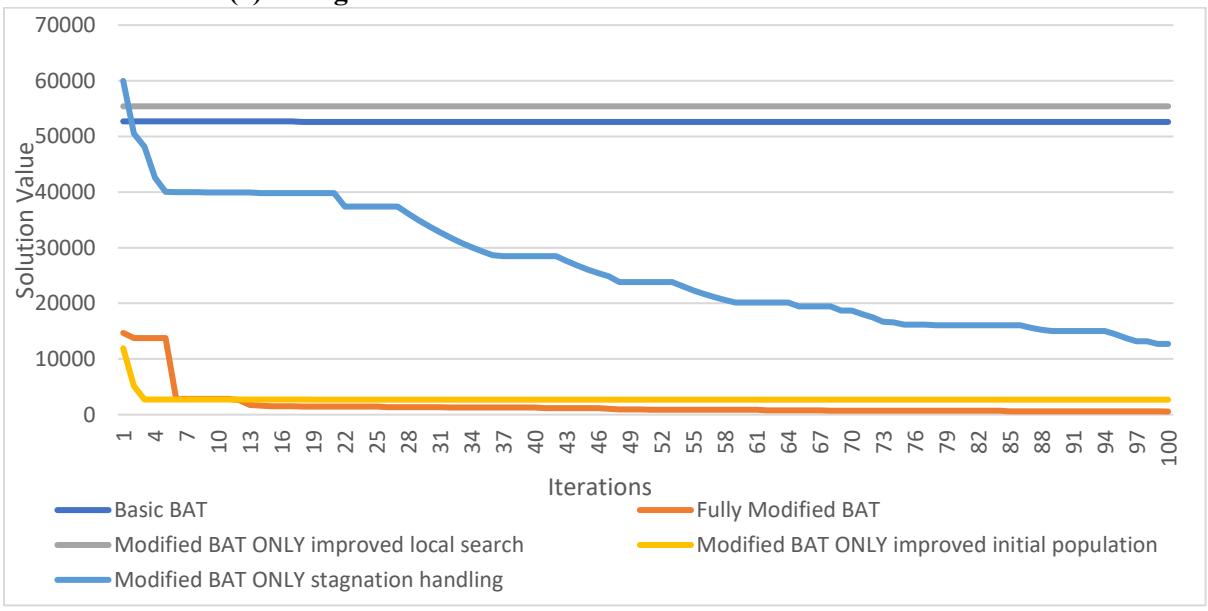
Case 5: Partially Improved BAT algorithm but WITHOUT improved local search.

Case 6: Partially Improved BAT algorithm but WITHOUT stagnation handling.

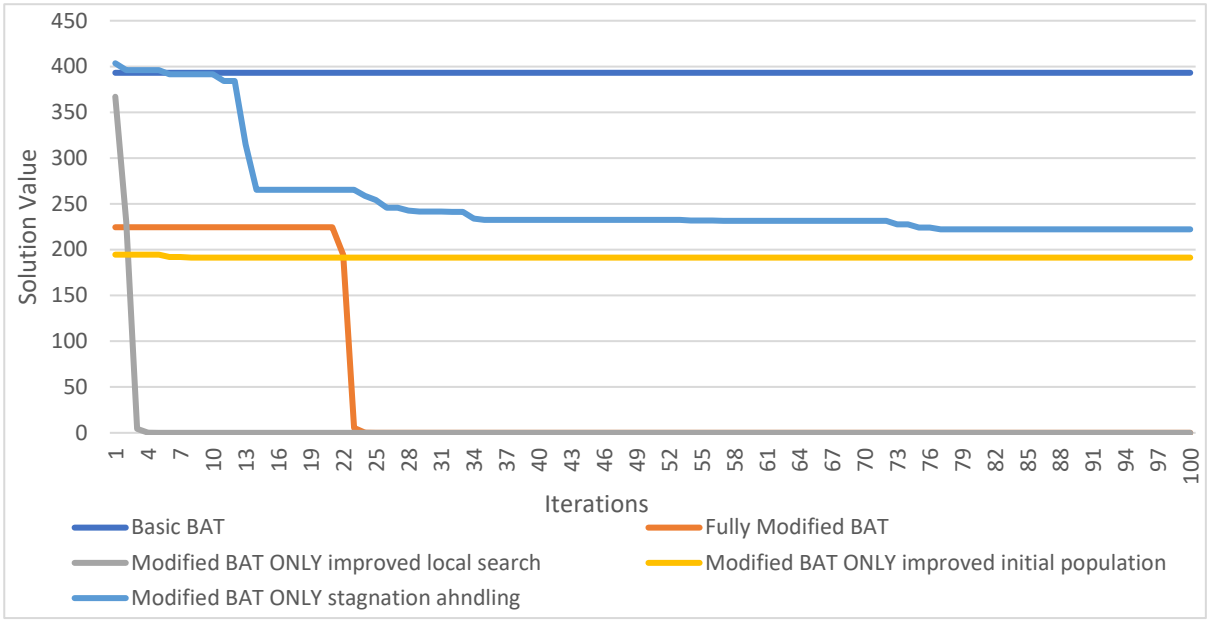
Figure 4.8 shows the performance of the partially improved BAT (first three cases) against the original BAT and fully improved BAT (with all three improvements together), while Figure 4.9 shows the performance of the partially improved BAT (second three cases) against the original BAT and fully improved BAT (with all three improvements together)



(a) Using Griewank Multimodal Benchmark Test Functions



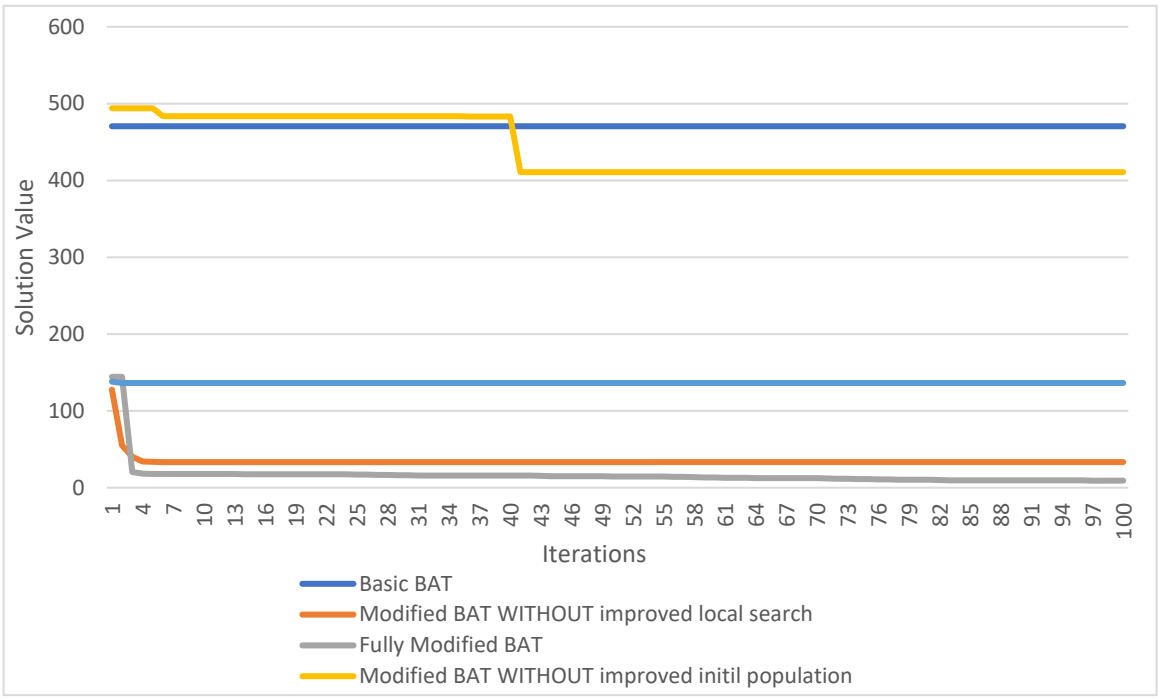
(b) Using Step Unimodal Benchmark Test Functions



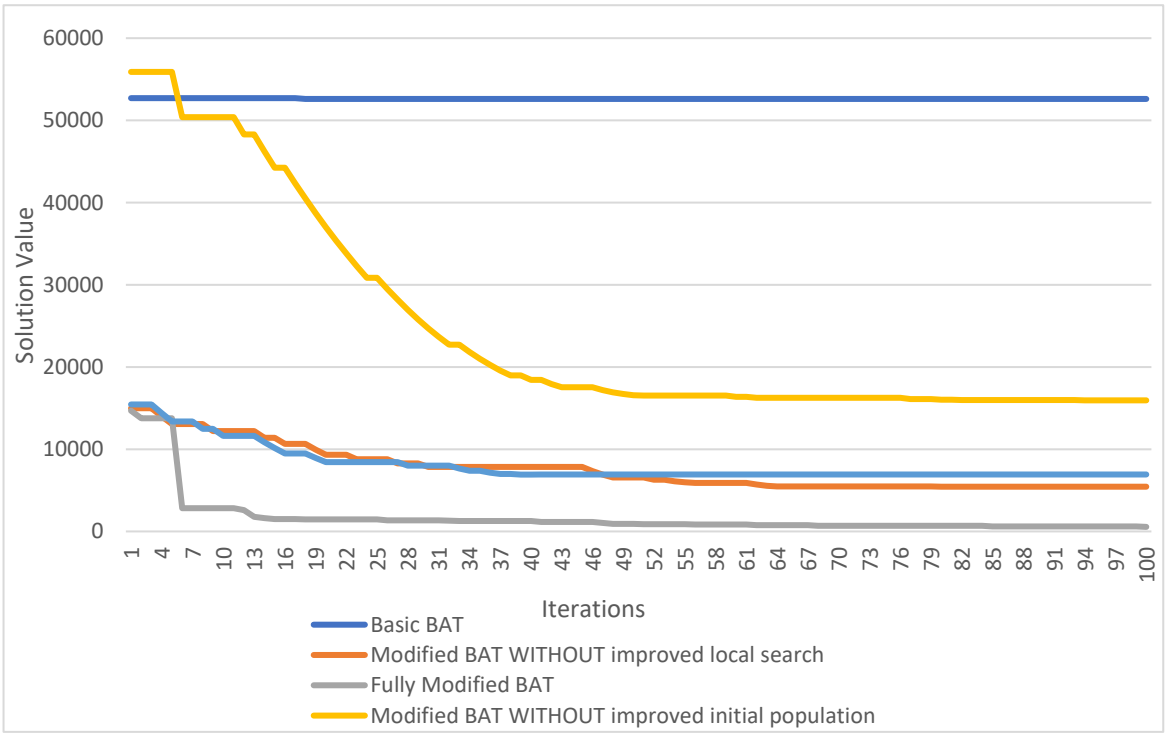
(c) Using Rastrigin Multimodal Benchmark Test Function

(d) Using Alpine Multimodal Benchmark Test Function

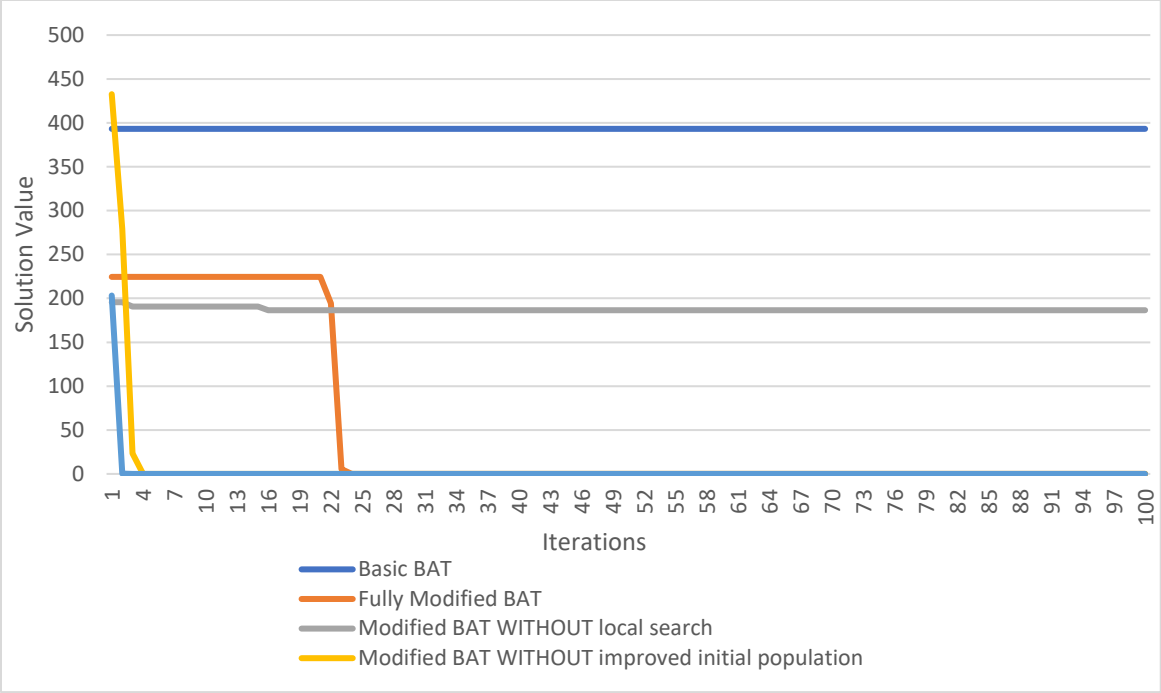
Fig. 4.8: Convergence Curves Using Four Benchmark Test Functions Over Original BAT, Fully Improved BAT, and Cases 1, 2, and 3



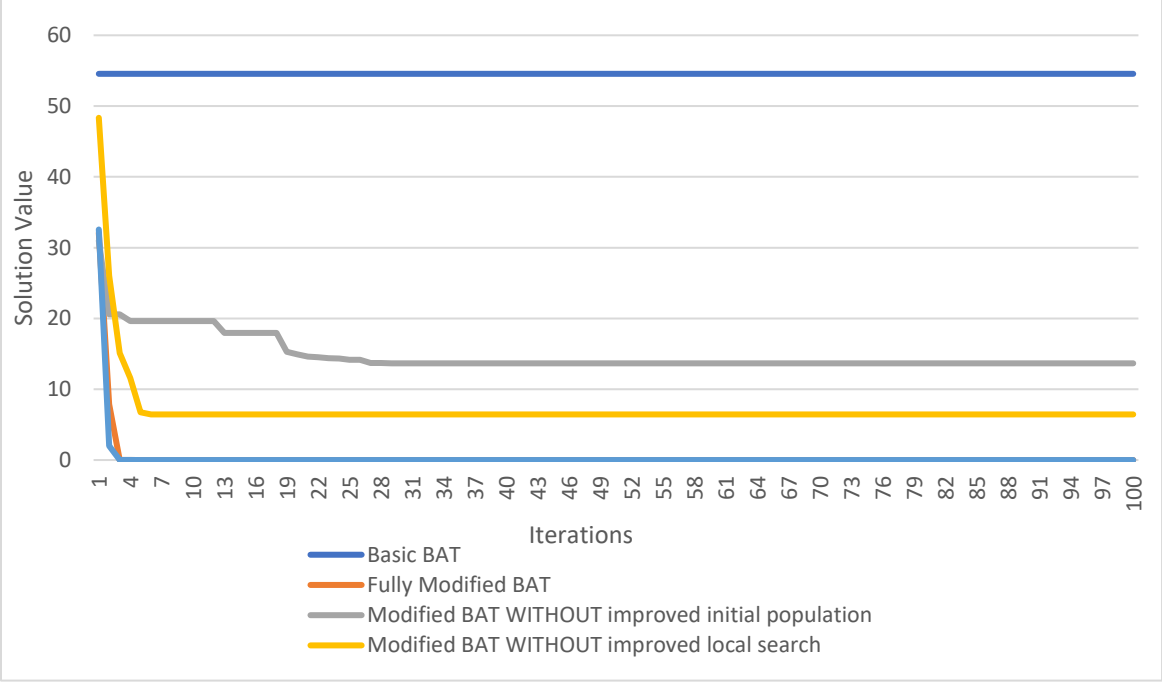
(a) Using Griewank Multimodal Benchmark Test Functions



(b) Using Step Unimodal Benchmark Test Functions



(c) Using Rastrigin Multimodal Benchmark Test Function



(d) Using Alpine Multimodal Benchmark Test Function

Fig. 4.9: Convergence Curves Using Four Benchmark Test Functions Over Original BAT, Fully Improved BAT, and Cases 4, 5, and 6

As noticed clearly from Figure 4.8, the most significant role in the improvement is given to the population initialization phase, where the performance of the BAT algorithm with only an improved initial population is the first better performance after the fully improved BAT algorithm in three out of four test functions (Griewank, Step, and Alpine). The second significant role is equally due to improved local search and stagnation handling. In the Alpine test function, the performance of the BAT with only improved local search is the second better performance after Full improved BAT algorithm and BAT with only improved initial population. In the Rastrigin test function, the performance of the BAT with only improved local search is better than even the BAT algorithm with the improved initial population.

BAT algorithm with only stagnation handling has the second better performance after Full improved BAT algorithm and BAT with only improved initial population in two out of four test functions.

From another perception, the same thing is noticed in Figure 4.9, the improved BAT WITHOUT improved initial population has the worst performance after the Basic BAT algorithm in three out of four test functions (Griewank, Step, and Alpine), which means that the improved initial population has the most significant role in the improvement of BAT algorithm.

The convergence is improved in case there is an improvement in the initial population (in whatever case, with handling the stagnation or not, and with improving the local search or not).

The second significant role or effect is using chaotic maps to improve the local search (balancing the exploration and exploitation).

4.5 Comparing the Proposed Edge Nodes Placement Method with the Benchmark Placement Methods

The performance of the proposed placement method based on improved BAT metaheuristic algorithm will be evaluated by comparing it with the Random and Top-k benchmark placement methods as the following subsections.

4.5.1 The Numbers of Deployed Edge Nodes

As a primary setting to the proposed placement methods is the number of candidate edge nodes. Improved BAT Algorithm will take this number as the number of its individuals. The first population is generated randomly from all the BSs of the day, and the second one will be generated depending on the produced clusters. The improved population will enhance the selection of the edge servers' locations.

Each cluster will participate in the population depending on its number of BSs members to the total number of BSs of the day. This gives a guarantee that each cluster will get its need of resources from the deployed edge server within.

When conducting the improved BAT algorithm, there is a different number of candidate edge for each time period, that is because of the different workload heaviness. For example, for the first day in the dataset there is 1895 BS, so the number of candidate edge nodes for each time period for the improved BAT and the random and Top-K benchmark edge servers placement methods is as shown in Figure 4.10, where the number of candidate edge servers is fixed for all three period for both benchmark placement method, and varies with the proposed improved BAT algorithm.

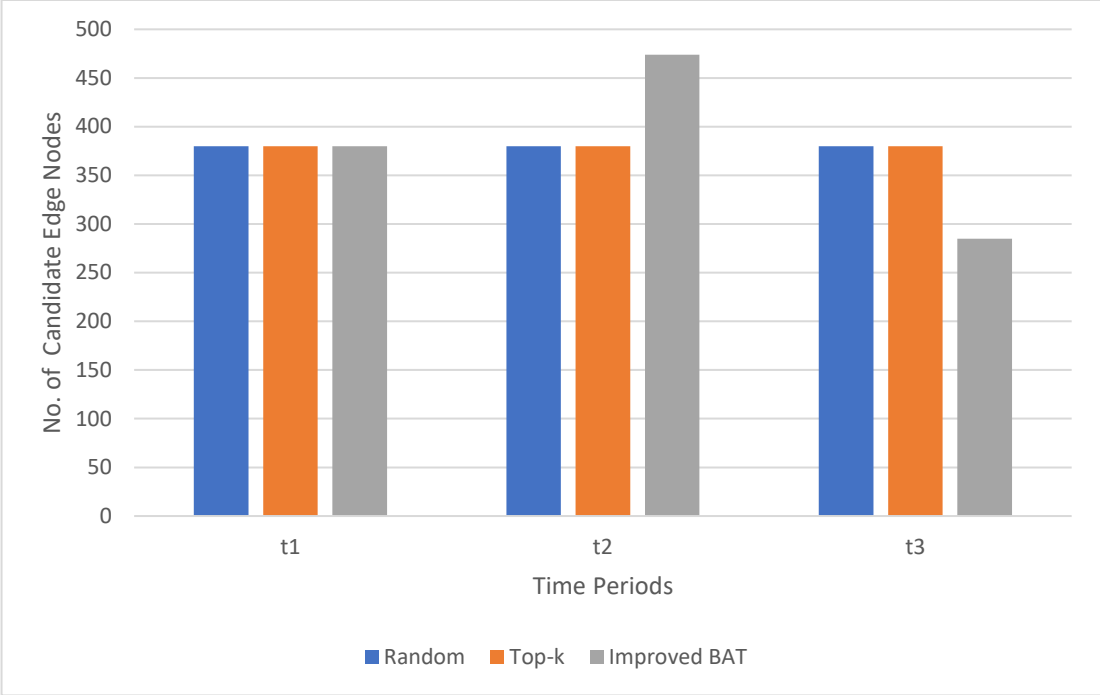


Fig. 4.10: The Number of Candidate Edge Nodes in the First Day of the Dataset

The participate ratio of each six clusters for the 1st, 2nd and 3rd periods is as shown in Tables 4.20, 4.21 and 4.22 respectively, where the population size for the 1st period is equal to 376 (20% the total number of BS within the first day), and the population size for the 2nd period is equal to 470 (25% the total number of BS within the first day), while the population size for the 3rd period is equal to 280 (15% the total number of BS within the first day).

Table 4.20: Participation Ratio of Each Cluster in the Population for 1st Time Period

Cluster ID	No. of Members	Participate Ratio in the Population	No. of Individuals provided by each Cluster
1	937	0.49	**186**
2	95	0.05	**19**
3	98	0.05	**19**
4	79	0.04	**15**
5	658	0.35	**133**
6	28	0.01	**4**

Table 4.21: Participation Ratio of Each Cluster in the Population for 2nd Time Period

Cluster ID	No. of Members	Participate Ratio in the Population	No. of Individuals provided by each Cluster
1	937	0.49	**232**
2	95	0.05	**24**
3	98	0.05	**24**
4	79	0.04	**19**
5	658	0.35	**166**
6	28	0.01	**5**

Table 4.22: Participation Ratio of Each Cluster in the Population for 3rd Time Period

Cluster ID	No. of Members	Participate Ratio in the Population	No. of Individuals provided by each Cluster
1	937	0.49	**139**
2	95	0.05	**14**
3	98	0.05	**14**
4	79	0.04	**11**
5	658	0.35	**99**
6	28	0.01	**3**

Both Random and Top-k placement approaches deploy the same number of edge nodes for each day which distributed randomly among all the BSs for serving all BSs within their coverage area.

On the contrast, the improved BAT algorithm deploys much smaller number of edge node during the day, that is the proposed approach aims to balance the workload between edge servers and aims to better utilize each edge servers. That is of course will cause longer latency time than Random and Top-k approaches, but achieves better edge system utilization and better-balanced workload, and generally better edge system performance. Figure 4.11 shows the number of deployed edge servers within a day.

It's clearly shown that the 2nd time period has the highest number of deployed edge servers due to its heaviest workload throughout the

day, while the 3rd period has the lowest number of deployed edge server due to its lower workload demand.

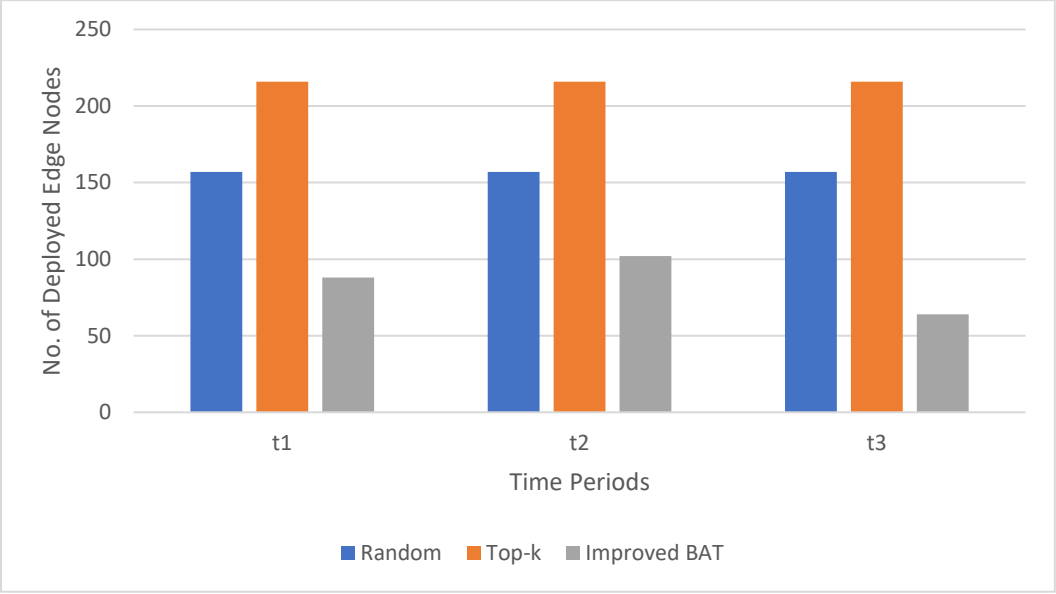


Fig. 4.11: The Number of Deployed Edge Nodes in the First Day of the Dataset

4.5.2 The System Latency

Latency is measured as the sum of distances between each edge node and all BSs attached to it. This distance is computed using Euclidean distance. As the location of each BS is represented by the format *mm/dd/yyyy hh:mm:ss"* 24-Hour format, so the distance will represent the differenced time in second. Figure 4.12 shows that the proposed edge nodes' placement model has near similar latency as the both benchmark approaches.

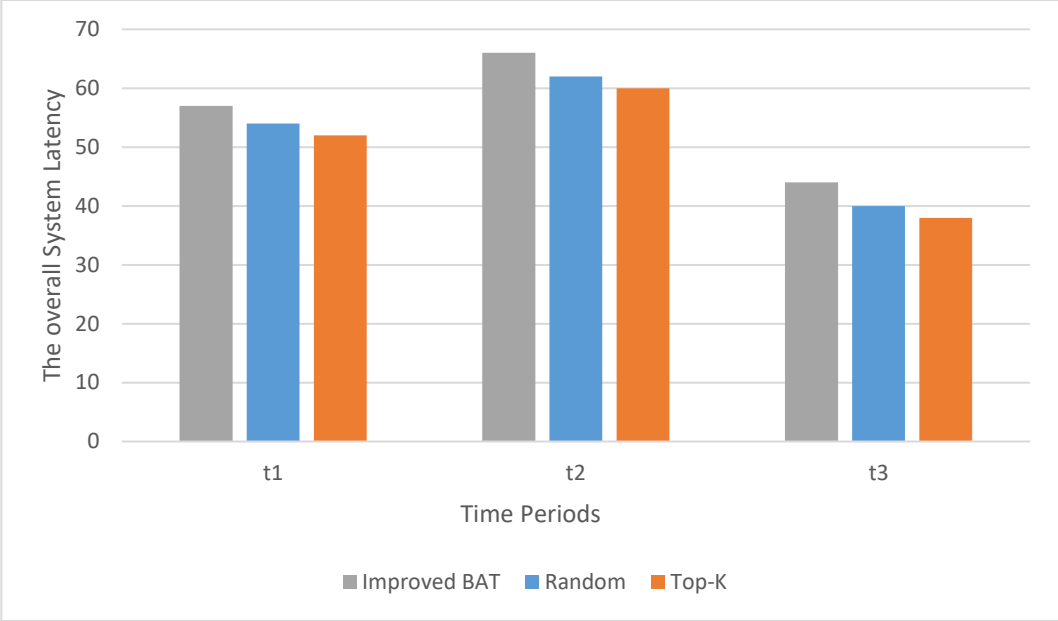


Fig. 4.12: The System Latency

4.5.3 The Edge Nodes Utilization

Figure 4.13 shows the utilization of the edge servers using the proposed BAT algorithm against Random and Top-k benchmark placement methods.

As shown from the Figure 4.13 the best edge system utilization is achieved with the proposed improved BAT placement method. It gets (30.7% - 50%) improvement on average than the Top-k approach, and gets (30.1% - 40.5%) improvement on average than the Random approach.

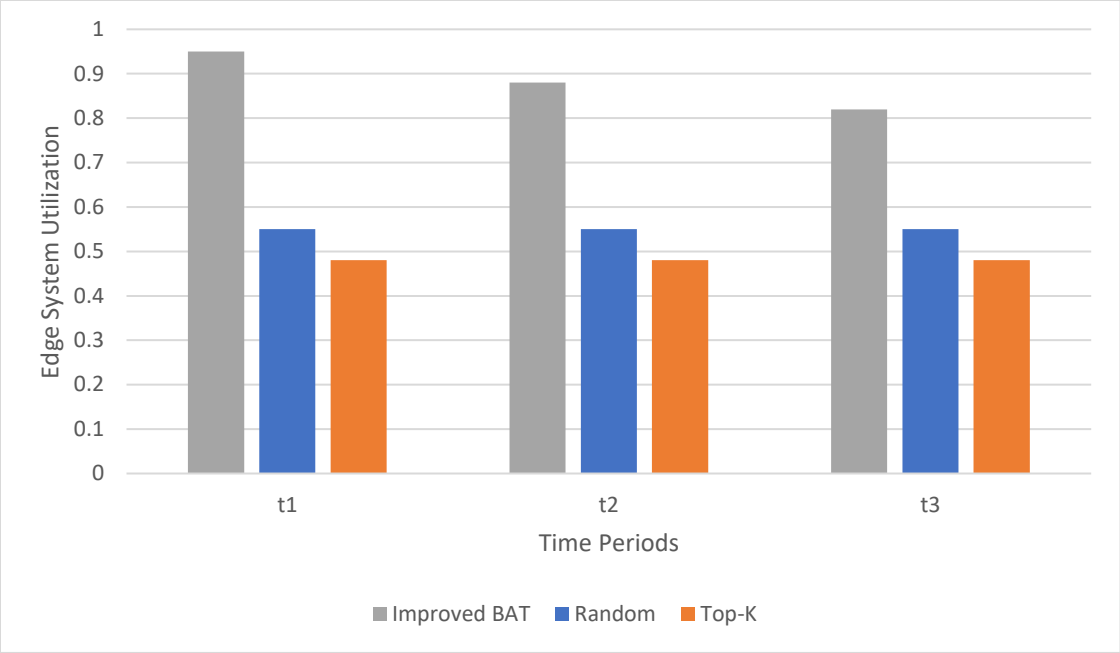


Fig. 4.13: Edge System Utilization

4.5.4 Balanced Workload

The improved BAT method for edge nodes' placement results in smaller number of edge nodes than both benchmark placement methods, this will result in a balanced workload (workload difference) among the edge nodes. Figure 4.14 shows the balanced workload among edge servers using the proposed BAT algorithm against Random and Top-k benchmark placement methods.

The best balancing workload is found with the proposed improved BAT algorithm, where it has much lower workload difference among the deployed edge servers than both benchmark placement methods.

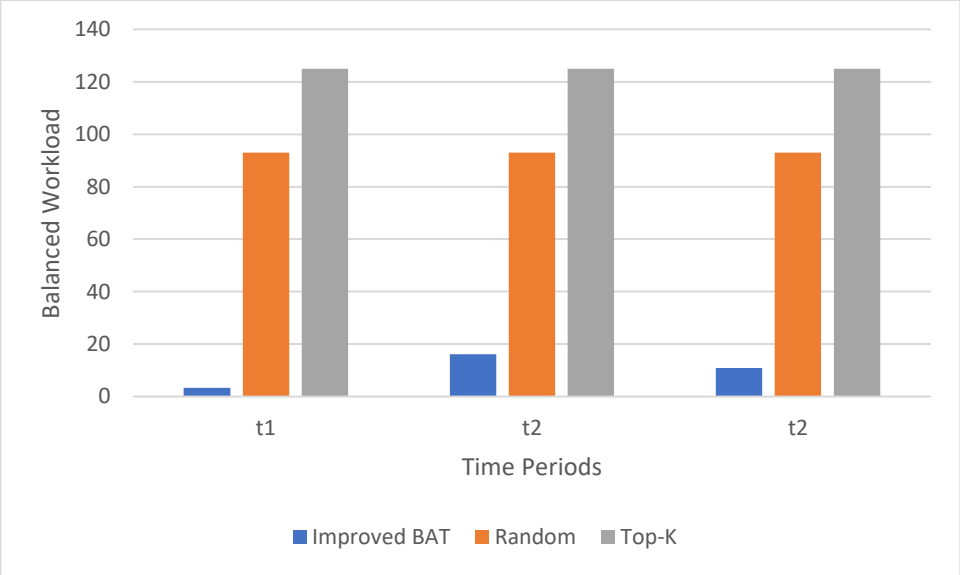


Fig. 4.14: The Balanced Workload

Generally, the results of the proposed edge node placement method reflect its efficiency in terms of network latency, edge system utilization and balanced workload. The proposed system deployed a smaller number of edge node (the only required number of edge nodes), and this will save the system energy.

CHAPTER FIVE

CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

Edge and Fog computing are both beneficent technologies that complement traditional and central cloud computing. They improve the performance of IoT devices by minimizing latency, especially for delay-critical applications. Edge and Fog nodes' deployment is considered a critical issue in the IoT and 5G systems, so defining the influencing parameters of the deployment is essential to get the perfect and optimal one.

This dissertation proposes an edge nodes' placement approach depends on an improved BAT metaheuristic algorithm with the aid of density-based clustering and chaotic strategies, with three improvements to the original BAT algorithm to improve its exploration and exploitation abilities and enhance its performance. The main conclusions are:

1- The greater improvement impact is going to the improving of the initial population, that is the better the population quality, the better the results are.

2- Handling the stagnation issue has the second impact on the improvement, since it prevents the algorithm from being stuck in the local optima.

3- Using of chaotic maps has the lower impact on the improvement of the BAT algorithm.

4- Using of clustering technique make an essential role in the approach, since it groups the same features' points in a cluster with some known requirement. Depending on these requirements, the proposed approach assigns the appropriate number and location of edge nodes.

5- Using more than one objective in the decision making of the number and location of the edge nodes, make the approach more accurate

and beneficial. In case of using a single objective, the decision will be biased toward that objective and neglect the other, while they are important also.

5.2 Future Works

1- The proposed edge nodes' placement approach is centralized and relies on the existence of the base broker. As a future work, we consider proposing edge nodes' placement system with more than one base broker to ensure the reliability. If one base broker fails, the other do the required actions.

2- Considering the distributed edge nodes placement system beside the centralized one.

3- Edge and Fog computing plays a vital role in our daily lives and activities. As edge/fog node placement is crucial, other resource allocation and management problems are also essential. As a future scope, we consider building a model which gets an optimal edge nodes placement and simultaneously considers balancing the workload between these deployed nodes by scheduling users' tasks, offloading some of these tasks to the underutilized edge/fog nodes, and so achieving load balancing through the network.

References

[1] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge Computing for the Internet of Things: A Case Study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, 2018, doi: 10.1109/JIOT.2018.2805263.

[2] C.-C. Lin, D.-J. Deng, S. Suwatcharachaitiwong, and Y.-S. Li, "Dynamic Weighted Fog Computing Device Placement Using a Bat-Inspired Algorithm with Dynamic Local Search Selection," *Mob. Networks Appl.*, vol. 25, no. 5, pp. 1805–1815, 2020, doi: 10.1007/s11036-020-01565-9.

[3] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand, "IFogStor: An IoT Data Placement Strategy for Fog Infrastructure," *Proc. - 2017 IEEE 1st Int. Conf. Fog Edge Comput. ICFEC 2017*, pp. 97–104, 2017, doi: 10.1109/ICFEC.2017.15.

[4] H. A. Alharbi, T. E. H. Elgorashi, and J. M. H. Elmirghani, "Energy efficient virtual machines placement over cloud-fog network architecture," *IEEE Access*, vol. 8, pp. 94697–94718, 2020, doi: 10.1109/ACCESS.2020.2995393.

[5] Y. Liang *et al.*, "Interaction-Oriented Service Entity Placement in Edge Computing," *IEEE Trans. Mob. Comput.*, vol. 20, no. 3, pp. 1064–1075, 2021, doi: 10.1109/TMC.2019.2952097.

[6] A. Asensio *et al.*, "Designing an efficient clustering strategy for combined Fog-to-Cloud scenarios," *Futur. Gener. Comput. Syst. Int. J. ESCIENCE*, vol. 109, pp. 392–406, 2020, doi: 10.1016/j.future.2020.03.056.

[7] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, "Exploring Placement of Heterogeneous Edge Servers for Response Time Minimization in Mobile Edge-Cloud Computing," *IEEE Trans. Ind. Informatics*, vol. 17, no. 1, pp. 494–503, 2021, doi: 10.1109/TII.2020.2975897.

[8] O. Chukhno, N. Chukhno, G. Araniti, C. Campolo, A. Iera, and A. Molinaro, "Optimal placement of social digital twins in edge iot networks," *Sensors (Switzerland)*, vol. 20, no. 21, pp. 1–17, 2020, doi: 10.3390/s20216181.

[9] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5853–5863, 2019, doi: 10.1109/JIOT.2019.2907605.

[10] S. W. Yuanzhe Li, Ao Zhou, Xiao Ma, "Profit-aware Edge Server Placement," *IEEE INTERNET THINGS J.*, vol. 4662, no. c, pp. 66–73, 2021, doi: 10.1109/jiot.2021.3082898.

[11] X. Zhang, S. Huang, H. Dong, and Z. Bao, "Edge Node Placement with Minimum Costs: When User Tolerance on Service Delay Matters," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 13121 LNCS, pp. 765–772, 2021, doi: 10.1007/978-3-030-91431-8_53.

[12] X. Zhao, Y. Zeng, H. Ding, B. Li, and Z. Yang, "Optimize the placement of edge server between workload balancing and system delay in smart city," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 6, pp. 3778–3792, 2021, doi: 10.1007/s12083-021-01208-0.

[13] I. Leyva-Pupo, A. Santoyo-González, and C. Cervelló-Pastor, “A framework for the joint placement of edge service infrastructure and user plane functions for 5G,” *Sensors (Switzerland)*, vol. 19, no. 18, 2019, doi: 10.3390/s19183975.

[14] G. Cui, Q. He, X. Xia, F. Chen, H. Jin, and Y. Yang, “Robustness-oriented k Edge Server Placement,” *Proc. - 20th IEEE/ACM Int. Symp. Clust. Cloud Internet Comput. CCGRID 2020*, pp. 81–90, 2020, doi: 10.1109/CCGrid49817.2020.00-85.

[15] T. Lähderanta *et al.*, “Edge computing server placement with capacitated location allocation,” *J. Parallel Distrib. Comput.*, vol. 153, pp. 130–149, 2021, doi: 10.1016/j.jpdc.2021.03.007.

[16] B. Li, K. Wang, D. Xue, and Y. Pei, “K-Means based edge server deployment algorithm for edge computing environments,” in *Proceedings - 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations, SmartWorld/UIC/ATC/ScalCom/CBDCo*, 2018, pp. 1169–1174, doi: 10.1109/SmartWorld.2018.00203.

[17] J. Lu, J. Jiang, V. Balasubramanian, M. R. Khosravi, and X. Xu, “Deep reinforcement learning-based multi-objective edge server placement in Internet of Vehicles,” *Comput. Commun.*, vol. 187, pp. 172–180, 2022, doi: https://doi.org/10.1016/j.comcom.2022.02.011.

[18] F. Luo, S. Zheng, W. Ding, J. Fuentes, and Y. Li, “An Edge Server Placement Method Based on Reinforcement Learning,” *Entropy*, vol. 24, no. 3, pp. 1–14, 2022, doi: 10.3390/e24030317.

[19] M. K. Kasi, S. A. Ghazalah, R. N. Akram, and D. Sauveron, “Secure mobile edge server placement using multi-agent reinforcement learning,” *Electron.*, vol. 10, no. 17, pp. 1–19, 2021, doi: 10.3390/electronics10172098.

[20] X. Xu *et al.*, “Edge Server Quantification and Placement for Offloading Social Media Services in Industrial Cognitive IoV,” *IEEE Trans. Ind. Informatics*, vol. 17, no. 4, pp. 2910–2918, 2021, doi: 10.1109/TII.2020.2987994.

[21] Z. H. Lv and L. Qiao, “Optimization of collaborative resource allocation for mobile edge computing,” *Comput. Commun.*, vol. 161, pp. 19–27, 2020, doi: 10.1016/j.comcom.2020.07.022.

[22] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C. H. Hsu, “User allocation-aware edge cloud placement in mobile edge computing,” *Softw. - Pract. Exp.*, vol. 50, no. 5, pp. 489–502, 2020, doi: 10.1002/spe.2685.

[23] G. Manasvi, A. Chakraborty, and B. S. Manoj, “Social Network Aware Dynamic Edge Server Placement for Next-Generation Cellular Networks,” *2020 Int. Conf. Commun. Syst. NETworkS, COMSNETS 2020*, pp. 499–502, 2020, doi: 10.1109/COMSNETS48256.2020.9027421.

[24] G. Cui, Q. He, F. Chen, H. Jin, and Y. Yang, “Trading off between User Coverage and Network Robustness for Edge Server Placement,” *IEEE Trans. Cloud Comput.*, vol. XX,

no. X, pp. 1–12, 2020, doi: 10.1109/TCC.2020.3008440.

[25] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *Sensors (Switzerland)*, vol. 19, no. 1, pp. 1–21, 2019, doi: 10.3390/s19010032.

[26] S. Yilmaz and E. U. Küçüksille, "A new modification approach on bat algorithm for solving optimization problems," *Appl. Soft Comput. J.*, vol. 28, pp. 259–275, 2015, doi: 10.1016/j.asoc.2014.11.029.

[27] X.-S. S. Yang and M. Karamanoglu, *Nature-Inspired Metaheuristic Algorithms Second Edition*, vol. 4, no. C. Luniver Press, United Kingdom, 2013.

[28] Z. Haruna and S. A. T. Mu'azu, Muhammad B., Kabir A. Abubilal, "Development of a Modified Bat Algorithm using Elite Opposition – Based Learning," *IEEE 3rd Int. Conf. Electro-Technology Natl. Dev. Dev.*, pp. 144–151, 2017.

[29] X. Shan, K. Liu, and P. L. Sun, "Modified Bat Algorithm Based on Lévy Flight and Opposition Based Learning," *Sci. Program.*, vol. 2016, 2016, doi: 10.1155/2016/8031560.

[30] M. R. Chen, Y. Y. Huang, G. Q. Zeng, K. Di Lu, and L. Q. Yang, "An improved bat algorithm hybridized with extremal optimization and Boltzmann selection," *Expert Syst. Appl.*, vol. 175, no. March, p. 114812, 2021, doi: 10.1016/j.eswa.2021.114812.

[31] S. Yilmaz, E. U. Kucuksille, and Y. Cengiz, "Modified bat algorithm," *Elektron. ir Elektrotechnika*, vol. 20, no. 2, pp. 71–78, 2014, doi: 10.5755/j01.eee.20.2.4762.

[32] D. Tansui and A. Thammano, "An Enhanced Bat Algorithm with Random Walk for Solving Continuous Optimization Problems," *Proc. - 20th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2019*, pp. 39–44, 2019, doi: 10.1109/SNPD.2019.8935679.

[33] X. Wang, W. Wang, and Y. Wang, "An Adaptive Bat Algorithm," *IIntelligent Comput. Theor. Technol. 9th Int. Conf. ICIC 2013, Nanning, China, July 28-31, 2013. Proc. 9. Springer Berlin Heidelberg, 2013.*, pp. 216–223, 2013.

[34] J. Huang and Y. Ma, "Bat algorithm based on an integration strategy and gaussian distribution," *Math. Probl. Eng.*, vol. 2020, pp. 1–22, 2020, doi: 10.1155/2020/9495281.

[35] S. S. Guo, J. S. Wang, and X. X. Ma, "Improved Bat Algorithm Based on Multipopulation Strategy of Island Model for Solving Global Function Optimization Problem," *Comput. Intell. Neurosci.*, vol. 2019, 2019, doi: 10.1155/2019/6068743.

[36] A. Rezaee Jordehi, "Chaotic bat swarm optimisation (CBSO)," *Appl. Soft Comput. J.*, vol. 26, pp. 523–530, 2014, doi: 10.1016/j.asoc.2014.10.010.

[37] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, pp. 27–33, 2010, doi: 10.1109/AINA.2010.187.

[38] E. Balevi and R. D. Gitlin, "Optimizing the Number of Fog Nodes for Cloud-Fog-Thing Networks," *IEEE Access*, vol. 6, pp. 11173–11183, 2018, doi: 10.1109/ACCESS.2018.2808598.

[39] Y. Li and S. Wang, “An energy-aware edge server placement algorithm in mobile edge computing,” *Proc. - 2018 IEEE Int. Conf. Edge Comput. EDGE 2018 - Part 2018 IEEE World Congr. Serv.*, pp. 66–73, 2018, doi: 10.1109/EDGE.2018.00016.

[40] R. A. C. da Silva and N. L. S. da Fonseca, “On the location of fog nodes in fog-cloud infrastructures,” *Sensors (Switzerland)*, vol. 19, no. 11, 2019, doi: 10.3390/s19112445.

[41] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, “A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges,” *IEEE Commun. Surv. Tutorials*, vol. 20, no. 1, pp. 416–464, 2018, doi: 10.1109/COMST.2017.2771153.

[42] H. Sami and A. Mourad, “Dynamic On-Demand Fog Formation Offering On-the-Fly IoT Service Deployment,” *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 2, pp. 1026–1039, 2020, doi: 10.1109/TNSM.2019.2963643.

[43] A. V Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, *Chapter 4 - Fog Computing: principles, architectures, and applications*, vol. i. Elsevier Inc.

[44] M. Aazam and S. Korea, “Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT,” 2015, doi: 10.1109/AINA.2015.254.

[45] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi, and M. Guizani, “Edge and fog computing for IoT: A survey on current research activities & future directions,” *Comput. Commun.*, vol. 180, pp. 210–231, 2021, doi: 10.1016/j.comcom.2021.09.003.

[46] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014, doi: 10.1109/JIOT.2014.2306328.

[47] S. Dash, S. Biswas, D. Banerjee, and Atta-Ur-Rahman, “Edge and fog computing in healthcare - A review,” *Scalable Comput.*, vol. 20, no. 2, pp. 191–206, 2019, doi: 10.12694/scpe.v20i2.1504.

[48] P. Hu, S. Dhelim, H. Ning, and T. Qiu, “Survey on fog computing: architecture, key technologies, applications and open issues,” *J. Netw. Comput. Appl.*, vol. 98, pp. 27–42, 2017, doi: 10.1016/j.jnca.2017.09.002.

[49] V. Pande, C. Marlecha, and S. Kayte, “A Review- Fog Computing and Its Role in the Internet of Things,” *Int. J. Eng. Res. Appl.*, vol. 6, no. 10, pp. 7–11, 2016, [Online]. Available: https://doaj.org/article/56d8bfb360744411ac6b95163af341e8.

[50] S. O. Ogundoyin and I. A. Kamil, “Optimization techniques and applications in fog computing: An exhaustive survey,” *Swarm Evol. Comput.*, vol. 66, p. 100937, 2021, doi: https://doi.org/10.1016/j.swevo.2021.100937.

[51] N. Wang, Y. Cai, J. Fu, and J. Xu, “Privacy-Preserving Efficient Data Retrieval in IoMT Based on Low-Cost Fog Computing,” *Complexity*, vol. 2021, 2021, doi: 10.1155/2021/6211475.

[52] M. N. O. Sadiku, M. Tembely, and S. M. Musa, “Fog Computing: A Primer,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 7, no. 7, p. 405, 2017, doi: 10.23956/ijarcsse.v7i7.165.

[53] A. Santoyo-Gonzalez and C. Cervello-Pastor, “Edge Nodes Infrastructure Placement Parameters for 5G Networks,” *2018 IEEE Conf. Stand. Commun. Networking, CSCN 2018*, 2018, doi: 10.1109/CSCN.2018.8581749.

[54] J. Bellendorf and Z. Á. Mann, “Classification of optimization problems in fog computing,” *Futur. Gener. Comput. Syst.*, vol. 107, pp. 158–176, 2020, doi: 10.1016/j.future.2020.01.036.

[55] H. Djigal, J. Xu, L. Liu, and Y. Zhang, “Machine and Deep Learning for Resource Allocation in Multi-Access Edge Computing: A Survey,” *IEEE Commun. Surv. Tutorials*, vol. 24, no. 4, pp. 2449–2494, 2022, doi: 10.1109/COMST.2022.3199544.

[56] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, “Capacitated cloudlet placements in Wireless Metropolitan Area Networks,” *Proc. - Conf. Local Comput. Networks, LCN*, vol. 26-29-Octo, pp. 570–578, 2015, doi: 10.1109/LCN.2015.7366372.

[57] G. Premsankar, B. Ghaddar, M. Di Francesco, and R. Verago, “Efficient placement of edge computing devices for vehicular applications in smart cities,” *IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World, NOMS 2018*, pp. 1–9, 2018, doi: 10.1109/NOMS.2018.8406256.

[58] A. Mazloomi, H. Sami, J. Bentahar, H. Otrok, and A. Mourad, “Reinforcement Learning Framework for Server Placement and Workload Allocation in Multi-Access Edge Computing,” *IEEE Internet Things J.*, p. 1, 2022, doi: 10.1109/JIOT.2022.3205051.

[59] S. Dutta, “Optimization in Chemical Engineering,” *Optim. Chem. Eng.*, no. September, 2016, doi: 10.1017/cbo9781316134504.

[60] N. P. Theodorakatos, “Optimal Phasor Measurement Unit Placement for Numerical Observability Using Branch-and-Bound and a Binary-Coded Genetic Algorithm,” *Electr. Power Components Syst.*, vol. 47, no. 4–5, pp. 357–371, 2019, doi: 10.1080/15325008.2019.1605635.

[61] N. P. Theodorakatos, M. Lytras, and R. Babu, “A Generalized Pattern Search Algorithm Methodology for solving an Under-Determined System of Equality Constraints to achieve Power System Observability using Synchrophasors,” *J. Phys. Conf. Ser.*, vol. 2090, no. 1, 2021, doi: 10.1088/1742-6596/2090/1/012125.

[62] X. Li, C. Magnant, and Z. Qin, “Computational Complexity,” *SpringerBriefs Math.*, pp. 137–138, 2018, doi: 10.1007/978-3-319-89617-5_12.

[63] F. Rossi, V. Cardellini, F. Lo Presti, and M. Nardelli, “Geo-distributed efficient deployment of containers with Kubernetes,” *Comput. Commun.*, vol. 159, pp. 161–174, 2020, doi: 10.1016/j.comcom.2020.04.061.

[64] A. Santoyo-González and C. Cervelló-Pastor, “Latency-aware cost optimization of the service infrastructure placement in 5G networks,” *J. Netw. Comput. Appl.*, vol. 114, pp. 29–37, 2018, doi: https://doi.org/10.1016/j.jnca.2018.04.007.

[65] J. Xiao, H. Wen, B. Wu, X. Jiang, P. Ho, and L. Zhang, “Joint Design on DCN Placement and Survivable Cloud Service Provision over All-Optical Mesh Networks,” vol. 62, no. 1,

pp. 235–245, 2014.

[66] Q. Fan and N. Ansari, “On cost aware cloudlet placement for mobile edge computing,” *IEEE/CAA J. Autom. Sin.*, vol. 6, no. 4, pp. 926–937, 2019, doi: 10.1109/JAS.2019.1911564.

[67] A. Van Breedam, “Improvement heuristics for the Vehicle Routing Problem based on Simulated Annealing,” vol. 2217, no. 1979, 2020.

[68] P. Taylor, A. Gogna, and A. Tayal, “Journal of Experimental & Theoretical Metaheuristics : review and application,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 4. Taylor & Francis, pp. 503–526, doi: 10.1080/0952813X.2013.782347.

[69] H. Deghbouch and F. Debbat, “A hybrid bees algorithm with grasshopper optimization algorithm for optimal deployment of wireless sensor networks,” *Intel. Artif.*, vol. 24, no. 67, pp. 18–35, 2021, doi: 10.4114/intartif.vol24iss67pp18-35.

[70] M. Bouet and V. Conan, “Mobile Edge Computing Resources Optimization: A Geo-Clustering Approach,” *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 2, pp. 787–796, 2018, doi: 10.1109/TNSM.2018.2816263.

[71] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, “Exploring Placement of Heterogeneous Edge Servers for Response Time Minimization in Mobile Edge-Cloud Computing,” *IEEE Trans. Ind. Informatics*, vol. 17, no. 1, pp. 494–503, 2021, doi: 10.1109/TII.2020.2975897.

[72] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, “Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing,” *Concurr. Comput. Pract. Exp.*, vol. 29, no. 16, pp. 1–9, 2017, doi: 10.1002/cpe.3975.

[73] J. Gedeon *et al.*, “From cell towers to smart street lamps: Placing cloudlets on existing urban infrastructures,” *Proc. - 2018 3rd ACM/IEEE Symp. Edge Comput. SEC 2018*, pp. 187–202, 2018, doi: 10.1109/SEC.2018.00021.

[74] S. Mondal, G. Das, and E. Wong, “CCOMPASSION: A Hybrid Cloudlet Placement Framework over Passive Optical Access Networks,” *Proc. - IEEE INFOCOM*, vol. 2018-April, pp. 216–224, 2018, doi: 10.1109/INFOCOM.2018.8485846.

[75] S. Mondal, G. Das, and E. Wong, “An Analytical Cost-Optimal Cloudlet Placement Framework over Fiber-Wireless Networks with Quasi-Convex Latency Constraint,” *ELECTRONICS*, vol. 8, no. 4, 2019, doi: 10.3390/electronics8040404.

[76] X. J. Guan, X. L. Wan, T. J. Wang, and Y. F. Li, “A Long-Term Cost-Oriented Cloudlet Planning Method in Wireless Metropolitan Area Networks,” *ELECTRONICS*, vol. 8, no. 11, 2019, doi: 10.3390/electronics8111213.

[77] M. Jia, J. Cao, and W. Liang, “Optimal Cloudlet Placement and User to Cloudlet Allocation in Wireless Metropolitan Area Networks,” *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, 2015, doi: 10.1109/tcc.2015.2449834.

[78] A. Ceselli, M. Premoli, and S. Secci, “Cloudlet network design optimization,” *Proc. 2015 14th IFIP Netw. Conf. IFIP Netw. 2015*, 2015, doi: 10.1109/IFIPNetworking.2015.7145315.

[79] H. Xiang *et al.*, "An adaptive cloudlet placement method for mobile applications over GPS big data," *2016 IEEE Glob. Commun. Conf. GLOBECOM 2016 - Proc.*, pp. 0–5, 2016, doi: 10.1109/GLOCOM.2016.7841576.

[80] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal Placement of Cloudlets for Access Delay Minimization in SDN-Based Internet of Things Networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1334–1344, 2018, doi: 10.1109/JIOT.2018.2811808.

[81] P. Kochovski, R. Sakellariou, M. Bajec, P. Drobintsev, and V. Stankovski, "An architecture and stochastic method for database container placement in the edge-fog-cloud continuum," in *Proceedings - 2019 IEEE 33rd International Parallel and Distributed Processing Symposium, IPDPS 2019*, 2019, pp. 396–405, doi: 10.1109/IPDPS.2019.00050.

[82] L. Ma, J. Wu, and L. Chen, "DOTA: Delay bounded optimal cloudlet deployment and user association in WMANs," *Proc. - 2017 17th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. CCGRID 2017*, pp. 196–203, 2017, doi: 10.1109/CCGRID.2017.34.

[83] H. Sinky, B. Khalfi, B. Hamdaoui, and A. Rayes, "Adaptive edge-centric cloud content placement for responsive smart cities," *IEEE Netw.*, vol. 33, no. 3, pp. 177–183, 2019, doi: 10.1109/MNET.2019.1800137.

[84] M. Laha, S. Kamble, and R. Datta, "Edge nodes placement in 5G enabled urban vehicular networks: A centrality-based approach," 2020, doi: 10.1109/NCC48643.2020.9056059.

[85] R. A. Ghalehtaki, S. Kianpisheh, and R. Glitho, "A Bee Colony-based Algorithm for Micro-cache Placement Close to End Users in Fog-based Content Delivery Networks," *2019 16th IEEE Annu. Consum. Commun. Netw. Conf. CCNC 2019*, pp. 1–4, 2019, doi: 10.1109/CCNC.2019.8651773.

[86] Q. Fan and N. Ansari, "Cost Aware cloudlet Placement for big data processing at the edge," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6, doi: 10.1109/ICC.2017.7996722.

[87] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C. H. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, 2019, doi: 10.1016/j.jpdc.2018.06.008.

[88] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient Algorithms for Capacitated Cloudlet Placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, 2016, doi: 10.1109/TPDS.2015.2510638.

[89] J. Meng, W. Shi, H. Tan, and X. Li, "Cloudlet Placement and Minimum-Delay Routing in Cloudlet Computing," *Proc. - 2017 3rd Int. Conf. Big Data Comput. Commun. BigCom 2017*, pp. 297–304, 2017, doi: 10.1109/BIGCOM.2017.58.

[90] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: a comprehensive survey," *Artif. Intell. Rev.*, vol. 52, no. 4, pp. 2191–2233, 2019, doi: 10.1007/s10462-017-9605-z.

[91] J. H. Holland, "Genetic algorithms," *Sci. Am.*, vol. 267, no. 1, pp. 66–72, 1992, doi: 10.1038/scientificamerican0792-66.

[92] L. M. G. Dorigo, M, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *Belgium TR/IRIDIA/1996-*, vol. 1, no. 1, p. 53, 1997, [Online]. Available: http://people.idsia.ch/~luca/acs-ec97.pdf.

[93] James Kennedy and Russell Eberhart, "Particle Swarm Optimisation," *Proc. ICNN'95 - Int. Conf. Neural Networks, Perth, WA, Aust.*, vol. 4, pp. 1942–1948, 1995, doi: 10.1109/ICNN.1995.488968.

[94] X. S. Yang, "A new metaheuristic Bat-inspired Algorithm," *Stud. Comput. Intell.*, vol. 284, pp. 65–74, 2010, doi: 10.1007/978-3-642-12538-6_6.

[95] S. S. Rao, *Engineering optimization: Theory and practice, Fifth Edition*. John Wiley & Sons, Inc, 2020.

[96] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms Second Edition*. Luniver Press, United Kingdom, 2010.

[97] N. Kamel and T. El-Omari, "Sea Lion Optimization Algorithm for Solving the Maximum Flow Problem," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 20, no. 8, p. 30, 2020, doi: 10.22937/IJCSNS.2020.20.08.5.

[98] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, *Metaheuristic algorithms: A comprehensive review*. Elsevier Inc., 2018.

[99] J. Doering, R. Kizys, A. A. Juan, À. Fitó, and O. Polat, "Metaheuristics for rich portfolio optimisation and risk management: Current state and future trends," *Oper. Res. Perspect.*, vol. 6, no. August, p. 100121, 2019, doi: 10.1016/j.orp.2019.100121.

[100] S. Annealing, "Simulated annealing: Theory and applications," *Math. Comput. Simul.*, vol. 30, no. 1–2, pp. 7–15, 1988, doi: 10.1016/0378-4754(88)90140-1.

[101] X. S. Yang, "Engineering Optimization: An Introduction with Metaheuristic Applications," *Hoboken: Wiley*, 2010.

[102] A. H. Alsaeedi, A. H. Aljanabi, M. E. Manna, and A. L. Albukhnefis, "A proactive metaheuristic model for optimizing weights of artificial neural network," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 20, no. 2, pp. 976–984, 2020, doi: 10.11591/ijeecs.v20.i2.pp976-984.

[103] R. Sagban, H. A. Marhoon, and R. Alubady, "Hybrid bat-ant colony optimization algorithm for rule-based feature selection in health care," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 6, pp. 6655–6663, 2020, doi: 10.11591/ijece.v10i6.pp6655-6663.

[104] J. E. Smith and A. E. Eiben, *Introduction to evolutionary computing*, vol. 28. Springer Berlin Heidelberg, 2015.

[105] E.-G. Talbi, *METAHEURISTICS FROM DESIGN TO IMPLEMENTATION*, vol. 6, no. May. John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[106] H. T. Rauf, M. Hadi, and A. Rehman, "Bat algorithm with Weibull walk for solving global optimisation and classification problems," *Int. J. Bio-Inspired Comput.*, vol. 15, no. 3, pp. 159–170, 2020, doi: 10.1504/IJBIC.2020.107470.

[107] Y. Ye, X. Zhao, and L. Xiong, “An improved bat algorithm with velocity weight and curve decreasing,” *J. Supercomput.*, vol. 78, no. 10, pp. 12461–12475, 2022, doi: 10.1007/s11227-022-04368-9.

[108] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, “A novel population initialization method for accelerating evolutionary algorithms,” *Comput. Math. with Appl.*, vol. 53, no. 10, pp. 1605–1614, 2007, doi: 10.1016/j.camwa.2006.07.013.

[109] I. Fister, D. Fister, and X. S. Yang, “A hybrid bat algorithm,” *Elektroteh. Vestnik/Electrotechnical Rev.*, vol. 80, no. 1–2, pp. 1–7, 2013.

[110] M. Z. Rodriguez *et al.*, “Clustering algorithms: A comparative approach,” *PLoS One*, vol. 14, no. 1, pp. 1–31, 2019, doi: 10.1371/journal.pone.0210236.

[111] G. Verma, “Chapter-04 D,” *Jaypees Dent. Dict.*, pp. 126–151, 2009, doi: 10.5005/jp/books/10428_4.

[112] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, “An overview of clustering methods,” *Intell. Data Anal.*, vol. 11, no. 6, pp. 583–605, 2007, doi: 10.3233/ida-2007-11602.

[113] T. S. Madhulatha, “An overview of clustering methods,” *IOSR J. Eng.*, vol. 2(4), pp. 719–725, 2012, doi: 10.3233/ida-2007-11602.

[114] S. Patel, S. Sihmar, and A. Jatain, “A study of hierarchical clustering algorithms,” *2015 Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2015*, vol. 3, no. 10, pp. 537–541, 2015.

[115] A. Rajaraman and J. D. Ullman, “Mining of massive datasets,” *Min. Massive Datasets*, vol. 9781107015, pp. 1–315, 2011, doi: 10.1017/CBO9781139058452.

[116] G. Karypis, E. H. Han, and V. Kumar, “Chameleon: Hierarchical clustering using dynamic modeling,” *Computer (Long. Beach. Calif).*, vol. 32, no. 8, pp. 68–75, 1999, doi: 10.1109/2.781637.

[117] S. Guha, R. Rastogi, and K. Shim, “CURE: An efficient clustering algorithm for large databases,” *Inf. Syst.*, vol. 26, no. 1, pp. 35–58, 2001, doi: 10.1016/S0306-4379(01)00008-4.

[118] J. A. S. Almeida, L. M. S. Barbosa, A. A. C. C. Pais, and S. J. Formosinho, “Improving hierarchical cluster analysis: A new method with outlier detection and automatic clustering,” *Chemom. Intell. Lab. Syst.*, vol. 87, no. 2, pp. 208–217, 2007, doi: 10.1016/j.chemolab.2007.01.005.

[119] J. H. Xin Jin, “Partitional Clustering,” *Sammut, C., Webb, G.I. Encycl. Mach. Learn. Springer, Boston, MA.*, pp. 167–173, 2011, doi: 10.1109/ICCCIS51004.2021.9397141.

[120] H. S. Park and C. H. Jun, “A simple and fast algorithm for K-medoids clustering,” *Expert Syst. Appl.*, vol. 36, no. 2 PART 2, pp. 3336–3341, 2009, doi: 10.1016/j.eswa.2008.01.039.

[121] X. X. Martin Ester, Hans-Peter Kriegel, Jörg Sander, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” *Proc. 2nd Int. Conf. Knowl. Discov. Data Min.*, vol. 96, no. 34, pp. 226–231, 1996, doi: 10.11901/1005.3093.2016.318.

[122] D. Deng, “DBSCAN Clustering Algorithm Based on Density,” *Proc. - 2020 7th Int. Forum*

Electr. Eng. Autom. IFEEA 2020, pp. 949–953, 2020, doi: 10.1109/IFEEA51475.2020.00199.

[123] J. Hou, H. Gao, and X. Li, "DSets-DBSCAN : A Parameter-Free Clustering Algorithm," *DSets-DBSCAN A Parameter-Free*, vol. 25, no. 7, pp. 3182–3193, 2016.

[124] D. Barbara and P. Chen, "Fractal Mining - Self Similarity-based Clustering and its Applications Daniel," *Data Min. Knowl. Discov. Handbook, 2nd ed.*, 2010, doi: 10.1007/978-0-387-09823-4.

[125] D. Al-Shammary, I. Khalil, Z. Tari, and A. Y. Zomaya, "Fractal self-similarity measurements based clustering technique for SOAP Web messages," *J. Parallel Distrib. Comput.*, vol. 73, no. 5, pp. 664–676, 2013, doi: 10.1016/j.jpdc.2013.01.005.

[126] D. Barbará and P. Chen, "Using the fractal dimension to cluster datasets," *Proceeding Sixth ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 260–264, 2000, doi: 10.1145/347090.347145.

[127] F. R. Almeida, A. Brayner, J. P. C. Rodrigues, and J. E. Bessa Maia, "Improving multidimensional wireless sensor network lifetime using pearson correlation and fractal clustering," *Sensors (Switzerland)*, vol. 17, no. 6, pp. 1–24, 2017, doi: 10.3390/s17061317.

[128] S. S. Alkafagi, "Build Network Intrusion Detection System based on combination of Fractal Density Peak Clustering and Artificial Neural Network," *J. Al-Qadisiyah Comput. Sci. Math.*, vol. 15, no. 1, pp. 1–14, 2023, doi: https://doi.org/10.29304/jqcm.2023.15.1.1151 1.

[129] Y. Zhang, G. Ji, Z. Dong, S. Wang, and P. Phillips, "'An Investigation into the Performance of Particle Swarm Optimization with Various Chaotic Maps,'" *Math. Probl. Eng.*, vol. 2014, pp. 11–14, 2015, doi: 10.1155/2015/815370.

[130] D. He, C. He, L. G. Jiang, H. W. Zhu, and G. R. Hu, "Chaotic characteristics of a one-dimensional iterative map with infinite collapses," *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.*, vol. 48, no. 7, pp. 900–906, 2001, doi: 10.1109/81.933333.

[131] M. S. Tavazoei and M. Haeri, "Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms," *Appl. Math. Comput.*, vol. 187, no. 2, pp. 1076–1085, 2007, doi: 10.1016/j.amc.2006.09.087.

[132] M. A. Elaziz and S. Mirjalili, "A hyper-heuristic for improving the initial population of whale optimization algorithm," *Knowledge-Based Syst.*, vol. 172, pp. 42–63, 2019, doi: 10.1016/j.knosys.2019.02.010.

[133] X.-S. Yang, "Appendix A: Test Problems in Optimization," *Eng. Optim.*, no. 2010, pp. 261–266, 2010, doi: 10.1002/9780470640425.app1.

[134] R. W. Garden and A. P. Engelbrecht, "Analysis and classification of optimisation benchmark functions and benchmark suites," *Proc. 2014 IEEE Congr. Evol. Comput. CEC 2014*, vol. 1, pp. 1641–1649, 2014, doi: 10.1109/CEC.2014.6900240.

[135] K. Hussain, M. N. M. Salleh, S. Cheng, and R. Naseem, "Common benchmark functions for metaheuristic evaluation: A review," *Int. J. Informatics Vis.*, vol. 1, no. 4–2, pp. 218–223, 2017, doi: 10.30630/joiv.1.4-2.65.

[136] M. Jamil and X. S. Yang, “A literature survey of benchmark functions for global optimisation problems,” *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013, doi: 10.1504/IJMMNO.2013.055204.

[137] I. The MathWorks, “MATLAB version: 9.13.0 (R2022b).” The MathWorks Inc., Natick, Massachusetts, United States, 2022, [Online]. Available: accessed: September 01, 2023. Available: https://www.mathworks.com.

[138] D. Bhatta and L. Mashayekhy, “Generalized cost-aware cloudlet placement for vehicular edge computing systems,” *Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom*, vol. 2019-Decem, pp. 159–166, 2019, doi: 10.1109/CloudCom.2019.00033.

[139] A. Ceselli, M. Premoli, and S. Secci, “Mobile edge cloud network design optimization,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1818–1831, 2017, doi: 10.1109/TNET.2017.2652850.

[140] N. Mohan, A. Zavodovski, P. Zhou, and J. Kangasharju, “Anveshak: Placing edge servers in the wild,” *MECOMM 2018 - Proc. 2018 Work. Mob. Edge Commun. Part SIGCOMM 2018*, no. 1, pp. 7–12, 2018, doi: 10.1145/3229556.3229560.

[141] Y. Shao, Z. Shen, S. Gong, and H. Huang, “Cost-Aware Placement Optimization of Edge Servers for IoT Services in Wireless Metropolitan Area Networks,” *Wirel. Commun. Mob. Comput.*, vol. 2022, 2022, doi: 10.1155/2022/8936576.

[142] ali asghari, H. Azgomi, and Z. Darvishmofarahi, “Multi-Objective Edge Server Placement Using the Whale Optimization Algorithm and Game Theory,” *SSRN Electron. J.*, 2022, doi: 10.2139/ssrn.4185182.

[143] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, “Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach,” *IEEE Trans. Mob. Comput.*, vol. 20, no. 3, pp. 939–951, 2021, doi: 10.1109/TMC.2019.2957804.

Appendix A

Published Papers

(1)

Received: 30 November 2022 | Revised: 2 March 2023 | Accepted: 20 May 2023

DOI: 10.1049/ntw2.12087

IET Networks

The Institution of Engineering and Technology WILEY

REVIEW

A survey on edge and fog nodes' placement methods, techniques, parameters, and constraints

Samraa Adnan Al-Asadi[1] | Safaa O. Al-Mamory[2]

[1]Department of Information Networks / College of Information Technology, University of Babylon, Hillah, Iraq

[2]College of Business Informatics, University of Information Technology and Communications, Baghdad, Iraq

Correspondence
Samraa Adnan Al-Asadi.
Email: samraa.alasadi@uobabylon.edu.iq

Abstract

Within Edge and Fog computing, edge and fog nodes must be optimally located at the network edge to minimise the network's overall latency. This survey addresses all aspects of these nodes' placement problems. Literature on edge and fog nodes' placement is collected from reputable databases (IEEE Xplore digital library, Scopus, ScienceDirect, and Web of Science) using a search query. Manual search using keywords and the snowball method is also used to get as many related papers as possible. According to defined inclusion criteria, retrieved documents are filtered to 64 articles for eight years (2015–2022). Depending on the optimisation method used, literature is classified into six categories. The first relies on Integer programming, accounting for 20.3% (13/64). The second category depends on heuristic and metaheuristic methods, accounting for 20.3% (13/64). The third category depends on hybrid methods between the two aforementioned categories accounting for 18.7% (12/64). Forth category depends on clustering methods, accounting for 11% (7/64). The fifth category depends on reinforcement learning, accounting for 6.3% (4/64). And the final category depends on the hybrid methods between two or more methods mentioned above, accounting for 23.4% (15/64). Papers have been analysed to get information like the optimisation problem, the method used for solving it, considered parameters, objectives, constraints, implementation tools, and evaluation methods.

KEYWORDS

edge computing, edge/ fog nodes placement, fog computing, optimisation, resource allocation

1 | INTRODUCTION

The data generated by mobile and Internet of Things (IoT) devices has grown significantly. Their limited energy and computational resources characterise IoT devices like sensors, smartphones, and wearable gadgets. These limitations are addressed by offloading processing and storage from resource-constrained devices to the cloud. Because of the scalable and on-demand nature of the cloud, it is considered an ideal solution for computation offloading [1]. However, the centralised, far away cloud server location is causing a too-long latency to respond to the requests from a vast number of IoT sensors distributed in a large working area, especially for applications with latency-critical needs, such as e-health and the Internet of Vehicles (IoV). This type of central data processing also causes colossal network traffic as the number of services and objects increases [2–5].

Therefore, the first question that needs to determine is: *"How to eliminate the drawbacks of the cloud and provide that specific need?"*

Mobile Edge Computing (MEC) and Mobile Fog Computing (MFC) platforms are both network architectures that complement the central cloud by bringing cloud functionalities to the network edge using edge nodes. Edge nodes are small-scale edge servers/cloudlets geographically distributed near end-user devices for more efficient service access [6, 7].

The number of smart devices at the network edge is increasing rapidly, and many IoT applications have specific

Published Paper

(2)

Inteligencia Artificial 26(72), 102-123
doi: 10.4114/intartif.vol26iss72pp102-123

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

An Improved BAT Algorithm Using Density-Based Clustering

Samraa Adnan Al-Asadi[1], Safaa O. Al-Mamory[2]

[1] Department of Information Networks / College of Information Technology, University of Babylon
Email: samraa.alasadi@uobabylon.edu.iq

[2] College of Business Informatics, University of Information Technology and Communications
Email: salmamory@uoitc.edu.iq

Abstract. BAT algorithm is a nature-inspired metaheuristic algorithm that depends on the principle of the echolocation behavior of bats. However, due to poor exploration, the algorithm suffers from being stuck in the local optima. An improved BAT algorithm based on the density-based clustering technique is proposed to enhance the algorithm's performance.

In this paper, the initial population is improved by generating two populations, randomly and depending on the clusters' center information, and by getting the fittest individuals from these two populations, the initial improved one is generated. The random walk function is improved using chaotic maps instead of the fixed-size movement, so the local search is improved, as well as the global search abilities, by diversifying the solutions. Another improvement is dealing with stagnation by partitioning the search space into two parts depending on the generated clusters' information to obtain the newly generated solution, comparing their quality with the previously generated solution, and choosing the best.

The performance of the proposed improved BAT algorithm is evaluated by comparing it with the original BAT algorithm over ten benchmark optimization test functions. Depending on the results, the improved BAT outperforms the original BAT by obtaining the optimal global solutions for most of the benchmark test functions.

Keywords: Metaheuristic Algorithms, BAT Algorithm, Density-based Clustering, Chaotic Strategies

1 Introduction

Due to technological development, many complicated optimization problems appear in many different fields, like engineering, manufacturing, information technology, and economic management. [1]. These problems are solved using optimization algorithms. Optimization algorithms are categorized as deterministic algorithms and stochastic algorithms. Deterministic algorithms, like the simplex method in linear programming, usually need gradient information. The problems with one global optimum are effectively solved by deterministic algorithms, but for problems with many local optima or problems with unavailable gradient information, they may be invalid. On the other hand, stochastic algorithms require only the information of the objective function. Heuristic and metaheuristic are generally the two types of stochastic algorithms [2]. Heuristic approaches are used to find the optimum or at least near-optimum solutions. These approaches do not guarantee to reach that global optimum solution. A high-level heuristic is called a metaheuristic [3].

Metaheuristic algorithms, for example, Simulated Annealing (SA) [4] and Particle Swarm Optimization (PSO) [5], are very powerful in solving hard optimization problems, so they have been applied in almost all significant areas of engineering and science and industrial applications [6]. For example, the optimum weights of the artificial neural network are found by using PSO instead of using the Backpropagation algorithm due to its better classification accuracy and faster processing time when compared with the Backpropagation algorithm [7], and BAT algorithm is used for attributes and features selection [8]. Generally, some prominent applications of metaheuristic algorithms are [9][10]: ***Combinatorial Optimization,*** where some metaheuristic algorithms like

ISSN: 1137-3601 (print), 1988-3064 (on-line)

الخلاصه

نظرا لوجود اعداد كبيره من الاجهزة التي تطلب العديد من الخدمات المختلفه, فإن الخوادم السحابيه تتطلب وقتا طويلا لأرسال ومعالجة ومن ثم استلام الإجابه لهذه الطلبات. لحل مشكلة التأخير بالوقت, تم اعتماد الحوسبه الضبابيه لتقريب موارد الخوادم السحابيه من حافة الشبكه ومن ثم المستخدمين.

الحوسبه الضبابيه هي الحل الأمثل للتخلص من هذا الوقت الطويل, لكن تحديد العدد والموقع الأمثل للخوادم الضبابيه يعد مشكله تحتاج الى وقت طويل لحلها وبالتالي فإن الطريقة الأمثل لتحديد عدد وموقع هذه الخوادم الضبابيه هو بإستخدام الطرق التقريبيه للحل الامثل.

تم تمثيل المشكله رياضيا بانها مشكله متعددة الاهداف, ذلك لان تحديد العدد الامثل والموقع الامثل له عده اهداف اهمها تقليل وقت التأخير, والاستغلال الامثل للخوادم وتوزيع الطلبات بصورة مثلى بين الخوادم. ثم تم حل المشكله باستخدام خوارزمية الخفاش المستوحاه من الطبيعه ومن سلوك الخفاش وتعامله مع الاجسام وتحديد قربها وموقعها عن طريق الصدى. تم تطوير خوارزمية الخفاش بالاستعانه بطرق التجميع. هذا التطوير جاء بسبب ضعف الخوازميه بأكتشاف مساحة البحث بصورة كفوءه كما انها تعاني من الركود وبالتالي فأن الحل الامثل قد لا نحصل عليه.

تعتمد جودة الحلول الناتجه على المجتمع الأولي الذي تبتدأ الخوارزميه عملها منه, لذلك فأن تطوير المجتمع الاولي له دور كبير.. تم الاعتماد على المجموعات المتشكله من خوارميات التجميع بتحديد مراكزها وبالتالي تحديد افضل الافراد الذين يكونون متجمعين حول المراكز.. ولكي نضمن الوصول لكل مساحة البحث فأن المجتمع الاولي يعتمد ايضا على العشوائيه في تحديد الافراد. حركة الفرد الواحد ضمن المجتمع ستكون فوضويه بدلا من الخطوات الثابتة الحجم, حيث ان النمط الفوضوي سيعطي توازنا للخوارزميه من ناحيه الاستكشاف والاستثمار. وفي حالة وصول الخوارزميه لحالة ركود, فان عمليه توليد حلول موقته بالاعتماد على المجموعات وتقسيمها لقسمين ومن ثم مقارنه الحل الناتج مع الموجود اصلا واختيار الافضل سيكون له دورا في التخلص من هذه الحاله.

تم مقارنه اداء الخوارزميه المقترحه مع الطرق الاصليه (التوزيع العشوائي, والتوزيع حسب الجهد الاكبر) وكانت النتائج افضل بكثير للخوارزميه المطوره وذلك بالاعتماد على مجموعه من المقاييس المعتمده للاختبار, حيث ان الخوارزميه المطوره تحدد عدد اقل من الخوادم بإستغلال امثل وبتوزيع جهد ومهام بصوره افضل من الطرق التقليديه.

تم استخدام الخوارزميه المطوره في تحديد العدد الأمثل والموقع الأمثل للخوادم السحابيه بعد أن اثبتت كفائتها في الحل حيث تم مقارنه ادائها مع مجموعه من الطرق التقليديه المعتمده للاختبار مثل التوزيع العشوائي للخوادم والتوزيع ضمن الاجهزه الاكثر تعرضا للطلبات. اختبار الاداء اعتمد على بيانات حقيقيه من شبكة اتصالات شانغهاي تيليكوم. الخوارزميه المطوره تخصص عدد أقل من الخوادم بإستغلال أمثل وتوزيع جهد امثل بين الخوادم مقارنة بالطرق التقليديه.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنلوجيا المعلومات
قسم شبكات المعلومات

تحسين خوارزمية الخفاش لحل مشكلة الموقع فى الحوسبة الحافه

أطروحة

مقدمة إلى مجلس كلية تكنولوجيا المعلومات في جامعة بابل والتي هي جزء من متطلبات الحصول على درجة الدكتوراه فلسفه في تكنولوجيا المعلومات \ شبكات المعلومات

مقدمة من قبل
سمراء عدنان عبدمسلم جاسم

باشراف
أ. د. صفاء عبيس مهدي مظلوم

هـ 1445 م 2023