**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

Moisés Manuel Borba Roriz Ramires

# Two-level Fingerprint-based Indoor Positioning using advanced Machine Learning

November 2022

**Universidade do Minho**
Escola de Engenharia
Departamento de Informática

Moisés Manuel Borba Roriz Ramires

# Two-level Fingerprint-based Indoor Positioning using advanced Machine Learning

Master dissertation
Master's in Informatics Engineering

Dissertation supervised by
**Adriano Moreira and Joaquín Torres-Sospedra**

November 2022

## ACKNOWLEDGEMENTS

I am grateful to both my supervisors for the opportunity given, and for the continuous support and guidance in this work. Their motivation and encouragement made it possible not only to finish the work presented but also to evolve as a student and person. I always had the support of my supervisors and their views on my ideas. To my supervisors, Adriano Moreira and Joaquín Torres-Sospedra, thank you for everything. I also want to express my gratitude towards my family, who without them I would never be where I am now. Their belief in me and support allowed me to get here, and to continue going further.

## STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

## ABSTRACT

These days, positioning systems are Global Navigation Satellite System (GNSS) based – such as Global Positioning System (GPS) or the European Galileo – have been deployed worldwide, due to their efficiency, reliability, and need. Today using GPS for navigation or localization is quite common, this technology shaped our world and it is now part of our life.

However, these satellite-based positioning systems fail to provide good results inside infrastructures. If someone is inside a building, walls and other objects inside will attenuate the signals, making them unreliable for obtaining a position.

For example, an Indoor Positioning System (IPS) offering localization services inside a hospital could bring a lot of benefits, namely patient orientation, locating doctors and nurses for emergency responses, or immediately locating critical instrumentation, among others. In the case of a warehouse, it can be used for better logistics, optimization of resources and autonomous vehicle driving. Other related contexts can be found in airports, museums and shopping malls, where IPSs can be used to support indoor navigation.

There is a large number of solutions created for this challenge, using technologies such as Bluetooth Low Energy (BLE), Wi-Fi, Ultra-Wideband (UWB), Light Detection and Ranging (LIDAR), and Infrared, among many others. These are usually associated with techniques such as proximity, trilateration, triangulation, and fingerprinting.

This work will focus on using Wi-Fi technology using the fingerprinting technique, i.e., Wi-Fi Fingerprinting for large indoor enviroments.



KEYWORDS Fingerprinting, Clustering, RSSI Averaged Positioning Error (APE)

## RESUMO

Atualmente, os sistemas de posicionamento baseados em GNSS – como o GPS ou o Galileo europeu – foram implementados em todo o mundo devido à sua eficiência, confiabilidade e necessidade. Hoje usar o GPS para navegação ou localização é bastante comum, esta tecnologia moldou nosso mundo e agora faz parte de nossa vida.

No entanto, esses sistemas de posicionamento baseados em satélite não fornecem bons resultados dentro das infraestruturas. Se alguém estiver dentro de um prédio, paredes e outros objetos no interior atenuarão os sinais, tornando-os pouco confiáveis para estimar uma posição.

Por exemplo, um IPS que ofereça serviços de localização dentro de um hospital pode trazer muitos benefícios, como orientação ao paciente, localização de médicos e enfermeiros para respostas de emergência, localização imediata de instrumentação crítica, entre outros. No caso de um armazém, pode ser utilizado para uma melhor logística, otimização de recursos e condução autónoma de veículos. Outros contextos relacionados podem ser encontrados em aeroportos, museus e shopping centers, onde o IPSs pode ser usado para dar suporte à navegação interna.

Existe um grande número de soluções criadas para este desafio, utilizando tecnologias como BLE, Wi-Fi, UWB, LIDAR, Infrared, entre muitas outras. Estas geralmente estão associados a técnicas como proximidade, trilateração, triangulação e Fingerprinting.

Este trabalho foca-se no uso da tecnologia Wi-Fi e o método Fingerprinting, ou seja, Fingerprinting Wi-Fi para grandes ambientes indoor.



PALAVRAS-CHAVE Fingerprinting, Clustering, RSSI APE

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ALGORITHMS

# ACRONYMS

**$k$-NN** $k$-Nearest Neighbour. xi, 2, 3, 14, 18, 19, 23, 24, 27, 44, 45, 46, 47, 48
**Wi-Fi** IEEE 802.11 Wireless LAN. 10, 14, 54

**AET** Averaged Execution Time. 46
**AI** Artificial Intelligence. 7
**AoA** Angle of Arrival. viii, 6, 8, 9, 11, 12
**AP** Access Point. viii, 3, 9, 10, 11, 14, 18, 19, 22, 24, 25, 26, 27, 29, 30, 31, 34, 46, 52, 54
**APC** Affinity Propagation Clustering. 23, 46
**APE** Averaged Positioning Error. iii, iv, 19, 33, 35, 36, 37, 38, 39, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54

**BLE** Bluetooth Low Energy. iii, iv, 1, 5, 6, 8, 10
**BSC** Base Station Clustering. v, viii, ix, xi, 3, 4, 20, 29, 30, 31, 32, 35, 38, 39, 40, 41, 44, 45, 48, 49, 50, 51, 52, 53, 54, 63, 66, 68, 71, 73, 76, 78, 81, 83

**FTM** Fine Timing Measurement. 9

**GNSS** Global Navigation Satellite System. iii, iv, 1
**GPS** Global Positioning System. iii, iv, 1, 7

**IMU** Inertial Measurement Units. 7, 8, 10
**INS** Inertial Navigation Systems. 7
**IoT** Internet of Things. 6
**IP** Indoor Positioning. 6, 9
**IPS** Indoor Positioning System. iii, iv, 1, 3, 7, 8, 11, 14
**IPS** Indoor Positioning System. 9, 24
**IR** Infrared. 8, 10

**LIDAR** Light Detection and Ranging. iii, iv, 1, 7, 8, 10
**LOS** Line Of Sight. 7, 10

**MAD** Median Absolute Deviation. 7, 17

**NLOS** No Line Of Sight. 7, 11, 12

**PIR** Passive Infrared. 8

**RF** Radio Frequency. 4, 9, 10

**RFID** Radio Frequency Identification. 13
**RSS** Received Signal Strength. 1, 11, 12, 13, 14, 18, 19
**RSSI** Received Signal Strength Indicator. 6, 14, 18, 24, 25, 29, 30, 31, 52
**RTT** Round Trip Time. 9

**SAS** Strongest AP Set. v, vi, viii, ix, xi, 3, 4, 20, 22, 23, 24, 25, 26, 27, 28, 29, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 63, 65, 68, 70, 73, 75, 78, 80, 83
**SLAM** Simultaneous Localization and Mapping. 7, 8
**SVC** Support Vector Clustering. 19
**SWOT** Strengths, Weaknesses, Opportunities, and Threats. 7

**TDoA** Time Difference of arrival. 6, 11, 12
**ToA** Time of arrival. viii, 6, 7, 10, 11, 12
**ToF** Time of Flight. 8, 10
**TWTF** two-way time of flight. 7

**UAV** Unmanned Air vehicle. 8
**UGV** Unmanned Ground Vehicle. 7
**UWB** Ultra-Wideband. iii, iv, 1, 6, 7, 10, 11, 18

**WKNN** Weighted K Nearest Neighbors. 18, 19

Part I

# INTRODUCTORY MATERIAL

# 1

# INTRODUCTION

## 1.1 INTRODUCTION TO INDOOR LOCALIZATION

Positioning systems based on GNSS – such as GPS or the European Galileo – have been deployed worldwide, due to their efficiency, reliability, and need. Today using GPS for navigation or localization is quite common, this technology shaped our world and it is now part of our life.

However, these satellite-based positioning systems fail to provide good results inside infrastructures. If someone is inside a building, walls and other objects inside will attenuate the signals, making them unreliable for obtaining a position.

An IPS offering localization services inside a hospital could bring a lot of benefits, namely patient orientation, locating doctors and nurses for emergency responses, or immediately locating critical instrumentation, among others. In the case of a warehouse, it can be used for better logistics, optimization of resources and autonomous vehicle driving. Other related contexts can be found in airports, museums and shopping malls, where IPSs can be used to support indoor navigation.

There is a large number of solutions created for this challenge, using technologies such as BLE, Wi-Fi, UWB, LIDAR, and Infrared, among many others. These are usually associated with techniques such as proximity, trilateration, triangulation, and fingerprinting.

This work will focus on using Wi-Fi technology using the fingerprinting technique, i.e., Wi-Fi Fingerprinting.

## 1.2 WI-FI FINGERPRINTING

The reason behind using Wi-Fi fingerprinting is due to the ubiquity of Wi-Fi networks. i.e., Wi-Fi is a commonly available technology in buildings for communication purposes, therefore a Wi-Fi-based indoor positioning system can be developed using the already in-place infrastructure without additional costs.

Fingerprinting is a technique that consists of measuring the environment to create a reference dataset in the offline phase, also known as radio map. In the online phase, it applies an estimator to estimate the current position based on the data included in the reference dataset.

In this case, a fingerprint is a set of Received Signal Strength (RSS) measurements detected from Access Points and the coordinates where these were taken. For the offline phase, one or many fingerprints are collected

in reference locations over the entire operational area. The resulting dataset is commonly referred to as a radio map, being the set of all fingerprints collected and corresponding positions.

In the operational phase, a new fingerprint is collected at an unknown location. Then, the pattern matching algorithm $k$-NN is usually applied to obtain the most similar fingerprints and compute their centroids to provide a position estimation.

Traditional fingerprinting is working well, providing positioning errors of a few meters. However, there are some challenges with this approach. If a radio map has a lot of fingerprints or it covers a very large area, there can be unnecessary processing of data in the operational phase. In some cases, the time it takes to estimate the position of a person/device might be prohibitive.

In the literature, there have been different approaches to tackle this problem by restricting the matching step to similar related fingerprints or dividing the radio map into disjoint smaller radio maps by means of unsupervised techniques, i.e., clustering.

## 1.3 CLUSTERING

Clustering can be described as dividing something into multiple sets, where each element in a set has more in common with elements from the same set than with elements belonging to the other sets. The idea is to divide data into smaller sets of data, allowing for better computational costs and improving, in our case, positioning error. There are multiple clustering algorithms developed and tested, providing efficient results for different problems, such as $K$-Means, DBSCAN, Affinity Propagation, and more, shown in Figure 1:

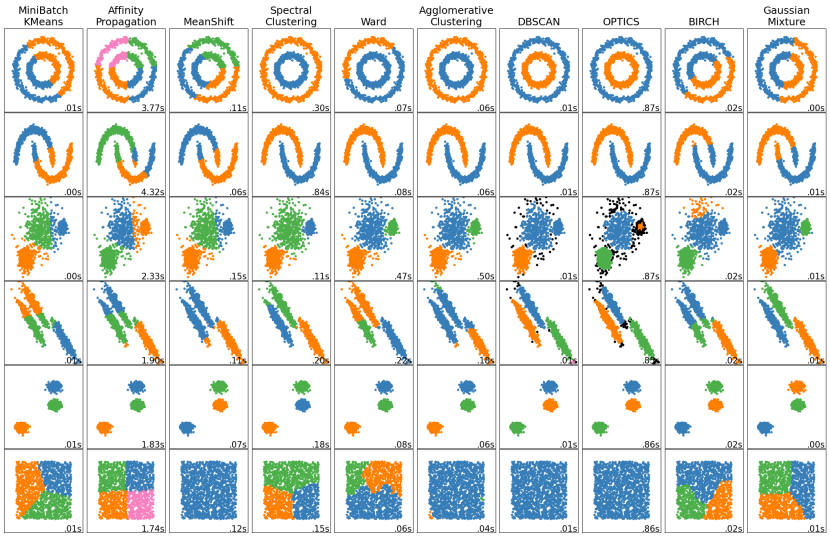



Figure 1: Clusters algorithms, Source https://scikit-learn.org/stable/modules/clustering.html

However, these clustering models should be considered generic algorithms that do not take into account any specific domain knowledge, in this case the characteristics of radio signals propagation.

## 1.4 RESEARCH OBJECTIVES

The purpose of this work is to create a reliable and efficient fingerprint-based IPS that can scale to large areas, and the use of multiple simultaneous devices. Being able to scale means being able to be deployed without concerns about large radio maps, and therefore making indoor positioning more affordable in terms of computation.

The main problem of fingerprinting comes with the use of very large radio maps, which usually translates into a high computational cost for the $k$-NN model in the operational phase. Our hypothesis is that applying clustering to our radio maps, this cost can be significantly reduced. However, our implementation must abide by two metrics:

- Reduce computational time
- Have a low penalty in the positioning error or even a gain

## 1.5 RESEARCH APPROACH

The core of this work is to explore clustering – i.e., dividing a radio map into sub-radio maps – by developing and experimenting with different clustering and cluster identification solutions. To evaluate the solutions, a set of diverse datasets has been gathered, and they will be used to experiment with the solutions. This ensures that our proposed model is not attached to one single location and generalizes to different scenarios.As illustrated in Figure 1, different clustering methods perform differently in different datasets.

## 1.6 SAS & BSC

Since the known clustering algorithms can be categorized as generic algorithms, this means they were not designed for a specific context, which means they do not consider aspects of the indoor positioning challenge. This work proposes two new clustering algorithms, developed in particular for the indoor positioning context: the Strongest AP Set (SAS) and the Base Station Clustering (BSC), which will make use of knowledge on indoor positioning and signal propagation in order to provide a better clustering outcome.

In particular, the proposed SAS clustering method exploits the concept that the strongest AP is the closest AP and, somehow, indicates the region where the user is located at. The idea behind SAS is to take the set of the $N$ strongest APs for each fingerprint, and then cluster them according to the set. BSC offers a more expensive approach in terms of memory to alleviate the computational cost of the operational phase, also making use of the strongest AP concept.

## 1.7 WORK CONTRIBUTION

This works contributes by creating two 2 new clustering algorithms with context awareness for indoo positioning. One of them, SAS, has been presented at the VTC Fall 2022 conference that took place in London in September 2022, and is described in the corresponding paper Ramires et al. (2022).

## 1.8 DISSERTATION STRUCTURE

The rest of this document is organized as follows:

**Chapter 2:** (State of the Art) introduces a series of previous works on Indoor Positioning, which are analysed, from an early implementation of Radio Frequency (RF) systems for Indoor Positioning, all the way to clustering solutions.

**Chapter 3:** (SAS) describes the new SAS algorithm.

**Chapter 4:** (BSC) describes the new BSC algorithm.

**Chapter 5:** (Validation Methodology) shows the methodology used to test the implemented algorithms, along with some dataset description.

**Chapter 6:** (Results) The current results of the work.

**Chapter 7:** (Conclusions and Future Work) has the final thoughts on the work conducted, and some possible future avenues to be explored.

# 2

# STATE OF ART

## 2.1 INTRODUCTION

This chapter introduces a review of the state of art in Indoor Positioning by focusing on three dimensions: positioning technologies, the kind of measurements and the positioning techniques. In addition, we also review the works related to Wi-Fi fingerprinting and Clustering.

## 2.2 POSITIONING TECHNOLOGIES

Indoor positioning has been a topic of research for a long time. In Bahl and Padmanabhan (2000) a solution based on RF was proposed in the year 2000, presenting results for an early implementation of indoor localization using RF, achieving a median accuracy of 2–3 meters, inside a $22.5\,\mathrm{m} \times 43.5\,\mathrm{m}$ floor. This is one of the first results presented with some good accuracy, showing us the potential of Indoor Positioning based on Wi-Fi fingerprinting.

As Indoor positioning became more and more a topic of research, more solutions, both in terms of technologies and techniques emerged. Surveys like Luca Mainetti (2014), Brena et al. (2017), Al-Ammarz et al. (2014), Alkhawaja et al. (2019) and Khan et al. (2021) describe several technologies, techniques and algorithms, along with their disadvantages and benefits.

### 2.2.1 *BLE*

BLE is highly researched due to low power consumption and good accuracy. BLE is studied in detail in Gomez et al. (2012), describing it as "BLE has been designed as a low-power solution for control and monitoring applications", going into deep detail about this solution.

BLE radio is designed for very low power operation. It transmits data over 40 channels in a wireless personal area network technology, using three of them for broadcasting messages (exploited by the Eddystone [1] and iBeacon [2] protocols for positioning purposes). BLE is now also widely used as a device positioning technology to address the increasing demand for high accuracy indoor location services. Initially supporting simple presence

1 https://www.mokoblue.com/all-about-eddystone-beacon/
2 https://pt.wikipedia.org/wiki/IBeacon

and proximity capabilities, BLE now supports Bluetooth Direction Finding and soon, high-accuracy distance measurement [3].

Several works, including Jiw et al. (2015), Kalbandhe and Shailaja.C.Patil (2016), Zili et al. (2014), Spachos and Plataniotis (2020) approach this technology in detail in the context of Indoor Positioning (IP), presenting some possible system architectures.

Jiw et al. (2015) starts with a brief analysis of BLE characteristics in general, starting by comparing it to classic Bluetooth. It also states that it has up to seven times smaller range and that it does not support voice transmission because of its low transmission capacity. After, BLE is analysed from the IP view using BLE beacons, in more detail going about signal attenuation in relation to distance and trying to obtain a path loss model for BLE in IP, using both a random Topology for spreading BLE beacons and a grid Topology.

Kalbandhe and Shailaja.C.Patil (2016) proposes a system divided into 3 components: BLE Tags, Mobile Applications and the system communication interface. The BLE Tags continuously transmit packets of data in regular intervals, and when a mobile Application enters the Tag range, the Indoor Positioning technique extracts the Received Signal Strength Indicator (RSSI) detected. The article also addresses some challenges of this technology, such as the nature of the signal, its attenuation and noise. Finally, the authors achieved an accuracy of around $4\,\mathrm{m}$, stating that the use of BLE Tags provides better results than Wi-Fi and Trilateration localization.

Zili et al. (2014) presents some improvements to enhance positioning performance, including Gaussian filter and Least Squares based piecewise fitting for offline training; weighted sliding windows and weighted distance filter for real-time RSSI processing; and Taylor series expansion based cooperative localization. The authors concluded that their positioning methods provide a more robust solution to Indoor localization than previous solutions.

Spachos and Plataniotis (2020) actually provides us with a real-world scenario, using BLE Beacons for IP inside an Internet of Things (IoT)-based Smart Museum. It keeps track of visitors' positions, and when they approach an exhibit they receive a notification about the exhibit. The authors concluded that the range/accuracy of the beacons was sufficient for their application.

### 2.2.2 *Ultra Wide-Band*

UWB is one of the most promising technologies for indoor positioning, being able to achieve errors in the range of centimetres ($\mathrm{cm}$). UWB is a short-range, wireless communication protocol that operates through radio waves, it operates at very high frequencies and can be used to capture highly accurate spatial and directional data. Using larger channel bandwidth ($500\,\mathrm{MHz}$) with short pulses (two nanoseconds each), UWB is able to achieve greater accuracy compared to other alternatives such as Wi-Fi fingerprinting.

Alarifi et al. (2016) primary focus is UWB, but it describes a lot of other IP technologies and compares them to each other. The authors highlight as an advantage the fact that UWB has high accuracy, yet it uses high-cost equipment. It also discusses and describes some algorithms for this technology, ToA, AoA, Time Difference of

3 https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

arrival (TDoA), Fingerprinting, among others. This study ends with a Strengths, Weaknesses, Opportunities, and Threats (SWOT) analysis of UWB to identify its strengths and weakness.

Hol et al. (2010) uses ToA measurements to estimate a user position, its main objective is a novel calibration method to determine the clock parameters of the UWB receivers as well as their 3D positions, stating that the method is capable of accurately calibrating a UWB setup in minutes.

Dabove et al. (2018) make use of two-way time of flight (TWTF). Their system was deployed using a Tag on the user, and 4 UWB beacons on the corners at a height of $2\,\mathrm{m}$ of a room with $7\,\mathrm{m}\times 9\,\mathrm{m}$ dimension achieving an accuracy of $125\,\mathrm{mm}$ indoor, and inside a narrow corridor, $1.8\,\mathrm{m}\times 6.8\,\mathrm{m}$, getting results in the range of $150\,\mathrm{mm}$ to $200\,\mathrm{mm}$. They conclude that their proposed system has a great error-wise result, but present limitations in terms of cost, time and infrastructure equipment.

Cheng et al. (2020) integrates the use of Artificial Intelligence (AI) with the UWB technology for an IPS, using two-layer Kalman filters, Line Of Sight (LOS) and No Line Of Sight (NLOS) classification, and an outlier detection based on Median Absolute Deviation (MAD). The proposed system achieves am averaged positioning error of around $6\,\mathrm{cm}$ for two-dimensional data sets, and $14\,\mathrm{cm}$ for three-dimensional sets.

Angelis et al. (2010) focus on using Inertial Navigation Systems (INS) in conjunction with UWB to provide a scalable indoor navigation system, obtaining an accuracy of $5\,\mathrm{cm}$ inside a controlled environment.

Silvia et al. (2018) creates a system for indoor positioning with UWB , achieving an accuracy of $6\,\mathrm{cm}$.

### 2.2.3 *Lidar*

LIDAR is a remote sensing method that uses light in the form of a pulsed laser to measure ranges. It can provide high accuracy in terms of indoor positioning, and it is an easy-to-implement solution. This is a technology that is usually used for vehicle position and velocity estimation. Many known robot vacuums cleaners already use this technology for indoor navigation[4].

Jiang et al. (2021) uses LIDAR to obtain the coordinates of a moving robot inside a controlled environment, reaching errors in terms of X-axis and Y-axis of $93\,\mathrm{mm}$ and $77\,\mathrm{mm}$ accordingly.

Sánchez et al. (2019) creates a method based on the LIDAR technology for an IPS, using a vehicle moving at $5\,\mathrm{km\,h^{-1}}$ to $40\,\mathrm{km\,h^{-1}}$ and a set of infrastructure markers, achieving an average error of $40\,\mathrm{mm}$ at the lowest speed and an average error of $100\,\mathrm{mm}$ at $40\,\mathrm{km\,h^{-1}}$ .

Li et al. (2014) Proposes a method to integrate the measurements from LIDAR and a MEMS Inertial Measurement Units (IMU) for a Simultaneous Localization and Mapping (SLAM) solution, ending up with more accurate results than the original LIDAR.

Gao et al. (2015) makes use of LIDAR and GPS in order to provide periodic corrections to a INS, using a controlled environment to experiment with a Unmanned Ground Vehicle (UGV), getting an average error of $0.44\,\mathrm{m}$ in total with a Tightly coupled system, for inside and outside environments.

4 https://www.rollingstone.com/product-recommendations/electronics/best-lidar-robot-vacuums-1224185/

Kumar et al. (2017) proposes an indoor mapping and localization solution for a Unmanned Air vehicle (UAV) with the use of LIDAR and IMU sensors, also using Kalman filters to fuse LIDAR data. The paper ends by showing and stating that the solution is viable for UAV in SLAM.

Wang et al. (2018) only uses LIDAR for indoor localization, arriving at a result of $53.7\,\mathrm{mm}$ average error for their solution with $100\,\mathrm{mm}$ interpolation and $60.5\,\mathrm{mm}$ with $200\,\mathrm{mm}$ interpolation, in a controlled environment.

### 2.2.4 *Infrared*

Infrared (IR) indoor localization systems use IR light pulses to locate objects inside a building. IR receivers are installed in every room, and when the IR tag pulses, it is read by the IR receiver device.

IR can guarantee room-level accuracy. It uses light instead of radio waves, which can not go through walls. Radio-based systems have some trouble with false positives, as the radio waves can sometimes be picked up by other readers through walls.

In Yucel et al. (2012) an IPS using both IR technology and Ultrasonic, and the measurement of Time of Flight (ToF) is proposed and tested. By setting a couple of Transmitters for the Ultrasonic and IR signals, and equipping a receiver unit to the user, they obtain a maximum error less than $2\,\mathrm{cm}$ inside a $2\,\mathrm{m}$ to $2\,\mathrm{m}$ area.

Arai et al. (2019) uses IR to mimic a real-world scenario of retailing where a customer is tracked, by spreading a set of IR receivers on the ceiling and each costumer would carry a IR beacon. They also experimented with BLE for comparison purposes and achieved an average error of $648\,\mathrm{mm}$ using BLE and an average error of $147\,\mathrm{mm}$ mm with IR.

Lee et al. (2004) utilizes IR and the measurement of AoA to determine an indoor position, by using a set of emitters spread in the environment in the shape of a regular triangle with $40\,\mathrm{cm}$ length, and a receiver on the user, stating the experimental results were acceptable.

Cahyadi et al. (2019) also uses IR technology for an IPS. The major difference from previous solutions is that it is based on the available surveillance cameras as receivers and a single invisible IR LED beacon installed in smartphones as a transmitter, achieving a mean error of $6\,\mathrm{cm}$.

Yang et al. (2018) proposes an indoor solution based on Passive Infrared (PIR) sensor using a grid-based accessibility map and an A-star algorithm. The environment had a $12\,\mathrm{m}$ to $7.2\,\mathrm{m}$ dimension, and used a total of 10 PIR sensors, obtaining an average error of $0.21\,\mathrm{m}$.

Martín-Gorostiza et al. (2019) utilizes a fusion of cameras and IR sensors to develop their IPS. Their solution covered a $4\,\mathrm{m}$ to $3\,\mathrm{m}$ cell with 5 infrared detectors on the ceiling and one single camera, achieving an average error of $1.4\,\mathrm{cm}$, and a maximum of $2.5\,\mathrm{cm}$.

### 2.2.5 *Wi-Fi based*

Wi-Fi is a great technology for indoor positioning due to its overall presence around us, it does not need any additional infrastructure like other technologies, but it also doesn't provide as good accuracy as some of the other technologies.

Yang and Shao (2015) enumerates some factors that make Wi-Fi-Based Indoor Positioning challenging, and how Wi-Fi, due to its wide deployment, might become a prominent tool for indoor positioning. In their IP system they use Wi-FI APs as anchors, and any mobile platform with Wi-Fi capability, proposing a new AoA approach, reaching a maximum result of $2.2\,\mathrm{m}$ error.

He and Chan (2016) discusses various papers about Wi-Fi fingerprinting, analysing the basics of Fingerprint Localization, exploiting Spatial and Temporal signal patterns, which are the geographical distribution of signals and the Wi-Fi signal sequence patterns during walking in the indoor environment, also reporting a lot of data about where to use Spatial and Temporal signalling. It moves then to Collaborative Localization, presenting two categories: Distance-based, which measures the distance between users, and Proximity-based, since the accurate distance between users may not always be available. It finishes with Motion-Assisted localization. Then the subject changes from analysing techniques to improving the efficiency of the system, discussing the reduction of the offline site survey, adapting to Fingerprints changes, as in change of APs location, change in transmission power, crowds of people and many more. They also discuss talking about the calibration of Heterogeneous Devices and the concern about energy efficiency for Mobiles.

Ma et al. (2015) focuses on an improved Wi-Fi Indoor Positioning algorithm via Weighted Fusion, based on a traditional fingerprinting algorithm, making use of an improved Euclidean distance algorithm and weighted fusion for estimating a user position, achieving an average error of $1.54\,\mathrm{m}$.

Liu and Yang (2011) approaches the challenge of a multi-floor environment, combining characteristics of RF trilateration and fingerprinting, reaching $100\,\%$ accuracy when estimating the floors.

Gentner et al. (2020) uses a Fine Timing Measurement (FTM) protocol, Wi-Fi Round Trip Time (RTT), to estimate a user position, using both a robot and a person to simulate a user trajectory, having $92\,\%$ of the robot errors below 1 meter, and for the person $70\,\%$ below $1\,\mathrm{m}$.

LASHKARI et al. (2010) proposes a classic Wi-Fi Indoor Positioning System (IPS), using filtering techniques when collecting signal strengths, interpolation techniques to reduce the time spent training the system, and the Euclidean distance algorithm

Nicolas Le Dortz and Zetterberg (2012) uses probability distributions for estimating positions. In the offline phase, the signal strength distributions are estimated for the locations. In the online phase, a set of samples are collected and their signal strength distribution is calculated and compared to the ones in the radio map, reaching an error lower than $3\,\mathrm{m}$.

### 2.2.6 *Discussion*

Table 1 summarizes the accuracy of the positioning technologies as well as their advantages and disadvantages. The table contains 4 columns, the first one indicates the technology, the second the accuracy range, the third the advantages of this technology and the fourth the disadvantages.

| Technology | Accuracy | Advantages | Disadvantages |
|---|---|---|---|
| BLE | 2-3 m | Built into most smartphones, low power consumption, low cost. | Lower range than classic Bluetooth, subject to radio interference. |
| UWB | cm | High accuracy and does not interfere with already deployed RF systems | Expensive hardware, subject to interference from metal and liquid objects. |
| LIDAR | mm | High accuracy, can be combined with IMU sensors, cameras, sonar and others | Expensive, problems penetrating dense materials |
| IR | cm | Guaranteed room level accuracy | Cannot go through walls, needs to add sensors in every room, is subject to interference from sunlight, requires LOS |
| IEEE 802.11 Wireless LAN (Wi-Fi) | 3-5 m | Technology already installed and with good coverage most of the time, not needing additional hardware, built into most smartphones, low cost. | Vulnerable to AP changes, subject to radio interference. |

Table 1: Algorithms classification accuracy

## 2.3 MEASUREMENTS

### 2.3.1 *Time of Arrival*

ToA, also known as Time of Flight (ToF), is the instant when a radio signal from a transmitter reaches a receiver. Usually, this measurement, since the velocity of a radio signal is known, uses 3 emitters to estimate a position for a receiver based on its ToA, as shown in Figure 2. The 3 distinct ToA from different emitters are received, by intersecting the range of the 3 emitter for the corresponding ToA we can end up with a single point.

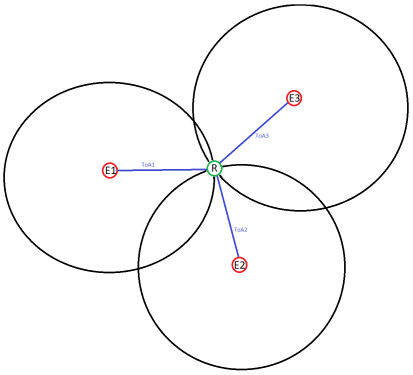



Figure 2: Example of multilateration using ToA

In the previously described work, Hol et al. (2010), the author utilize UWB along with the ToA for their IPS. This technique is accurate, but it is complex to implement, as it requires strict time synchronization of all the devices involved.

### 2.3.2 *Time Difference of Arrival*

TDoA measures the time differences between received ToA from multiple anchors. Then distance differences and multilateration are used to provide the position estimation. This type of measurements has the advantage that only the base stations need o be synchronized, but it is affected by multipath propagation of the signals.

### 2.3.3 *Received Signal Strength*

The Received Signal Strength (RSS) is the intensity of the signal emitted by an AP at the receiver end. The RSS value and a signal propagation model enables calculating the distance to the AP that send the signal. A common technique associated with this type of measurement is either Fingerprinting or multilateration. It is simple to deploy and does not require special hardware besides a wireless network interface card. However, the signal fluctuation is significant due to obstacles and other disturbances. Fingerprinting, in addition, requires a site survey, which is a time-consuming procedure.

### 2.3.4 *Angle of Arrival*

AoA is a measurement based on the angle of the signal received. The way it works is by using multiple beacons, and using lines with mentioned angles, the point of intersection is the estimated position, as exemplified in Figure 3.

This measurement does not require time synchronization, but it needs complex hardware to measure the AoA, and is highly affected by NLOS conditions.

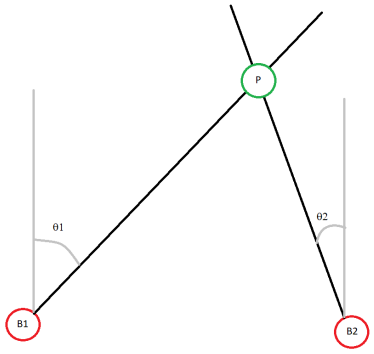



Figure 3: Example of AoA

### 2.3.5 *Discussion*

Table 2 consists of 3 columns, which depict the measurements in question and the advantages/disadvantages of them:

| Measurement | Advantages | Disadvantages |
|---|---|---|
| ToA | Accurate. | Requires time synchronization from all devices involved. |
| TDoA | Only requires synchronization on base stations. | Affected by multipath signals. |
| RSS | Simple and no need for special hardware besides a wireless network interface card. | Signal fluctuation due to environment, requires a site survey. |
| AoA | No need for synchronization. | Affected by NLOS, and requires complex hardware to detect AoA. |

Table 2: Types of measurements advantages and disadvantages

## 2.4 TECHNIQUES

### 2.4.1 *Trilateration*

Trilateration is a technique based on the position of the beacons and the distance from the user/device to those beacons. For this to work properly in 2 dimensions, there is the need to have at least three beacons. With just

one beacon, we end up with a circumference of possible positions as shown in Figure 4 (a), With two beacons, we end up with two possible positions, where the distance range of the beacon intersects, as in Figure 4 (b). With three beacons, we end up with one point where they all intersect, as seen in Figure 4 (c). Similarly, in the case of 3 dimensions, we need a minimum of 4 beacons. In a real case, the range measurements have errors and, therefore, the intersection of the circumferences might not be a single point. This is addressed by using more elaborated estimators.

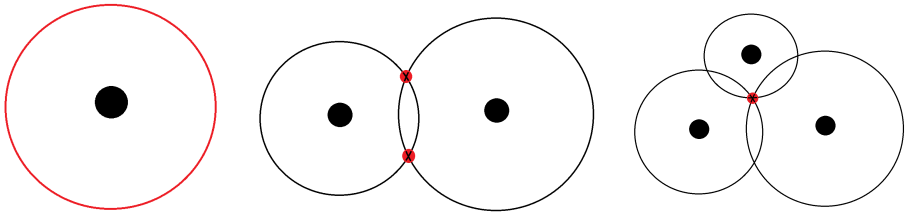



Figure 4: Range-based positioning with (a) 1 Beacon, (b) 2 Beacons and (c) 3 beacons.

### 2.4.2 *Triangulation*

Triangulation utilizes the geometric properties of triangles in order to estimate positions. Using two points with known coordinates, and using the angle from the object/user to the points it can determine its position in 2 dimensions, as seen in Figure 3.

### 2.4.3 *Proximity*

Proximity consists of providing the closest anchor/emitter location instead of estimating an accurate location. Bluetooth and Radio Frequency Identification (RFID) are two technologies associated with this technique.

### 2.4.4 *Fingerprinting*

Fingerprinting is a common technique used for indoor positioning. It consists of two phases, the offline phase and the online phase.

The offline phase corresponds to the construction of the radio map and the calibration of algorithms to use. The radio map is a set of the signals information, usually the RSS measurements of the signals and the coordinates where these were captured, these are then commonly referred to as fingerprints.

The second phase, the online phase, corresponds to the actual deployment of the positioning system. Both, the radio map previously collected and the tuned algorithm, are used to estimate the position of an incoming fingerprint from a user. The calibration of an algorithm generally uses two radio maps, one for training and another for validation. We try to estimate the positions of the validation radio map fingerprints, using the fingerprints of

the training radio map and our algorithm, for example, $k$-NN. The $k$-NN model takes the $k$ nearest fingerprints in terms of RSS levels and uses them to estimate a position.

## 2.5 WI-FI FINGERPRINTING

Wi-Fi Fingerprinting is a common solution for IPS, since Wi-Fi is a ubiquitous communications technology and may provide good coverage, meaning there will not be a need for additional hardware deployment most of the time. Additionally, fingerprinting is a technique easy to deploy that provides good accuracy for several applications.

The idea of Wi-Fi fingerprinting consists of applying the Fingerprinting technique on the measurement RSSI using Wi-Fi technology. The radio map consists of fingerprints that have the RSSI detected for each AP and the coordinates where these measurements were taken.

There are many works that use this style of IPS, such as He and Chan (2016), Nicolas Le Dortz and Zetterberg (2012), Torres-Sospedra et al. (2015), Ninh et al. (2020), Xia et al. (2017), Qin et al. (2021), Meng et al. (2011), Hossain et al. (2012), Costilla-Reyes and Namuduri (2014) and Torteeka and Chundi (2014), all of them providing results and alternative algorithms for this kind of data.

But there are some challenges with this approach, if a radio map has a significantly high number of fingerprints covering a large area, there can be unnecessary processing of data, leading to a long time for estimating a position.

Assume a radio map that covers 3 buildings, a library, class building 1, and class building 2, as depicted in Figure 5. If a user is located in the library, fingerprints from class building 2, might no have that much in common. In fact, they will not be used to estimate the user position, since they are so far apart, but they are still processed and compared against the user fingerprints, even though it is futile.

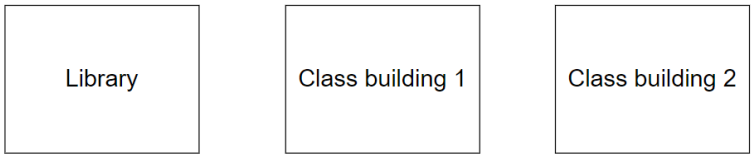



Figure 5: Example of a real-world scenario

We do not actually need to process every fingerprint to extract the most similar/closest fingerprints in the radio map. Implementing a method that prevents this behaviour can greatly decrease the time it takes to compute a position estimate, but if not correctly implemented can also increase the positioning error as relevant fingerprints might be filtered out.

Clustering is a solution for this challenge, since a radio map can be divided into multiple smaller radio maps, and each fingerprint is compared to a smaller amount of fingerprints this way. However, with traditional cluster-

ing models, such as $K$-Means, the generated subsets are disjoint, reducing the information available when an operational fingerprint lies near the cluster boundaries.

## 2.6 CLUSTERING

There are several known clustering techniques, which provide good performance and an efficient partition of a data set. Algorithms like $K$-Means, DBSCAN, Affinity propagation, Mean shift, Spectral Clustering, Ward, Agglomerative Clustering, OPTICS, BIRCH, and Gaussian Mixture, are good examples of clustering. Some of them have been integrated into indoor positioning as stated in Subsection 2.6.3 about Indoor Localization Clustering.

### 2.6.1 *K-Means*

$K$-Means, MacQueen (1967), is a well-known unsupervised algorithm used for clustering. The way it works is by defining an initial $K$, which represents the number of clusters to form. Then, the initial $K$ centroids are usually selected randomly from within the dataset samples, i.e., the radio map in fingerprinting. However, some authors consider this random initialization inefficient as spreading out the $K$ initial cluster centroids should be better. All remaining samples are assigned to the nearest centroid. Anyway, when all samples are assigned to a cluster, the centroid of each cluster is recalculated and the samples are again assigned to the nearest centroid. The process is repeated until the cluster distribution becomes stable, which means no samples (fingerprints) change from one cluster to another. Some implementations stop the clustering process after a given number of iterations or when the difference in the cluster distribution is marginal.

We provide an illustrative example in Figure 6 for a toy dataset, see Figure 6(a), and $K = 3$ on our $K$-Means. The algorithm starts by selecting $K$ samples. In the example, the 3 randomly selected samples are highlighted as red, green and blue points in Figure 6(b). Those three points correspond to the initial centroids. Then, all samples are assigned to the closest centroid as shown in Figure 6(c). These are the initial clusters, the next step is to calculate the median of each cluster, represented with a hollow circle in Figure 6(d), and get the new centroids. Then, the samples have to be reassigned to the closer cluster centroid. This process is repeated until the clusters stop changing.

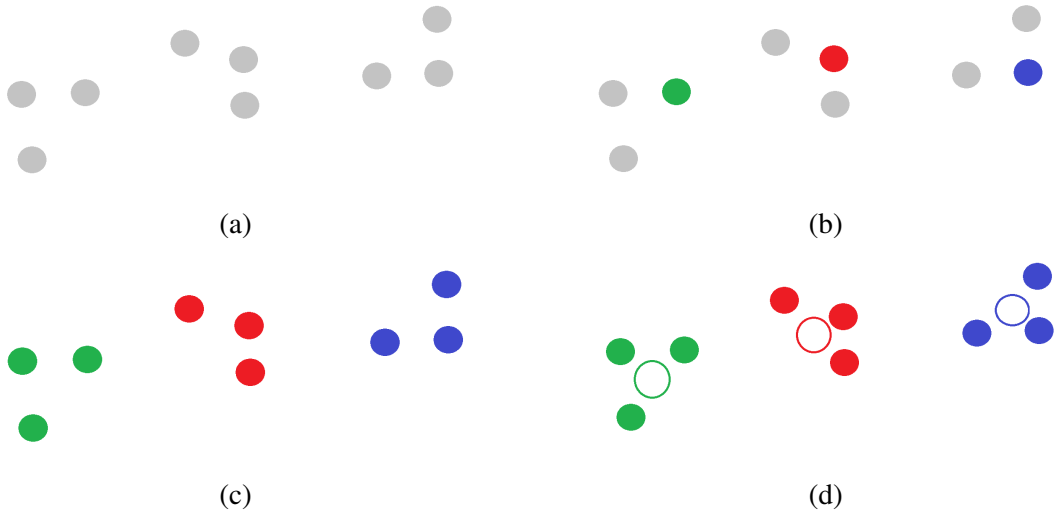



Figure 6: Clustering with $K$-Means: (a) Unclustered Point, (b) Initial clusters centroids, (c) Initial Clusters, and (d) Clusters and centroids

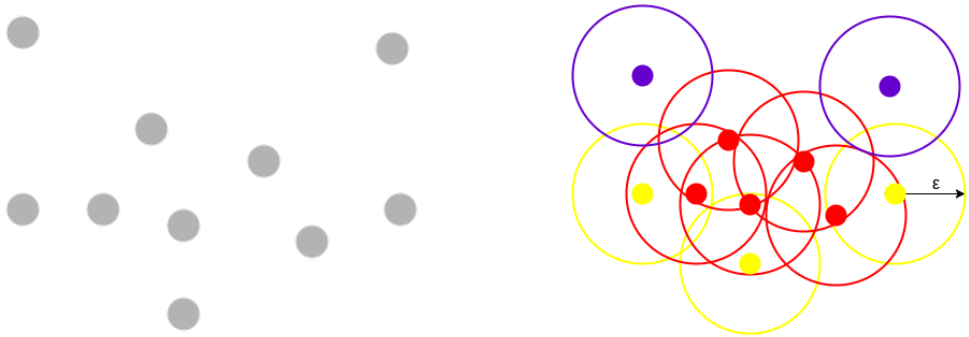
### 2.6.2 *DBSCAN*

DBSCAN, Ester et al. (1996), is a density-based clustering algorithm that groups samples that are close to each other. It also relies on the Euclidean Distance to determine if two samples are close. This clustering model has 2 hyper-parameters, namely *minPoints* and *EPS*. The *EPS* is the radius of a circle to create around each sample. The *minPoints* represents how many samples must be inside an EPS range for a sample to be considered a Core point.

The method is quite simple once understood, but somewhat difficult to explain properly, so it will be explained via an illustrative example[5].

Suppose that we have the set of samples, in a 2-dimensional feature space, depicted in Figure 7. The way DBSCAN works is by creating a circle (a hypersphere for $n$ dimensions) with a radius of $EPS$ around each sample, and classifying them into Core, Border and Noisy points.

5 Both images were inspired from [https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/]




(a) (b)

Figure 7: Example of DBSCAN clustering[5].

In the example, *minPoints* is set to 3. Therefore, all the samples that have 3 other samples in the EPS range are considered Core samples, represented in Red in the figure. Core samples can be used to add new samples to the cluster. In contrast, the Border samples are those samples that fall inside the Core samples, but do not have enough *minPoints*. Border samples, represented in yellow in the figure, are also considered part of the cluster but will not be used to add new samples to the cluster. Finally, those samples that are not in the range of Core points and do not satisfy the *minPoints* requisite are consider Noisy points, represented as Purple in the figure. Noisy points are usually discarded after clustering as they are considered outliers.

### 2.6.3 *Clustering in Indoor Localization Systems*

A literature review in clustering for indoor positioning shows that there are several studies around applying clustering models in indoor positioning.

Sadhukhan et al. (2021) proposes a clustering strategy for fingerprint-based positioning systems to search overhead incurred by such systems, reduce the storage overhead, and introduce a robust outlier mitigation technique. Their outlier technique uses the Hample Filter with some data from MAD. After removing the outliers from the radio map they proceed to apply a two-step clustering strategy. In the first step, it partitions all the training locations into several disjoint clusters based on the working principle of one-way hierarchical clustering strategy (1-way HCS) proposed in a previous work Saha and Sadhukhan (2015).

The rule of 1-way HCS is that the set of training locations belonging to a cluster receives the strongest signal strength from a particular AP only. So, the number of clusters created by the proposed strategy is equal to the number of APs deployed in the localization area. The second step consists of first calculating the Euclidean distance between any pair of the RSS patterns belonging to the same cluster and then adjoining those patterns whose euclidean distances are bounded by a certain value (called threshold) into a single RSS pattern.

They compare this clustering and outlier strategy to Affinity propagation, $K$-Means, 2-way HCS, SOM and NNSS, achieving the lowest positioning error, of $7.5\,\mathrm{m}$, and the lowest computational time with their solution.

Tian et al. (2013) utilizes Affinity propagation in order to cluster a radio map, but they give priority to samples based on the median of the RSS values. The dimension of their test area was $66\,\mathrm{m}$ to $22\,\mathrm{m}$. A total of 4 implementations were tested: 1) $K$-Means with $k$-NN, with results of mean error of $3.38\,\mathrm{m}$ and max error of $6.72\,\mathrm{m}$; 2) $K$-Means with Probability Distribution, achieving results of a mean error of $3.11\,\mathrm{m}$ and max error of $6.70\,\mathrm{m}$; 3) Affinity propagation with $k$-NN, obtaining a mean error of $2.66\,\mathrm{m}$ m and a max error of $6.67\,\mathrm{m}$; and finally 4) their proposed solution, which achieved a mean error of $2.23\,\mathrm{m}$ and a max error of $6.52\,\mathrm{m}$.

Hu et al. (2015) combines AP sets similarity and RSS distance when calculating fingerprint distance, also making use of a semi-supervised Affinity propagation clustering algorithm in combination with detection of isolated points to address problems of the time-varying nature of wireless signals, and Weighted K Nearest Neighbors (WKNN). The first step is to calculate a fingerprint AP similarity and average RSS distance to all other fingerprints. Then $k$-NN is utilized in order to cluster the radio map based on the nearest neighbours using the semi-supervised Affinity propagation. They then compared the proposed solution with $K$-Means and using only WKNN. For $K$-Means, the best results were obtained with $k = 3$, with a mean error of $2.28\,\mathrm{m}$, a standard deviation of $1.57\,\mathrm{m}$ and a median of $1.94\,\mathrm{m}$. For single WKNN, the mean error was $2.59\,\mathrm{m}$, with a standard deviation of $2.08\,\mathrm{m}$ and a median of $1.92\,\mathrm{m}$. For their proposed solution, the best result had $1.85\,\mathrm{m}$ of mean error, a standard deviation of $1.39\,\mathrm{m}$, and a median of $1.46\,\mathrm{m}$.

Arsan and Hameez (2019) implements $K$-Means, fuzzy cmeans, and mean shift for clustering with a Kalman filter, using UWB technology. For the $K$-Means solution, they obtain an average of $9.27\,\mathrm{cm}$, for the fuzzy cmeans $9.32\,\mathrm{cm}$, for the mean shift $9.89\,\mathrm{cm}$ and for the raw data $6.34\,\mathrm{cm}$ average error.

Li et al. (2021) proposes a Cluster-Based Principal Component Analysis to extract the main components of a sample and cluster them together. This solution is then compared to: 1) $k$-NN-based clustering, which achieved results of $0.75\,\mathrm{m}$ average error and a maximum error of $2.30\,\mathrm{m}$; 2) $K$-Means with an average error of $0.66\,\mathrm{m}$ and a maximum error of $2.02\,\mathrm{m}$; 3) Weighted $K$-Means, with $0.60\,\mathrm{m}$ m average error and a maximum error of $1.88\,\mathrm{m}$; 4) Fuzzy $c$-means, with an average error of $0.43\,\mathrm{m}$ and a maximum error of $1.31\,\mathrm{m}$; and, finally, 5) the proposed solution with an average error $0.39\,\mathrm{m}$ and a maximum error of $0.85\,\mathrm{m}$ m.

REN et al. (2019) proposes a variant of the cmeans algorithm, the improved public cmeans, which handles the samples near cluster boundaries. At each cluster boundary, it creates a public section, which contains the boundary samples from both clusters, and if the probabilities of a point being part of both clusters are similar, the sample is classified as being part of the public section. They compared their proposed solution to $K$-Means and fuzzy cmeans, only the accuracy within $2\,\mathrm{m}$ to $3\,\mathrm{m}$ is accounted for, achieving $50\,\%$ and $76\,\%$ for $K$-Means, $53\,\%$ and $78\,\%$ for fuzzy cmeans, and $57\,\%$ and $84\,\%$ for the proposed solution.

Chen et al. (2015) makes use of the APs similarity to cluster. The idea is to start and create a centroid for every AP, the centroid is the sample where that AP has the highest RSSI value detected in the whole radio map. The second step is to cluster all the other samples according to their Euclidean distance to the previously established centroids, and the final step is to go through each cluster and identify the best possible centroid for them. This solution is compared to the use of plain WKNN without any clustering and to the use of $K$-Means, achieving an average error of $0.88\,\mathrm{m}$ for WKNN, $0.82\,\mathrm{m}$ for the $K$-Means, and $0.77\,\mathrm{m}$ for the proposed solution.

Lee et al. (2013) presents a novel support vector machine based clustering approach, which uses the margin between two canonical hyperplanes for classification instead of using the Euclidean distance between two centroids of reference locations. The proposed algorithm reduced the mean error when compared to $K$-Means by $25.34\,\%$, $25.21\,\%$ compared to Affinity propagation, and $26.01\,\%\%$ when compared to Support Vector Clustering (SVC).

Bai and Wu (2013) utilizes $K$-Means to cluster and principal component analysis for the fingerprints, ending with $k$-NN for estimating a position, stating that the computational time has been reduced without a major penalty on the positioning error.

MA et al. (2008) proposes a Cluster Filtered $k$-NN, it clusters fingerprints by their physical location, improving the average positioning error of the solo use of $k$-NN by $1.36\,\%$.

Cramariuc et al. (2016) analyses different clustering methods and introduces the Penalized Logarithmic Gaussian Distance metric to improve the clustering performance. They analyse the use of Affinity propagation and $K$-Means, with and without the proposed metric over two datasets, achieving their best result of $6.8\,\mathrm{m}$ average error on the first dataset and $3.6\,\mathrm{m}$ average error on the second dataset, both with the use of the $K$-Means plus the proposed metric.

Zhang et al. (2016) proposes a Domain Clustering, the idea is to identify the room where the target is located based on the visibility of the APs. A visible AP is one that can be detected in all the calibration points on a room, and the room with the most common visible APs with the target is selected, and then apply the $k$-NN to identify the nearest neighbours and estimate the target position. The proposed solution is tested on and compared to WKNN and the Naive Bayes Classifier, improving the result of both algorithms. In Naive Bayes, the average error is improved by $0.7\,\mathrm{m}$, and with the WKNN it improves only by $0.02\,\mathrm{m}$.

Suroso et al. (2011) utilizes fuzzy cmeans to cluster, and validates its ability in clustering, obtaining errors with less than $1\,\mathrm{m}$.

Subedi et al. (2019) uses two-level fingerprinting, using Affinity propagation clustering, Weighted Centroids and a propagation model to convert scanned RSS to distances. The weight is based on mentioned distances, claiming that their method reduced their radio map size by removing distant beacons, and that during the online phase the use of Affinity propagation minimizes the computational cost, obtaining an average error of $1.38\,\mathrm{m}$.

All the works provide decent errors, but the works lack the diversity of multiple real world scenarios, and most of them focus on the APE as the main criteria, which is an important metric, but leaves the question if the methods are actually scalable. Moreover, the different methods are difficult to directly compare since they were evaluated in very different conditions. With this work we aim to develop methods that not only reduce the APE, but also reduce the computational cost associated, breaking the traditional error-time trade-off, while supporting scallability to large environments.

## 2.7 SUMMARY

There are many works on the subject of Clustering for Indoor location fingerprinting, most of them apply known algorithms or variants of this to tackle this challenge, while some try and develop methods that take into ac-

count the challenges of indoor fingerprinting. However, most of the works do not consider the use of multiple datasets, and those that do usually stay in the range of 2–3 datasets, not providing, enough data on their algorithm's reliability in different uncontrolled environments. i.e., the current state-of-the-art works fail in ensuring the generalizability of the proposed algorithms.

In this work, the SAS and BSC algorithms will be assessed over multiple diverse datasets, and compared to other traditional clustering algorithms, in order to assert their efficiency and reliability in multiple diverse environments.

# Part II

# CORE OF THE DISSERTATION

# 3

# SAS

## 3.1 INTRODUCTION

In this chapter the SAS algorithm is introduced, along with its description, visual samples are also presented for further clarification The chapter ends with some variants developed to either better understand SAS or to try and improve it.

## 3.2 DESCRIPTION

SAS utilizes a set of the strongest APs for each sample to perform clustering. This approach should, in theory, provide a more reliable clustering, since we use the strongest signals which are more dependable.

Even though classic clustering techniques have been applied to indoor positioning, the reduction in computational cost is usually achieved at the expense of a slightly higher positioning error Torres-Sospedra et al. (2022). With SAS the objective is to have a considerable reduction on the computational time, without having a penalty associated with the error, even in some cases having a gain in positioning error.

Similar to fingerprinting, SAS also works in two phases. In the offline phase, once the radio map is built, the clusters for the radio map are formed. In the online phase, for any incoming operational fingerprint, the position estimate is done in two steps. The first step is to identify which cluster (or set of clusters) the operational fingerprint belongs to. Then, once we know which cluster(s) the sample belongs to, we use the fingerprints belonging to them to estimate the position as in plain fingerprinting without clustering.

### 3.2.1 *Basics of Clustering in fingerprinting*

As mentioned before, fingerprinting requires two phases. In the offline phase, reference fingerprints ($s^t$) are collected in a set of locations whose position is known in advance, generating a radio map ($\mathcal{T}$). In the online phase, the operational fingerprints (collected at unknown positions) are compared to the fingerprints stored on the radio map. Their position is estimated using the locations of the most similar fingerprints in the radio map, usually computing their centroid.

Let $\mathcal{V}$ be the set of all the operational fingerprints ($s^v$) used to assess a fingerprint method. The cost of estimating the position for all operational fingerprints in $|\mathcal{V}|$ is $O(|\mathcal{T}| \times |\mathcal{V}|)$, since every operational fingerprint must be compared to each fingerprint in the radio map. This may not be efficient in large radio maps and/or for a large number of simultaneous users.

Clustering adds a new step to the offline phase (see (1) in Algorithm 1), which is devoted to generating the groups of similar fingerprints and a representative sample for each cluster. This step is just run once per radio map, and it has no impact on the operational time. However, some clustering models, such as Affinity Propagation Clustering (APC) are very demanding at this stage. In those dynamic systems where the radio map is updated regularly, the time to generate the clusters may be a critical factor. i.e., clustering should not take minutes or hours.

Clustering also modifies the operational phase to perform this two-step search. First, the most relevant cluster is identified (see (2) in Algorithm 1). In most of the clustering models, each cluster is represented by its centroid or a"popular" sample. Thus, the coarse search corresponds to finding the most similar cluster representative. Once, the centroid is selected, the fine-grained search is done over the samples belonging to that cluster (see (3) in Algorithm 1), being the reduced radio map $\hat{\mathcal{T}}$ much smaller that the full radio map $\mathcal{T}$ ($|\hat{\mathcal{T}}| << |\mathcal{T}|$). The most significant gains in terms of estimation time are expected to come from the use of a much smaller radio map.

The changes introduced by clustering in the traditional fingerprinting method are highlighted with (1)–(3) in Algorithm 1.

**Algorithm 1** Pseudocode of $k$-NN for positioning

```
input T, V, k
(1) offline pre-processing of training datasets (radio map)
for i = 1 to |V| do
    (2) Identify most relevant cluster
    (3) Generate reduced radio map, T̂, using T and s_i^v
    for j = 1 to |T̂| do
        Compute similarity between s_i^v and s_j^t
    end for
    Sort radio map samples by similarity
    Select the k closest candidates ( better similarity)
    Estimate building, floor and position
end for
Return: Estimated positions, floors, buildings
```

### 3.2.2 *The Strongest AP Set (SAS) Clustering*

Although traditional clustering models have been able to tackle indoor positioning, the efficiency has often been achieved at the cost of slightly worse positioning accuracy Torres-Sospedra et al. (2022). Some knowledge-

based rules can narrow the problem and help us in generating better clusters (step 1) and have better ways to identify the most relevant cluster (Step 2) for a given operational fingerprint.

In order to perform clustering over radio maps that takes into account the challenges of RSSI-based positioning, we have developed SAS. SAS takes advantage of the link between the strongest AP and the sub-region in the operational area when creating a cluster. i.e., the set of strongest APs in an operational fingerprint is the key to indicating the coarse-grained region where it has been collected since those strong signals cannot be measured anywhere else (in regions far from the APs that transmitted them).

To cover a wide set of scenarios, SAS utilizes 2 hyperparameters, $N$ and $P$, to operate. The former represents the length of the set of strongest APs in a sample, and the latter indicates the minimum number of common strongest APs between two samples required for them to belong to the same cluster. Both hyperparameters have to be individually set for every dataset.

SAS follows the algorithmic structure of the traditional fingerprint-based Indoor Positioning Systems (IPSs) as described in Bahl and Padmanabhan (2000) and introduced in Algorithm 1, where the $k$-NN model is applied over the reduced radio map, $\hat{\mathcal{T}}$, to estimate the position.

*Creating the clusters*

In the offline phase, SAS clusters the radio map by exploiting the strongest APs according to Algorithm 2. First, the set of strongest APs is identified for every reference sample in the radio map as follows:

1. For each reference fingerprint in the radio map, $\mathcal{T}$, get the identifiers of the $N$ strongest APs and the strongest absolute RSSI value;

2. The APs with undetected RSSI values are excluded from the strongest AP set and the identifier is replaced with the value $-1$; this happens when the number of observed APs in the reference fingerprint is shorter than $N$;

3. Fingerprints with $P$ or less detected APs in their strongest AP set, are considered noisy samples and, therefore not included in the filtered radio map $\dot{\mathcal{T}}$;

4. Sort fingerprints in descending order according to the strongest RSSI value and provide sorted filtered radio map $\ddot{\mathcal{T}}$ and corresponding sets of strongest APs $SAS$.

A visual example is provided in Fig.8 with $N$=3 and $P$=1.

Generating the clusters requires iterating through the sorted list of fingerprints. Marked fingerprints, already members of at least one cluster, cannot form their own cluster. In the iterative process, if a fingerprint is not marked, then:

1. It is marked and it starts its own cluster, $i$, and its set of strongest APs is used as the cluster $i$ representative.

2. Its set of strongest APs will be compared to the sets of strongest APs of all the other reference fingerprints.

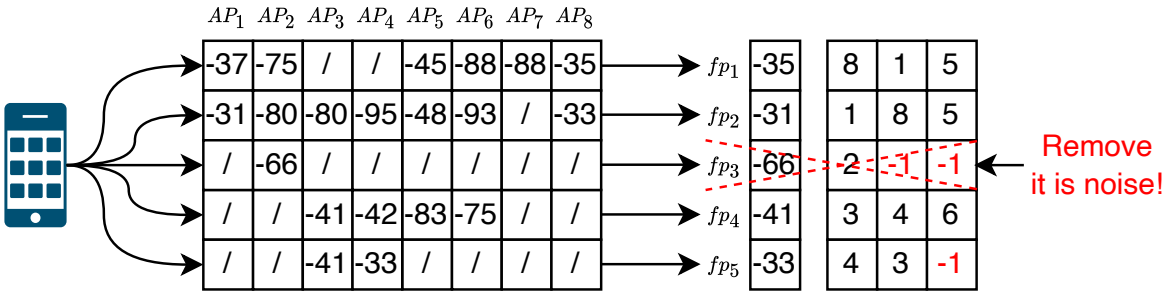



Figure 8: Identification of the Strongest AP Sets for cluster generation(N=3, P=1)

3. Those reference fingerprints that have $P$ or more APs in common with the representative of the created cluster (APs with identifier $-1$ are not considered) are also marked and assigned to the new created cluster, $i$.

Let us define $S_i$ as the set of the N strongest AP identifiers in $s_i^t$: $ap_1, ap_2, \ldots, ap_N$, and $\Omega$ as the set of all $S_i$ values. For clarification, we provide a graphical example in Fig.9. The full algorithmic description of the clustering process is provided in Algorithm 2.

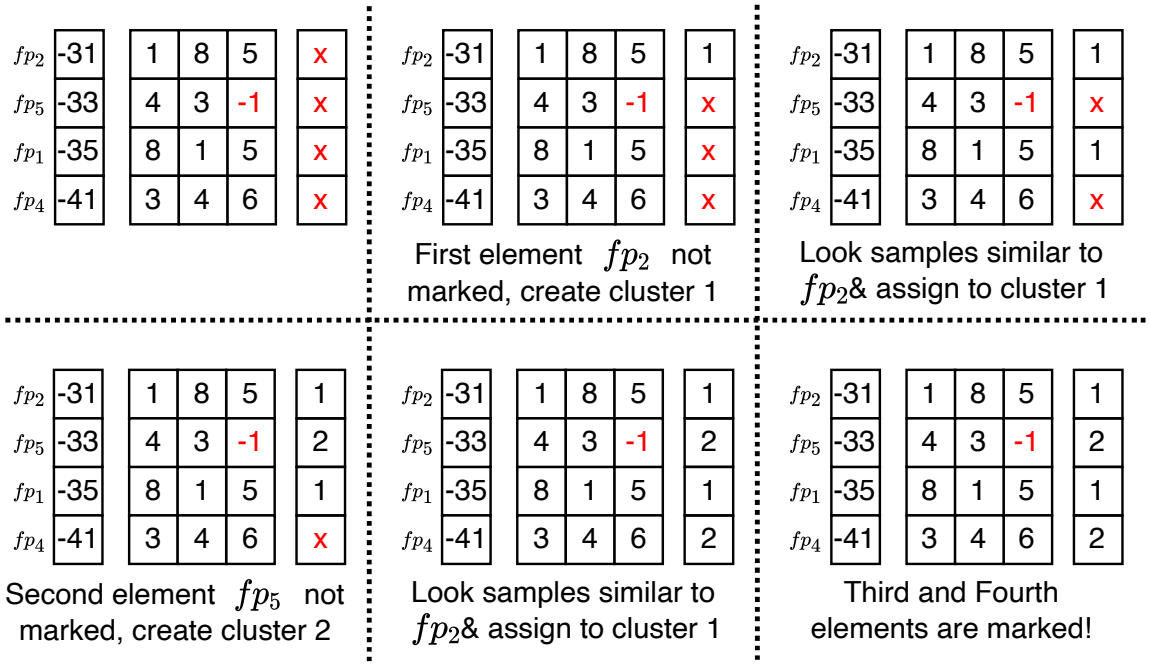
**Algorithm 2** SAS Clustering

**Input:** $\mathcal{T}$ (ordered in descendent order of strongest AP), $\Omega$, $N$, $P$
// Initialize Variables
$C = \{\}$ // Initialize set of clusters
$CR = \{\}$ // Initialize cluster representatives
$N_{cl} = 0$ // Initialize number of current clusters
**for** $i = 1$ to $|\mathcal{T}|$ **do**
    // Process fingerprint if it doesn't belong to any cluster
    **if** $s_i^t \notin \bigcup_k C[k]$ **then**
        $N_{cl} = N_{cl} + 1$
        $C[N_{cl}] = \{s^t \in \mathcal{T} : |S_j \cap S_i| > P, \forall j \in \{1, 2, \ldots, |\Omega|\}\}$
        $CR[N_{cl}] = S_i$
    **end if**
**end for**
**Return:** $C$, $CR$

*Cluster identification in the operational phase*

In the operational phase, when a new fingerprint is collected, the system should identify which is the most relevant cluster (or set of clusters) for that sample. The corresponding procedure is given by Algorithm 3 and is made of the following steps:

1. Get the identifier of the $N$ strongest APs of the current operational fingerprint, the APs with undetected RSSI values are excluded from the set and the identifier is replaced with $-2$.




$$C[1] = \{1,3\} \qquad CR[1] = [1,8,5]$$
$$C[2] = \{2,4\} \qquad CR[2] = [4,3,-1]$$

Figure 9: Iterative process to generate the Clusters with SAS

2. Compare the set of strongest APs to all cluster representatives and get how many APs they have in common, again undetected APs are not considered.

3. Select the cluster with the highest similarity (highest number of common APs). In the case of a tie, select all the clusters with the highest similarity.

**Algorithm 3** SAS Cluster Identification

**Input:** $CR$, $S^{\mathcal{V}}$
$D = \{\}$ //is the set of selected clusters
$E = \{\}$ // is the set of similarities
**for** $i = 1$ to $|CR|$ **do**
    $E[i] = \{|S^{\mathcal{V}} \cap CR[i]|, i\}$
**end for**
$\dot{E}$ =sort E in descending order of the first element (similarity)
$\ddot{E}$ =select the first element of $\dot{E}$ and all other elements with the same similarity
$D = \bigcup_k C[\ddot{E}[k,2]]$
**Return:** $D$

*Generate the reduced radio map*

Once the most similar cluster (or set of clusters) are identified, the fingerprints to generate the reduced radio map, $\hat{\mathcal{T}}$, can be easily retrieved. Then, the $k$-NN algorithm is applied to estimate the position using the reduced radio map.

As the cluster identification may provide multiple "*most similar*" clusters for an operational fingerprint, the fingerprints of each cluster have to be inserted into the reduced radio map. This raises a question about those fingerprints that belong to multiple clusters. In SAS, multiple instances of the same fingerprint are not allowed in the reduced radio map and only one instance is included.

## 3.3 VARIATIONS

In order to gain more insight into the results obtained with the original SAS algorithm, and to experiment with some variations of the same, a set of derived algorithms from SAS were created. In particular, we have considered 4 SAS variations: the **Classic** one as described in previous sections, the **Heavy** variant, the **RCR – Recalculate Cluster Representatives**, and the **3C – Calculate Cluster Centroids**

These variations are attempts to either bring down the computational cost or the average error of classic SAS. Their description are provided in the next subsections.

### 3.3.1 *Heavy*

The purpose of this variant is to find out if SAS is selecting the most optimal cluster, or if it needs to consider more clusters in order to bring down the average error. With respect to the classic SAS, the main difference relies on the cluster selection during the online phase.

In this variant instead of selecting only the clusters with the highest similarity, it also selects the ones with the second highest similarity. By forcing the selection of multiple clusters, we can expect an increase in computational cost, but we can asses if the clusters being selected by the classic SAS are actual the ideal and contain the best samples, or if this version can somehow provide better results in terms of average positioning error.

### 3.3.2 *Recalculate Cluster Representatives (RCR)*

RCR cluster the same way as SAS, but after forming all the clusters, it iterates through them and creates a new cluster representative for each of them. The objective is to create a cluster representative that represents better the set of samples inside the cluster.

In this case, the cluster Representatives will not actually be derived from a real sample, but rather from all of the samples in the cluster. We take all of the $N$ strongest APs for each sample in the cluster, and after retrieving them all we take the $N$ most common ones and form our new cluster representative.

This approach may offer some good results, without a necessary computational cost increase.

### 3.3.3 *Calculate Cluster Centroids (3C)*

Similar to the previous version, 3C, clusters just like SAS, but in the end it calculates the cluster centroid for each cluster, using the average of samples from the said cluster.

During the online phase, it uses the Euclidean distance, in the RSSI space, to every centroid to determine the best suitable cluster for the input sample.

# 4

# BSC

## 4.1 INTRODUCTION

In this chapter, the BSC algorithm is introduced, along with its description, and visual samples are also presented for further clarification.

## 4.2 DESCRIPTION

Base Station Clustering (BSC) is another clustering algorithm designed specifically for indoor positioning. The way it works is by being more expensive in terms of memory to alleviate the computational cost while trying to not have a penalty for the error or even provide a gain in it.

It utilizes two hyperparameters, $N$, which stands for the number of strongest APs and $T$ which is the RSSI value to be used as a threshold when determining the clusters of a given sample.

The clusters are stored in a matrix, where the columns correspond to the reference APs and the rows to RSSI levels. Each cluster is stored in a cell and contains the samples where the reference AP has a valid value and they are in the range $[RSSI - T, \ldots, RSSI + T]$. In contrast to other clustering models, a sample can belong to multiple clusters in BSC, and in SAS. But a cluster is only formed if there is a sample with that column that has its strongest AP, so if no sample has that AP as the strongest one, no cluster is formed for that column, even if it is detected in some samples.

### 4.2.1 *Clustering*

The first step in the clustering stage is to iterate over all the samples in the radio map and identify the AP providing the strongest RSSI. Then, for each sample in the radio map, the second step is to identify all the samples in the radio map where the RSSI difference for the strongest AP is not higher than a threshold $T$. All samples meeting that criteria are added to the clusters. The full algorithmic description is provided in Algorithm 4, whereas a visual example is provided in Figure 10

To explain how BSC works, we provide an illustrative example in Figure 10 and Figure 11, where we assume that the radio map is composed of $3$ samples and the threshold, $T$, is set to one ($T = 1$).

**Algorithm 4** BSC Clustering

**Input:** $\mathcal{T}$, $T$
// Initialize Variables
$Clusters = []$ // Initialize set of clusters as an empty matrix
**for** $i = 1$ to $|\mathcal{T}|$ **do**
    // Process fingerprint
    ID = $i$ strongest AP ID
    RSSI = $i$ strongest AP RSSI
    **for** $c$ in [RSSI-T, RSSI +T] **do**
        **for** $j = 1$ to $|\mathcal{T}|$ **do**
            **if** $j \notin$ Clusters[ID][$c$] **then**
                rssi = $j[ID]$
                **if** $rssi \geq RSSI - T \wedge rssi \leq RSSI + T$ **then**
                    Clusters[ID][$c$] add $j$
                **end if**
            **end if**
        **end for**
    **end for**
**end for**
**Return:** $C$, $CR$

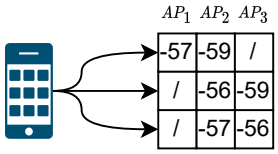



Figure 10: Samples collected

In the visual example, Figure 11, the cluster matrix is initially empty. For the first sample, $s1$, the strongest AP is AP1 with $-57\,\mathrm{dBm}$. Only $s1$ has a value close to $-57\,\mathrm{dBm}$ and AP1, therefore we only add $s1$ to clusters $c_{-56,AP1}$, $c_{-57,AP1}$ and $c_{-58,AP1}$. For the second sample, $s2$, the strongest AP is AP2 with $-56\,\mathrm{dBm}$. In this case $s2$ and $s3$ have a value close to $-56\,\mathrm{dBm}$ and AP2, therefore we add $s2$ to clusters $c_{-55,AP2}$, $c_{-56,AP2}$ and $c_{-57,AP2}$, and we also add $s3$ to clusters $c_{-56,AP2}$ and $c_{-57,AP2}$. As the value for $s3$ and AP2 is $-57\,\mathrm{dBm}$, we do not add $s3$ to clusters $c_{-55,AP2}$ because the difference is $2\,\mathrm{dB}$, which is above $T$. Finally, for the third sample, we only add $s3$ to clusters $c_{-55,AP3}$, $c_{-56,AP3}$ and $c_{-57,AP3}$.

### 4.2.2 *Cluster identification in the operational phase*

In contrast to other classic clustering models where the distance to centroids has to be calculated or the set of strongest APs is intersected, selecting the clusters in BSC is simple and straightforward. It just requires to access a cell of a 2-dimensional matrix, where the rows are related to RSSI values and the columns to the APs. Thus the $N$ strongest APs in the operational fingerprint and their corresponding RSSI are both used to retrieve the

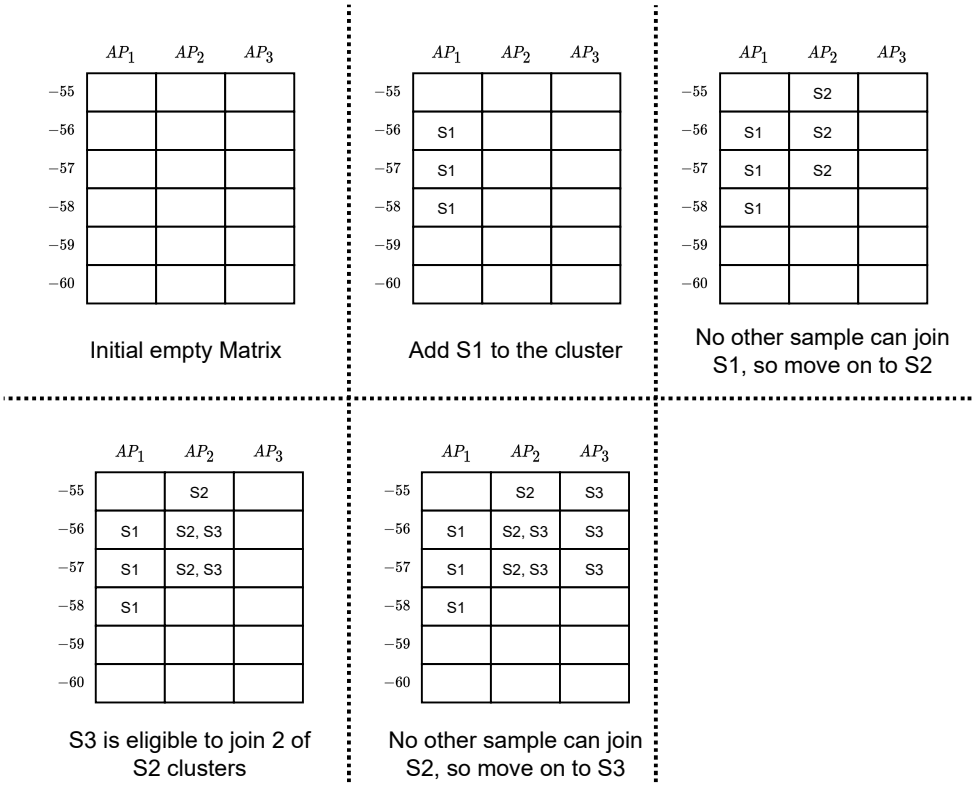



Figure 11: BSC Clustering process

lists of elements belonging to the cluster. This way we are selecting $N$ clusters, by just retrieving the information of $N$ cells.

In case one of the accessed cells is void, we will retrieve the cluster by increasing the RSSI until we meet the maximum value of RSSI or a non-void cluster. If no cluster is found, we will repeat the process, but, by decreasing the RSSI. If no cluster is formed for a specific AP then we utilize the full radio map.

Bellow is a visual example of cluster identification with $N = 2$, using the sample in Fig. 12, and the cluster matrix in Fig. 13.

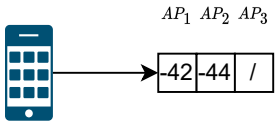



Figure 12: Sample collected

The first step is to take the N strongest APs and their RSSI values from the sample, $StrongestAPs = [(AP_1, -42), (AP_2, -44)]$. Now we iterate the strongest APs and add the cluster corresponding to their RSSI value to our reduced radio map, so first we add the cluster with row $AP_1$ and RSSI value $-42$, as in the second step of Fig. 13, meaning our current reduced radio map consist of cluster C21. Now we continue this process until we have the N strongest APs iterated, in this case, one more step, since $N = 2$, corresponding to

**Algorithm 5** BSC Cluster Identification

**Input:** $Clusters$, $N$, $sample$
// Initialize Variables
$SelectedClusters = []$
$StongestAPs$ //The array with the N strongest APs for the $sample$
**for** $i$ in $StrongestAps$ **do**
    $rssi = sample(i)$ rssi value
    **if** $Clusteres[i][rssi] \neq \emptyset$ **then**
        $SelectedClusteraddClusteres[i][rssi]$
    **else**
        Search the closest cluster for the same AP, by increasing the $rssi$, until the upper boundary is meet.
        If none is found do the same by decreasing AP, until the lower boundary is meet.
    **end if**
**end for**
**if** $SelectedClusters = \emptyset$ **then**
    $SelectedClusters = Clusters$
**end if**
**Return:** $SelectedClusters$

the last step of Fig. 13, where we add the cluster C42. And our reduce radio map is the merge of clusters C21 and C42.

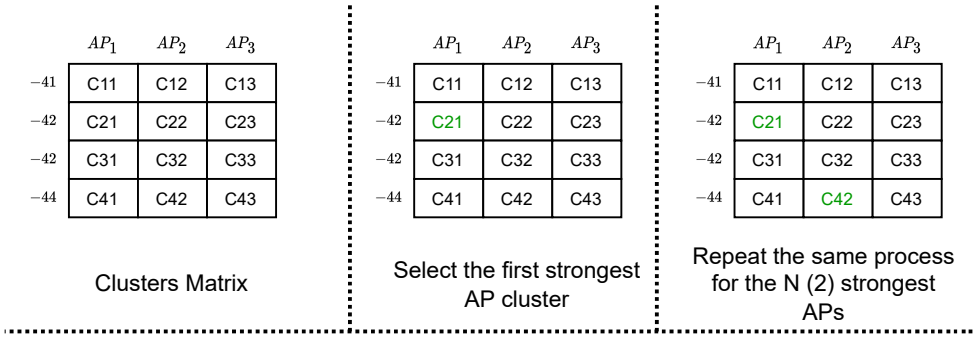



Figure 13: BSC Cluster Identification process

# 5

# EVALUATION METHODOLOGY

This section holds great importance, describing how we handle data, test algorithms and providing details about the experimental setup, which all enable reproducibility and replicability of the experiments.

## 5.1 EVALUATION METRICS

After defining and implementing our algorithms, we need a way to evaluate their effectiveness. To do so, a set of fingerprinting datasets were selected. They reflect multiple diverse real-world scenarios and are publicly available. The objective is to run the algorithms on all of these datasets and evaluate them in accordance with the following metrics:

- APE - Average Positioning Error, in meters $[\mathrm{m}]$, ISO/IEC 18305:2016 .
- Maximum Error - The maximum error in meters $[\mathrm{m}]$.
- Minimum Error - The minimum error in meters $[\mathrm{m}]$.
- Clustering Time - The time it takes for the clustering algorithms to create the clusters in seconds $[\mathrm{s}]$.
- Normalized Clustering Time - The time it takes for the clustering algorithms to create the clusters, divided by the number of samples, to asses scalability, in milliseconds $[\mathrm{ms}]$.
- Cluster Identification Time - The time it takes for identifying the most suitable cluster(s) for a given sample, in milliseconds $[\mathrm{ms}]$.
- Matching Time - The time it takes to estimate a position using the $k$-NN algorithm, in milliseconds $[\mathrm{ms}]$.

Besides the previous metrics, the CDF for each individual dataset will also be provided to better observe the distribution of error.

## 5.2 DATASET DESCRIPTION

To test the algorithms and compare them to $K$-Means and Plain $k$-NN, a set of datasets publicly available was used. In particular, we have selected a total of $10$ diverse datasets. These datasets are varied among them-

selves, they have a different number of samples in testing and training, and a different sample density per reference point. Some of them are multi-floor multi-building datasets, while others are all collected on the same floor and building. In addition, some have unprocessed RSS data, while others averaged RSS data per reference point or grid cell. Finally, some datasets were collected with a single device, but others consider device diversity.

All datasets were subject to a soft data cleanse, we removed all the APs that did not have any detected value above $-90\,\mathrm{dBm}$, and remove all the samples that did not have any detected AP. The 10 datasets are: DSI1 Moreira et al. (2020) which was collected in the Departamento de Sistemas de Informação in Minho University; UJI1 Torres-Sospedra et al. (2014) and LIB1-2 Mendoza-Silva et al. (2018) which were collected in the University Jame I, Spain; MAN1 King et al. (2008); King et al. (2008) which was collected in University of Mannheim, Germany; SAH1 Lohan et al. (2021) and TUT5-6 Richter et al.; Lohan (2020) which were collected in Tampere University, Finland; UTS1 Song et al. (2019) which was collected in University of Technology Sydney, Australia; and, finally, MINT1 Moreira et al. (2019) which was collected in PIEP in Minho University.

Table 3 introduces the definition of the parameters, whereas Table 4 presents a brief summary of each dataset in relation to their training and testing sizes, area dimensions, number of floors and buildings.

Table 3: Parameters Definition

| | |
|---|---|
| $\lvert\mathcal{T}\rvert$ | Number of Training samples |
| $\lvert\mathcal{V}\rvert$ | Number of Testing/Validation samples |
| $RP$ | Number of Reference Points |
| $D$ | Average number of Fingerprint per Reference Point |
| $f$ | Number of Floors |
| $b$ | Number of Buildings |

Table 4: Datasets descriptions

| | $\lvert\mathcal{T}\rvert$ | $\lvert\mathcal{V}\rvert$ | Dimension | Area | $f$ | $b$ | $RP$ | $D$ |
|---|---|---|---|---|---|---|---|---|
| DSI1 | 1369 | 348 | 100m × 18m | 1800m$^2$ | 1 | 1 | 230 | 6 |
| LIB1 | 3120 | 576 | 15m × 10m | 150m$^2$ | 2 | 1 | 48 | 12 |
| LIB2 | 3120 | 576 | 15m × 10m | 150m$^2$ | 2 | 1 | 48 | 12 |
| MAN1 | 14 300 | 460 | 50m × 36m | 1800m$^2$ | 1 | 1 | 130 | 110 |
| TUT5 | 442 | 975 | 85m × 145m | 12.325m$^2$ | 3 | 1 | 446 | 1 |
| TUT6 | 3107 | 7237 | 135m × 62m | 8.775m$^2$ | 4 | 1 | 3116 | 1 |
| UTS1 | 9108 | 388 | – | 44 000 m$^2$ | 1 | 1 | 1466 | 3 |
| SAH1 | 9274 | 156 | – | 4184 m$^2$ | 1 | 1 | 9291 | 1 |
| MINT1 | 4973 | 810 | 50 m×20 m | 1000 m$^2$ | 1 | 1 | 189 | 19 |
| UJI1 | 19 369 | 1111 | | 108.702m$^2$ | 4 to 5 | 3 | 933 | 20 |

Since the datasets are so varied among themselves it means that they attest for how the algorithms can handle multiple diverse scenarios.The experiments were run in a computer with Intel® Core™ i7-4710MQ CPU

$2.50\,\mathrm{GHz}$ and Python 3.9.4, on an Ubuntu environment, all of the datasets were stored in CSV files, the $K$-Means algorithm was implemented via the sklearn library [1].

## 5.3 HYPERPARAMETERS

Using the full radio map or a reduced radio map both depend on the $k$-NN algorithm, so the first initial step is to find the optimal value of $k$ for the $k$-NN algorithm in each dataset.

To actual obtain the results the methodology is to estimate the positions using the Plain $k$-NN, with multiple $k$ values and select the one that obtains the best APE, in other words using brute-force to find the ideal $k$ value.

After obtaining the best $k$ value for $k$-NN, we apply the same procedure to $K$-Means, SAS and BSC, by brute-forcing their $K$, $N$, $P$ and $T$ values to find the most suitable combinations for each algorithm.

Brute-force is not the most efficient way to find the optimal values, but when dealing with new algorithms, like SAS and BSC it offers robustness to the results. We have to evaluate all possible combinations and then select the one providing the best results.

In the earliest stages of the work, two methods were used in SAS to find the ideal $N$ and $P$ combination, by working with the cluster and determining if they were good or bad clusters. The methods used were the Calinski-Harabasz Index (Variance Ratio Criterion) Caliński and Harabasz (1974) and the Davies-Bouldin Index Davies and Bouldin (1979). For Calinski-Harabasz Index, the score is defined as the ratio between the within-cluster dispersion and the between-cluster dispersion. The bigger the index, the better. The Davies-Bouldin Index is defined as the average similarity measure of each cluster with its most similar cluster. The similarity is the ratio of within-cluster distances to between-cluster distances. In this way, clusters which are farther apart and less dispersed will lead to a better score. Lower values mean better clustering performance.

Both these metrics would bring some insight about how good the clusters were. However, the results were not conclusive to a point where we could easily select the best hyperparameters combination. Usually, there would be a set of combinations that had good scores, having the best combination in terms of APE and computational cost among them, but the best score was not attributed to the best combination, and therefore, the brute-force method was implemented to be certain we were selecting the best possible combinations in terms of APE and computational cost, inside a given interval.

After defining the best hyperparameters for each algorithm and dataset, we compare the best results on each dataset with all the algorithms in accordance with the metrics defined above.

### 5.3.1 *DSI1 Hyperparameter selection*

#### *k-NN*

The graphs 14 show the results of varying the value of $k$ for the $k$-NN in terms of matching time and Averaged Positioning Error (APE).

1 https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

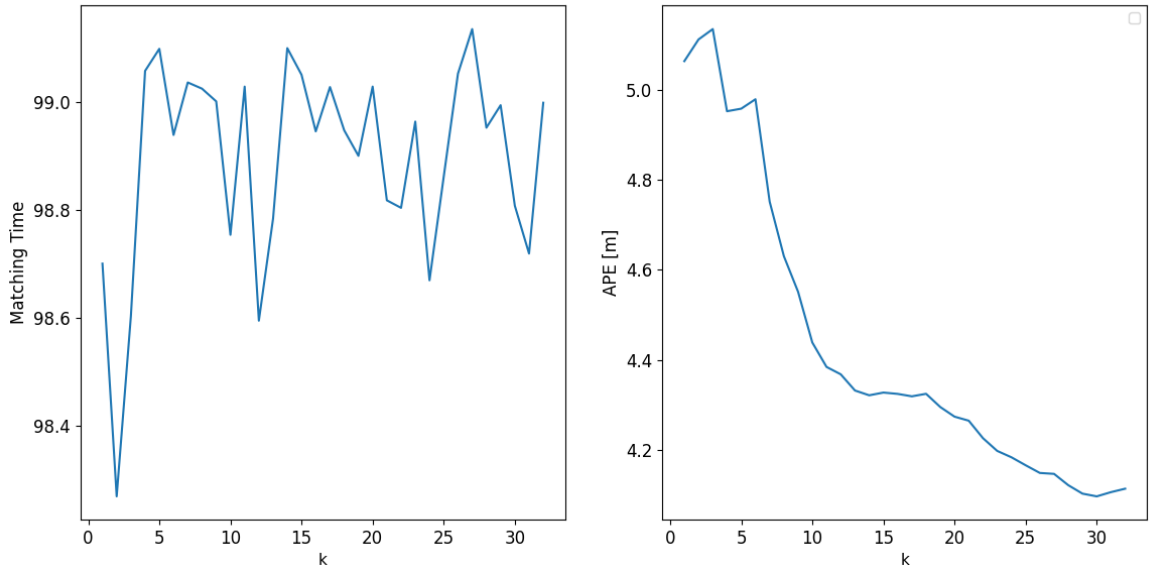



Figure 14: DSI1 $k$-NN parameter

In relation to the $k$-NN results, we can see that the computational cost is not that affected, always staying inside the $98\,\mathrm{ms}$ to $100\,\mathrm{ms}$, and we can see that the APE curve has a steep drop and starts converging as the values of $k$ increases.

*$K$-Means Clustering*

The graphs in Fig. 15 show the result of varying the value of$K$ for the $K$-Means in terms of the number of clusters, APE in meters, the Cluster Identification time in milliseconds and the Matching Time also in milliseconds on the y-axis (From top left to bottom right) and the value of $k$.

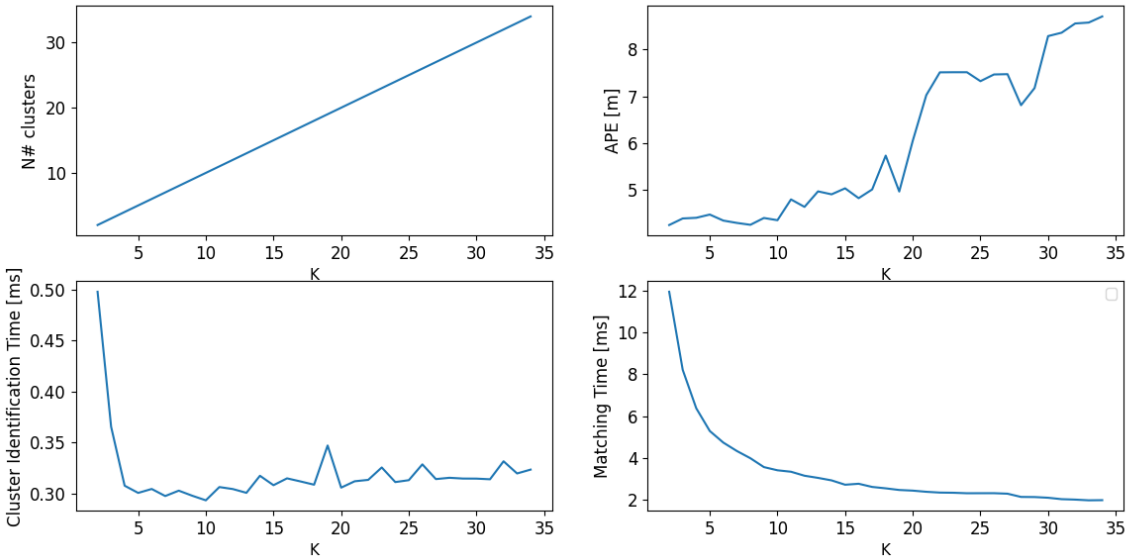



Figure 15: DSI1 $K$-Means hyperparameters

The number of clusters shows the $K$ in relation to the number of clusters, being $K$ representing the number of clusters in $K$-Means, so we end up with the line. In relation to the APE, we can see that the bigger the $K$ the bigger the error, therefore in terms of APE a smaller $K$, $K \in [1, \ldots, 10]$ is preferred for this dataset.

In relation to the Cluster Identification Time, as $K$ increases the time decreases for the first two values ($K = 2$ and $K = 3$). Then, it converges and starts to slightly increase as expected. This behavior is probably due to the library using some thread processing.

The Matching Time is the actual opposite of this, the bigger the $K$ the less computational cost, this is due to the $K$-Means clusters being split into Voronoi cells, meaning the cluster on average will be smaller. However, as we cannot control how the samples are distributed over the clusters, we may end up in having unbalanced clusters.

Anyway, these graphs show us the classic trade-off, where we can sacrifice the APE in order to obtain a better computational cost, and vice-versa.

*SAS Clustering*

The graphs shown in Fig. 16 show the result of varying the $N$ and $P$ for SAS.

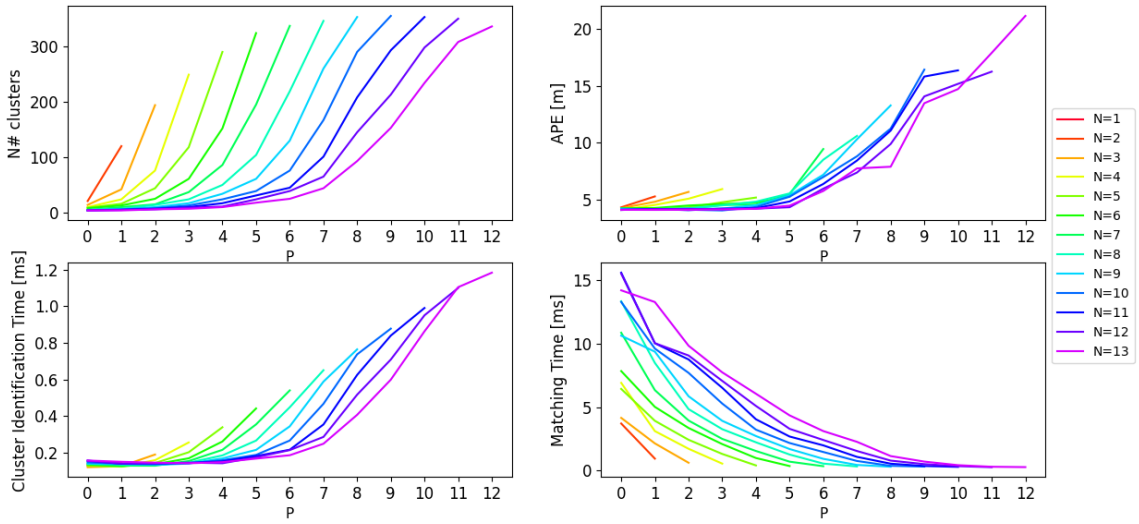



Figure 16: DSI1 SAS hyperparameters

The structure is similar to the $K$-Means ones, but the $x$-axis in the plots corresponds to values of $P$ and each line corresponds to a value of $N$. Concerning the number of clusters, it is clear the larger the $N$ and $P$ the more cluster we end up with, this is due to the cluster being more restrictive, consequentially creating more and smaller clusters.

Concerning the APE, the larger the $N$ and $P$, the larger the error. Also, is important to notice that the ideal APE is achieved when $P$ is small in comparison to $N$. Regarding the Cluster Identification Time, we can see that it scales the larger the $N$ and $P$, this is related to the graph above it, which shows that the larger $N$ and $P$ the more clusters we have in the end, resulting in a larger Cluster Identification Time. The last graph, Matching Time, shows the opposite trends with respect to the previous one. This is expected as the number of clusters is higher and, therefore, we end up with smaller clusters, which in turn reduces the execution time.

*BSC Clustering*

The graphs in Fig.17 shows the result of varying the $N$ and $T$ for the BSC.

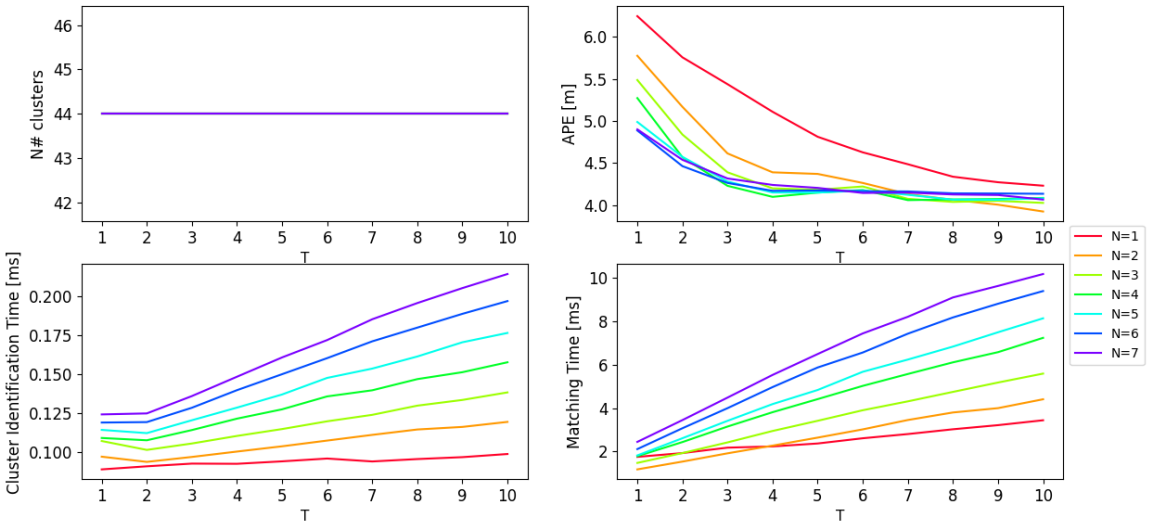



Figure 17: DSI1 BSC hyperparameters

The number of clusters generated is always the same, the only change by altering $N$ and $T$ is the actual size of the clusters. In terms of the APE, we can observe that the larger the $N$ and $T$ the smaller the APE becomes.

The Cluster Identification time is always below $0.23\,\mathrm{ms}$ and does not suffer any drastic changes. The matching time starts increasing with $N$ and $T$, which is justified since, we have the same number of clusters, but they have more samples inside each one of them.

*Discussion about hyperparameters selection for DSI1*

The ideal combination of hyperparameters for all solutions –plain $k$-NN, $K$-Means, SAS and BSC– was based on the pre-established metrics APE and Matching time.

### 5.3.2 *Resume*

The same procedure to select the hyperparameters for DSI dataset was followed to select the best combinations for the remaining datasets. For the sake of clarity, the results can be consulted in Appendix iii.

Table 5 shows the final hyperparameters selected for each dataset and positioning/clustering model.

Table 5: Hyperparameters for the executed models

| | | DSI1 | LIB1 | LIB2 | MAN1 | MINT1 | UJI1 | SAH1 | TUT5 | TUT6 | UTS1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|\mathcal{T}\|$ | 1369 | 3120 | 3120 | 14300 | 4973 | 19369 | 9274 | 442 | 3107 | 9108 |
| $k$-NN | $k$ | 30 | 11 | 24 | 14 | 12 | 5 | 4 | 4 | 1 | 24 |
| $K$-Means | $K$ | 2 | 26 | 30 | 3 | 31 | 19 | 5 | 32 | 2 | 30 |
| SAS | $N$ | 10 | 7 | 6 | 6 | 4 | 3 | 3 | 18 | 10 | 10 |
| | $P$ | 3 | 3 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| | #$C$ | 13 | 28 | 7 | 6 | 5 | 71 | 108 | 10 | 30 | 30 |
| BSC | $N$ | 2 | 8 | 1 | 4 | 1 | 2 | 1 | 3 | 7 | 4 |
| | $T$ | 10 | 7 | 6 | 1 | 1 | 3 | 7 | 14 | 5 | 5 |
| | #$C$ | 44 | 35 | 33 | 16 | 11 | 247 | 156 | 178 | 280 | 328 |

For SAS it is worth noting that all datasets report similar plots, and the best trade-off between positioning error and Matching time happens when $P \leq \frac{N}{2}$ in all datasets. By analyzing the data we can conclude that selecting the best $N$ and $P$ combination is challenging. $P$ must be small in comparison to $N$ to achieve the lowest positioning errors, but these solutions will not be the best in terms of computational cost. Selecting the optimal combination of hyperparameters requires finding a middle ground between this trade-off.

Concerning BSC its more difficult to create a relation between $N$ and $T$, since the relation between them fluctuates so much, in some cases $N$ being bigger, in others $T$ is the largest. In some cases, they are very close, being the same in dataset MINT1. In contrast, in some cases they are very apart, like in TUT5 where $N = 3$ and $T = 14$. And of the justifications for this difference compared to SAS is that $P$ was dependent on $N$, but for BSC $T$ and $N$ are not related as they are used in different stages, which makes the task of reaching a relation between both more difficult.

# 6

# RESULTS

In this section, we provide and analyse the empirical results obtained. We compare the results of the plain $k$-NN model, $K$-Means clustering, Strongest AP Set (SAS) and Base Station Clustering (BSC).

First, we provide an assessment of the SAS variants introduced in Chapter 3. Second, we compare the results provided by the proposed clustering models to plain $k$-NN and $K$-Means clustering. Third, we provide an analysis of the large errors by focusing on a few samples providing an extremely large positioning error.

## 6.1 ASSESSMENT OF THE SAS VARIANTS

One of the first steps is to decide on which SAS variant to use. To do so, we evaluate the performance, in terms of positioning error and computational time, of all of the variants only in the UJI1 dataset. We selected this dataset as it is a reference dataset in the literature and it was collected in a multi-building and multi-floor environment by means of different users and smartphones. In addition, for baseline comparison, we also provide the performance of the Plain $k$-NN. The results, according to the metrics defined in Section 5.1, are shown in Table 6, whereas Figure 18 introduces the CDF plot for the positioning error of Plain $k$-NN and all SAS variants. 18.

Table 6: Main results: SAS Variants Comparision.

| | APE[m] | CT[s] | CId[ms] | Matching[ms] | Min Error[m] | Max Error[m] |
|---|---|---|---|---|---|---|
| Plain $k$-NN | 8.84 | 0 | 0 | 612.42 | **0** | 208.99 |
| SAS | **8.21** | **3.47** | **0.56** | 30.0 | **0** | 147.63 |
| 3C | 9.18 | 4.83 | 2.48 | **21.54** | **0** | 177.50 |
| RCR | 8.25 | 3.78 | 0.59 | 41.73 | **0** | **88.05** |
| Heavy | 8.31 | 5.00 | 0.61 | 33.19 | **0** | 92.71 |

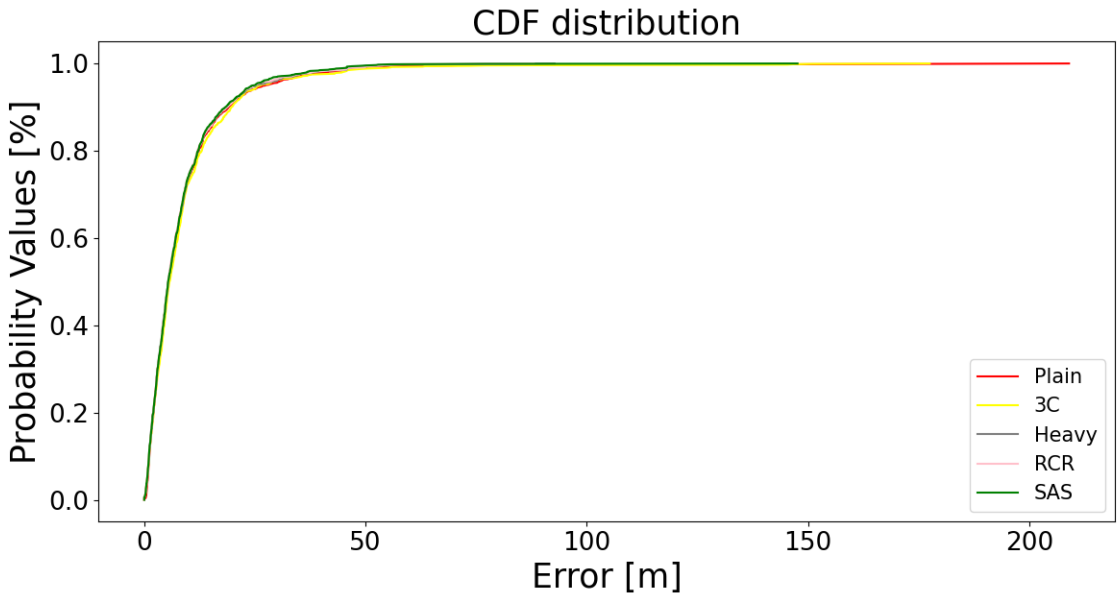



Figure 18: CDF plot for SAS Variants on UJI

By analysing the results, the classic SAS is providing good results in the main evaluation metrics. It is providing the best averaged positioning error –$8.21\,\mathrm{m}$–, being $63\,\mathrm{cm}$ better than the Plain $K$-NN. Also, it provides excellent results for the matching time –$30.0\,\mathrm{ms}$–, being $20$ times faster than Plain $k$-NN. In contrast, the 3C variant is the fastest one with a matching time $21.54\,\mathrm{ms}$, but this gain in efficiency is at the expense of a much higher averaged positioning error, which is $9.18\,\mathrm{m}$. The RCR variant is similar to the classic one, having a much lower maximum error but a slightly worse matching time. Similarly, the Heavy variant is providing similar results. Comparing RCR and Heavy, the former is providing better positioning errors, whereas the latter is providing better computational costs.

Regarding the CDF, the classic SAS (the green line) is reporting the best behaviour. Despite all algorithms providing similar results until percentile $80$, we see moderate/large differences with respect to plain $k$-NN in the highest percentile values. Our approach is not only reducing the computational costs but also improving the error in extremely large cases. We consider that this information may be useful in further developments as, for instance, detecting when a position estimate may incur in a large positionig error.

In order to visualise better the trade-off between positioning error and computational cost, we provide a visual representation in Figs. 19–20. Fig. 19 contains a scatter plot with absolute values, whereas Fig. 20 contains the values normalized by the Plain $k$-NN results.

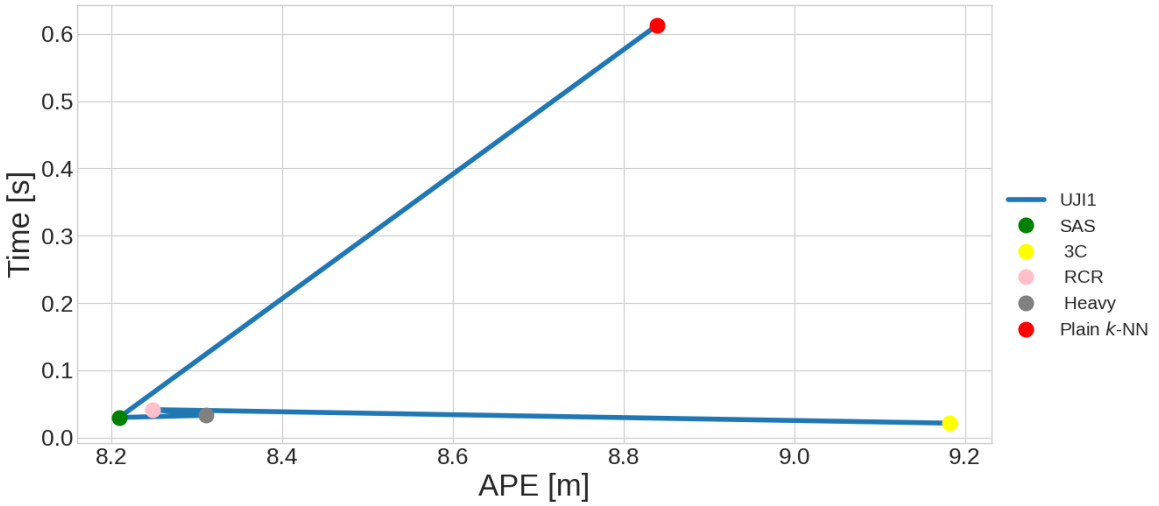



Figure 19: Absolute values graph for SAS Variants on UJI

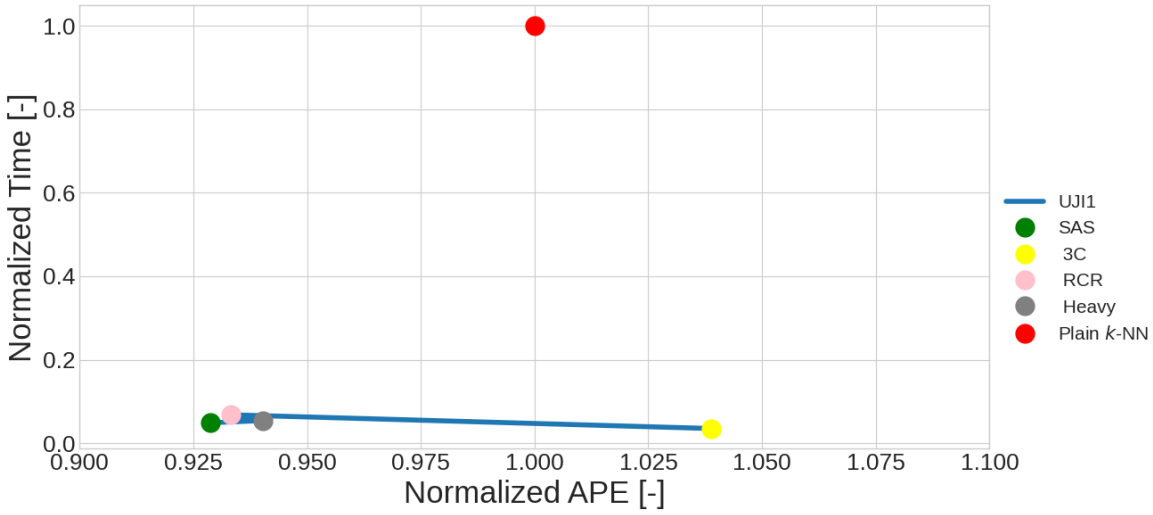



Figure 20: Relative values graph for SAS Variants on UJI

Both figures clearly show that the classic SAS is the best alternative for UJI1 dataset, being Heavy and RCR quite similar. The classic SAS is significantly improving the $k$-NN model in both dimensions, but specially on the computational costs. In the remaining of this Thesis, we will refer as SAS to the classic SAS, discarding other variants.

## 6.2 CLUSTERING RESULTS

### 6.2.1 *Full Results*

The final results for SAS, BSC, $K$-Means and Plain $k$-NN are presented in this section.Tables 7–12 present the main results in terms of Averaged Positioning Error (APE), clustering time, averaged cluster identification time and averaged matching time, respectively.

Table 7: Full Results: Average Positioning Error [m].

| | DSI1 | LIB1 | LIB2 | MAN1 | MINT1 | UJI1 | SAH1 | TUT5 | TUT6 | UTS1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Plain $k$-NN | 4.10 | 2.44 | 2.92 | 2.25 | 2.50 | 8.84 | 6.25 | 6.25 | 2.08 | 7.51 |
| SAS | 4.05 | 2.36 | 2.78 | **2.24** | 2.48 | 8.21 | **5.87** | 6.22 | **2.04** | **7.14** |
| BSC | **3.92** | 2.44 | **2.48** | 2.29 | **2.17** | **7.89** | 6.44 | **6.12** | 2.06 | 7.32 |
| $K$-Means | 4.25 | **2.33** | 2.81 | **2.24** | 2.45 | 8.87 | 6.04 | 6.45 | 2.09 | 7.41 |

In relation to the APE, Table 7 it is clear the advantage of clustering the radio map, in all of the datasets the best results are always obtained by the clustering approaches, not only that but SAS and BSC are providing the best overall results.

Table 8: Full Results: Clustering Time [s].

| | DSI1 | LIB1 | LIB2 | MAN1 | MINT1 | UJI1 | SAH1 | TUT5 | TUT6 | UTS1 |
|---|---|---|---|---|---|---|---|---|---|---|
| SAS | 0.09 | 0.52 | 0.18 | 0.27 | 0.08 | 3.47 | 2.77 | 0.09 | 0.75 | 2.26 |
| BSC | 64.61 | 527.75 | 405.22 | 5283.11 | 415.63 | 5038.03 | 3030.45 | 2.64 | 104.77 | 2133.88 |
| $K$-Means | 0.54 | 0.87 | 0.96 | 2.65 | 4.49 | 6.72 | 2.93 | 0.17 | 0.74 | 0.57 |

Regarding the Clustering time, Table 8, plain $k$-NN does not run this step, so only $K$-Means, BSC and SAS are assessed. SAS and $K$-Means provide acceptable costs for the clustering stage, but BSC does not offer the same type of results and shows that it is unscalable for large datasets.

Table 9: Full Results: Cluster Identification Time [ms].

| | DSI1 | LIB1 | LIB2 | MAN1 | MINT1 | UJI1 | SAH1 | TUT5 | TUT6 | UTS1 |
|---|---|---|---|---|---|---|---|---|---|---|
| SAS | 0.14 | 0.29 | 0.18 | 0.54 | 0.16 | 0.56 | 0.85 | 0.32 | 0.48 | 0.61 |
| BSC | 0.12 | 0.59 | 0.14 | 0.37 | 0.07 | 0.27 | 0.46 | 0.29 | 0.44 | 0.43 |
| $K$-Means | 0.50 | 0.30 | 0.30 | 0.40 | 0.28 | 0.50 | 0.57 | 0.42 | 2.93 | 0.52 |

Concerning Table 9, the Cluster Identification time, all the results are low values and do not surpass the mark of 3 milliseconds.

Table 10: Full Results: Matching Time [ms].

| | DSI1 | LIB1 | LIB2 | MAN1 | MINT1 | UJI1 | SAH1 | TUT5 | TUT6 | UTS1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Plain $k$-NN | 98.81 | 48.24 | 46.65 | 130.85 | 36.70 | 612.42 | 482.46 | 15.29 | 120.83 | 349.90 |
| SAS | 5.42 | 7.15 | 18.38 | 76.72 | 18.52 | 30.00 | 19.60 | 4.65 | 16.09 | 53.07 |
| BSC | 4.41 | 32.41 | 8.75 | 22.12 | 3.50 | 6.21 | 48.96 | 4.96 | 16.01 | 22.50 |
| $K$-Means | 11.95 | 3.16 | 2.96 | 44.01 | 2.39 | 51.24 | 147.54 | 1.73 | 82.18 | 20.39 |

The matching time, Table 10, in every dataset is better for the clustering solutions, always providing a lower computational cost to obtain a position estimate for a fingerprint, thus backing this sort of approach.

Table 11: Full Results: Maximum Positioning Error [m].

| | DSI1 | LIB1 | LIB2 | MAN1 | MINT1 | UJI1 | SAH1 | TUT5 | TUT6 | UTS1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Plain $k$-NN | **12.11** | **7.80** | 9.01 | **8.39** | **9.13** | 208.99 | 17.25 | **22.81** | 78.77 | 41.84 |
| SAS | 13.17 | **7.80** | 9.01 | **8.39** | **9.13** | 147.63 | 23.43 | **22.81** | **57.37** | 40.68 |
| BSC | 19.95 | **7.80** | **8.16** | 8.56 | 12.68 | **146.55** | 23.72 | 26.12 | 78.77 | **35.80** |
| $K$-Means | 20.29 | **7.80** | 8.46 | **8.39** | 12.68 | 208.99 | **16.35** | 26.57 | 78.77 | 41.84 |

Focusing on the maximum error, Table 11, we can see that in some cases the clustering algorithms offer worse results than the actual Plain $k$-NN in some datasets, the biggest difference being in the DSI1 dataset with $K$-Means increasing the error by $8.18\,\mathrm{m}$, however, in other cases the clustering algorithms provide a great improvement over the Plain $k$-NN, as in the UJI1 dataset where we have improvements over $50\,\mathrm{m}$ for both SAS and BSC.

Table 12: Full Results: Minimum Positioning Error [m].

| | DSI1 | LIB1 | LIB2 | MAN1 | MINT1 | UJI1 | SAH1 | TUT5 | TUT6 | UTS1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Plain $k$-NN | **0.18** | 0.17 | 0.16 | **0.02** | **0** | **0** | 0.61 | **0.17** | **0.04** | **0.20** |
| SAS | **0.18** | 0.17 | 0.14 | **0.02** | **0** | **0** | **0.48** | **0.17** | **0.04** | **0.20** |
| BSC | 0.29 | 0.17 | **0.08** | 0.04 | **0** | **0** | 0.61 | **0.17** | **0.04** | 0.48 |
| $K$-Means | **0.18** | **0.09** | 0.24 | **0.02** | **0** | **0** | 0.61 | **0.17** | **0.04** | **0.20** |

Regarding the minimum error, Table 12 all the results are low in terms of error and acceptable for both solutions.

The results show that the clustering solutions are superior to the Plain $k$-NN results, the problem is which one to select for each dataset and if they are all equally good or if some stands out from the others.

All of the results will be further analyzed by comparing the algorithms in pairs, over the next sections, to attempt an answer these problems.

### 6.2.2 *Detailed comparison between SAS and KMEANS*

First, the results show the diversity of datasets in terms of APE, Table 7, with errors ranging from $2.04\,\mathrm{m}$ to $8.81\,\mathrm{m}$. $K$-Means is providing slightly worse results than the plain $k$-NN in four datasets and slightly better

results on the remaining six. $K$-Means splits the radio map into Voronoi cells, so fingerprints near the cluster boundaries may have less information available and, therefore, worse results. In contrast, SAS is always providing the best results than plain $k$-NN and $K$-Means, with just two cases where it is outperformed by $K$-Means.

It is worth mentioning the outstanding performance provided by SAS in the challenging datasets UJI1, UTS1, SAH1, and TUT5, where the APE provided by SAS is significantly lower than plain $k$-NN and/or $K$-Means.

Concerning clustering time, Table 8, SAS provides better results than $K$-Means in all datasets, except for TUT6, where they have a difference of $0.01\,\mathrm{s}$ and in UTS1, where $K$-Means is the winner. In contrast to Affinity Propagation Clustering (APC), which requires tight memory and computation resources Torres-Sospedra et al. (2020), SAS scales to large datasets.

In relation to the averaged cluster identification time, Table 9, plain $k$-NN also does not run this step. $K$-Means and SAS are providing similar results throughout the datasets. The identification time depends on the dataset, being in the worst case $3\,\mathrm{ms}$. An important aspect is that despite the time being lower for $K$-Means, this trend is the opposite in the smallest datasets.

Regarding the matching time, Table 10, the plain $k$-NN does not scale. In the largest dataset (UJI1), it requires more than $0.6\,\mathrm{s}$ to provide a position estimate. SAS is providing a matching time below $53\,\mathrm{ms}$ in all datasets, except MAN1. MAN1 is the dataset with the highest density of samples with 110 fingerprints per reference point and many samples will be located in the sub-region dominated by the set of strongest APs. In the largest datasets (UJI1, SAH1, TUT6), with the exception of UTS1, the computational cost of SAS is not only satisfactory but better than $K$-Means. It seems that SAS is promising for large operational areas.

Regarding the maximum error, Table 11, there are some cases, DSI and SAH1, where SAS actual obtains a worse error than the Plain $k$-NN, but in the scenario where the error is the largest, UJI1, SAS provides an improvement of $51.36\,\mathrm{m}$ compares to Plain $k$-NN and $K$-Means. $K$-Means provides better results than SAS in 2 datasets, LIB2 and SAH1, they tie in 2 other datasets, LIB1 and MAN1, and in the remaining 6 SAS always provides better results than $K$-Means.

In relation to the minimum error, Table 12, the results are low in all datasets except SAH1, always bellow $0.21\,\mathrm{m}$ for SAS and Plain $k$-NN, and bellow $0.24\,\mathrm{m}$ for $K$-Means, and in some cases even being nonexistent ( error = $0\,\mathrm{m}$), but in the SAH1 dataset we get minimum errors of about half a meter, $0.48\,\mathrm{m}$ for SAS and $0.61\,\mathrm{m}$ for the Plain $k$-NN and $K$-Means.

According to all presented results, it seems that $K$-Means is more efficient than SAS in terms of computational cost in the operational phase. This is in part due to the cluster identification in SAS being more sophisticated. SAS generates clusters that may overlap and multiple clusters can be assigned as the "*most similar cluster*" in the operational phase of SAS. Still, the computational cost of SAS is significantly lower with respect to plain $k$-NN, especially in the datasets covering very large areas. In contrast, SAS is providing better accuracy than $K$-Means, and always improving in relation to plain $k$-NN.

In order to analyze the trade-off between the computational costs and positioning error, we provide a scatter plot in Fig.21 (top), where Averaged Positioning Error (APE) and Averaged Execution Time (AET) are compared for each dataset. Averaged Execution Time (AET) is the cluster identification time plus the matching time. As the datasets are diverse, it is hard to see a pattern. Thus, both metrics are normalized with respect to a baseline for

each dataset resulting in the scatter plot shown in Fig.21 (bottom) with relative values with respect to the plain $k$-NN.

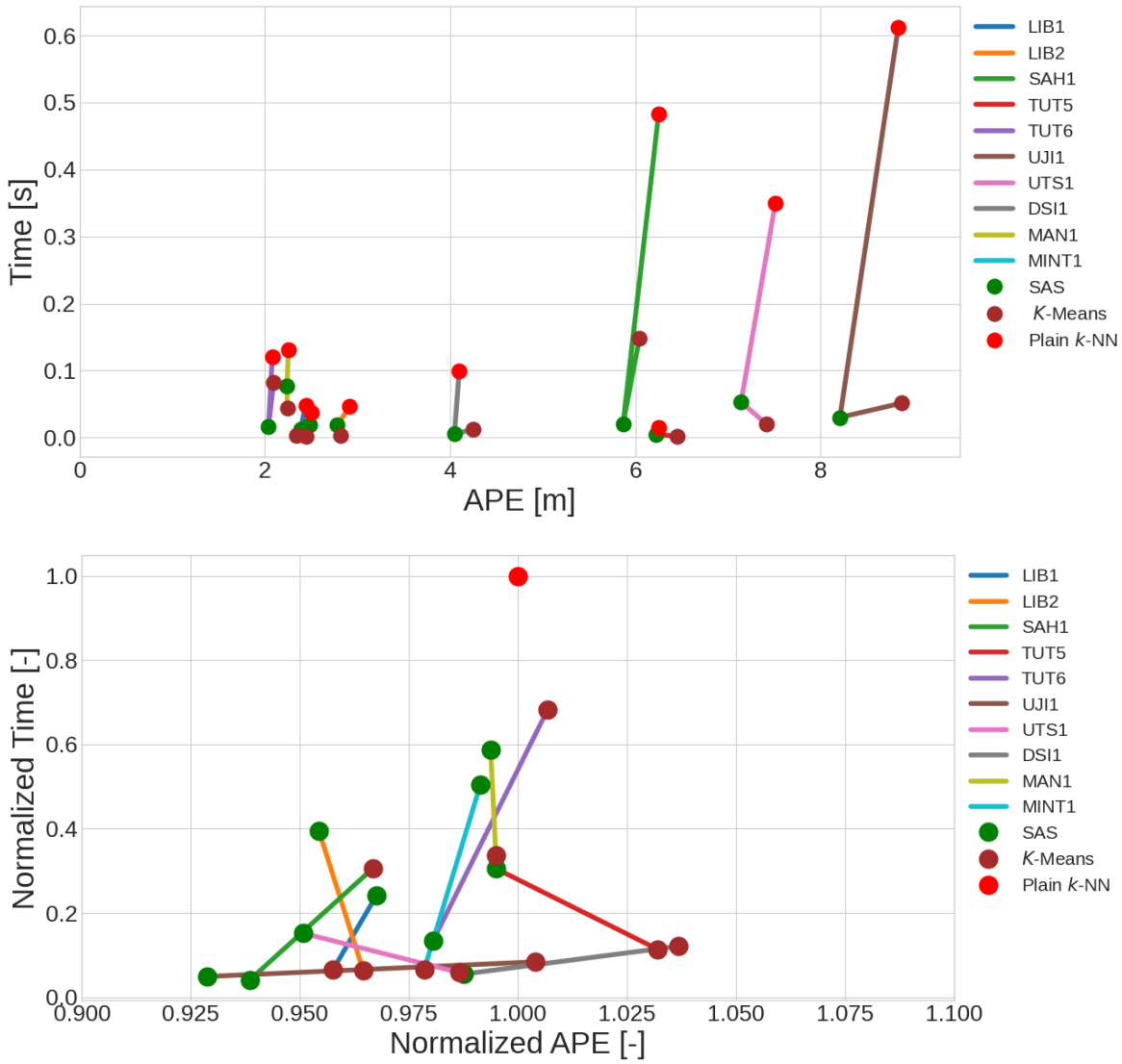



Figure 21: Efficiency vs. Accuracy: absolute values (top), relative values (bottom)

Fig. 21 identifies 3 scenarios, the first one is where SAS is worse than $K$-Means both in terms of average error and in terms of computational cost, and this happens for the datasets LIB1, MAN1 and MINT1. Should be noted that even if SAS is losing, it still has an improvement in the average error with respect to the plain $k$-NN. The second scenario is when $K$-Means has a better computational cost, but SAS has a better average error, and this is can be observed in the datasets LIB2, TUT5 and UTS1. The third scenario is when SAS beats $K$-Means in both fields, and this can happens in the datasets that cover large areas (UJI1, SAH1 and TUT6) and DSI1. In particular, Fig. 21 shows 3 main outputs.

1. $K$-Means is computationally more efficient than SAS in the operational stage at the expense of having slightly worse performance (APE) than the plain $k$-NN.

2. SAS provides a good efficiency/accuracy trade-off, giving good efficiency without sacrificing accuracy.

3. SAS has not only reduced the computational cost to a minimum expression in the two largest datasets, but also the positioning error has been significantly decreased in around $7.5$–$12.5\%$. It seems that SAS is a promising method for datasets involving large operational areas.

### 6.2.3 *Detailed comparison between BSC and KMEANS*

This section focuses only on the BSC and $K$-Means results regarding the previously established metrics.

Table 7 present the APE results, and BSC has very unstable results, in the UJI1 dataset it improves the error by 0.95 meters and in LIB2 it improves by 0.44 meters, which are significant error improvements, however, there are datasets where the error actual has a penalty for BSC, like in SAH1 which has a penalty of $0.19\,\mathrm{m}$ or in MAN1 it a penalty of $0.04\,\mathrm{m}$. When compared to $K$-Means, in most cases BSC is the winner, in 7 out of 10 datasets also beating the Plain $k$-NN results in these datasets, and for LIB1 BSC is providing worse results than $K$-Means, but the same as Plain $k$-NN, and for the last 2 datasets, MAN1 and SAH1, BSC is worse than both algorithms.

Regarding the Clustering Time, Table 8 show the results, and there is no competition, BSC is not scaling with the datasets, and it has too much of a computational cost compared to $K$-Means. BSC clearly fails on this step compared to $K$-Means, even if in the smaller datasets the computational cost may be acceptable, in the largest dataset it is to expensive, like in UJI1, SAH1, UTS1 and MAN1.

Concerning the Cluster Identification Time, Table 9, BSC provides good results on pair with $K$-Means, always bellow 0.6 milliseconds, and for the TUT6 providing an improvement from 2,93 milliseconds to 0.44.

Focusing on Table 10, Matching Time, there are some datasets where BSC prevails and others where $K$-Means surpasses it. In DSI1, MAN1, UJI1, SAH1 and TUT6 BSC is the winner, especially in the UJI1, SAH1 and TUT6, where there is a considerable drop in the Matching Time even when compared to $K$-Means, UJ1 goes from 51.24 milliseconds in $K$-Means to 6.21 ith BSC, for SAH1 it goes from 147.54 to 48.96, and in TUT6 from 82.18 to 16.01. In the remaining cases, $K$-Means provides better results, but even so, the most far apart case is in the LIB1, $K$-Means with 3.16 milliseconds, and BSC with 32.41 milliseconds, a significant increase, but nothing like the ones where BSC is the best algorithm.

Relating to the Maximum positioning error, Table 11, the results fluctuate a bit, in the UJI1 dataset we have an improvement of 62.44 meters for BSC and for $K$-Means has the same error as Plain $k$-NN, but in DSI1, MAN1, MINT1, SAH1 and TUT5 BSC is providing worse results than the Plain $k$-NN, not by a difference as big as UJI1, but still worse than Plain $k$-NN. $K$-Means is more reliable in this metric, only being worse than Plain $k$-NN in the datasets DSI1, Mint1 and TUT5, not having an incredible improvement on the other datasets, like BSC in UJI1, but providing results that are the same as Plain $k$-NN or even better.

Regarding the Minimum positioning error, shown in Table 12, the results for both, $K$-Means and BSC are overall close to the Plain $k$-NN results, with the biggest difference of $0.28\,\mathrm{m}$ in the UTS1 dataset, where BSC obtains an error of 0.48 meters, while $K$-Means and Plain $k$-NN have an error of 0.20 meters.

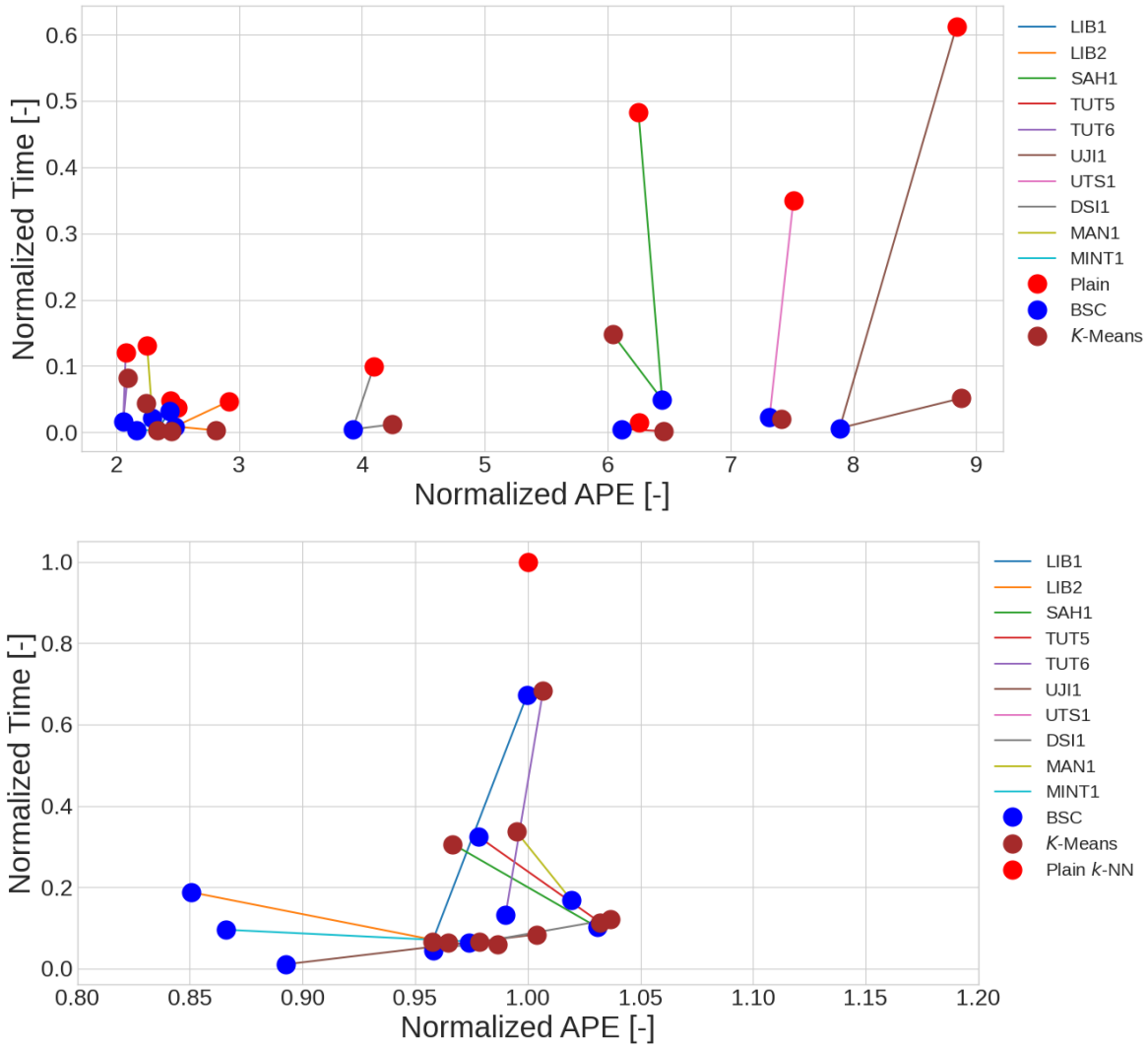



Figure 22: Efficiency vs. Accuracy: absolute values (top), relative values (bottom)

Fig. 22 identifies 4 scenarios, the first one where BSC is better than $K$-Means in both APE and Matching Time, on the datasets UJI1, TUT5 and DSI. The second scenario is the opposite, where $K$-Means ans surpasses BSC in the APE and Matching Time, this happens on the datasets LIB1 and SAH1. The third scenario is when BSC provides a betterAPE but $K$-Means is more computationally efficient, this occours in the datasets MINT1, LIB2, TUT5 and UTS1. The last scenario is when $K$-Means offers a better APE but BSC is more efficient and this happens in the MAN1 dataset. And there are four main output:

1. BSC is clearly inefficient in the offline phase, the clustering stage, compared to $K$-Means.

2. BSC offers good APE results, even in the worst case, SAH1 it suffers a considerable penalty but the drop in Matching Time obtained is significant, and its the only case with a considerable penalty.

3. BSC provides good Matching time in the biggest datasets.

4. BSC can provide great error improvements in some datasets.

### 6.2.4 *Detailed comparison between SAS and BSC*

This section compares the SAS and BSC results over the defined metrics.

Regarding Table 7, the APE, SAS offers more reliable results always having an improvement compared to the Plain $k$-NN results, while BSC has more unpredictable behaviour, sometimes providing very good results, like in UJI1, and sometime providing penalties in the APE, like in the SAH1 dataset.

BSC is providing the best results in UJI1, improving by almost 1 meter in errors, but SAS also provides a good improvement, of about 0.6 meters, and the same goes for LIB2 and MINT1, but in both of these cases SAS also provides a sligth error improvement, while in cases like SAH1 and MAN1, SAS is providing error improvements, while BSC offer a penalty.

In relation to the Cluster Identification Time, Table 9, this results are almost similar, we have to consider were talking about milliseconds, and the biggest difference between 2 results is in the dataset SAH1 with a difference of 0.39 milliseconds.

Regarding the Clustering time, Table 8, there is no doubt that SAS is the winner, and that BSC is quite inefficient and unscalable for this stage.

Focusing on the Matching Time, Table 10, both algortihms provide good results, but BSC is better in 7 out of 10 dataset, including UJ1, UTS1 and TUT6 (large datasets), being almost the same in the TUT5 dataset, and being worse in the LIB1 and SAH1 datasets.

Concerning the maximum error, Table 11, SAS is sligthly better than BSC the results are very close to each other with the exception of the TUT6 dataset where SAS obtains a result of 57.37 meters and BSC has a maximum error of 78.77 meters, having a difference of 21.4 meters.

Regarding the minimum positioning error, Table 12, the results are also very close, being the biggest difference in the UTS1 dataset, with a 0.28 meters deifference, in favor of SAS.

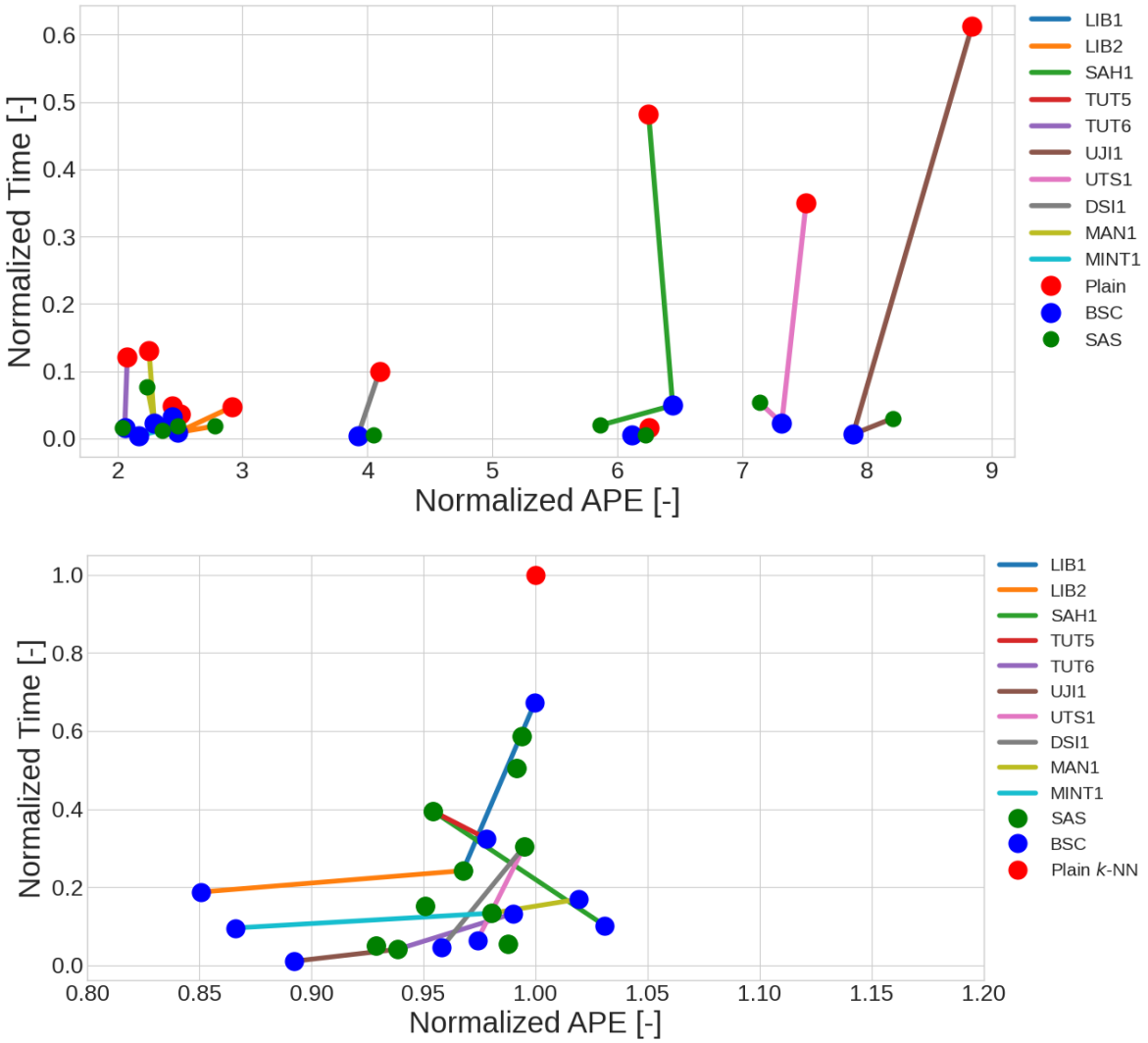



Figure 23: Efficiency vs. Accuracy: absolute values (top), relative values (bottom)

Fig. 23 shows us 4 scenarios, the first one is when SAS has a better APE and computational cost, and this happens in 2 datasets, LIB1 and SAH1. The second scenario is when BSC beats SAS in both metrics, and this happens on 4 datasets, DSI, LIB2, MINT1 and UJI. The third scenario is when SAS beats BSC in terms of APE but loses in computational efficiency, and this happens in 3 datasets, MAN1, TUT6 and UTS1. And the last scenario is the reverse of the previous one, when BSC beats SAS in terms of APE but loses in computational cost, and this happen only in TUT5. And we can formulate 4 main outputs from our data:

1. SAS is more reliable in terms of APE, all the results are improvements upon the Plain $k$-NN.
2. BSC has the potential to greatly improve the APE while also improving its computational cost, like in the UJI1 dataset, but its less reliable.
3. BSC is extremely inefficient compared to SAS when it comes to the offline phase, clustering.

4. BSC is computationally more efficient than SAS in the operational phase.

## 6.3 ERROR INSPECTION

Both SAS and BSC provide overall good results in terms of APE, with BSC achieving the best errors improvements and SAS always improving its error compared to Plain $k$-NN. But there are still some large errors according to the CDF plots, Appendix iii, and maximum error tables, Table 11.

In order to determine the cause of these large errors, the 3 worst samples in terms of positioning error, for SAS, BSC and Plain $k$-NN were analyzed in more detail.

The idea is to understand why this large error exist, is it because of SAS and BSC select the wrong clusters or is it because we can not offer a good position estimate with the current $k$-NN algorithm.

Since a sample can be simultaneously the worst sample for more than one algorithm it means, we will be working with a number of samples $\leq 6$. In the end a total of 5 samples were analyzed, 2 of them are the worst cases for both BSC and Plain $k$-NN, 1 belong to all three algorithms, and the last 2 are only part of the SAS algorithm worst estimations. Table 13 shows the final samples ID, the algortihm for which they are the worst cases, the number of detected APs and the average RSSI of the detected APs.

Table 13: Worst samples estimations

| ID | Algorithms | # APs detected | AVG RSSI detected |
|---|---|---|---|
| 1557 | BSC, Plain $k$-NN | 1 | -86 |
| 539 | BSC, Plain $k$-NN | 2 | -73 |
| 3662 | SAS, BSC, Plain $k$-NN | 1 | -90 |
| 3747 | SAS | 10 | -81 |
| 3747 | SAS | 13 | -84 |

The procedure to analyze the samples is to go to each individual one and find out if the clusters SAS and BSC selected have other training samples that are geometrically closer to the samples in question. In total we have 10 cases, two for each sample, since we have 2 algorithms, SAS and BSC.

And for 9 out of the 10 cases the problem was related to the $k$-NN not being able to handle the undetected values of samples, and not due to cluster misidentification. In 9 cases, the problem was not caused by selecting the wrong cluster, since geometrically close training samples were part of that cluster, but rather that there are other training samples that are more close in the feature space to the sample, and therefore are selected even if they are geometrically farther apart than other samples.

So the problem arises from the $k$-NN not handling the undetected values, since it considers them as regular values, and samples with lot of undetected values end up being closer in the feature space.

The only case where there was a cluster misidentification was for SAS in sample 3662, since the 5 closest geometrical samples did not belong to that cluster, and also 3 of them have a lower feature distance than the sample selected by $k$-NN.

Results show that the majority of this errors come from the $k$-NN algorithm not having any knowledge on how to handle the undetected values, in the 5 final samples, only one case out of 10 is due to cluster misidentification.

## 6.4 DISCUSSION

By analysing the results, the clustering approach offer better results than the simple Plain $k$-NN solution. In every dataset they are computationally more efficient and at least one of the algorithms provides a better APE. The biggest problem is which algorithm to select. SAS is overall the most reliable one, significantly dropping the computational cost, and always providing better APE that Plain $k$-NN. $K$-Means offers a better efficiency in terms of computational cost, but giving in to the traditional trade off between time and error, sometime even having worse errors than Plain $k$-NN. BSC is the algorithm that has the biggest error improvements, while also having very efficient computational cost, however it is not scalable during the clustering stage, and not ideal for situations where the radio map is regenerated with frequency, and also it has cases where the error is actually worse than Plain $k$-NN, even if very close to it.

SAS end up being the more reliable algorithm, while BSC being able to provide great improvements the results are not as guaranteed as it is the case with SAS. SAS provides good APE with acceptable computational cost associated and it is scalable, BSC suffers a lot from the offline phase of clustering, and therefore cannot be used as widely as SAS, not being scalable to large dataset for the offline phase. $K$-Means is a decent alternative but not as good as SAS and BSC when correctly applied.

Another important take is that a more advanced version of the $k$-NN algorithm, or a different distance algorithm, may handle the undetected values problem, and provide more accurate results, however this added complexity may result in a bigger computational cost, which means the clustering algorithms will provide even more efficient results compared to the Plain $k$-NN.

# 7

# CONCLUSIONS AND FUTURE WORK

In this work, rwo new clustering algorithms that possess context awareness for Indoor Positioning were developed. Both were compared to the Plain $k$-NN and $K$-Means algorithms on a variety of real-world scenarios inside the Wi-Fi Fingerprint model. The methodology itself is described in Chapter 5 to allow the recreation of the experiments and future work comparison.

The results attest to the algorithms efficiency, reliability, improvement and flexibility. SAS and BSC are able to provide a good efficiency while also obtaining a good APE.

SAS always provides an improvement in terms of APE and acceptable computational cost, being scalable and reliable in terms of both computational cost and APE.

BSC offers great computational efficiency, and in some datasets exceptional improvement in the APE, however, there are cases where this error is degraded, even if it is a very small difference from the Plain $k$-NN. But the biggest drawback of BSC is the fact that it is not scalable during the clustering stage, meaning that scenarios that update their radio maps with high frequency are not ideal for this algorithm.

A more abstract and important outcome of this work, is that the results also show that sometimes is better to focus on trustworthy information (stronger APs) instead of focusing on the whole picture.

There is still some work to be done concerning the algorithms developed and some future research approaches. Creating a more efficient SAS version without prejudicing the APE can make this algorithm the number one choice. Also continuing to study BSC in order to create a more reliable version, on which we can always obtain an improvement in terms of APE without increasing the computational cost and likewise addressing the clustering stage high and unscalable computational cost. A more advanced $k$-NN algorithm can also help improve the error by handling the undetected values, and creating a more realistic relation between the feature space and geometric space. And lastly having a heavier data cleanse to get rid of noisy samples, or samples that only prejudice the error is also another approach to explore.

All the code developed during this work is available in github [1]. The SAS algorithm has been shared with the scientific community through the paper published in the conference VTC 2022 Fall.

1 https://github.com/moisesramires/SAS

# BIBLIOGRAPHY


Mai A. Al-Ammarz, Suheer Alhadhrami, AbdulMalik Al-Salman, Abdulrahman Alarifiy, Hend S. Al-Khalifa, Ahmad Alnafessahy, , and Mansour Alsalehy. Comparative survey of indoor positioning technologies, techniques, and algorithms. *2014 International Conference on Cyberworlds*, 2014.

Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A. Al-Ammar, and Hend S. Al-Khalifa. Ultra wideband indoor positioning technologies: Analysis and recent advances. *MDPI*, 2016.

Fares Alkhawaja, Mohammad Jaradat, and Lotfi Romdhane. Techniques of indoor positioning systems (ips): A survey. *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, 2019.

Alessio De Angelis, John Olof Nilsson, Isaac Skog, Peter Händel, and Paolo Carbone. Indoor positioning by ultrawide band radio aided inertial navigation. *MMS*, 2010.

Taiga Arai, Takahiro Yoshizawa, Takuya Aoki, Keiichi Zempo, and Yukihiko Okada. Evaluation of indoor positioning system based on attachable infrared beacons in metal shelf environment. *2019 IEEE International Conference on Consumer Electronics (ICCE)*, 2019.

Taner Arsan and Mohammed Muwafaq Noori Hameez. A clustering-based approach for improving the accuracy of uwb sensor-based indoor positioning system. *Mobile Information Systems*, 2019.

Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. *Microsoft Research*, 2000.

Sidong Bai and Tong Wu. Analysis of k-means algorithm on fingerprint based indoor localization system. *2013 5th IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, 2013.

Ramon F. Brena, Juan Pablo García-Vázquez, Carlos E. Galván-Tejada, David Muñoz-Rodriguez, Cesar Vargas-Rosales, , and James Fangmeyer Jr. Evolution of indoor positioning technologies: A survey. *Journal of Sensors*, 2017.

Willy Anugrah Cahyadi, Yeon Ho Chung, and Trio Adiono. Infrared indoor positioning using invisible beacon. *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, 2019.

T. Caliński and J Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974. doi: 10.1080/03610927408827101.

Wei Chen, Qiang Chang, Hong tao Hou, and Wei ping Wang. A novel clustering and kwnn-based strategy for wi-fi fingerprint indoor localization. *4th International Conference on Computer Science and Network Technology*, 2015.

Long Cheng, Zhaoqi Wu, Bo Lai, Qiang Yang, Anguo Zhao, and Yuanting Wang. Ultra wideband indoor positioning system based on artificial intelligence techniques. *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, 2020.

Omar Costilla-Reyes and Kamesh Namuduri. Dynamic wi-fi fingerprinting indoor positioning system. *2014 International Conference on Indoor Positioning and Indoor Navigation*, 2014.

Andrei Cramariuc, Heikki Huttunen, and Elena Simona Lohan. Clustering benefits in mobile-centric wifi positioning in multi-floor buildings. *2016 International Conference on Localization and GNSS (ICL-GNSS)*, 2016.

Paolo Dabove, Vincenzo Di Pietra, Marco Piras, Ansar Abdul Jabbar, and Syed Ali Kazim. Indoor positioning using ultra-wide band (uwb) technologies: positioning accuracies and sensors' performances. *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018.

David Davies and Don Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1:224 – 227, 05 1979.

Martin Ester, Hans-Peter Kriegel, Joerg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 96:226–231, 01 1996.

Yanbin Gao, Shifei Liu, Mohamed M. Atia, and Aboelmagd Noureldin. Ins/gps/lidar integrated navigation system for urban and indoor environments using hybrid scan matching algorithm. *Sensors*, 2015.

Christian Gentner, Markus Ulmschneider, Isabel Kuehner, and Armin Dammann. Wifi-rtt indoor positioning. *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2020.

Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *MDPI Sensors*, 2012.

Suining He and S.-H Gary Chan. Wi-fi fingerprint-based indoor positioning recent advances and comparisons. *IEEE Communications Surveys and Tutorials*, 2016.

Jeroen D. Hol, Thomas B. Schon, and Fredrik Gustafssony. Ultra-wideband calibration for indoor positioning. *2010 IEEE International Conference on Ultra-Wideband*, 2010.

Sazzad Hossain, Sharifah H. S. Ariffin, Norsheila Fisal, Choong Khong Neng, N. S. Abu Hassan, and L. A. Accuracy enhancement of fingerprint indoor positioning system. *2012 Third International Conference on Intelligent Systems Modelling and Simulation*, 2012.

Xuke Hu, Jianga Shang, Fuqiang Gu, , and Qi Han. Improving wi-fi indoor positioning via ap sets similarity and semi-supervised affinity propagation clustering. *International Journal of Distributed Sensor Networks*, 2015.

ISO/IEC 18305:2016. Information technology — real time locating systems — test and evaluation of localization and tracking systems. Standard, International Organization for Standardization, 2016.

Ping Jiang, Liang Chen, Hang Guo, Min Yu, and Jian Xiong. Novel indoor positioning algorithm based on lidar/inertial measurement unit integrated system. *International Journal of Advanced Robotic Systems*, 2021.

Myungin Jiw, Jooyoung Kim, Juil Jeon, and Youngsu Cho. Analysis of positioning accuracy corresponding to the number of ble beacons in indoor positioning system. *ICATCT2015*, 2015.

Ankush A. Kalbandhe and Shailaja.C.Patil. Indoor positioning system using bluetooth low energy. *CAST*, 2016.

Misha Urooj Khan, Dr. Syed Azhar Ali Zaidi, Arslan Ishtiaq, Syeda Ume Rubab Bukhari, Sana Samer, and Ayesha Farman. A comparative survey of lidar-slam and lidar based sensor technologies. *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*, 2021.

T. King, T. Haenselmann, and W. Effelsberg. On-demand fingerprint selection for 802.11-based positioning systems. In *Int. Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2008.

Thomas King, Stephan Kopf, Thomas Haenselmann, Christian Lubberger, and Wolfgang Effelsberg. CRAWDAD dataset mannheim/compass (v. 2008-04-11), 2008. URL https://crawdad.org/mannheim/compass/20080411.

G. Ajay Kumar, Ashok Kumar Patil, Rekha Patil, Seong Sill Park, and Young Ho Chai. A lidar and imu integrated indoor navigation system for uavs and its application in real-time pipeline classification. *Sensors*, 2017.

ARASH HABIBI LASHKARI, BEHRANG PARHIZKAR, and MIKE NG AH NGAN. Wifi-based indoor positioning system. *Second International Conference on Computer and Network Technology*, 2010.

Chung-Wei Lee, Tsung-Nan Lin, Shih-Hau Fang, and Yen-Chih Chou. A novel clustering-based approach of indoor location fingerprinting. *IEEE 24th International Symposium on Personal, Indoor and Mobile Radio Communications: Mobile and Wireless Networks*, 2013.

Chunhan Lee, Yushin Chang, Gunhong Pakl, Jaeheon Ry, Seung-Gweon Jeong, Seokhyun Pak, Jae We Pak, and Hee Chang Lee. Indoor positioning system based on incident angles of infared emitters. *30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004*, 2004.

Ang Li, Jingqi Fu, Huaming Shen, and Sizhou Sun. A cluster-principal-component-analysis-based indoor positioning algorithm. *IEEE INTERNET OF THINGS JOURNAL*, 2021.

Rongbing Li, Jianye Liu, Ling Zhang, and Yijun Hang. Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments. *Inertial Sensors and Systems*, 2014.

Hung-Huan Liu and Yu-Non Yang. Wifi-based indoor positioning for multi-floor environment. *TENCON*, 2011.

Lohan. Additional TAU datasets for Wi-Fi fingerprinting- based positioning, May 2020. URL https://doi.org/10.5281/zenodo.3819917.

Elena Simona Lohan, Joaquín Torres-Sospedra, and Alejandro Gonzalez. WiFi RSS measurements in Tampere University multi- building campus, 2017, August 2021. URL https://doi.org/10.5281/zenodo.5174851.

Ilaria Sergi Luca Mainetti, Luigi Patrono. A survey on indoor positioning systems. *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2014.

Jun MA, Xuansong LI, Xianping TAO, and Jian LU. Cluster filtered knn:a wlan-based indoor positioning scheme. *2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2008.

Rui Ma, Qiang Guo, Changzhen Hu, and Jingfeng Xue. An improved wifi indoor positioning algorithm by weighted fusion. *MDPI Sensors*, 2015.

J. MacQueen. Some methods for classification and analysis of multivariate observations. *Berkeley Symposium on Mathematical Statistics and Probability, 1967: 281-297 (1967)*, PAMI-1, 1967.

Ernesto Martín-Gorostiza, Miguel A. García-Garrido, Daniel Pizarro, David Salido-Monzú, and Patricia Torres. An indoor positioning approach based on fusion of cameras and infrared sensors. *Sensors*, 2019.

Germán Martín Mendoza-Silva, Philipp Richter, Joaquín Torres-Sospedra, Elena Simona Lohan, and Joaquín Huerta. Long-term wifi fingerprinting dataset for research on robust indoor positioning. *Data*, 3(1), 2018.

Wei Meng, Wendong Xiao, Wei Ni, and Lihua Xie. Secure and robust wi-fi fingerprinting indoor localization. *2011 International Conference on Indoor Positioning and Indoor Navigation*, 2011.

Adriano Moreira, Maria João Nicolau, Ivo Silva, Joaquín Torres-Sospedra, Cristiano Pendão, and Filipe Meneses. Wi-Fi Fingerprinting dataset with multiple simultaneous interfaces, September 2019. URL https://doi.org/10.5281/zenodo.3342526.

Adriano Moreira, Ivo Silva, and Joaquín Torres-Sospedra. The DSI dataset for Wi-Fi fingerprinting using mobile devices, 2020. URL https://doi.org/10.5281/zenodo.3778646.

Florian Gain Nicolas Le Dortz and Per Zetterberg. Wifi fingerprint indoor positioning system using probability distribution comparison. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

Duong Bao Ninh, Jing He, Vu Thanh Trung, and Dang Phuoc Huy. An effective random statistical method for indoor positioning system using wifi fingerprinting. *Future Generation Computer Systems*, 2020.

Feng Qin, Tao Zuo, and Xing Wang. Ccpos: Wifi fingerprint indoor positioning system based on cdae-cnn. *Sensors*, 2021.

Moises Ramires, Joaquín Torres-Sospedra, and Adriano Moreira. Accurate and Efficient Wi-Fi Fingerprinting-Based Indoor Positioning in Large Areas. In *2022 IEEE 96th Vehicular Technology Conference: (VTC2022-Fall)*, pages 1–6, 2022.

JIN REN, YUNAN WANG, CHANGLIU NIU, WEI SONG, and SONGYANG HUANG. A novel clustering algorithm for wi-fi indoor positioning. *IEEE Access*, 2019.

Philipp Richter, Elena Simona Lohan, and Jukka Talvitie. WLAN (WiFi) rss database for fingerprinting positioning. URL https://zenodo.org/record/1161525.

Pampa Sadhukhan, Supriya Gain, Keshav Dahal, Samiran Chattopadhyay, Nilkantha Garain, and Xinheng Wang. An efficient clustering with robust outlier mitigation for wi-fi fingerprint based indoor positioning. *Applied Soft Computing*, 2021.

A. Saha and P. Sadhukhan. A novel clustering strategy for fingerprinting-based localization system to reduce the searching time. *Proc. of 2015 IEEE 2nd International Conference on Recent Trends in Information System (ReTIS'15)*, 2015.

Zuin Silvia, Calzavara Martina, Sgarbossa Fabio, and Persona Alessandro. Ultra wide band indoor positioning system: analysis and testing of an ips technology. *IFAC*, 2018.

Xudong Song, Xiaochen Fan, Chaocan Xiang, Qianwen Ye, Leyu Liu, Zumin Wang, Xiangjian He, Ning Yang, and Gengfa Fang. A novel convolutional neural network based indoor localization framework with wifi fingerprinting. *IEEE Access*, 7:110698–110709, 2019. doi: 10.1109/ACCESS.2019.2933921.

Petros Spachos and Konstantinos N. Plataniotis. Ble beacons for indoor positioning at an interactive iot-based smart museum. *IEEE Systems Journal*, 2020.

SANTOSH Subedi, HUI-SEON GANG, NAK YONG KO, SUK-SEUNG HWANG, , and JAE-YOUNG PYUN. Improving indoor fingerprinting positioning with affinity propagation clustering and weighted centroid fingerprint. *IEEE Access*, 2019.

Dwi Joko Suroso, Panarat Chemtanomwong, and Jun ichi Takada. Fingerprint-based technique for indoor localization in wireless sensor networks using fuzzy c-means clustering algorithm. *Interational Symposium on Intelligent Signal Processing ad Communication Systems (ISPACS)*, 2011.

E. Sánchez, M. Botsch, B. Huber, and A. García. High precision indoor positioning by means of lidar. *Inertial Sensors and Systems*, 2019.

Zengshan Tian, Xiaomou Tang, Mu Zhou, and Zuohong Tan. Fingerprint indoor positioning algorithm based on affinity propagation clustering. *EURASIP Journal on Wireless Communications and Networking*, 2013.

Joaquín Torres-Sospedra, Raúl Montoliu, Adoolfo Martínez-Usó, Joan P. Avariento, Tomás J. Arnau, Mauri Benedito-Bordonau, and Joaquín Huerta. Ujiindoorloc: A new multi-building and multi-floor dtabase for wlan fingerprint-based indoor localization problems. *International Conference o Indoor Positioning and Indoor Navgation*, 2014.

Joaquín Torres-Sospedra, Sergi Trilles Oliver, Oscar Belmonte Fernández, and Joaquín Huerta. Comprehensive analysis of distance and similarity measures for wi-fi fingerprinting indoor positioning systems. *Expert System with Applications*, 2015.

Joaquín Torres-Sospedra, Philipp Richter, Adriano Moreira, Germán M. Mendoza-Silva, Elena Simona Lohan, Miguel Matey-Sanz Sergio Trilles, and Joaquín Huerta. A comprehensive and reproducible comparison of clustering and optimization rules in wi-fi fingerprinting. *IEEE Transactions on Mobile Computing*, 2020.

Joaquín Torres-Sospedra, Philipp Richter, Adriano Moreira, Germán M. Mendoza-Silva, Elena Simona Lohan, Sergio Trilles, Miguel Matey-Sanz, and Joaquín Huerta. A comprehensive and reproducible comparison of clustering and optimization rules in wi-fi fingerprinting. *IEEE Transactions on Mobile Computing*, 2022.

Peerapong Torteeka and XIU Chundi. Indoor positioning based on wi-fi fingerprint technique using fuzzy k-nearest neighbor. *IBCAST*, 2014.

Yun-Ting Wang, Chao-Chung Peng, Ankit A. Ravankar, and Abhijeet Ravankar. A single lidar-based feature fusion indoor localization algorithm. *Sensors*, 2018.

Shixiong Xia, Yi Liu, Guan Yuan, Mingjun Zhu, and Zhaohui Wang. Indoor fingerprint positioning based on wi-fi: An overview. *MDPI*, 2017.

Chouchang Yang and Huai-Rong Shao. Wifi-based indoor positioning. *IEEE Communications Magazine*, 2015.

Dan Yang, Bin Xu, Kaiyou Rao, and Weihua Sheng. Passive infrared (pir)-based indoor position tracking for smart homes using accessibility maps and a-star algorithm. *Sensors*, 2018.

Hikmet Yucel, Taha Ozkir, Rifat Edzikan, and Ahmet Yazici. Development of indoor positioning system with ultrasonic and infrared signals. *2012 International Symposium on Innovations in Intelligent Systems and Applications*, 2012.

Wei Zhang, Xianghong Hua, Kegen Yu, Weining Qiu, and Shoujian Zhang. Domain clustering based wifi indoor positioning algorithm. *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2016.

Zhujianyongand Chen Zili, Luo haiyong, and Li zhaohui. Rssi based bluetooth low energy indoor positioning. *2014 International Conference on Indoor Positioning and Indoor Navigation*, 2014.

Part III

# APPENDICES

## .1 LIB1

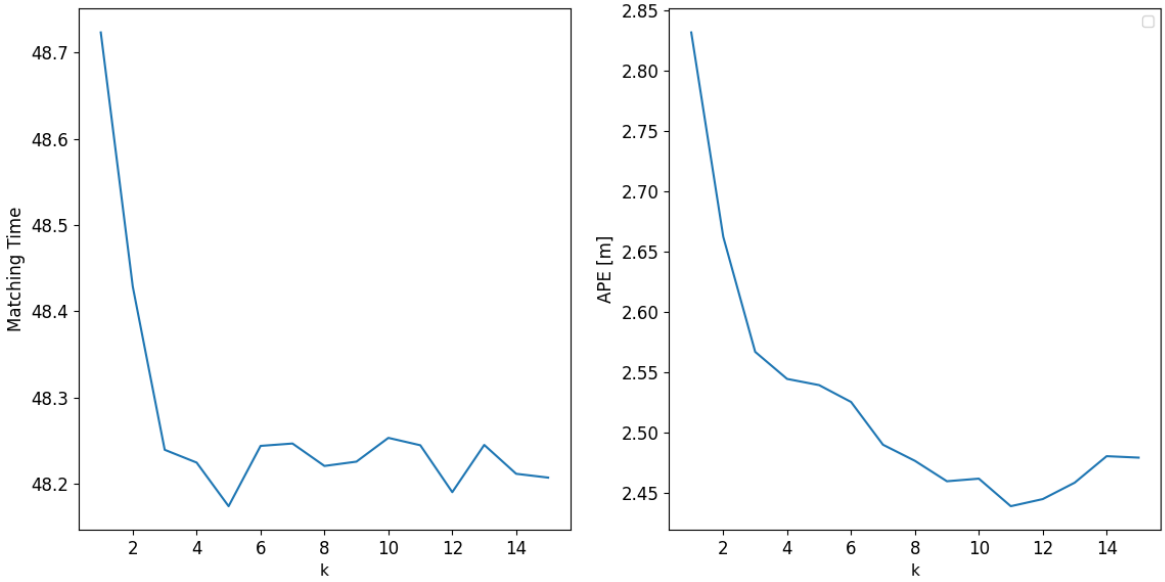



Figure 24: LIB1 *k*-NN parameter

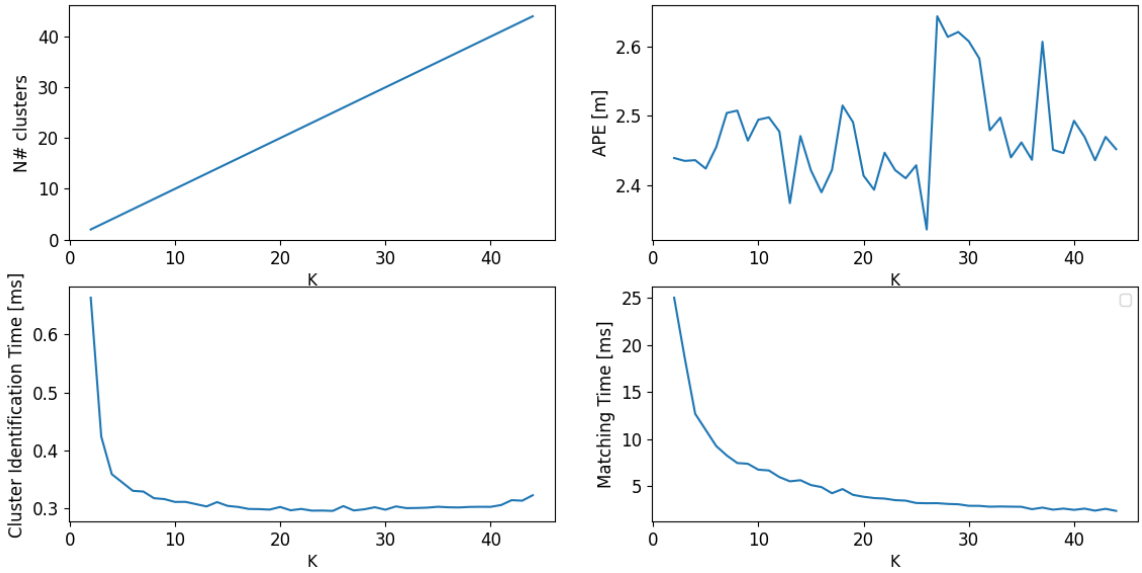



Figure 25: LIB1 *K*-Means hyperparameters

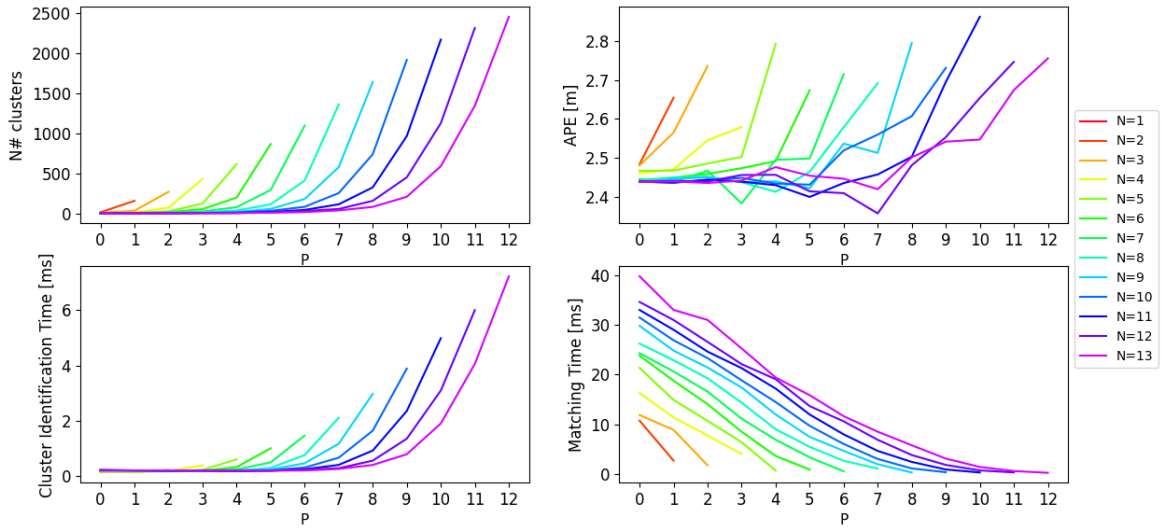



Figure 26: LIB1 SAS hyperparameters

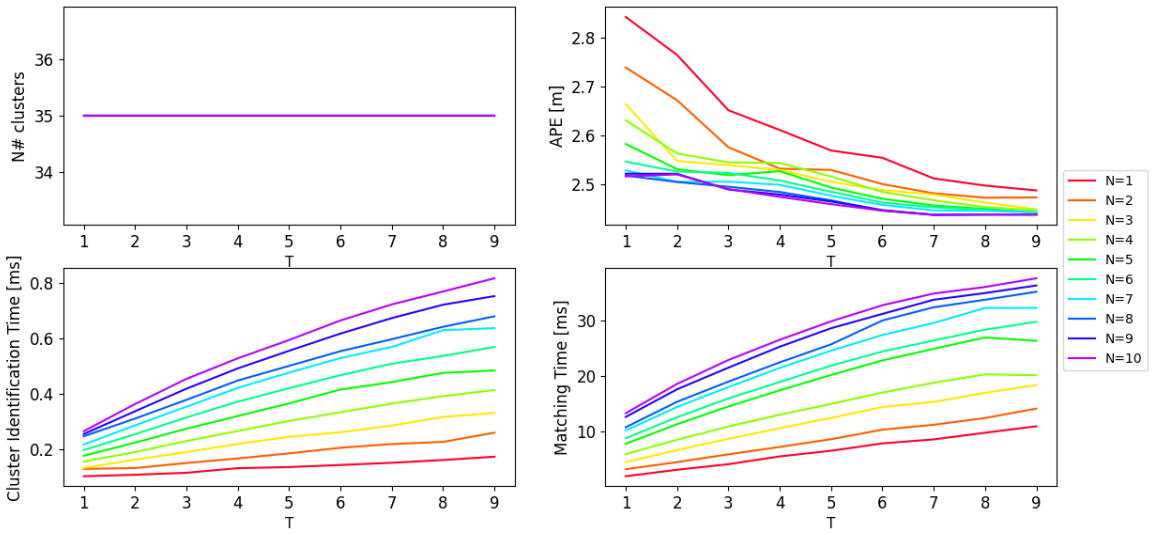



Figure 27: LIB1 BSC hyperparameters

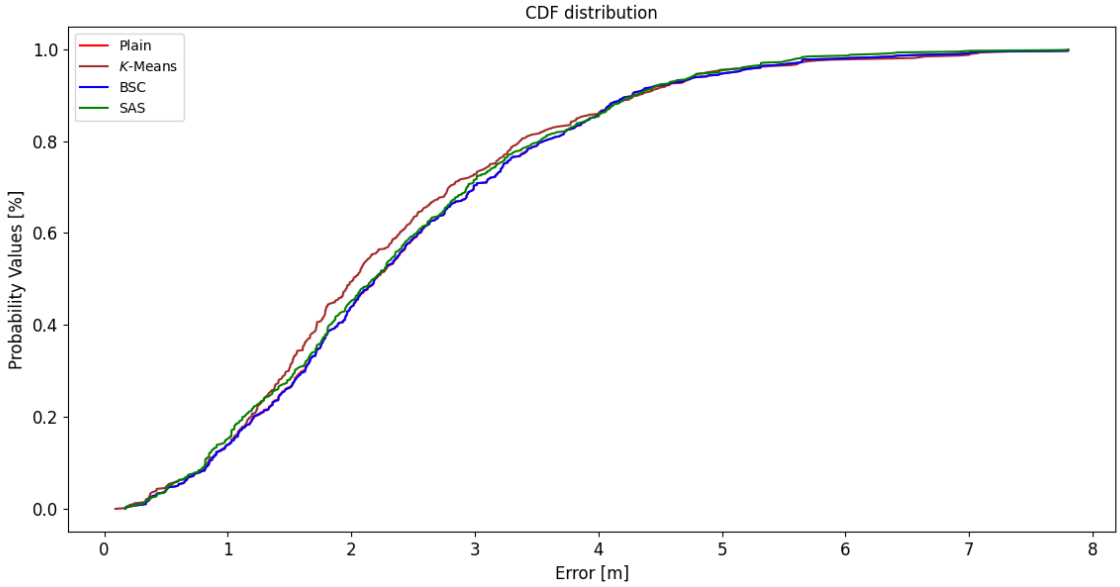



Figure 28: LIB1 CDF

## .2 LIB2

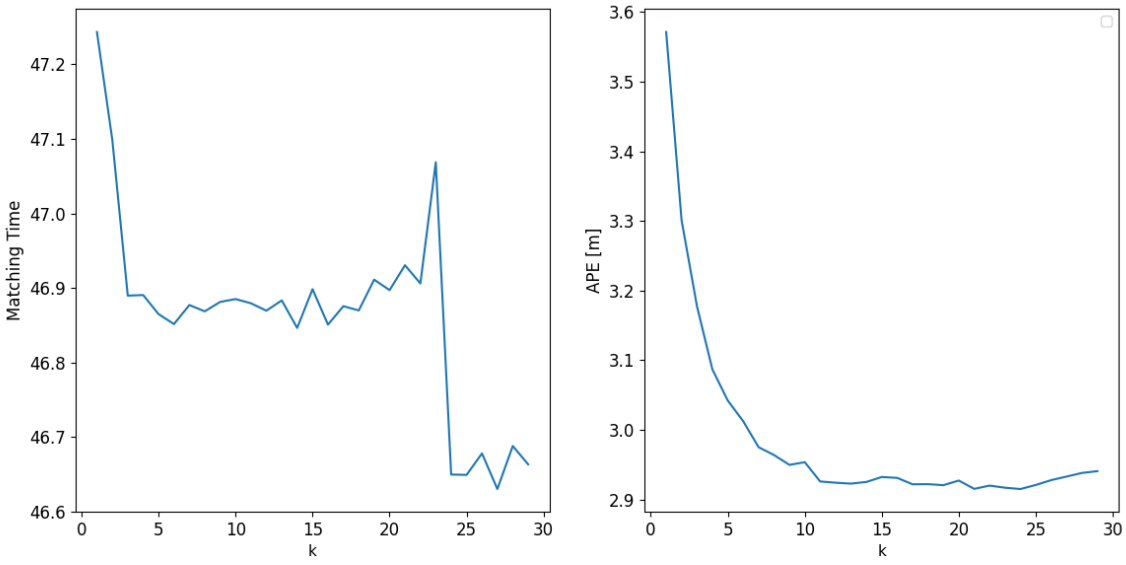



Figure 29: LIB2 $k$-NN parameter

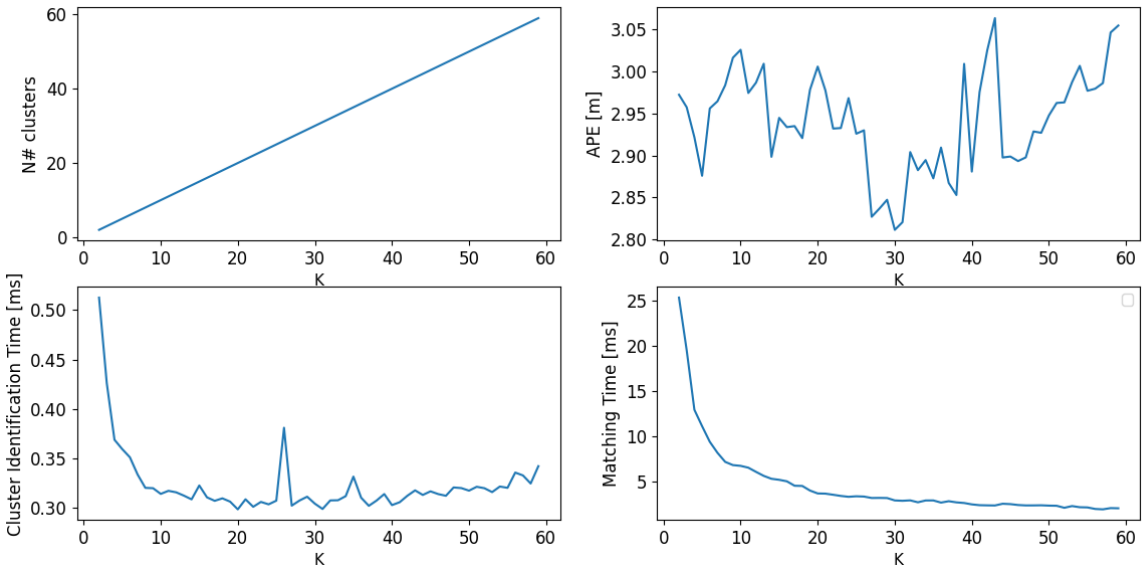



Figure 30: LIB2 *K*-Means hyperparameters

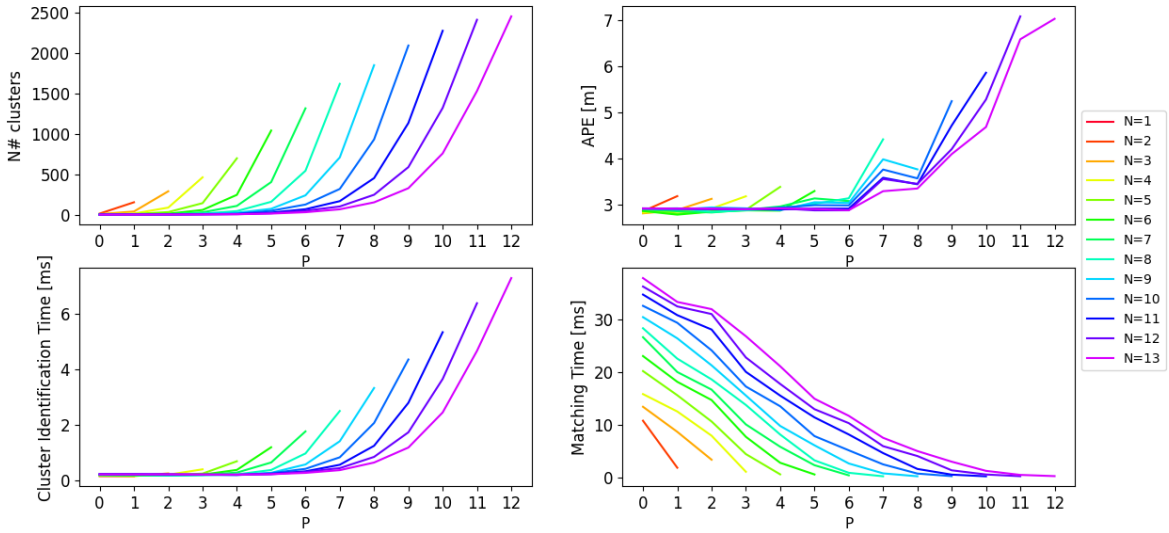



Figure 31: LIB2 SAS hyperparameters

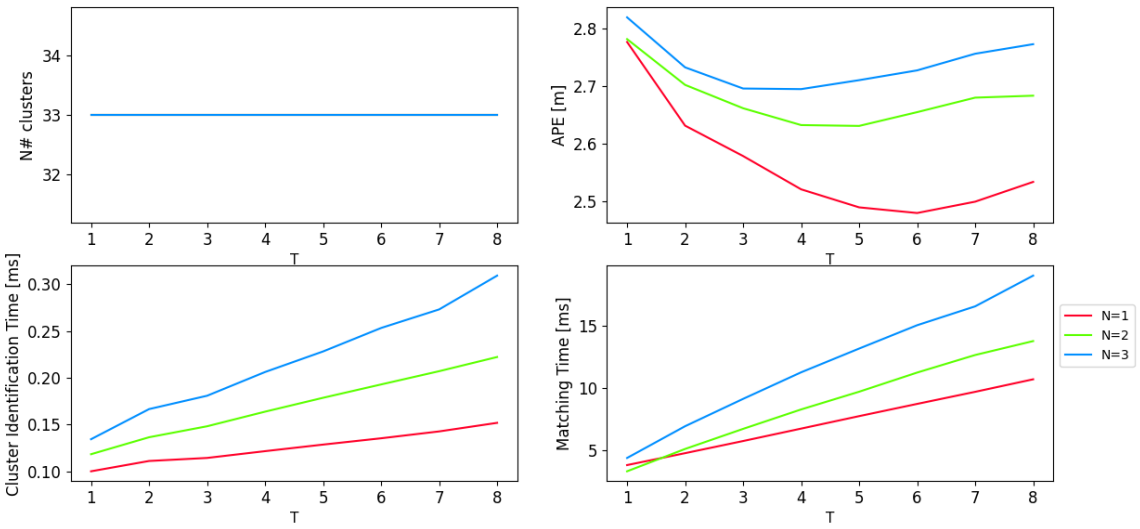



Figure 32: LIB2 BSC hyperparameters

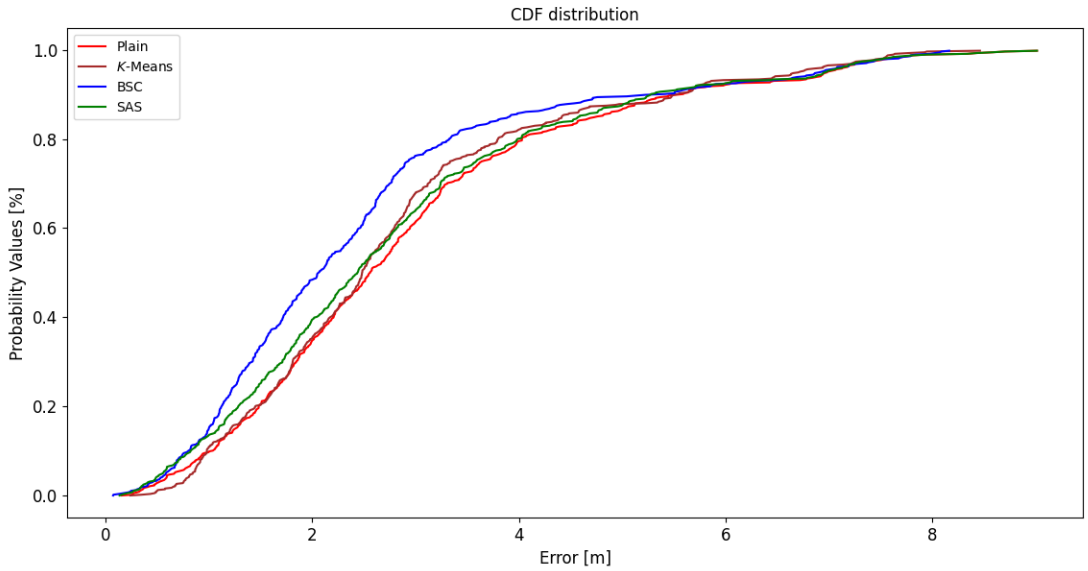



Figure 33: LIB2 CDF

## .3 MAN1

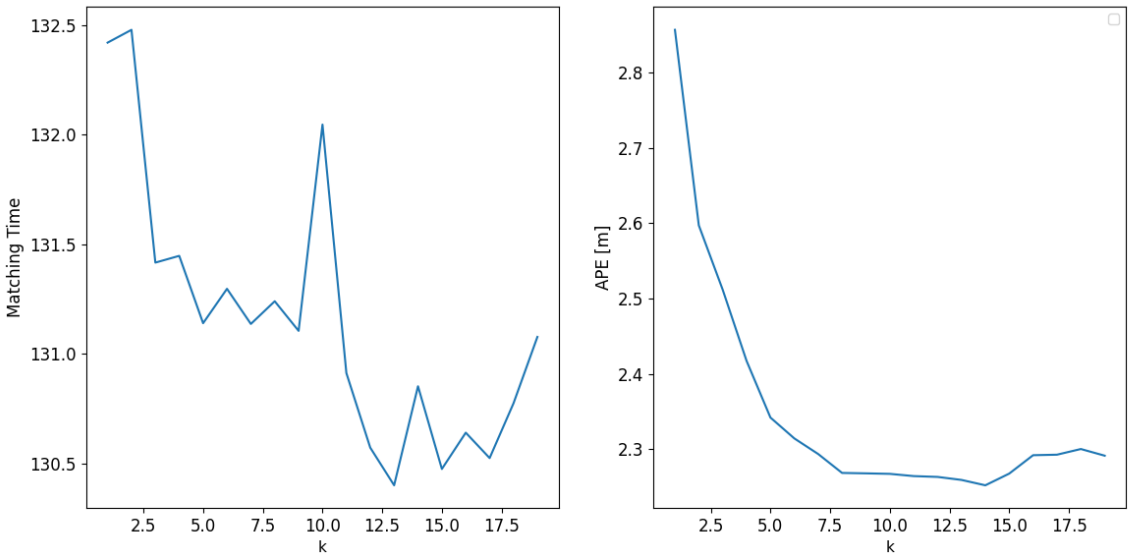



Figure 34: MAN1 $k$-NN parameter

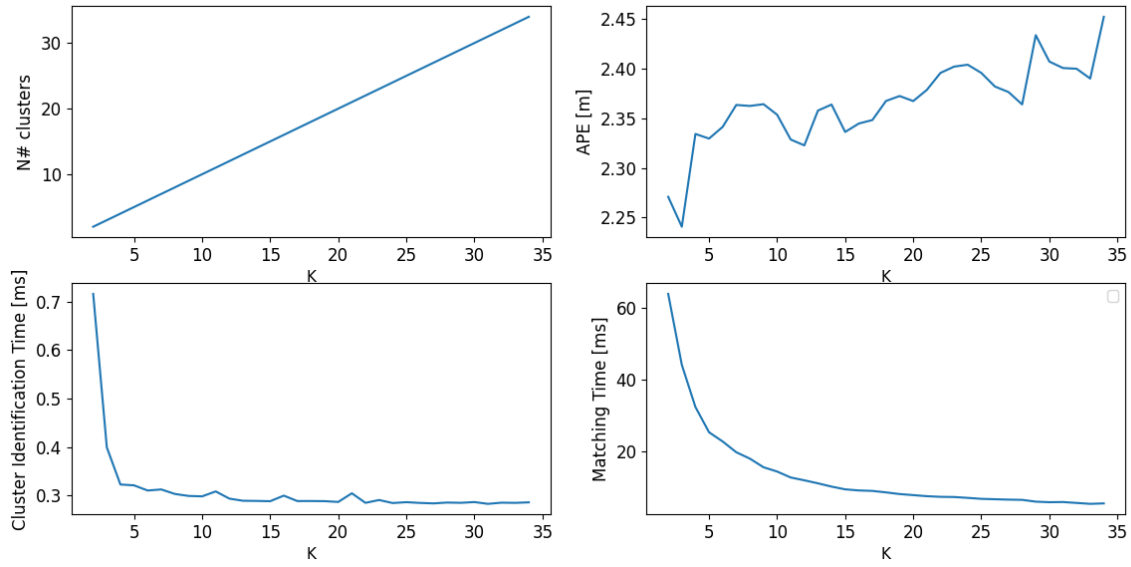



Figure 35: MAN1 $K$-Means hyperparameters

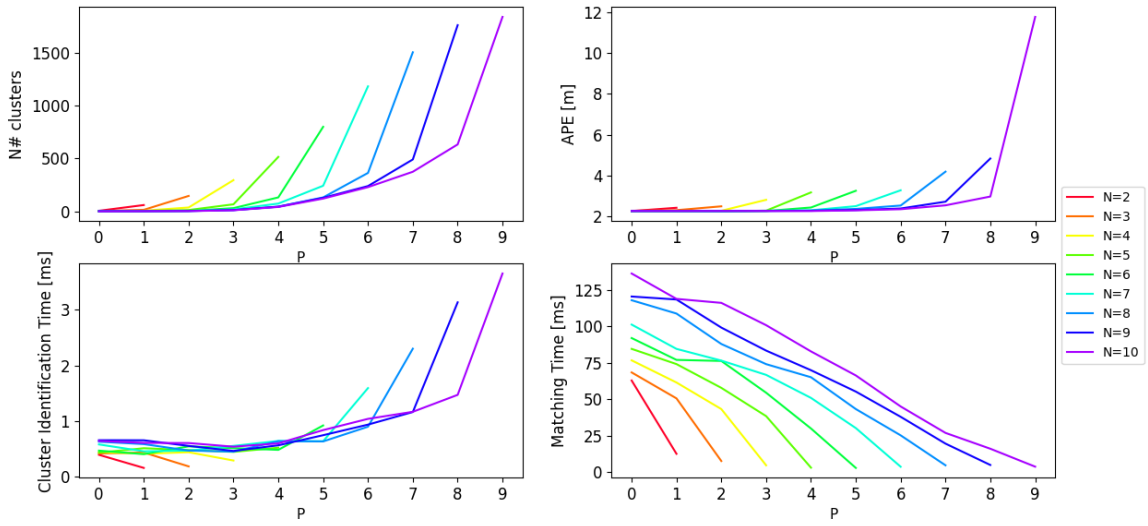



Figure 36: MAN1 SAS hyperparameters

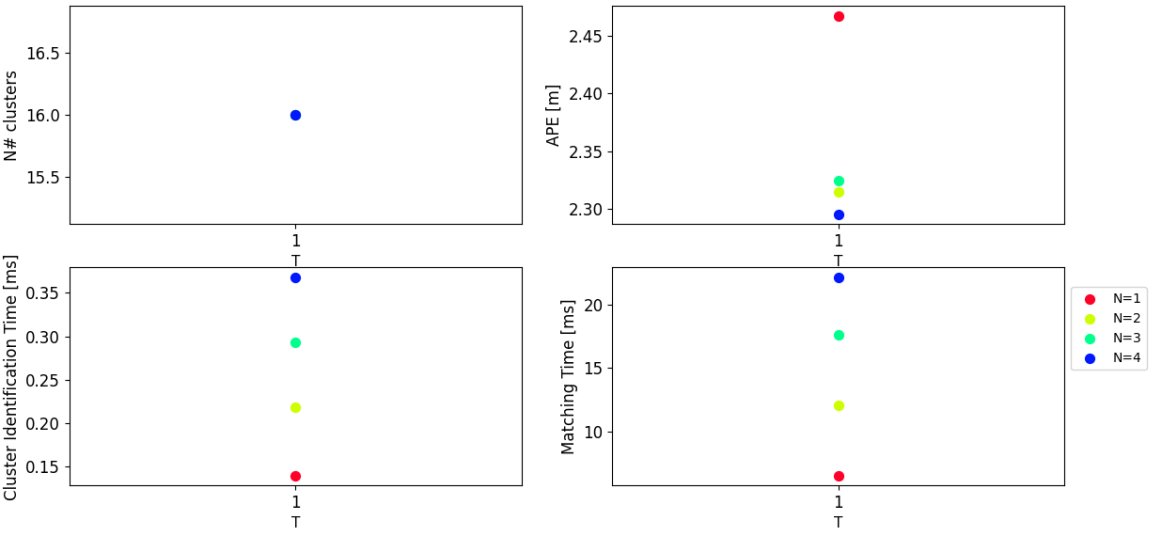



Figure 37: MAN1 BSC hyperparameters

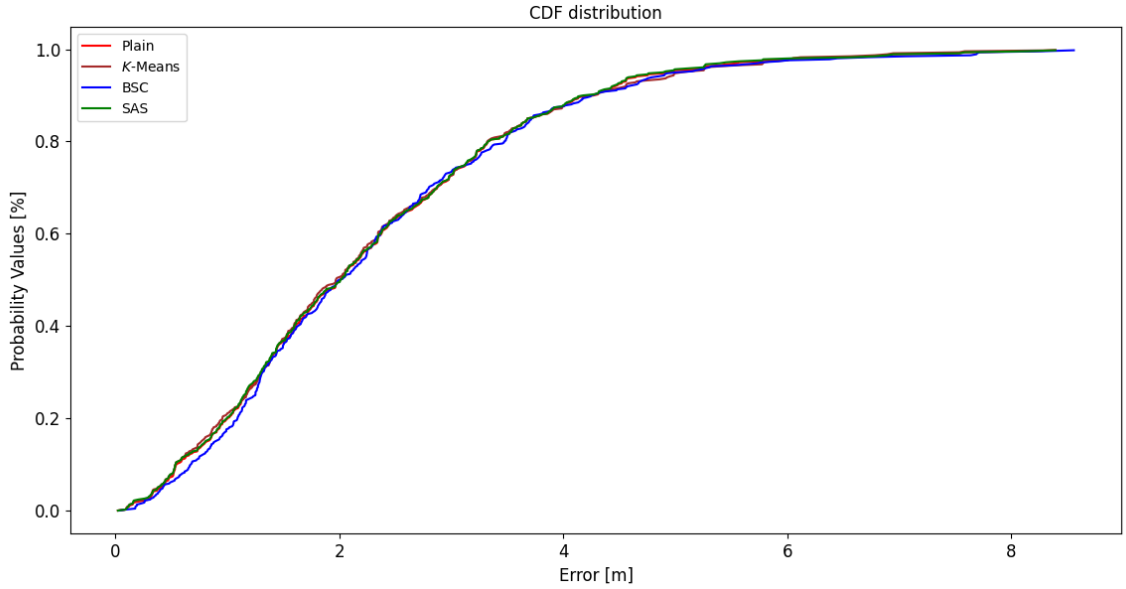



Figure 38: MAN1 CDF

## .4 MINT1

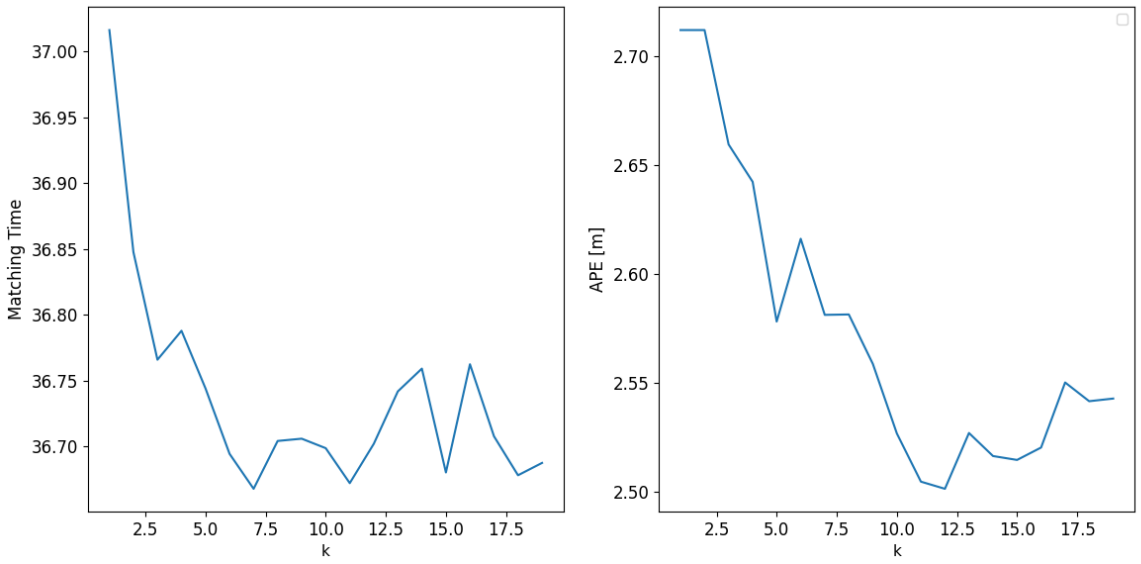



Figure 39: MINT1 *k*-NN parameter

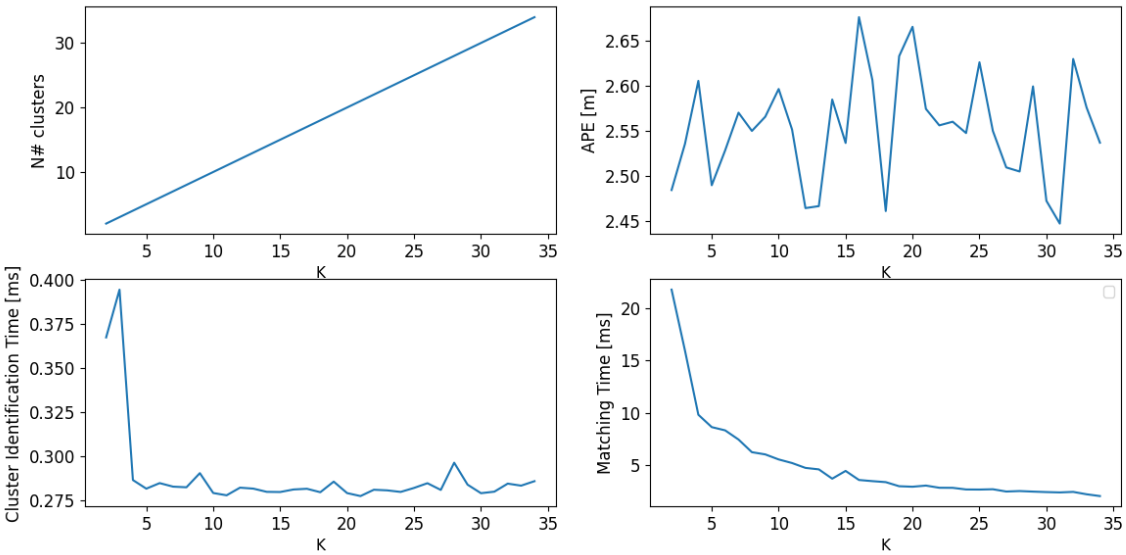



Figure 40: MINT1 *K*-Means hyperparameters

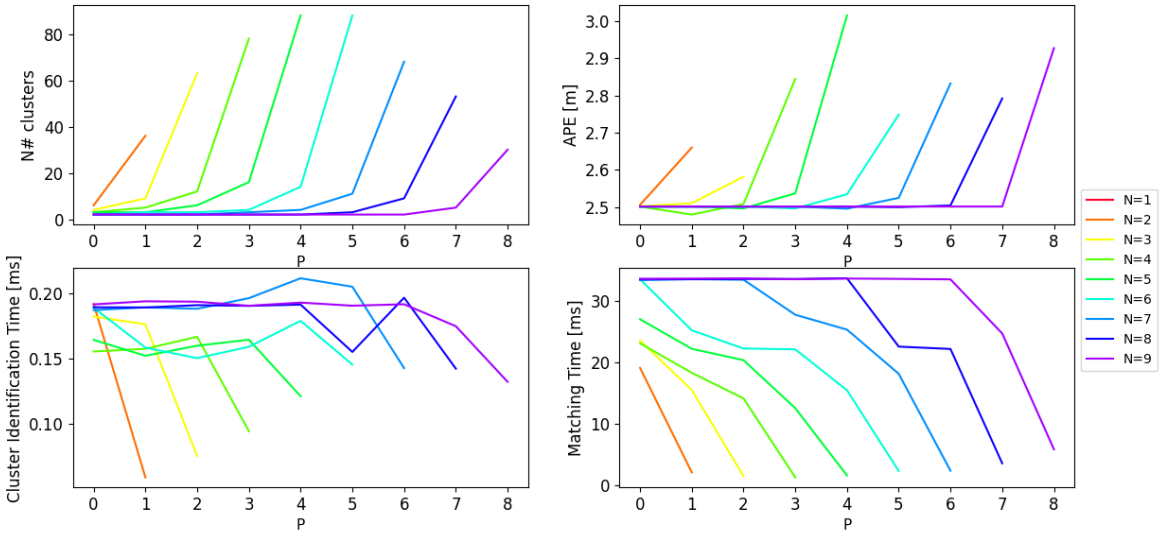



Figure 41: MINT1 SAS hyperparameters

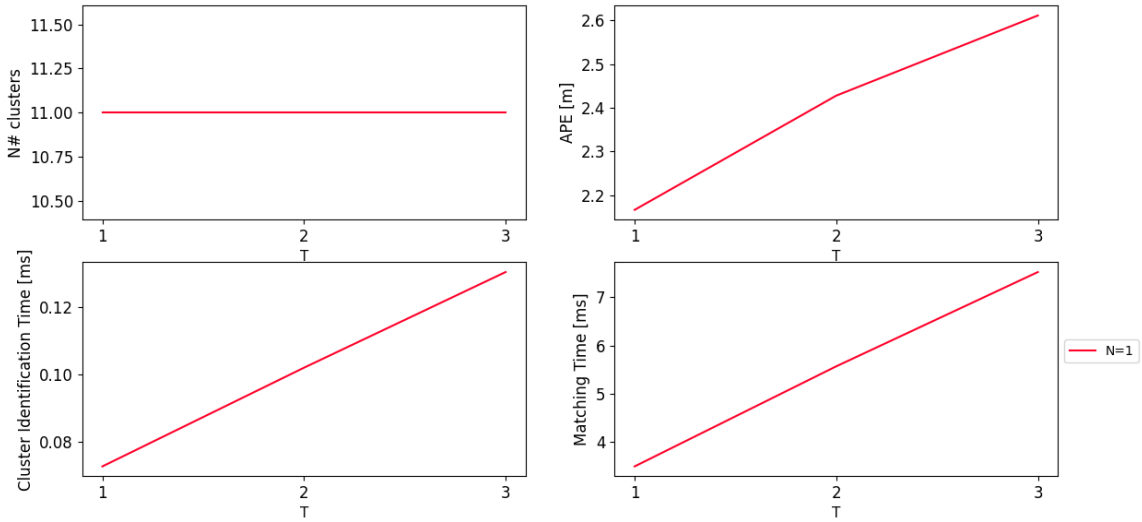



Figure 42: MINT1 BSC hyperparameters

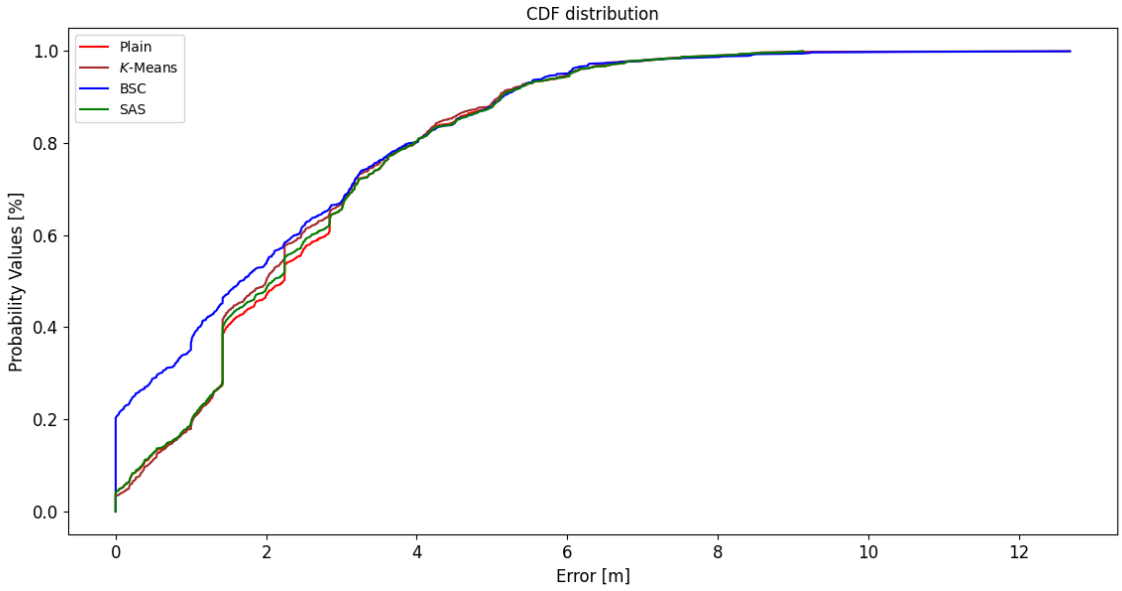



Figure 43: MINT1 CDF

.5 UJI1

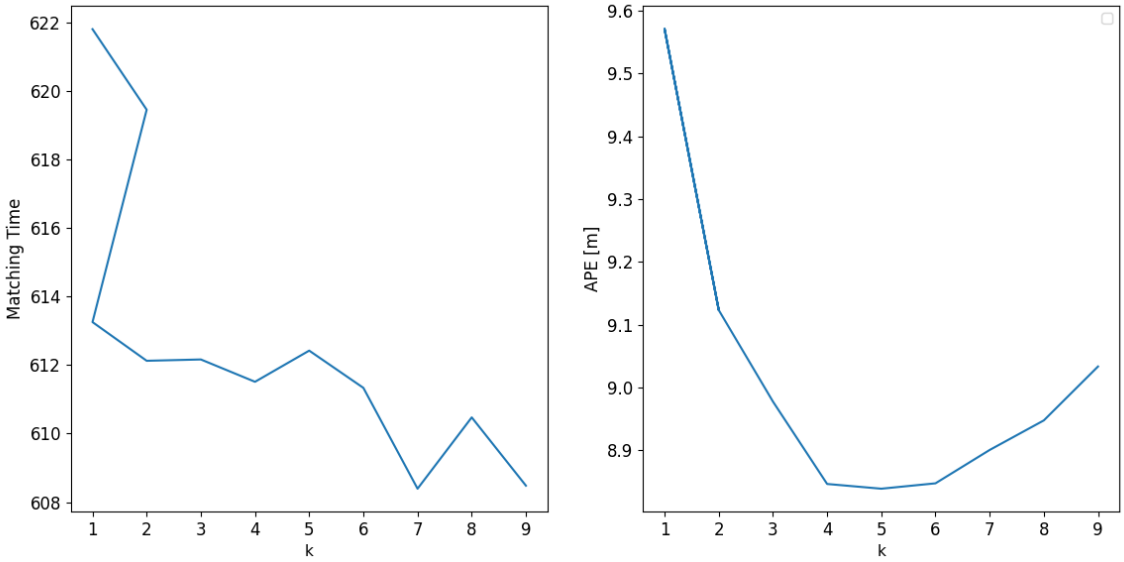



Figure 44: UJI1 *k*-NN parameter

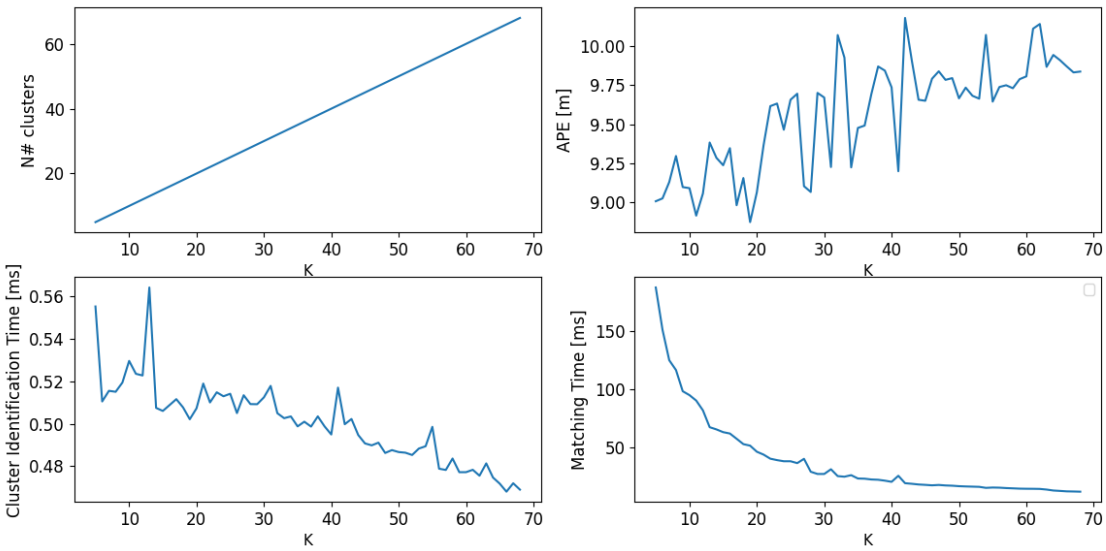



Figure 45: UJI1 *K*-Means hyperparameters

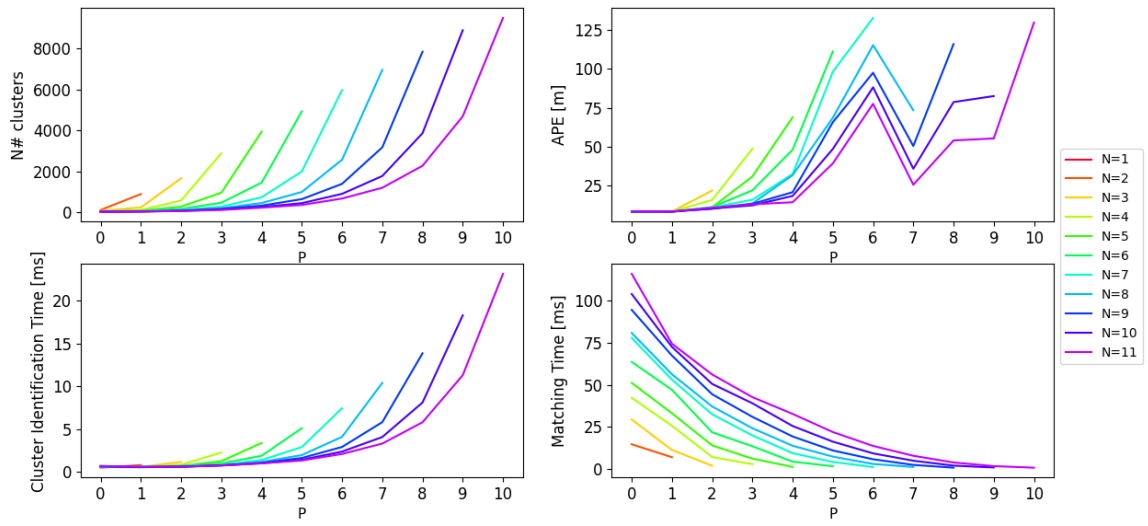



Figure 46: UJI1 SAS hyperparameters

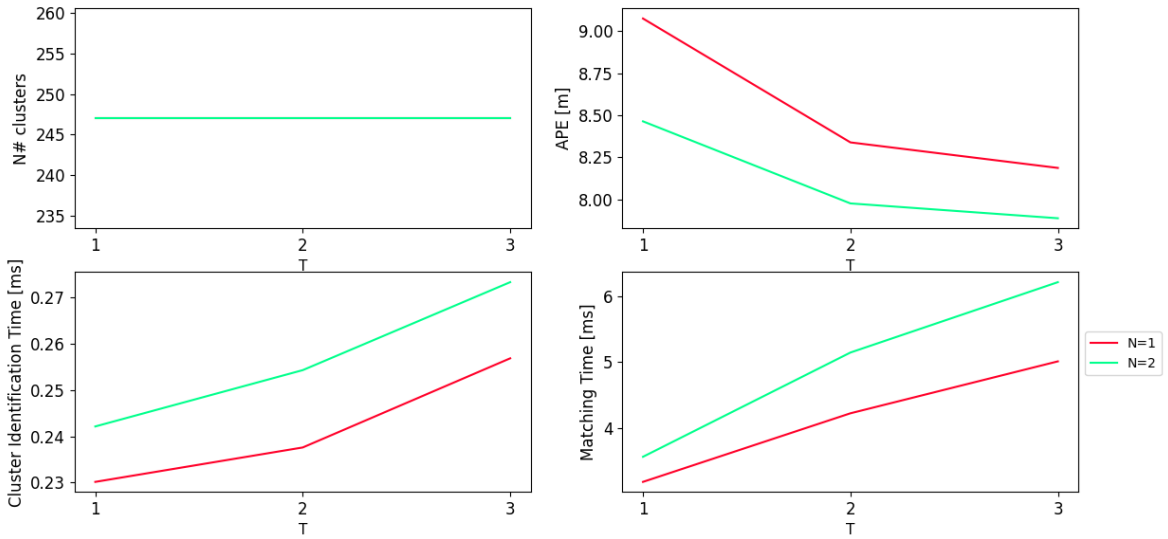



Figure 47: UJI1 BSC hyperparameters

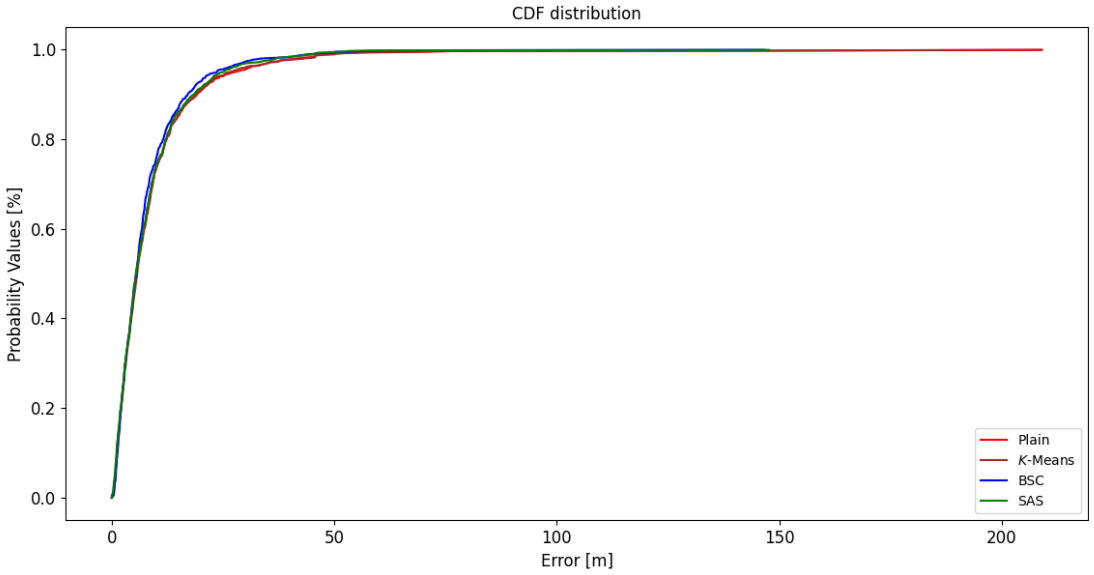



Figure 48: UJI1 CDF

## .6 SAH1

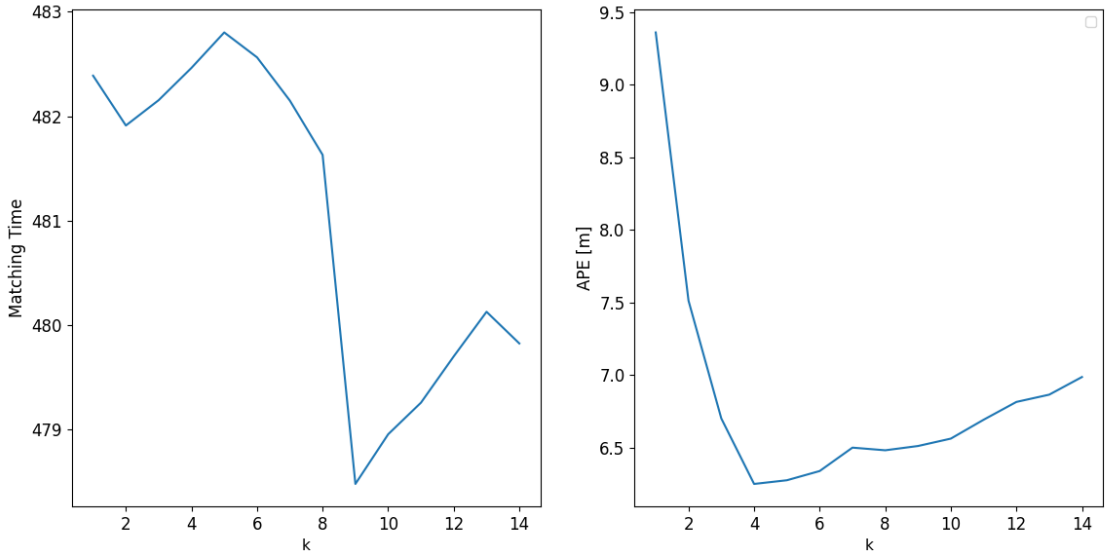



Figure 49: SAH1 *k*-NN parameter

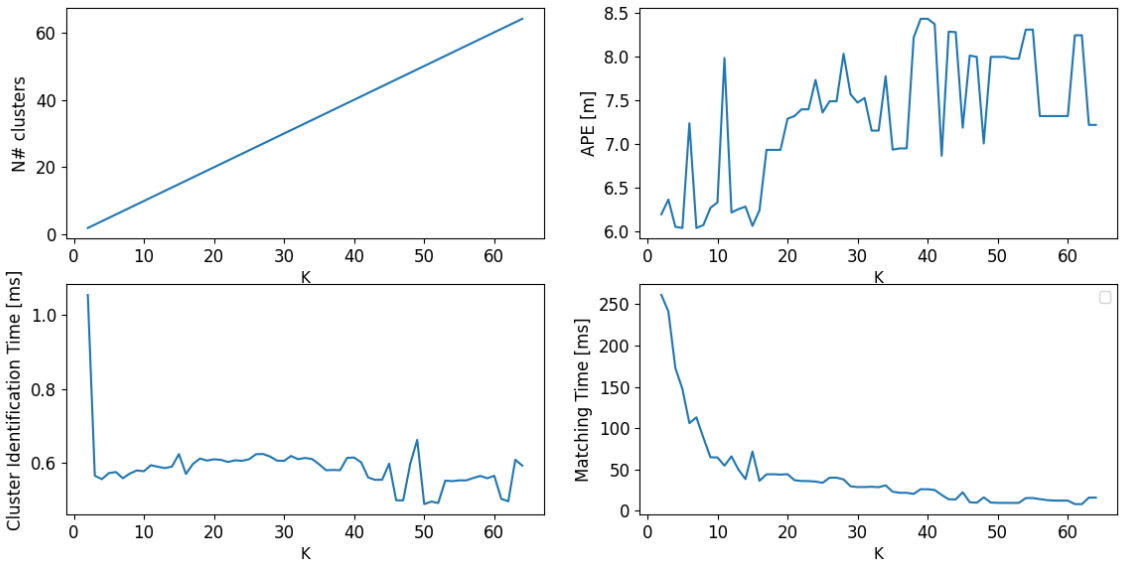



Figure 50: SAH1 *K*-Means hyperparameters

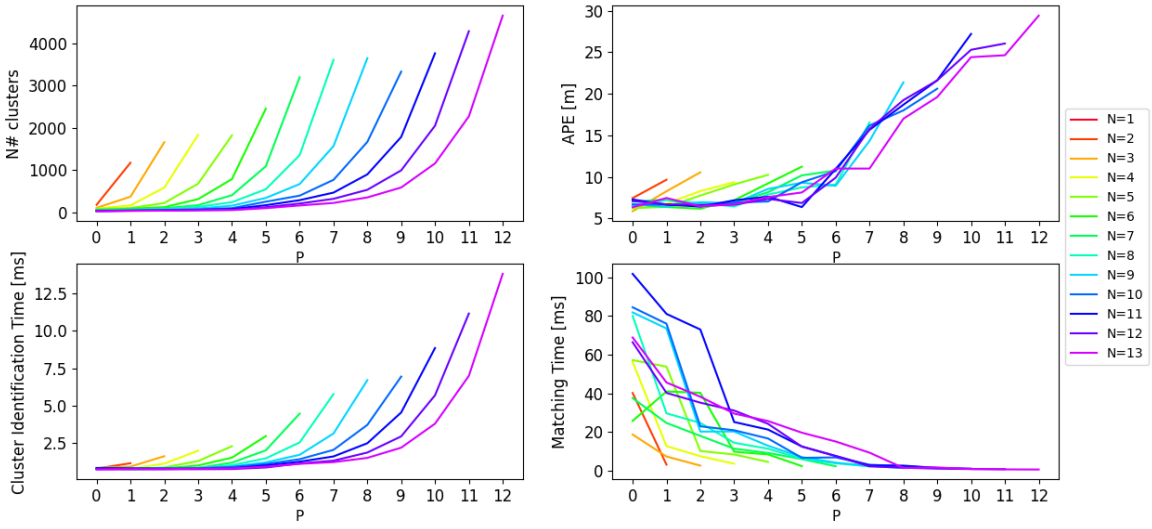



Figure 51: SAH1 SAS hyperparameters

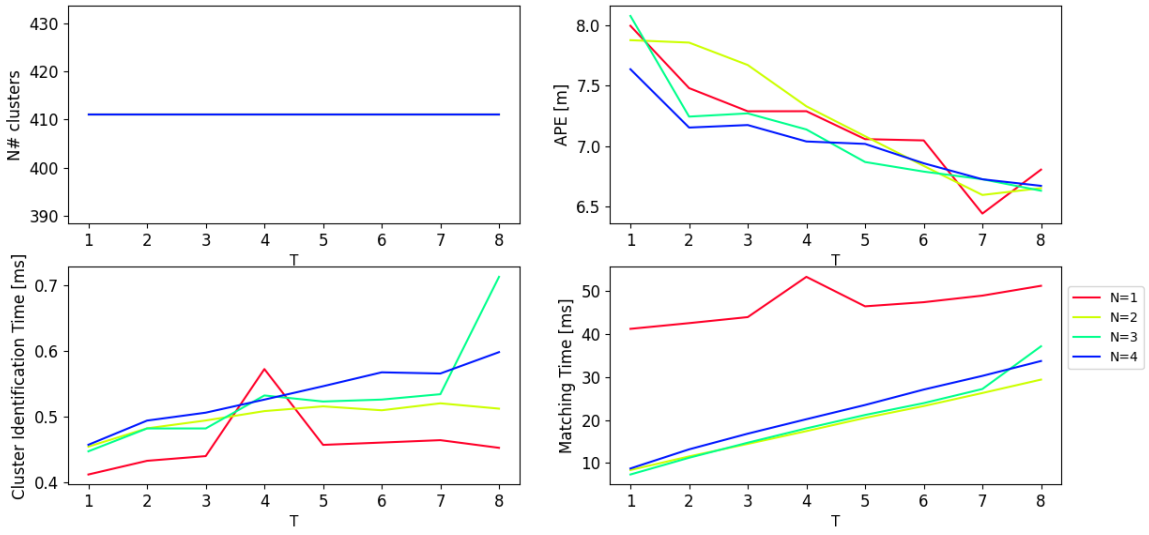



Figure 52: SAH1 BSC hyperparameters

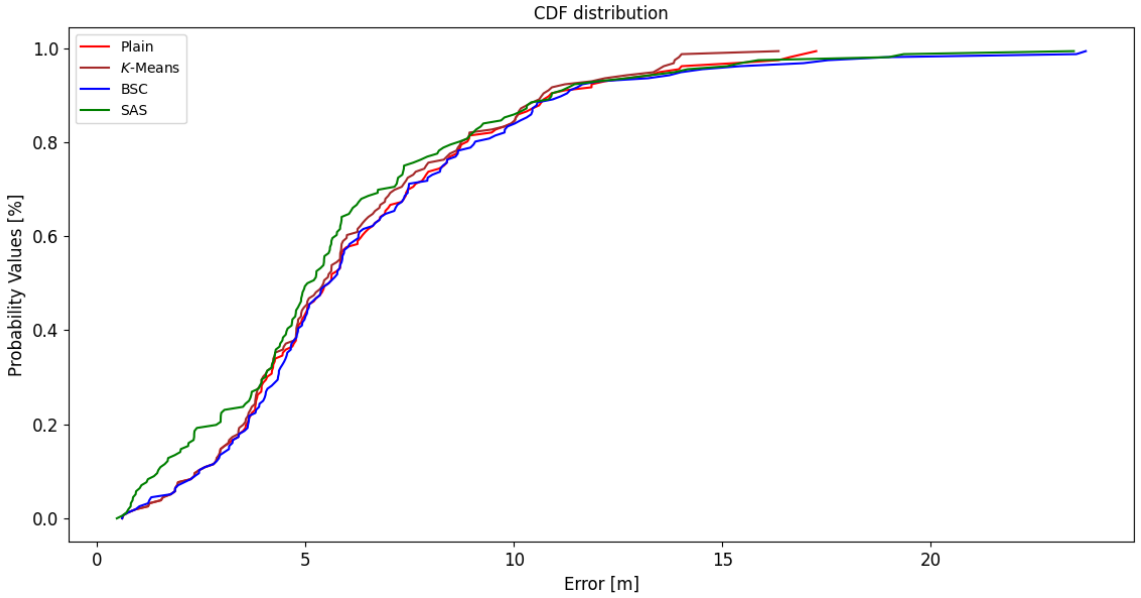



Figure 53: SAH1 CDF

## .7 TUT5

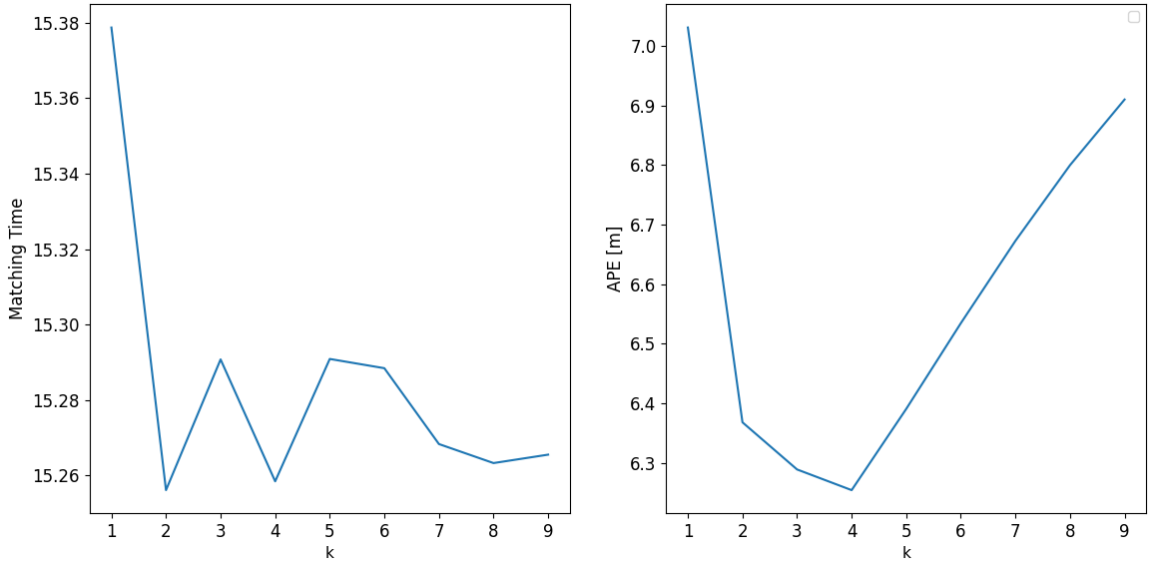



Figure 54: TUT5 *k*-NN parameter

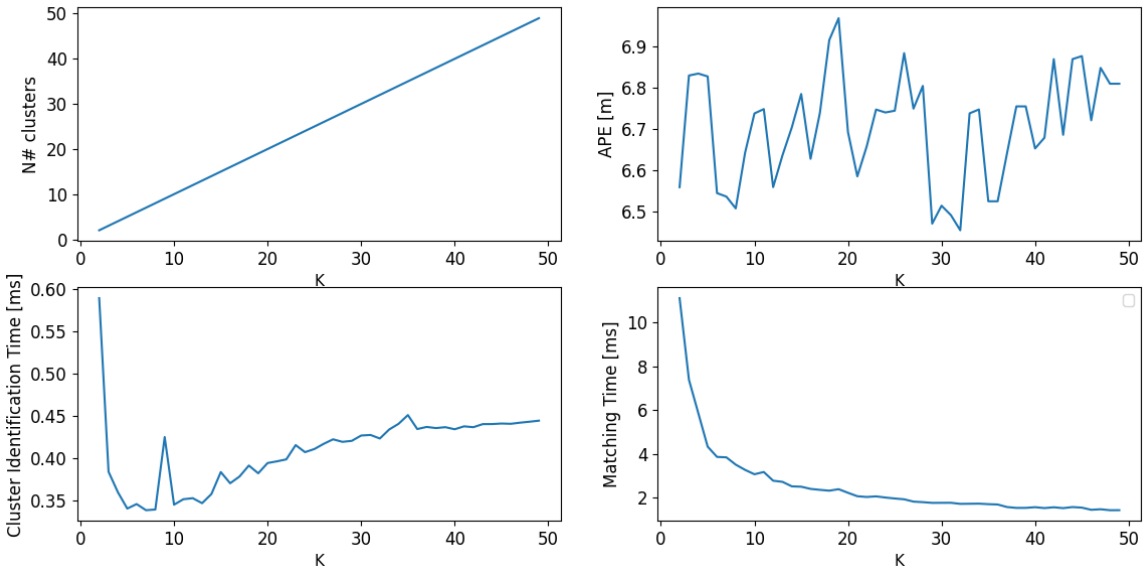



Figure 55: TUT5 *K*-Means hyperparameters

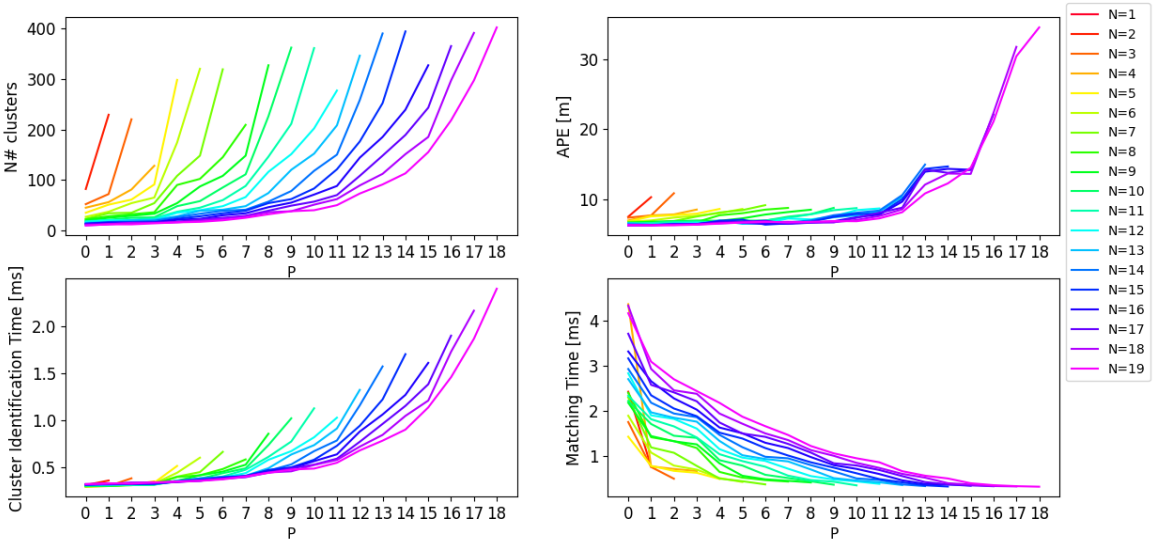



Figure 56: TUT5 SAS hyperparameters

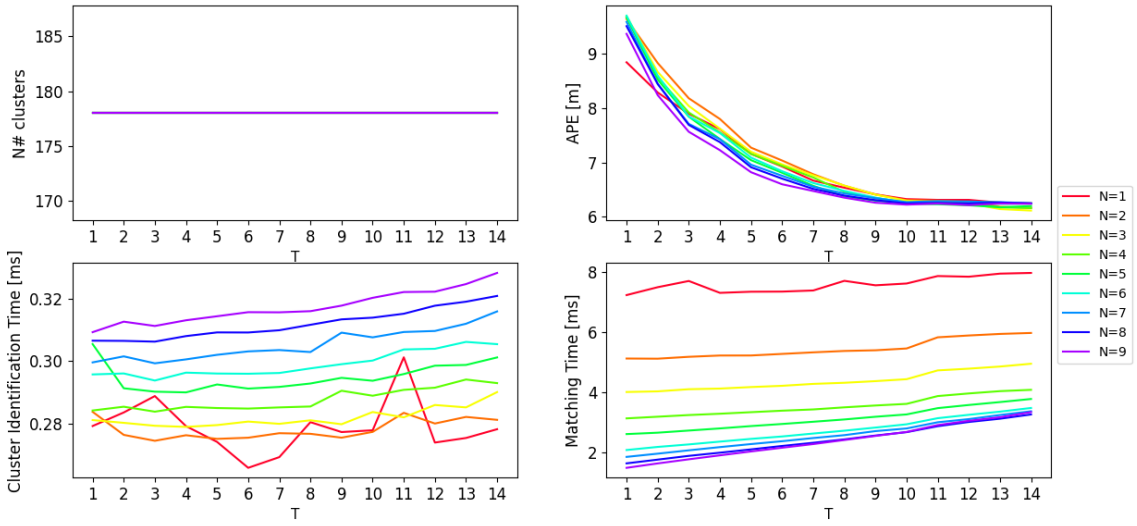



Figure 57: TUT5 BSC hyperparameters

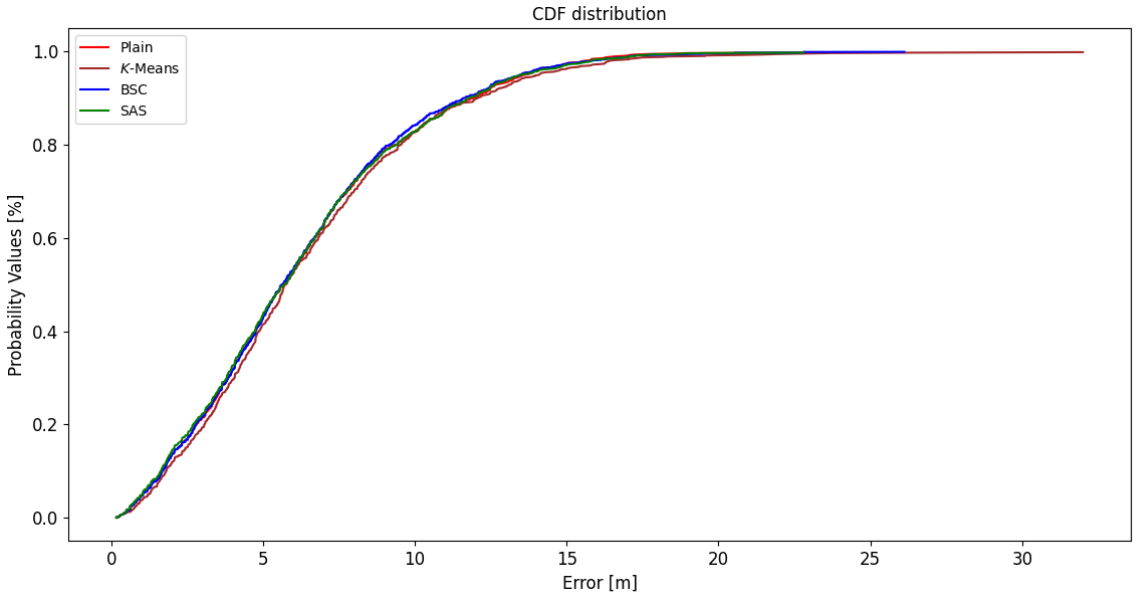



Figure 58: TUT5 CDF

## .8 TUT6

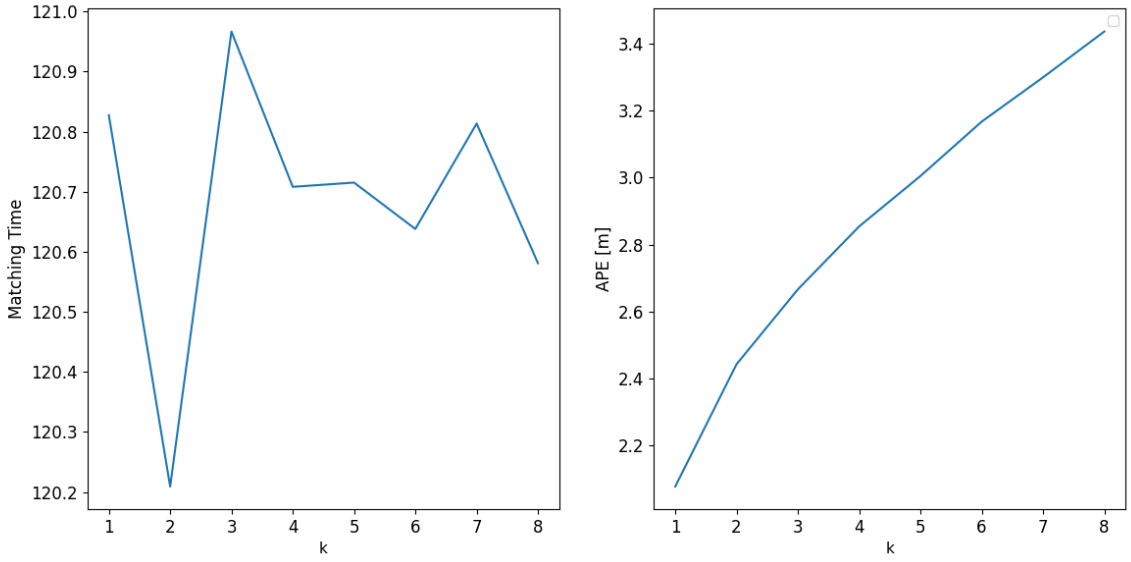



Figure 59: TUT6 *k*-NN parameter

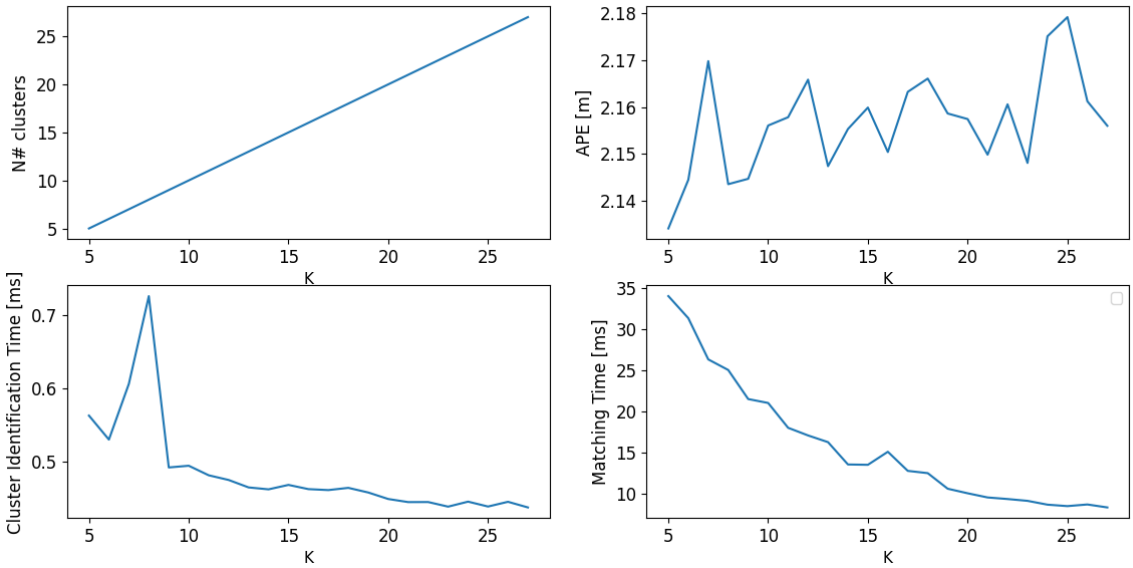



Figure 60: TUT6 $K$-Means hyperparameters

The graphs 61 shows the result of varying the $N$ and $P$ for SAS.

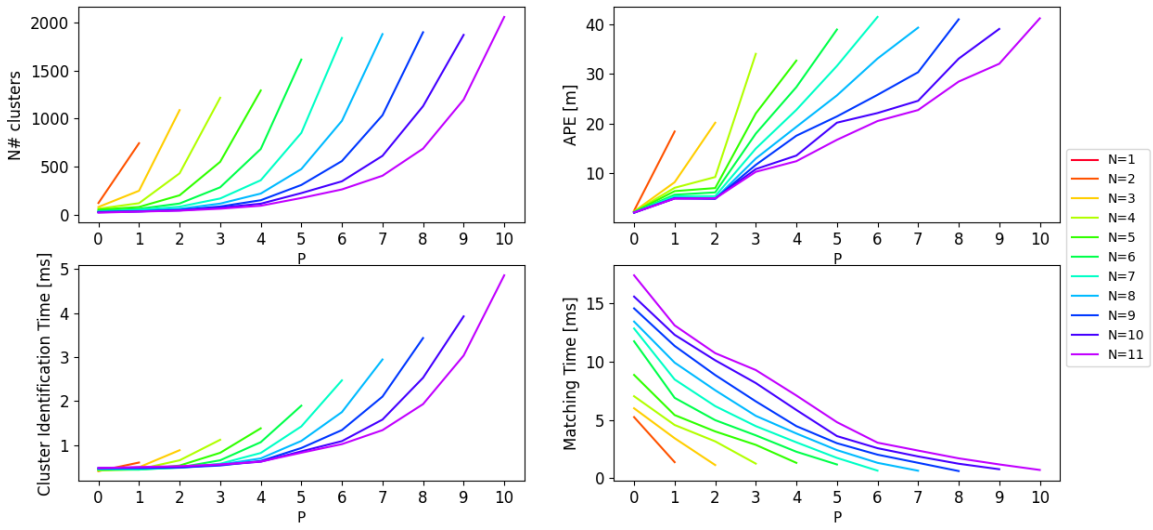



Figure 61: TUT6 SAS hyperparameters

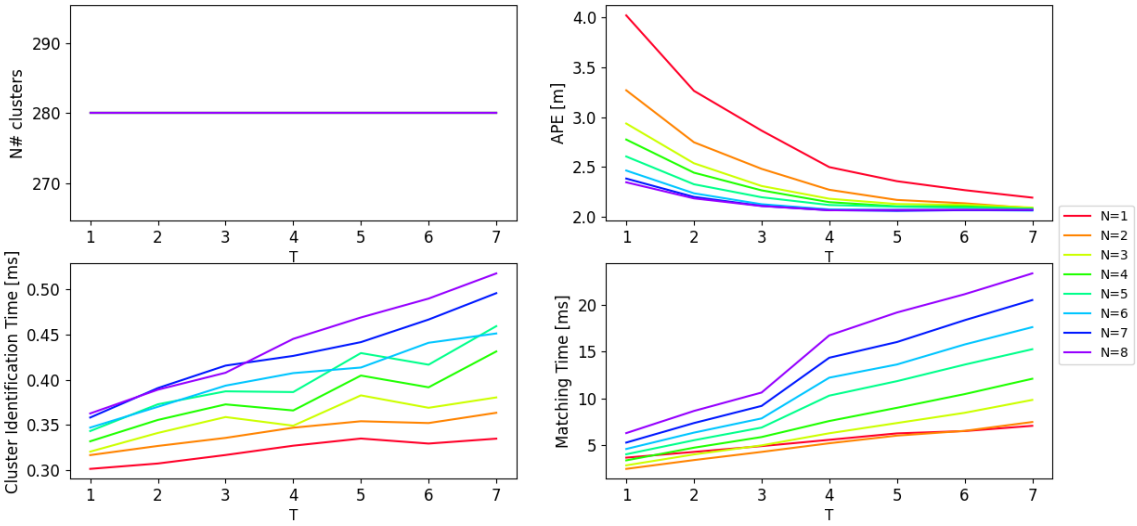



Figure 62: TUT6 BSC hyperparameters

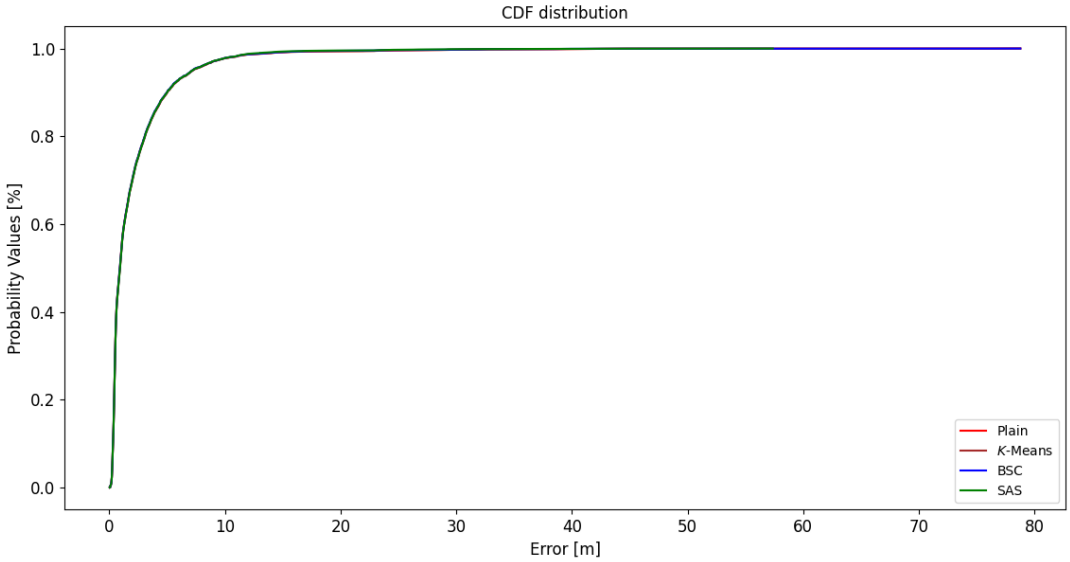



Figure 63: TUT6 CDF

## .9 UTS1

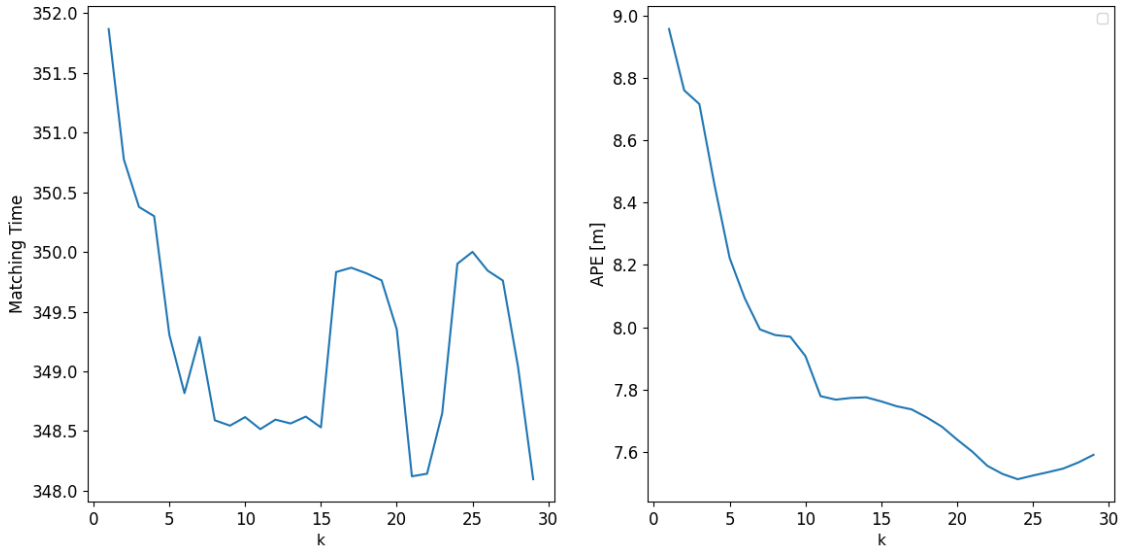



Figure 64: UTS1 *k*-NN parameter

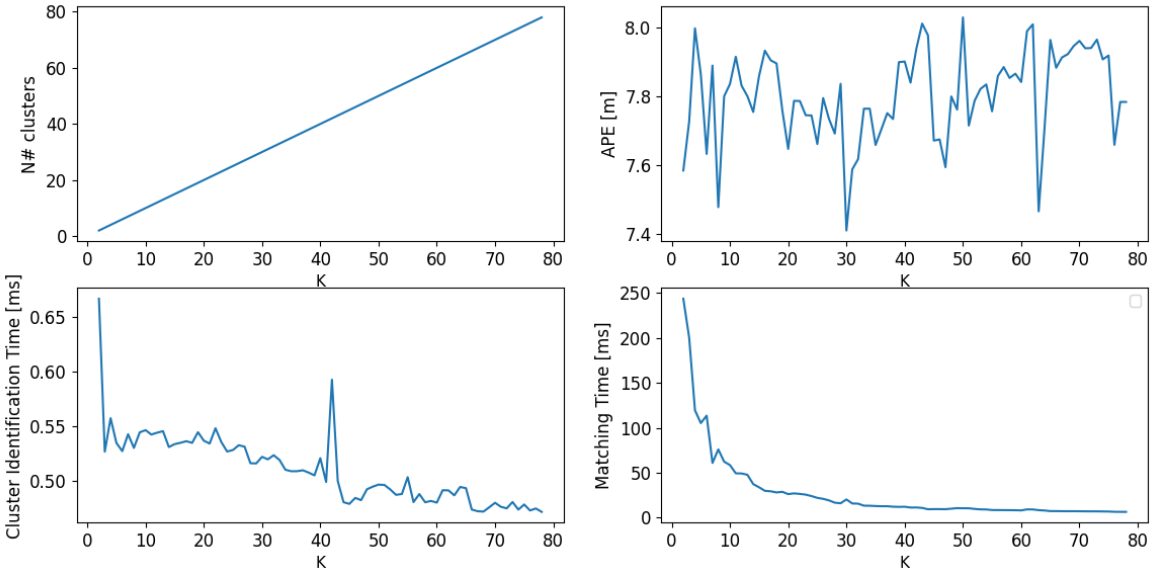



Figure 65: UTS1 *K*-Means hyperparameters

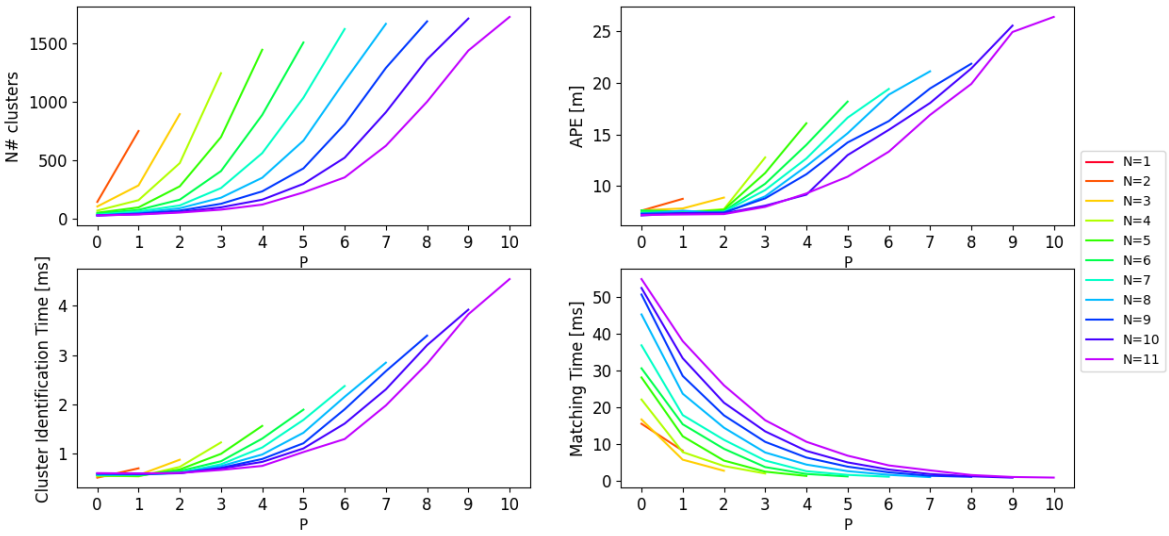



Figure 66: UTS1 SAS hyperparameters

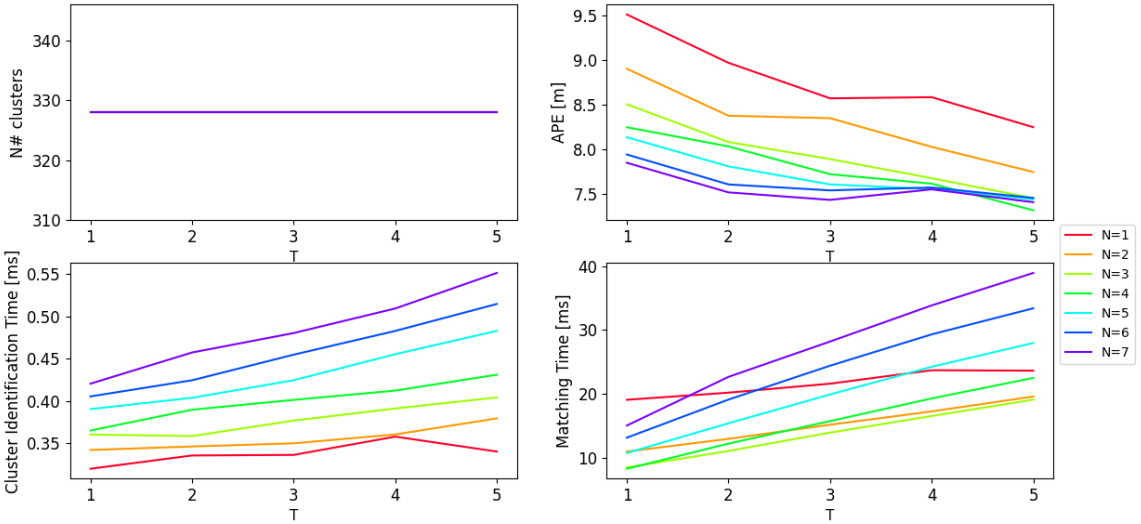



Figure 67: UTS1 BSC hyperparameters

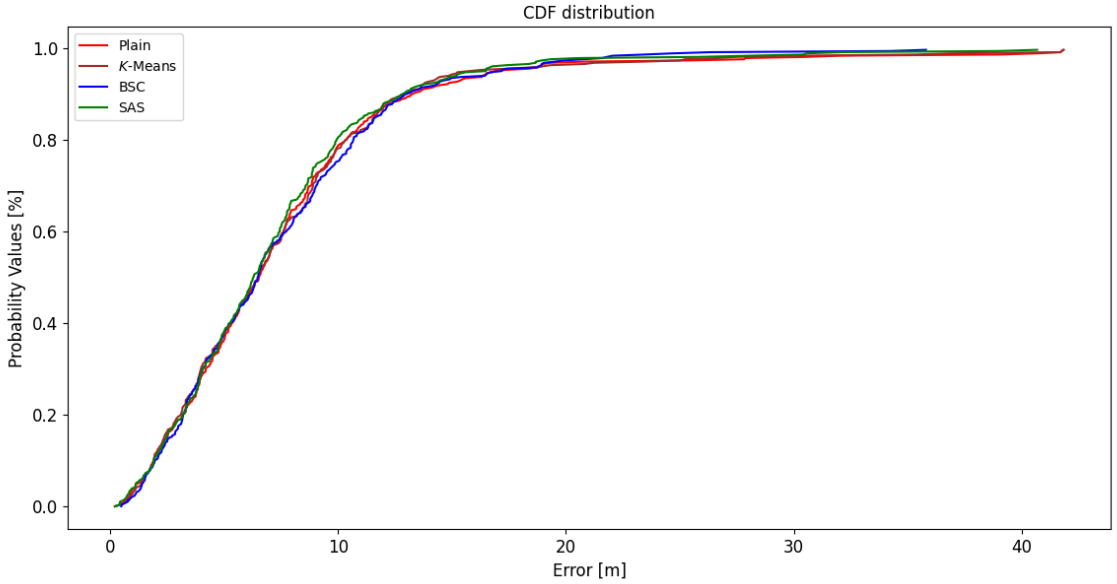



Figure 68: UTS1 CDF