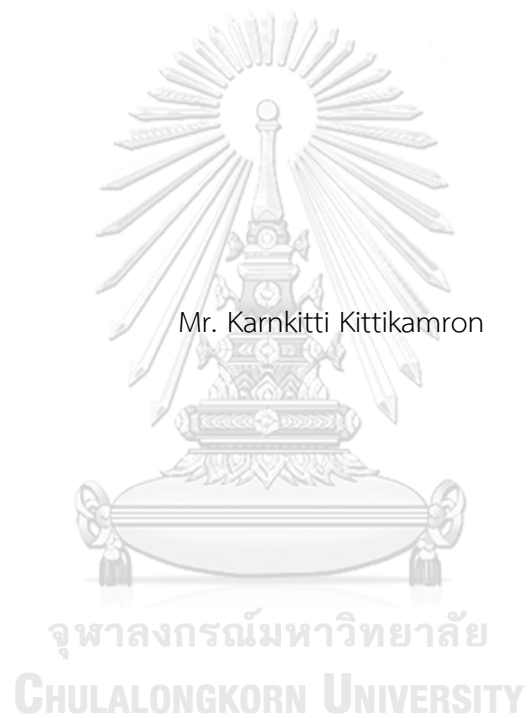


Service Placement Optimization for Location-Based Service



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Computer Engineering

Department of Computer Engineering

FACULTY OF ENGINEERING

Chulalongkorn University

Academic Year 2022

Copyright of Chulalongkorn University

การเพิ่มประสิทธิภาพตำแหน่งบริการสำหรับบริการตามตำแหน่ง

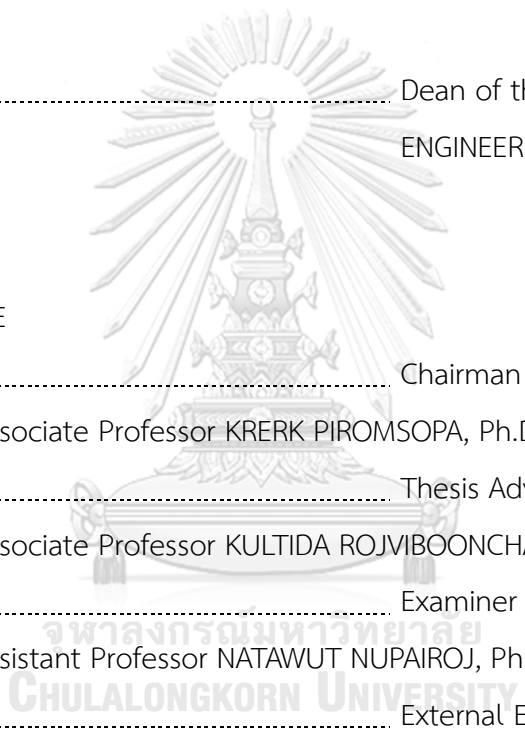


วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

Thesis Title Service Placement Optimization for Location-Based
Service
By Mr. Karnkitti Kittikamron
Field of Study Computer Engineering
Thesis Advisor Associate Professor KULTIDA ROJVIBOONCHAI, Ph.D.

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in
Partial Fulfillment of the Requirement for the Master of Engineering

..... Dean of the FACULTY OF
ENGINEERING
()
THEESIS COMMITTEE
..... Chairman
(Associate Professor KRERK PIROMSOPA, Ph.D.)
..... Thesis Advisor
(Associate Professor KULTIDA ROJVIBOONCHAI, Ph.D.)
..... Examiner
(Assistant Professor NATAWUT NUPAIROJ, Ph.D.)
..... External Examiner
(Associate Professor Anan Phonphoem, Ph.D.)



กานต์กิตติ กิตติคำรณ : การเพิ่มประสิทธิภาพตำแหน่งบริการสำหรับบริการตาม
ตำแหน่ง. (Service Placement Optimization for Location-Based Service) อ.ที่
ปรึกษาหลัก : รศ. ดร.กุลธิดา โรจน์วิบูลย์ชัย

บริการตามตำแหน่ง (LBS) จำเป็นและมีประโยชน์สำหรับแอปพลิเคชันต่างๆ มากมาย เช่น ระบบการนำทาง และเกม แอปพลิเคชันเหล่านี้ต้องการความแม่นยำสูงและความล่าช้าต่ำ โดยทั่วไป ความซับซ้อนของอัลกอริธึมการระบุตำแหน่งภายในอาคารที่ใช้ใน LBS จะขึ้นอยู่กับขนาดของข้อมูลลายนิ้วมือ สิ่งนี้สามารถนำไปสู่ความล่าช้าที่ยาวนาน เมื่อใช้งานในพื้นที่ขนาดใหญ่ ในบทความนี้ เราเสนอกรอบงานการปรับแบบแผนสำหรับการวางตำแหน่งบริการที่ชอบ โดยมีเป้าหมายเพื่อลดต้นทุนโดยรวมของการปรับใช้ การประมวลผลที่ชอบและเวลาตอบสนองของบริการให้น้อยที่สุด กลยุทธ์ตำแหน่งของเราใช้เพื่อแก้ปัญหาการวางโหนดชอบ วิธีการหลอมจำลอง จะถูกนำมาใช้ในการสำรวจปริภูมิคำตอบ เพื่อค้นหาคำตอบที่เหมาะสมที่สุดอย่างมีประสิทธิภาพ ผลลัพธ์แสดงให้เห็นว่ากรอบงานที่เราเสนอสามารถทำงานได้ดีกว่างานที่มีอยู่ โดยมีปรับปรุงเวลาตอบสนองของบริการถึง 30.50% จากการใช้ข้อมูลจำลองในกรทดสอบ และถึง 63.25% จากการใช้ข้อมูลขนาดใหญ่ในโลกแห่งความเป็นจริงในการทดสอบ

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

สาขาวิชา วิศวกรรมคอมพิวเตอร์
ปีการศึกษา 2565

ลายมือชื่อนิสิต
ลายมือชื่อ อ.ที่ปรึกษาหลัก

6470421621 : MAJOR COMPUTER ENGINEERING

KEYWORD: localization service, indoor localization, large-scale, edge computing, optimization problem

Karnkitti Kittikamron : Service Placement Optimization for Location-Based Service. Advisor: Assoc. Prof. KULTIDA ROJVIBOONCHAI, Ph.D.

Location-based service (LBS) is necessary and useful for several applications including navigation and games. These real-time applications require high accuracy and low delay. In general, the complexity of indoor localization algorithms used in LBS depends on the size of fingerprint data. This can lead to long delays when operating in large-scale areas. In this paper, we propose a novel optimization framework for edge service placement, aiming at minimizing the overall cost of edge computing deployment and service response time. Our placement strategy is used to solve the formulated edge node placement problems. The simulated annealing approach is then used in solution space exploration to discover the optimal solution efficiently. The results show that our proposed framework can outperform the existing work with a 30.50% improvement in the service response time on the simulated data, and a 63.25% improvement in the service response time on the real-world large-scale data.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Field of Study: Computer Engineering

Student's Signature

Academic Year: 2022

Advisor's Signature

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deep appreciation and gratitude to the individuals who have played a significant role in the completion of my thesis. I am immensely grateful to my thesis advisor, Assoc. Prof. Kultida Rojviboonchai, Ph.D., for her unwavering guidance, expertise, and encouragement. Her insightful feedback, invaluable suggestions, and dedication to my intellectual growth have been instrumental in shaping the direction and quality of this work.

I extend my heartfelt appreciation to the members of my thesis committee, namely, Assoc. Prof. Krerk Piromsopa, Ph.D., Asst. Prof. Natawut Nupairoj, Ph.D., and Assoc. Prof. Anan Phonphoem, Ph.D., for their valuable insights, thoughtful critique, and expertise. Their constructive comments and suggestions have significantly enriched the scope and depth of my research.

I would like to extend my deepest appreciation to my family and friends for their unwavering support, understanding, and encouragement. Their belief in my abilities, patience, and continuous motivation have been my pillars of strength throughout this challenging endeavor.

I am grateful for the financial support provided by Chula Computer Engineering Graduate Scholarship for CP Alumni, awarded by the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University. Their generosity allowed me to focus on my research and provided essential resources for data collection and analysis.

Karnkitti Kittikamron

TABLE OF CONTENTS

	Page
.....	iii
ABSTRACT (THAI).....	iii
.....	iv
ABSTRACT (ENGLISH).....	iv
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS.....	vi
LIST OF TABLES.....	ix
LIST OF FIGURES.....	x
1. Background and Signification of the Research Problem.....	1
1.1. Design Goals.....	3
1.2. Scope and Assumption.....	3
1.3. Academic Values.....	4
1.4. Process of Thesis.....	4
1.5. Publication.....	5
1.6. Thesis Organization.....	5
2. Theoretical Background and Literature Review.....	6
2.1. Location-Based Services.....	6
2.2. Indoor Localization.....	6
2.3. Edge Node Placement Strategy.....	7
3. Research Methodology.....	9
3.1. Problem Formulation.....	9

3.2. Framework Architecture.....	10
3.3. Input Processing Module	11
3.4. Placement Strategy Module	12
3.5. Solution Evaluation Module	14
3.6. Solution Space Exploration Module.....	16
4. Measurement and Evaluation.....	19
4.1. Experimental Setup	19
4.1.1. Development Tools.....	19
4.1.2. Scenario	19
4.1.3. Input Dataset	19
4.1.3.1. Simulated Network Topologies.....	19
4.1.3.2. Real-World Large-Scale Data.....	20
4.1.4. Parameter Configuration.....	23
4.1.5. Workload Modeling.....	25
4.1.5.1. Exploring the relationship between the size of fingerprint data and the processing time experiment.....	25
4.1.5.2. Exploring the relationship between the size of workload and the processing time experiment	26
4.1.5.3. Workload Modeling in the Experiment	27
4.2. Metrics	28
4.2.1. Total cost.....	28
4.2.2. Average response time	28
4.2.3. The number of edge nodes.....	28
4.3. Benchmarks	28

4.3.1. Full edge deployment	28
4.3.2. Centralized deployment	28
4.3.3. EdgeOn Framework [15]	29
4.3.4. Betweenness Centrality [21]	29
4.3.5. Degree Centrality [21]	29
4.4. Experimental Results	29
4.4.1. Simulated Networks Topology	29
4.4.2. Real-World Large-Scale Data	31
4.5. Result Discussion	34
4.5.1. Workload Imbalance	34
4.5.2. Cost Analysis	36
5. Conclusion	38
5.1. Thesis Summary	38
5.2. Discussions on the Fingerprint Data	39
5.3. Discussions on Historical Usage Quantity Data	39
5.4. Discussions on Performance Trade-Off	40
5.4.1. Parameter Tuning	41
5.5. Discussions on Centralized Deployment	42
5.6. Discussions on Workload Imbalance	45
5.7. Discussions on Application	45
5.8. Discussions on Limitations and Future Works	46
REFERENCES	48
VITA	51

LIST OF TABLES

	Page
Table 1 Hybrid Simulated Annealing's Parameter Configuration	24
Table 2 Cloud and Edge Server Parameter Configuration	24
Table 3 The experimental results for the large-scale area consisting of 37 multi-floor buildings using linear workload modeling	32
Table 4 The experimental results for the Central Shopping Mall Group data using logarithmic workload modeling.....	33
Table 5 The result of our proposed framework tested on the Central Shopping Mall Group data from 4.4.2(b) in detail.....	34
Table 6 The summary of the number of requests served by an edge node and the size of the fingerprint of each edge node data from 4.4.2(b).....	35
Table 7 The cost analysis of the experimental results on the Central Shopping Mall Group data	37

LIST OF FIGURES

	Page
Figure 1 Edge node placement illustration	9
Figure 2 Problem formulation illustration	10
Figure 3 Framework Architecture	11
Figure 4 The example results of the Input Processing Module	11
Figure 5 The example of the simulated network topology.....	20
Figure 6 37 Buildings in Chulalongkorn University from Ref. 13	21
Figure 7 Map of Central Shopping Mall Group used in the experiment.....	22
Figure 8 The example of Central Shopping Mall Group data	23
Figure 9 Scatter plot of the size of fingerprint data and the processing time.....	25
Figure 10 Scatter plot of the number of requests and the processing time	27
Figure 11 The impact of topology sizes on (a) Average service response time; (b) Total cost; (c) Number of edge nodes.....	30
Figure 12 The histogram of the number of requests served by an edge node	35
Figure 13 The histogram of the size of fingerprint data of each edge node	36
Figure 14 The example of fingerprint data of an Adaptive Indoor Localization System for Large-Scale Area from Ref. 13	39
Figure 15 The impact of topology sizes of simulated data using the new centralized deployment parameter on (a) Average service response time; (b) Total cost; (c) Number of edge nodes	44
Figure 16 Bandwidth and latency requirements for different application [22]	46

1. Background and Signification of the Research Problem

As smartphones are the part of people's everyday life over the last few years, one of the most useful services in this story is location-based service. Location-based service is a service that uses geospatial data and content preference combined to provide suitable information to users based on their location [1]. This service has been used vastly in many applications such as turn-by-turn navigation, shopping mall advertising, games and entertainment (AR/VR), social network and assistive healthcare systems, etc. which are delay-sensitive applications and require low latency. Positioning is crucial to LBS and is mostly supported in both outdoor environments handled by GPS services and indoor environments handled by indoor location services.

Several techniques have been proposed for indoor localization recently including the techniques that do not require the pre-installation of infrastructure which is by far more popular due to cost saving and accuracy [2]-[3]. The most popular technique that does not require the pre-installation of infrastructure is the Wi-Fi fingerprint technique [4]. Wi-Fi fingerprints can achieve high accuracy by using the assumption that each location has a specific signal characteristic like fingerprints which can be used to represent that location. There are two phases in this technique: the training phase and localizing phase. In the training phase, a site survey is performed to construct fingerprints of all targeted positions and store them in the database. In the localizing phase, the user's Wi-Fi fingerprint is compared to all fingerprints in the database by using a localizing algorithm and then returns the user's position as a result.

When it comes to large-scale area scenarios like multi-floor building environments, the size of fingerprints can be massive due to the high resolution of location tags indicating which position on which floor of which building. As a result, the indoor localization system contains a large size of fingerprints which causes an increase in processing time usage of the localization process affecting overall performance [5]. This problem results from cloud-centric architecture deployment which is a conventional architecture used in the present application. To illustrate,

indoor localization only needs the fingerprint data of a focused area such as a building or a building cluster in localization; all the large fingerprint data stored in the server containing unnecessary fingerprints are processed in localization leading to a long processing time problem. Moreover, the service is usually deployed on computational resources on the cloud which is far away from users resulting in a long delay, large workload, and low scalability [6]. To sum up, the current architecture can cause an increase in response time including propagation delay and localization processing time.

Edge computing is a distributed computing paradigm bringing computational resources and data storage closer to end users at the edge of the network which can provide new opportunities for efficient operation such as reducing latency and saving bandwidth. Edge computing could be applied to enhance the performance of indoor localization in terms of latency by distributing services close to the user and in terms of processing time by implementing only necessary data according to data locality.

Although, deploying the service on the edge server at each building can solve the problems magnificently, in a real situation, not every building has enough resources or is the potential to place an edge server due to cost limits. When it comes to edge computing implementation, the edge node placement strategy is the crucial part to manage how to divide the workload for edge nodes and where to put them. There are several edge node placement strategies proposed in the past few years which mostly focus on optimizing the algorithm to balance targeted metrics such as latency, energy consumption, resource utilization, cost, workload, etc. [7]-[8]. In indoor localization in large-scale areas, the size of fingerprint data of each building is different. Therefore, the time used in the localization process varies at different buildings. Misplaced services with varied building fingerprint data can increase the size of the fingerprint at each node resulting in long processing time and poor network traffic. Moreover, usage quantity varies depending on each location in the system [9]. The absence of this consideration of usage quantity results in misplacing service at an infrequently used node. There is no edge node placement strategy considering the difference in time used in processing and usage quantity at each node which impacts the performance of the positioning service.

To overcome the above-mentioned limitation, in this paper, we proposed an edge service placement optimization framework aiming to achieve better latency, cost, and response time including localization processing time and propagation delay by considering data locality dependency and historical usage quantity data or Point of Interest (POI). We developed a framework for solving edge node placement and used indoor localization service as a use case example in testing. In addition, our proposed placement optimization could be applied to not only positioning services but also services whose performance relies on data locality and data size.

1.1. Design Goals

This research has been studied to propose an edge node placement framework for location-based service with the restriction of real-time response. The optimal placement solution needs to divide the workload and allocate the service at a suitable size and place it at the optimal location to satisfy the requirement of LBS. The optimization objectives include the overall cost of edge node network deployment, the number of edge nodes deployed in the network, and the overall response time including propagation delay and localization processing time.

1.2. Scope and Assumption

The scope of this thesis is limited to the followings:

- This thesis considers optimizing the placement strategy based on static placement optimization using the input information only once.
- In terms of the computational complexity of each task, the size of fingerprint data has been used to calculate the size of time and conclude the processing time.
- This thesis focuses on achieving the optimized service response time including localization processing time and propagation delay using the average value.
- This thesis did not aim at improving any localization methods.

- In order to evaluate the proposed framework, the framework was implemented as modules by using Python. The network environment is formed by using the NetworkX library.
- This thesis doesn't concern service replica; each service presents at only one node.
- The issue of node and link failures is not taken into consideration in this thesis.
- All server resources are only reserved for the target service in service placement.

1.3. Academic Values

1. Edge node placement optimization framework for location-based services can be used in not only positioning services but also services whose performance relies on data locality and data size.

2. Edge node placement optimization framework can be tailored to the specification of targeted applications by adjusting the weights of metrics used in optimization.

1.4. Process of Thesis

1. Study and research related topics about fingerprint-based indoor localization
2. Study and research related topics about edge node placement strategy
3. Preliminary experiment
3. Design the edge service placement framework
4. Framework development
5. Design the edge service placement experiments using indoor localization as a use case
6. Prepare the simulated network topology data set
7. Test the experiment on the simulated input data from 6.
8. Analyze and compare the results from 7.
9. Prepare the real-world large-scale data set
10. Test the experiment on the real-world large-scale data set from 9.

11. Analyze and compare the results from 10.
12. Summarize and discuss the experimental results

1.5. Publication

“Edge Service Placement Optimization for Location-Based Service” by Karnkitti Kittikamron, Natthanon Manop, Adsadawut Chanakitkarnchok, and Kultida Rojviboonchai published and presented at the international academic conference named “The 20th International Joint Conference on Computer Science and Software Engineering (JCSSE2023)” at Phitsanulok, Thailand on June 28th – July 1st 2023.

1.6. Thesis Organization

The rest of the dissertation is organized as follows. The next chapter describes the theoretical background of location-based service, indoor localization, and edge node placement strategy. This chapter also includes the literature review contributed to this thesis. Chapter 3 explains the research methodology including the problem formulation, the architecture of the proposed framework, and also each module of the proposed framework. In Chapter 4, we evaluate the performance of our proposed framework compared with the previous approaches. Finally, Chapter 5 concludes the thesis and discussion for further research.

2. Theoretical Background and Literature Review

2.1. Location-Based Services

There are three basic components in the service: positioning, which is the part that determines the user's location, modeling which is the part that models context and characteristics adapted to the location of users and communication which is the part that provides relevant information from LBS applications to the users [10].

2.2. Indoor Localization

Several techniques have been applied for indoor localization [11]-[12]. The most popular technique is the Wi-Fi fingerprint. Due to no additional infrastructure installation required and availability in commercial smartphones, Wi-Fi could offer more ease of use and high accuracy, even though time and manpower are taken for site surveys. In the large-scale area, long processing time and accuracy are common problems. Many indoor localization techniques using area classification have been proposed aiming to deal with out-of-scope area data which can cause computational resource waste.

Adaptive Indoor Localization System for Large-Scale Area [13] proposed an indoor localization system for the large-scale area containing three main parts. First, an area classification is designed for identifying an area of the user's queries by filtering out the unknown data and out-of-scope area queries sent from outdoors and locating a building of the queries. Second, a fingerprint-based indoor localization algorithm uses the information from the previous part to localize the exact location of those queries. Third, the missing-BSSID detector algorithm handles the changing environment by detecting the missing BSSID from queries and updating the database. The proposed indoor localization system can achieve high accuracy and reduce the overall processing time in changing environments. Nevertheless, the system shows that the size of fingerprint data can affect the fingerprint-based indoor localization algorithm processing time resulting from cloud-centric architecture which also affects service propagation delay.

2.3. Edge Node Placement Strategy

Madamori et al. [14] proposed a cost-effective edge node placement in smart cities that opportunistically leverage public transit network strategy aiming to minimize overall delivery latency within a budget. Not only single-objective optimization has been studied, but there are also many multiple-objective optimizations have been proposed for optimizing more than one metric such as latency, resource utilization, cost, energy consumption, QoE, etc.

EdgeOn is a framework tailored to a 5G-EC ecosystem based on three key aspects such as cost, number of edge nodes, and capacity usage ratio [15]. The framework was developed by ensuring ultra-low latency demand compliance to address the strict latency and reliability demands of 5G. The main module of the framework works with three core processing stages and an output stage. First, Input Processing takes network topology as input and normalizes it to meet the 5G use case requirements. Second, to handle real data input, Scenario Generation implemented a network emulator to provide test scenarios. Next, the Placement Optimization phase is the key of the framework to return an optimized placement solution comprising the Pre-Optimization module which aims to reduce the problem complexity by separating the region of interest and omitting isolated service consumer nodes, Placement Strategy module implements two different greedy and scored algorithms to greedily pair service consumer and edge (service provider) nodes considering consumer requirements, network usage, edge node capacities, etc. and Solution Space Exploration module proceeding the pairing algorithm conforming the strictly constrained and scoring solution considering multi-objective to determine the Pareto front out of NP-hard problems. In the Solution Space Exploration module, the Hybrid Simulated Annealing technique was applied combining the concept of Traditional Simulated Annealing and Tabu Search algorithm to obtain a strong ability to escape local optima throughout the solution space exploration. Finally, the Output stage returns the optimum solution containing the set of edge node locations to place the service infrastructure. The framework can achieve 30% less average ENs deployed and a 25% higher average usage ratio. Nevertheless, the framework provides the best solution for only general applications and not for the application

that has concern dependencies at the data level. Moreover, the framework doesn't consider the historical data of usage quantities in solution exploration.



3. Research Methodology

3.1. Problem Formulation

In a large-scale area like multi-floor buildings, cloud-centric architecture or centralized deployment cannot satisfy low delay requirements of location-based services due to limitations in high localization processing time, propagation delay, and bandwidth.

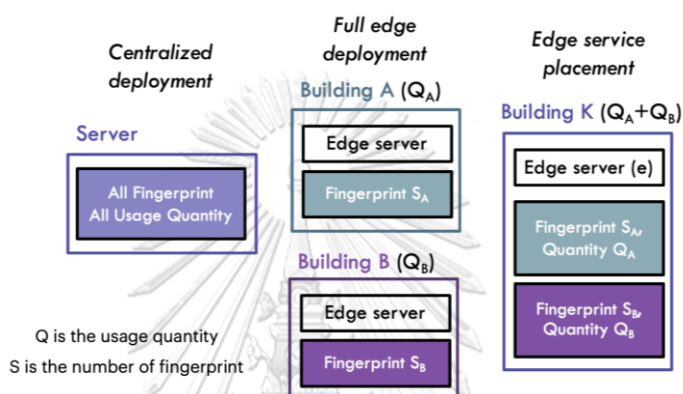


Figure 1 Edge node placement illustration

Edge computing brings computational resources and data storage closer to end users at the edge of the network which is suitable for low processing tasks and results in low latency and bandwidth. Integrating edge computing could be one of the good options. By deploying an edge server with service implemented at every building as shown in Figure 1, the stated problems are solved. However, not all buildings are the potential to place edge servers because of inadequate resources and limited budgets. This problem could be stated as an edge node placement problem where the problem can be solved by edge node placement strategies.

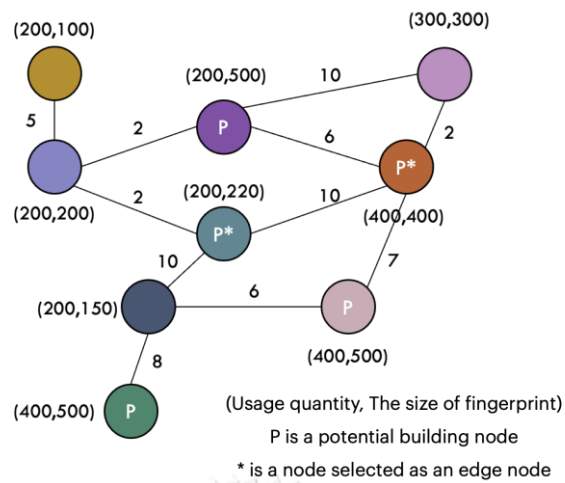


Figure 2 Problem formulation illustration

To illustrate, the goal of the solution is to select potential building nodes as edge nodes to place indoor localization services. As each building node holds a different size of the fingerprint which reflects the computational complexity and the different number of usage quantities shown in Figure 2, placing service needs to consider these characteristics concurrently. Therefore, this problem was formulated as a multi-objective edge node placement optimization problem.

$$\text{Min } w_1 * \text{Total cost} + w_2 * \text{Number of edge nodes} + w_3 * \text{Average response time} \quad 3.1.1$$

Equation 3.1.1 showed the objective function of the multi-objective optimization aiming at the minimized total cost, the minimized number of edge nodes, and the minimized average response time including localization processing time and propagation delay.

3.2. Framework Architecture

In this study, an edge service placement optimization framework for location-based service has been proposed composed of four main modules.

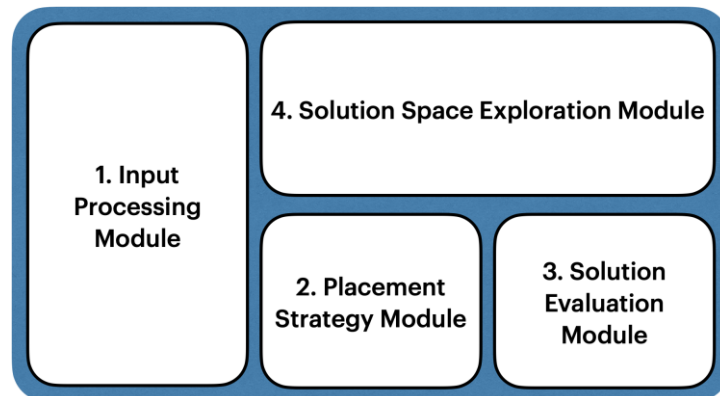


Figure 3 Framework Architecture

As shown in Figure 3, the proposed framework consists of four main modules working together to take a network topology as input and return the placement solution. The framework includes the Input Processing Module which handles the network topology input, the Placement Strategy Module which generates placement solutions, the Solution Evaluation Module which evaluates the quality of a solution in terms of score, and the Solution Space Exploration Module which explores the solution space to find an optimal solution by cooperating with the two prior modules.

3.3. Input Processing Module

This module transforms a network topology input into a simplified form and labels node and edge information.

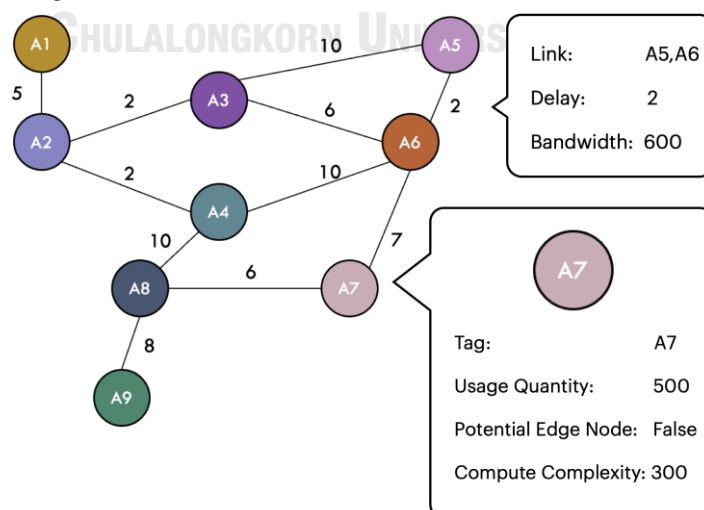


Figure 4 The example results of the Input Processing Module

Figure 4 shows the example of input network topology having been processed by the Input Processing Module. Each node represents a service node in the input network topology and holds attributes such as the number of historical usage quantities, potential edge node status, and compute complexity (the size of the fingerprint data). Each edge represents a link in the input network topology and holds link delay and bandwidth value.

3.4. Placement Strategy Module

There are nodes in the input network topology. At first, all nodes are unserved. An unserved node is a node that holds user demand and is not served. Then some nodes are selected to be edge nodes and serve the other unserved nodes. This module is responsible for solution generation by using a greedy algorithm that selects potential nodes as edge nodes and pairs them with unserved nodes. A potential node is a node that has adequate resources to place the service and to serve unserved nodes.

Algorithm 1 Edge Placement Algorithm

Input: a list of nodes, *delay* the maximum delay, an edge selection order, mode (*initial_mode*, *neighbor_mode*)

Output: an edge list (*El*), an edge selection order (*Eo*)

- 1: *sorted_P* \leftarrow list of potential nodes
- 2: *unserved_N* \leftarrow set of unserved nodes
- 3: **while** (*unserved_N* is not empty) **do**
- 4: **if** (*mode* is *initial_mode*) **then**
- 5: sort *sorted_P* by node's data size * node's usage quantity
- 6: **else**
- 7: *sorted_P* \leftarrow edge selection order
- 8: **end if**
- 9: *edge* \leftarrow the first node of *sorted_P*
- 10: *edge_list* \leftarrow add *edge*
- 11: *neighbor_unserved* \leftarrow all neighbor nodes of *edge* which is unserved reached within *delay* and its usage quantity not exceeding *edge*'s current capacity
- 12: sort *neighbor_unserved* by total data size * total usage quantity of each node and the *edge*
- 13: *served* \leftarrow the first node of *neighbor_unserved*
- 14: edge serving node operation
- 15: **end while**
- 16: **return** *El*, *Eo*

Our edge placement algorithm is shown in Algorithm 1. The purpose of the algorithm is to find an initial solution using a greedy algorithm. In the normal pairing process (*initial_mode*), a potential node with the least product of the usage quantity and the size of fingerprint data is selected as an edge node. Next, all nodes that are not edge nodes and have exceeding workload are filtered out, so that they could be selected as edge nodes later. Then the remaining nodes are sorted by the product of the sum of usage quantities between a node and edge node and the sum of the size of fingerprint data between the node and edge node. The one with the least product is chosen to pair with that edge node and becomes its served node. In the edge serving node, the shortest path between the edge node and served node is used, and all requests or the workload of the served node are served by the edge node. The service at the edge node also includes the fingerprint data of the served node. As the module process goes on, the order of edge selection is recorded. After the initial solution is retrieved, the module returns the answer consisting of an edge list

which is the list of edge nodes with their served nodes and the edge selection order. The algorithm can also be used in neighbor solution generation. If an edge selection order is given to the algorithm using *neighbor_mode*, in the pairing process, edge nodes are selected in order of the edge selection order input instead of the least product method.

3.5. Solution Evaluation Module

To know the quality of any solutions, each solution needs to be measured in terms of metrics and calculated as a score. This module evaluates solutions to make it be comparable to each other in different aspects such as total cost, number of edge nodes, and average response time including localization processing time and propagation delay. Each factor is defined as follows:

$$TC = \sum_{i=1}^n C_i * E_i \quad 3.5.1$$

where C_i is the compute complexity of node i ,

E_i is the status of node i which is 1 if node i is edge node and 0, otherwise.

$$Total\ network\ cost = \sum_{i=1}^n \sum_{j \in S_i} \sum_{p \in P_{ij}} k_p * E_i \quad 3.5.2$$

where P_{ij} is the set of links in the path from node i to node j ,

S_i is the set of nodes served by edge node i ,

k_p is the network cost of link p ,

E_i is the status of node i which is 1 if node i is edge node and 0, otherwise.

$$Total\ upfront\ cost = \sum_{i=1}^n U_i * E_i \quad 3.5.3$$

where U_i is the upfront deployment cost of node i ,

E_i is the status of node i which is 1 if node i is edge node and 0, otherwise.

$$Total\ cost = Total\ capacity\ cost + Total\ network\ cost + Total\ upfront\ cost \quad 3.5.4$$

$$\text{Total propagation delay (PD)} = \sum_{i=1}^n \sum_{j \in S_i} \sum_{p \in P_{ij}} d_p * E_i * Q_j \quad 3.5.5$$

where Q_i is the usage quantity of node i ,

d_p is the delay of link p ,

E_i is the status of node i which is 1 if node i is edge node and 0, otherwise

$$\text{Total localization processing time (LPT)} = \sum_{i=1}^n (E_i * (\sum_{j \in S_i} C_j * \sum_{k \in S_i} Q_k)) \quad 3.5.6$$

where Q_i is the usage quantity of node i ,

C_i is the compute complexity of node i ,

S_i is the set of nodes served by edge node i ,

E_i is the status of node i which is 1 if node i is edge node and 0, otherwise.

$$\text{Total usage quantity (UQ)} = \sum_{i=1}^n Q_i \quad 3.5.7$$

where Q_i is the usage quantity of node i

$$\text{Average localization processing time (avgLPT)} = \text{LPT} / \text{UQ} \quad 3.5.8$$

$$\text{Average response time (avgRT)} = (\text{LPT} + \text{PD}) / \text{UQ} \quad 3.5.9$$

$$\text{Number of edge nodes} = \sum_{i=1}^n E_i \quad 3.5.10$$

where E_i is the status of node i which is 1 if node i is edge node and 0, otherwise.

$$\text{Score} = w_1 * \text{Total cost} + w_2 * \text{Number of edge nodes} + w_3 * \text{Average response time} \quad 3.5.11$$

Equation 3.5.4 showed that total cost is the sum of the total capacity cost shown in Equation 3.5.1, which is the sum of the capacity unit of each edge node, and total network cost shown in Equation 3.5.2, which is the sum of the network unit used in network traffic between each edge node and its served nodes, and total upfront cost shown in Equation 3.5.3, which is the sum of edge server deployment cost of each edge node. The number of edge nodes is counted shown in Equation 3.5.10. Equation 3.5.9 showed that the average response time is the average of the sum of

total localization processing time from Equation 3.5.6 and the total propagation delay from Equation 3.5.5 over total usage quantity from Equation 3.5.7. To summarize the difference metrics into a score, each metric is normalized by using relative calculation with the initial solution. Then with provided weights that emphasize how much each metric is important, the score is calculated using a weighted sum approach as shown in Equation 3.5.11.

3.6. Solution Space Exploration Module

As the edge node placement problem was proved to be an NP-hard problem [16]-[17], the solution space can be vast that the optimal solution cannot be obtained in polynomial time. To find the optimal solution, the solution space needs to be explored efficiently.



Algorithm 2 Hybrid Simulated Annealing

Input: T_{min} minimum temperature, T initial temperature, I number of iterations, n number of neighbor solutions, NL a list of nodes, α_f fast T decrease, α_s slow T decrease

Output: an edge list (El)

- 1: $initial_sol \leftarrow placement(NL, initial_mode)$ // generate an initial solution
- 2: $best_sol \leftarrow initial_sol$
- 3: $best_sol_all \leftarrow initial_sol$
- 4: $worse_sol \leftarrow null$
- 5: $i \leftarrow 0$
- 7: **while** ($T > T_{min}$) **do**
- 8: **while** ($i < I$) **do**
- 9: $edge_orders \leftarrow mutate(best_sol, n)$ // generate n edge selection orders by mutating $best_sol$'s edge selection order
- 10: $N \leftarrow placement(NL, edge_orders, neighbor_mode)$ // generate n number of neighbor solutions
- 11: $S \leftarrow score(N)$ // score the solutions
- 12: **if** ($score(S[0]) < score(best_sol)$) **then**
- 13: $b_sol \leftarrow S[0]$
- 14: **else**
- 15: $p \leftarrow ap(T, score(S[0]), best_sol)$
- 16: **if** $p > random(0, 1)$ **then**
- 17: $best_sol \leftarrow S[0]$
- 18: **end if**
- 19: $worse_sol \leftarrow random(S)$
- 20: $i \leftarrow i + 1$
- 21: **end while**
- 22: **if** $score(best_sol) < score(best_sol_all)$ **then**
- 23: calculate the probability to adjust T , n , and $best_sol_all$
- 29: **else**
- 30: calculate the probability to adjust T and n
- 31: **end if**
- 32: **end while**
- 33: **return** $El(best_sol_all)$

This module applies the Simulated Annealing approach [18], which is a stochastic global search algorithm for combinatorial optimization problems and collaborates with Placement Strategy Module and Score Evaluation Module to find the optimal solution. The point of this method is to escape local optima and discover global

optima by keeping the bad solution in the possible solution list because good solutions may be generated from the bad solution.

Algorithm 2 shows the Hybrid Simulated Annealing. After the initial solution has been obtained by using Algorithm 1, the edge selection order is mutated into similar edge selection orders by shuffling, randomizing, swapping, and concatenating the initial edge selection order. Then neighbor solutions are generated using Algorithm 1 by taking the mutated edge selection orders as input. The weights for evaluating each metric of solutions are determined and sent together with the outcome solution to the Score Evaluation Module. The exploration continues by using the objective function as shown in Equation 3.1.1 through the score calculation as shown in Equation 3.5.11. The temperature of the Simulated Annealing is adjusted until it reaches the minimum temperature. In the end, the optimal solution is found and returned as the output. The output includes an edge list which is the list of edge nodes with their served nodes. This suggests where to deploy edge nodes and where to put which service to achieve the optimization goal.

The time complexity of simulated annealing from Algorithm 2 can be approximated as $O(k*N)$, where k is the number of iterations or steps taken by the algorithm and N is the size of the problem space. The size of the problem space can be analyzed from Algorithm 1. The placement strategy algorithm shows the size of the problem space is n^2 where n is the number of nodes in input topology. Therefore, the overall complexity of the framework is $O(k*n^2)$ where k is the number of iterations of the simulated annealing and n is the number of nodes in input topology. The time complexity analysis usually focuses on the expected performance over multiple runs rather than the worst-case scenario.

4. Measurement and Evaluation

4.1. Experimental Setup

4.1.1. Development Tools

To evaluate the performance of the proposed framework, the framework including the Input Processing module, the Placement Strategy module, the Score Evaluation module, and the Solution Space Exploration module was implemented using Python, a programming language. We also used Networkx, a Python package for the structure and functions of networks, to model the network topology.

4.1.2. Scenario

We used an indoor localization system as a use case in the edge node placement testing to test how the performance relies on data location and data size. The goal of the experiment was to find the nodes suitable for indoor localization service deployment among all network input nodes by considering cost and the metrics of concern using given network and application usage information.

4.1.3. Input Dataset

We need a network topology with the information of each edge including link bandwidth, and link delay, and the information of each node including the number of requests, the size of fingerprint data, and the potential to place service status for the experiment. To evaluate the framework in a large-scale area use case, we conducted experiments on simulated network topologies with different numbers of nodes and on large-scale data from an indoor localization service and from a country-scale shopping mall group data.

4.1.3.1. Simulated Network Topologies

For experiments on the simulated network topologies, 8 topologies were generated with the numbers of nodes ranging from 10 to 80 nodes. For each topology, nodes with coordinate pairs were plotted on a 2D grid map scatteredly and edges were randomly linked between nodes. The usage quantity and data size of each node was assigned with random values between 20,000 to 100,000 requests. All nodes have the potential to deploy edge servers. The link delay was calculated based on the Euclidean distance between two nodes. The maximum workload

capacity that each potential node can serve was set to be 3 times the maximum workload produced by each node. Each link holds the same bandwidth.

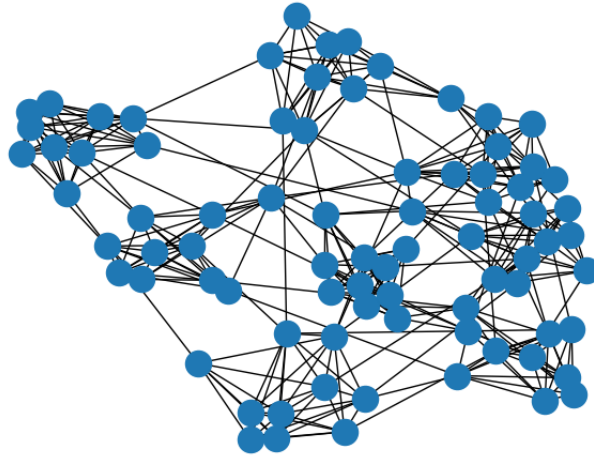


Figure 5 The example of the simulated network topology

Figure 5 depicts the example of the simulated network topology used in the experiment with the size of 80 nodes generated by using NetworkX. Each node's number of requests is randomly assigned with a value between 20,000 requests to 100,000 requests. Each node's size of fingerprint data is randomly assigned with a value between 50 to 500.

4.1.3.2. Real-World Large-Scale Data

To ensure that our proposed framework not only works in simulated network topologies, we used two real-world large-scale datasets as follows:

a) Wi-Fi fingerprint dataset of 37 multi-floor buildings at Chula Expo

The Wi-Fi fingerprint dataset of 37 multi-floor buildings collected at the Chula Expo Exhibition in 2017 was used to form input data [13]. To illustrate, each building represents each node in the network topology. Each node was plotted on a 2D grid map by its location.

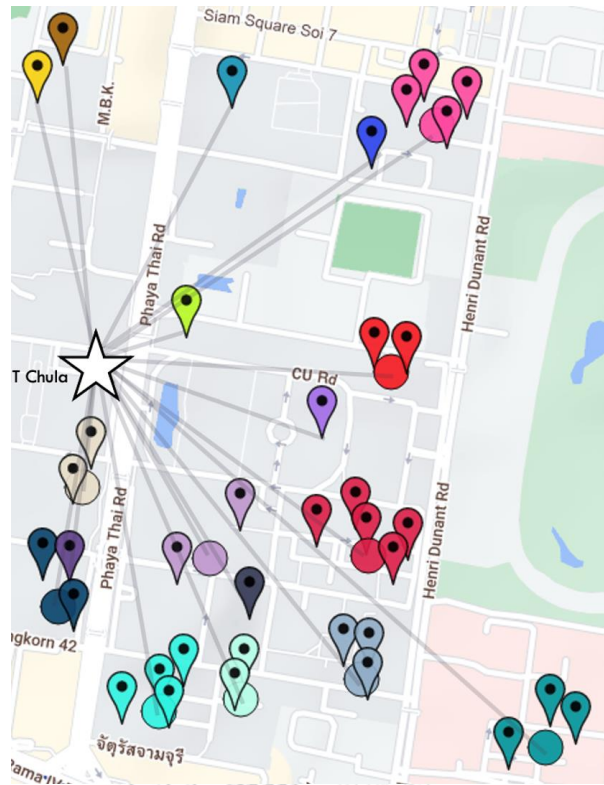


Figure 6 37 Buildings in Chulalongkorn University from Ref. 13

Figure 6 depicts the input topology used in the experiment showing the network topology occupying the campus over more than 10 clusters of buildings. For the network edges, there are links between each building node (Pin) and its faculty node (Circle) which is the centroid point of the cluster differentiated by colors and links between each faculty node and the IT center node. The link delay was calculated based on the geographical distance between buildings. To illustrate, the link with a longer distance will hold a longer propagation delay. The collected data includes (1) Wi-Fi fingerprint of each building retrieved from the access point scanning results, which represents the building id, floor, and position tag of each location, (2) the size of data of each building fingerprint which depends on the number of position tags within that building, (3) the network node topology consisting of each node and edge basic attributes, and (4) the historical service usage quantity obtained from the queries made by users on exhibition days at each building.

b) Central Shopping Mall Group

We also used another network topology dataset with country-scale size to emphasize the impact of propagation delay. A shopping mall is a place full of people

utilizing location-based services such as games, entertainment, advertising, navigation, etc. One of the largest and most popular shopping mall groups in Thailand is the Central Shopping Mall Group [19]. With marketing research and planning, each branch of the shopping mall is in a dense population and economic district area. The Central Shopping Mall Group is an interesting contender for the service placement strategy study due to the large number of branches and the topology scale which cover the whole part of Thailand. The information of Central Shopping Mall Group provided online is sufficient to use as network topology input of indoor localization use case in the experiment.

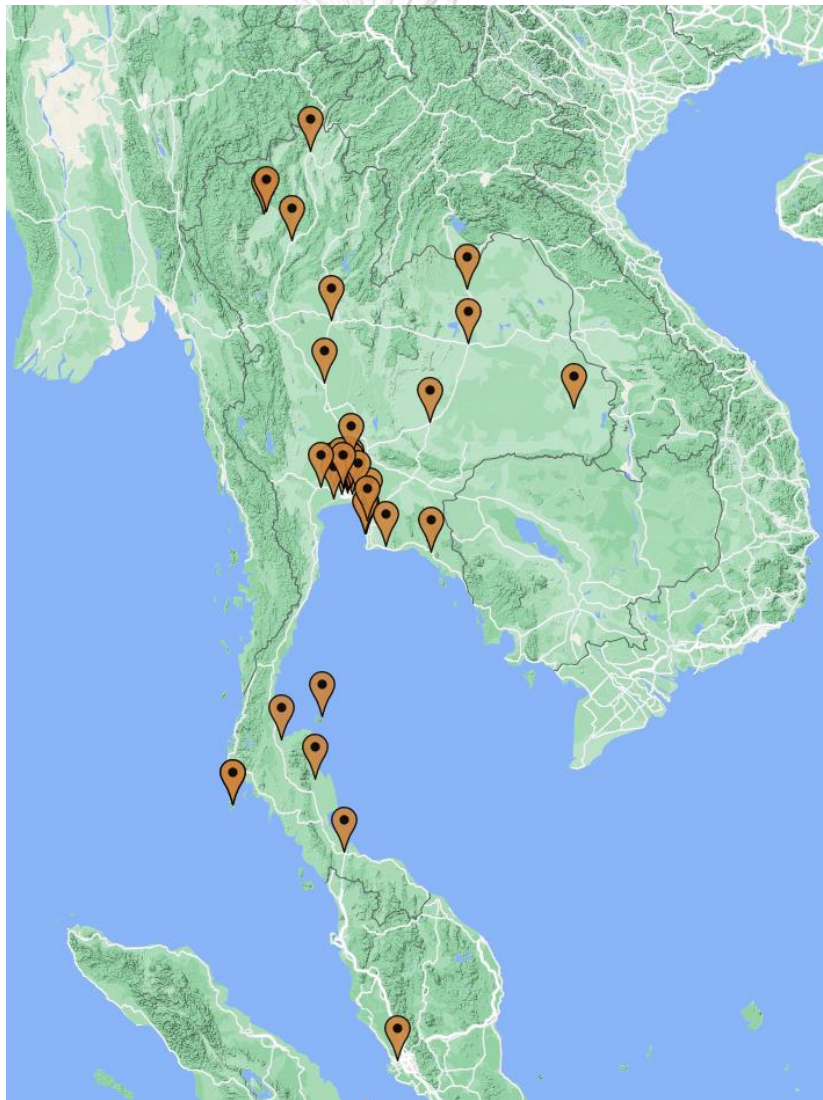


Figure 7 Map of Central Shopping Mall Group used in the experiment

Figure 7 depicts the network topology formed by using each shopping mall branch to represent each node. Each branch location was plotted on a real-world 2D map. On the map, there are 41 branches of Central Shopping Mall Group, for example, Central Lardprao located in Bangkok province in the central region of Thailand. As shown in the map, the number of branches is dense in some areas. For the network edges, there are links between each shopping mall node (Pin) and its region node (Circle) which is the centroid point of the cluster differentiated by colors. The link delay was calculated based on the geographical distance between the shopping mall. As the distance between each node is longer, we expected that the propagation delay would be more significant.

Name	Province	Lat	Lng	GLA (sq m)	Region	POI (Stores)	POI (Cust)
Central Lardprao	Bangkok	13.8161884	100.561057	78,700	C		339
Central Ramindra	Bangkok	13.8723394	100.6019646	23,500	C		145
Central Pinklao	Bangkok	13.7782234	100.4763161	104,500	C		255
Central Marina	Chonburi	12.945729	100.8902896	29,000	E		132
Central Chiangmai Airport	Chiang Mai	18.7693089	98.9752557	107,000	N		533
Central Rama III	Bangkok	13.6979914	100.537438	98,000	C		292
Central Bangna	Bangkok	13.66974489	100.634596	113,000	C		299
Central Rama II	Bangkok	13.6639306	100.4377225	161,500	C		350
CentralWorld	Bangkok	13.7460002	100.5399162	830,000	C		500
Central Rattanaibet	Nonthaburi	13.8661107	100.497032	105,000	C		216
Central Phuket – Festival	Phuket	7.8895826	98.3662517	300,000	S		300

Figure 8 The example of Central Shopping Mall Group data

For the node attributes, Figure 8 shows the example of each branch of Central Shopping Mall Group [19]. Besides the location of each node, we also retrieve the Gross leasable area (GLA) of each branch [19]. The Gross leasable area (GLA) data reflects the total floor area within a commercial property that is available for lease to tenants. It represents the space that can be rented out and generates rental income for the property owner or landlord. The usage quantity or the number of requests for each node was derived from the number of stores in each shopping mall. The size of fingerprint data was retrieved from the GLA data. All nodes were the potential nodes to place the service.

4.1.4. Parameter Configuration

In the experiment, the parameters of the Solution Space Exploration module in the proposed framework were defined.

Table 1 Hybrid Simulated Annealing's Parameter Configuration

Hybrid Simulated Annealing	
Minimum Temperature	1
Maximum Temperature	0.1
Temperature Iteration	30
The Number of Neighbors	12
Fast Alpha	0.80
Slow Alpha	0.95
The Weight of Total Cost (w1)	0.1875
The Weight of The Number of Edge nodes (w2)	0.625
The Weight of The Average Service Response Time (w3)	0.1875

For the Hybrid Simulated Annealing in the Solution Space Exploration Module, the parameters including the minimum temperature, the maximum temperature, the temperature iterations, the number of neighbors, the fast alpha, and the slow alpha were set as shown in Table 1. The weight values of w_1 , w_2 , and w_3 in the objective function of the Solution Space Exploration shown in Equation 3.1.1 and also in the score calculation of the Score Evaluation Module shown in Equation 3.5.11 are set to 0.1875, 0.625, and 0.1875, respectively.

Table 2 Cloud and Edge Server Parameter Configuration

Cost		
Server	Cloud	Edge
Computational Capability (unit)	10	1
Server Upfront Cost (unit)	1.5	1
Network Cost (unit)	1	0.1

Moreover, since we included the Centralized Deployment in the experiment benchmark, the parameters of the cloud server including the computational

capability, server upfront cost, and the network traffic cost were defined as shown in Table 2.

4.1.5. Workload Modeling

The localization processing time is one of the main factors in our optimization objective. When deploying service at only some edge nodes, some edge nodes need to have more than a single localization service that can serve not only their own building but also other buildings. As a result, the localization processing time increases due to the larger size of fingerprint data. To evaluate the total localization processing, we need to model a function to calculate the time used by each service for processing each request based on the size of fingerprint data.

4.1.5.1. Exploring the relationship between the size of fingerprint data and the processing time experiment

We implemented 37 localization services by using fingerprint data of each building from 37 multi-floor buildings data [13]. Then, requests to each service were made and the response time was recorded and calculated on average. The size of fingerprint data of each service varies between 38 to 513. The results were shown on a scatter plot between the size of fingerprint data and the processing time.

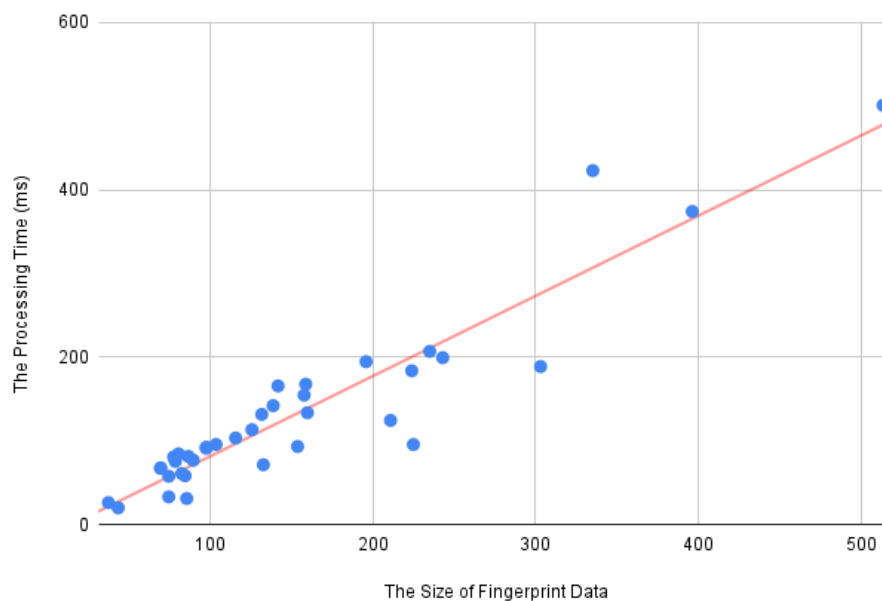


Figure 9 Scatter plot of the size of fingerprint data and the processing time

As shown in Figure 9, as the size of fingerprint data grows larger, the processing time tends to increase linearly according to the line of best fit (red). The linear equation of the line of best fit is $y = 0.9589x - 14.2306$, where x is the size of fingerprint data and y is the processing time. From the experimental result, we can conclude that the relationship between the size of fingerprint data and the processing time is linear.

There are other fingerprint-based indoor localizing algorithms with higher complexity than linear. For example, the complexity of a GraphSLAM-based crowdsourcing framework for indoor Wi-Fi fingerprinting is $O(N^2)$ where N is the number of the fingerprint data [12]. Therefore, we also modeled the workload with an exponential relationship between the size of fingerprint data and processing time to represent other complex applications as well.

In the real scenario, the workload size or the number of requests also affects the overall processing time. Consequently, the relationship between the processing time and workload is prone not to be a linear relationship due to high network traffic and limited resource utilization.

4.1.5.2. Exploring the relationship between the size of workload and the processing time experiment

We implemented a localization service for 37 multi-floor buildings [13]. Then, we deployed the service on a server. Next, load testing was done by requesting the service on the server to observe the change in the processing time at each workload. For the load testing, we use K6 which is an open-source load testing tool used to test the performance and scalability of web applications. It is designed to simulate virtual users or concurrent connections and generate load on the target system to measure its response time and throughput under different scenarios [20]. We assumed that the available time of service is 8 hours. The parameters of K6 were set as follows: the duration is 10 seconds, and the number of concurrent users varied between 10 users to 50,000 users.

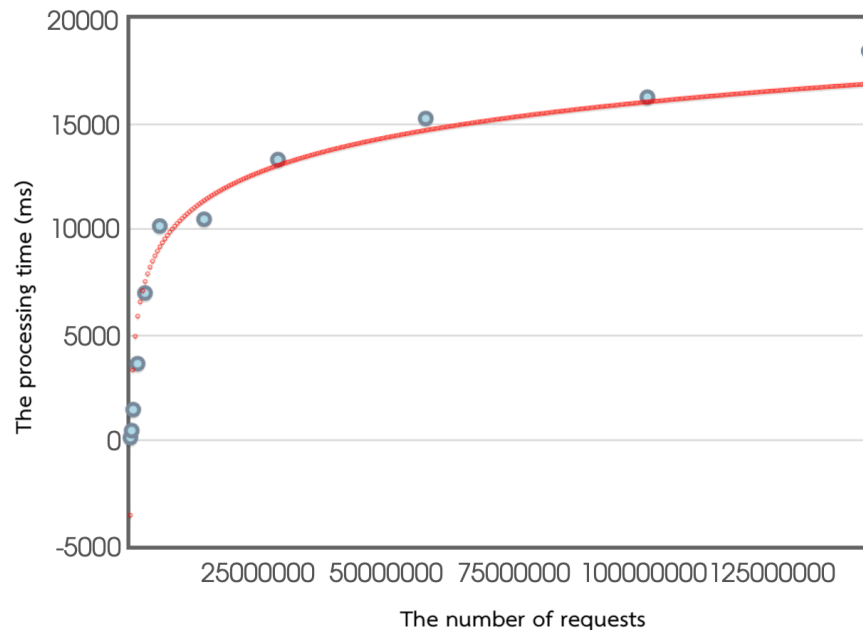


Figure 10 Scatter plot of the number of requests and the processing time

Figure 10 shows that as the number of requests grows larger, the processing time tends to increase rapidly at first and then slowly. According to the line of best fit (red), the regression equation of the line of best fit is $y = -28235.3892 + 2403.3221 \cdot \ln(x)$, where x is the number of requests and y is the processing time. We can conclude that the relationship between the size of the workload or the number of requests and the processing time is logarithmic.

4.1.5.3. Workload Modeling in the Experiment

As the relationship between the size of fingerprint data and the processing time has been proven to be linear, we will calculate the total localization processing time of any services in the experiment by using the sum of the size of fingerprint data. For the higher complexity application, we calculate the localization processing time using this equation: $y = 91.29498 \cdot 1.00442^x$ where x is the number of requests and y is the processing time.

As the relationship between the size of the workload or the number of requests and the processing time is logarithmic, we considered the impact of the size of the workload on the processing time by adding the processing time delay to the processing time. The processing time delay that we used in the experiment is

calculated by using two functions. The first equation is $y = 20.56652 \cdot 1.00002^x$ where x is the number of requests and y is the processing time used when the number of requests is lower than a million requests. The second equation is $y = -681.1851 + 108.75751 \cdot \ln(x)$, where x is the number of requests and y is the processing time. For the experiment on simulated network topologies, we rather used linear workload modeling as normal without considering the processing time delay instead.

4.2. Metrics

We evaluated the performance of the proposed framework by using three metrics corresponding to the objective functions.

4.2.1. Total cost

Total cost is the sum of overall capacity cost, network cost, and upfront edge server deployment cost using Equation 3.5.1-3.5.4.

4.2.2. Average response time

Average response time is the average amount of time taken to respond to a request including localization processing time and propagation delay using Equation 3.5.9.

4.2.3. The number of edge nodes

The number of edge nodes required for service placement is calculated by using Equation 3.5.10.

4.3. Benchmarks

We compared the experimental results of the proposed framework to five benchmarks to show the prominent point of each approach, and to which approach our proposed framework can outperform.

4.3.1. Full edge deployment

An approach in which every node on the network topology is used as an edge node with the service implemented.

4.3.2. Centralized deployment

An approach in which there is only one node that the service is implemented.

4.3.3. EdgeOn Framework [15]

An approach in which the service edge is placed by the solution from a multi-objective network-aware edge node placement model and solution strategy tailored to 5G scenarios. This framework shares the same goals with our proposed framework but does not consider the localization processing time and POI data.

4.3.4. Betweenness Centrality [21]

An approach in which the service edge is placed by the betweenness centrality which is the number of times a node acts as a bridge along the shortest path between two other nodes.

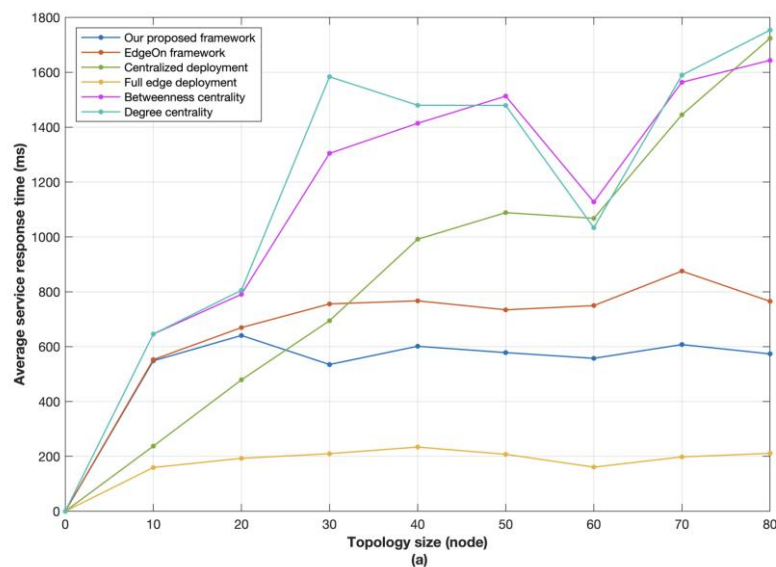
4.3.5. Degree Centrality [21]

An approach in which the service edge is placed by the number of links incident upon a node.

4.4. Experimental Results

4.4.1. Simulated Networks Topology

Fig. 4.1.1 shows the experimental results for the simulated network topologies with the topology sizes ranging from 10 nodes to 80 nodes. Fig. 4.1.1 shows the impact of topology sizes on the average service response time, the total cost, and the number of edge nodes.



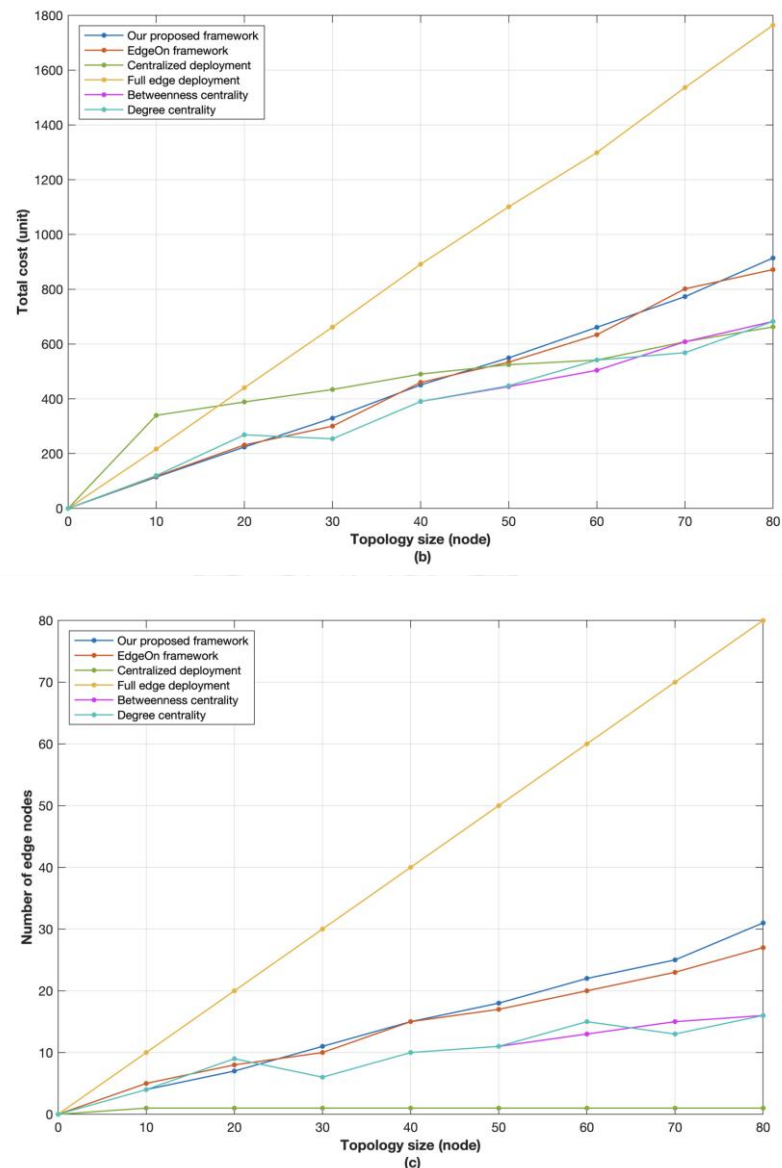


Figure 11 The impact of topology sizes on (a) Average service response time; (b) Total cost; (c) Number of edge nodes

It can be observed in Figure 11(a) that at the small topology size, the average response time of each approach slightly differs from the others due to the small number of fingerprints. As the topology size grows larger, all approaches outperform the centralized deployment. The average response time from all topology sizes of the EdgeOn framework is equal to 757.42 ms whereas that of our proposed framework is equal to 580.33 ms. This shows a 23.38% improvement on average and up to 30.50 %improvement. Our proposed framework can achieve less average

service response time than the EdgeOn framework at any topology size. This is because, unlike EdgeOn framework, our proposed framework considers the average service response time in the optimization. Moreover, the EdgeOn framework also focuses on maximizing the capacity usage ratio of each edge server which directly affects the growth of the data size and usage quantity at each edge server, leading to long service response time. The full edge deployment conquers all approaches with the least average service response time as expected. The betweenness centrality and the degree centrality provide the worst average service response time among approaches.

As shown in Figure 11(b), as the topology size increases, the total cost increases for all approaches. At the small topology size, the total cost of the centralized deployment is the highest due to the cloud service cost. At the large topology size, the total cost of the centralized deployment tends to be stable. The total cost of the full edge deployment is the highest at any topology size since the number of servers is the same number of topology sizes. The total cost of our proposed framework is about the same as that of EdgeOn.

As shown in Figure 11(c), our proposed framework provides a slightly higher number of edge nodes than the EdgeOn framework. The degree centrality can achieve the same number of edge nodes as that of the betweenness centrality.

The result reveals that our proposed framework can achieve a better average service response time than other approaches on the simulated topologies.

4.4.2. Real-World Large-Scale Data

a) Wi-Fi fingerprint dataset of 37 multi-floor buildings at Chula Expo

The performance of the proposed framework on real-world large-scale data could change due to the density of nodes, the sparsity of links and the realistic size of fingerprint data, and the number of requests.

Table 3 The experimental results for the large-scale area consisting of 37 multi-floor buildings using linear workload modeling

Approach	Metrics		
	Average service response time (ms)	Number of edge nodes	Total cost (unit)
Our proposed framework	100.93	21	656.82
EdgeOn framework	173.85	16	558.43
Centralized deployment	210.10	1	614.01
Full edge deployment	85.76	37	753.43

We carried out the edge node placement experiment on the 37 multi-floor buildings data using linear workload modeling with our proposed framework and other benchmarks. Table 3 shows the experimental results for the large-scale area consisting of 37 multi-floor buildings using linear workload modeling. As our results show, the EdgeOn framework achieves an average service response time of 173.85 ms whereas the centralized deployment achieves 210.10 ms. Our proposed framework achieves an average service response time of 100.93 ms which is 41.94% and 51.96% lower compared to the EdgeOn framework and the centralized deployment, respectively. In addition, the total cost of the EdgeOn framework is equal to 558.43 units whereas that of our proposed framework is equal to 656.82 units which are only 17.62 % higher. In terms of the number of edge nodes, the result of the EdgeOn framework is 16 nodes whereas that of our proposed framework is 21 nodes which is 31.25% higher.

The results reveal that our proposed framework can leverage the number of edge nodes and the total cost to achieve better average service response time than other approaches on the real-world large-scale data: Wi-Fi fingerprint dataset of 37 multi-floor buildings at Chula Expo using linear workload modeling.

b) Central Shopping Mall Group

The performance of the proposed framework on real-world large-scale data could be affected by the size of the topology in terms of distance. In the central shopping

mall group topology, the path with the longest propagation delay is the path between *Central i-City* and *Central Lampang* which is 143.49 ms. We expected that our proposed framework can provide the placement solution with the least average service response time while balancing the total cost and the number of edge nodes. We also used exponential workload modeling in this experiment to reflect the computational process behavior due to a large number of requests and a large size of fingerprint data.

Table 4 The experimental results for the Central Shopping Mall Group data using logarithmic workload modeling.

Approach	Metrics			
	Average service response time (ms)	Number of edge nodes	Total cost (unit)	Average propagation delay (ms)
Our proposed framework	655.44	26	627.48	2.51
EdgeOn framework	1,783.55	28	711.98	1.00

Table 4 shows the experimental results for the Central Shopping Mall Group data using logarithmic workload modeling. From the result, our proposed frameworks required 627.48 units of total cost whereas the EdgeOn framework required 711.98 units of total cost 84.5 units higher. In terms of the number of edge nodes required in service deployment, the proposed framework required less 2 edge nodes than the EdgeOn framework. The average service response time of our proposed framework is 655.44 ms which is 63.25% lower compared to that of EdgeOn framework. In addition, our proposed framework's solution can achieve the propagation delay at 2.51 ms on average per query on the large topology input. This shows that our proposed framework can provide the placement solution that can divide the user workloads and computational demand of each service efficiently and can preserve the optimized response time including both localization processing time and propagation delay.

The experimental results show that our proposed framework can be applied to applications with a linear relationship between the data size and the processing time and also the higher complexity applications with an exponential relationship between the data size and the processing time to provide a better average service response time than other approaches.

4.5. Result Discussion

4.5.1. Workload Imbalance

The main goal of the proposed framework is to minimize the service response time by including the average service response time in the objective function. In optimization, incorporating the average value into the goal or objective function is advantageous as it captures the overall value, covering both positive and negative aspects. Nevertheless, this approach may encounter challenges due to the extensive range between the minimum and maximum values, leading to potential imbalances. We designed the greedy algorithm of the Placement Strategy module to select the edge node by considering the least product of the number of requests and the size of fingerprint data of each node to place the small service first before combining the other services and becoming larger. Using the product methods, the number of requests is reciprocal to the size of fingerprint data. As a result, it is possible that there can be an edge node that serves a high workload whereas the other edge node serves a low workload leading to a workload imbalance problem.

Table 5 The result of our proposed framework tested on the Central Shopping Mall Group data from 4.4.2(b) in detail.

	Edge node detail (Number of requests)
Our proposed framework	(150000, 78.7), (100000, 300.0), (70000, 236.0), (100000, 311.0), (40000, 100.0), (70000, 186.0), (120000, 364.0), (70000, 255.0), (20000, 90.0), (40000, 225.0), (95000, 312.0), (105000, 340.0) (110000, 285.0), (25000, 155.0), (85000, 593.0), (100000, 128.0), (50000, 250.0), (60000, 270.0), (60000, 300.0), (40000, 104.953), (75000, 278.0), (100000, 107.0), (75000, 138.0), (90000, 181.0), (80000, 230.5), (130000, 830.0)

Table 5 shows the result of our proposed framework tested on the Central Shopping Mall Group data from 4.4.2(b) in detail. The number of requests served by

an edge node and the size of the fingerprint of each edge node are displayed in pair orders.

Table 6 The summary of the number of requests served by an edge node and the size of the fingerprint of each edge node data from 4.4.2(b)

	Minimum Value	Maximum Value	Standard Deviation
The number of requests	20,000	150,000	31582.99
The size of fingerprint data	78.7	830.0	158.69

Table 6 shows the summary of the number of requests served by an edge node and the size of the fingerprint of each edge node data from 4.4.2(b). The minimum number of requests served by an edge node is equal to 20,000 requests while the maximum number is equal to 150,000 requests. The difference between the maximum value and the minimum value is about 130,000 requests showing that there is a workload imbalance among edge nodes in some cases.

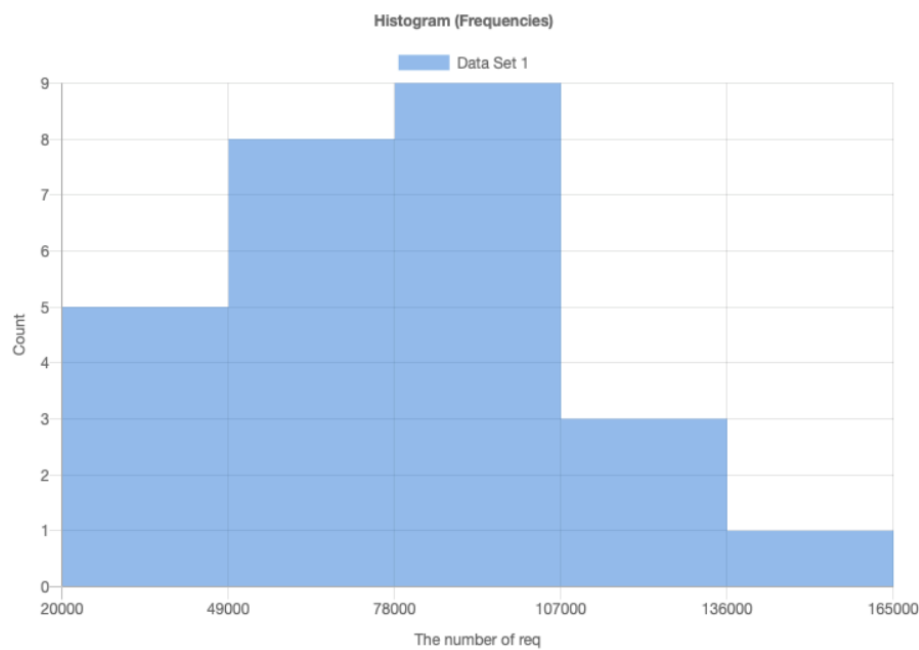


Figure 12 The histogram of the number of requests served by an edge node

Figure 12 shows that the characteristic of the histogram is a right-skewed distribution. Most of the number of requests served by an edge node is between 20,000 requests to 107,000 requests and only some are above.

In terms of the size of fingerprint data, from Table 6, the minimum value is equal to 78.7 and the maximum value is equal to 830.0. The standard deviation of the data is equal to 158.69.

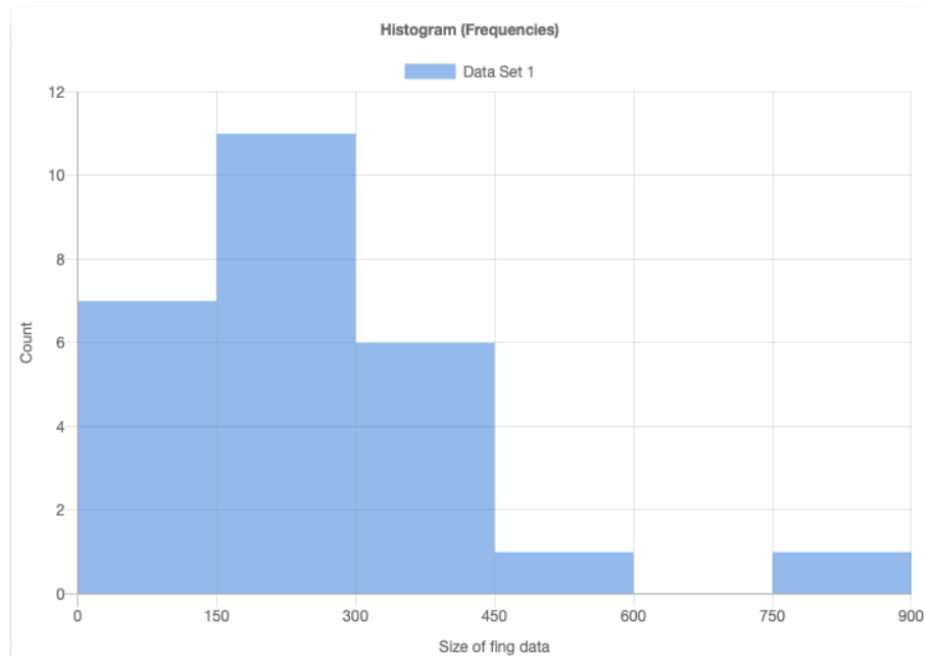


Figure 13 The histogram of the size of fingerprint data of each edge node

Figure 13 shows that the characteristic of the histogram is a right-skewed distribution. Only one outlier is presented which is the size of fingerprint data of 830.0. We have inspected the edge list and its serving node list and found that the node with that size of fingerprint data was serving only itself. This is necessary to prevent this node from being combined with the other nodes resulting in the larger size of the fingerprint data.

From the above results, it can be concluded that even though there is a difference in the number of requests among edge nodes, the size of the fingerprint data of each service which reflects the localization processing time is like each other. The workload imbalance issue presents. Our placement strategy can still achieve the optimization goal under such circumstances.

4.5.2. Cost Analysis

The overall cost is one of the objectives of the optimization goal. The experimental results in the aspect of the total cost on both simulated data and real-world data have been shown in units. The total cost only represents the sum of the

capacity cost, the network traffic cost, and the upfront deployment cost. The cost of the optimal placement solution has been recorded thoroughly to be analyzed later.

Table 7 The cost analysis of the experimental results on the Central Shopping Mall

Group data

Approach	Cost value			
	The capacity cost	The network cost	Total time the links used (times)	The upfront deployment cost
Our proposed framework	6,648.153	3,950,000	790,000	520
EdgeOn framework	6,648.153	4,550,000	910,000	560

Table 7 shows the cost analysis of the experimental results on the Central Shopping Mall Group data. The capacity cost is calculated from the total size of the fingerprint data of each node. Typically, using the EdgeOn approach which is focusing on maximizing the capacity usage ratio might lead to duplicated service placement; one node can be served by more than one edge node. The result shows that the capacity cost of our proposed framework and EdgeOn framework is the same which means that each non-edge node in EdgeOn's solution is served only once by only one edge node. The network cost is calculated by using the total times the links are used times the constant cost. The result shows that our proposed framework can manage the placement solution to use links in the network less than 120,000 times compared to the EdgeOn framework. The upfront deployment cost is the sum of the upfront deployment cost of each node which is set to the same for all nodes in this experiment.

From the results, we can conclude that our proposed framework outperforms the EdgeOn framework in any aspect of cost.

5. Conclusion

5.1. Thesis Summary

In this thesis, we proposed an edge service placement optimization framework for location-based services. The characteristics of location-based services have been studied. The finding expressed the importance of the localization process and its impact on the performance of location-based services in terms of real-time response. The most popular technique of localization technique. This technique struggles with the fingerprint size degrading the performance problem. In a large-scale area, like multi-story buildings, the size of the fingerprint could be massive leading to long localization processing time. The cloud architecture also resulted in long latency. Applying edge computing to deploy small services at the edge of the network close to users is the solution. Due to cost and resource limitations, the services can be placed at some edge nodes only. This is when the edge node placement strategies have been explored. Metrics can be included in the optimization goal of a placement strategy.

We included the edge deployment cost, the number of edge nodes, and the average service response time including localization processing time and propagation delay. The proposed framework consists of four main modules cooperating to generate placement solutions using a greedy algorithm and exploring the solution space using the Simulated Annealing approach.

The experiments were set up using indoor localization service as a use case and tested on both simulated data and real-world large-scale data. The relationship between the processing time and the size of the fingerprint data and workload modeling were also observed. Using our framework, the performance of the placement strategy for location-based service can be significantly improved. The results show that our proposed strategy outperforms the existing work, sharing the same goals in several aspects. The proposed framework can leverage the number of edge nodes and the total cost to achieve a lower average response time of up to 63.25% compared with the existing work. It can be tailored to other applications by adjusting the weights in the objective function. The proposed framework can also benefit other use cases whose performance relies on data locality and data size.

Since the framework aims at minimized response time and considers the Point of Interest data.

5.2. Discussions on the Fingerprint Data

The fingerprint-based technique is one of the most popular techniques used in indoor localization while using the concept that each location has its own unique signal characteristics which can be used to represent itself.

```
[
  {
    "sampling_id": 1,
    "fingerprint": [{ "RSSI": -55, "BSSID": "f4:f2:6d:ec:59:0a" }, { "RSSI": -89, "BSSID": "c8:1f:be:38:7b:d8" } ],
  },
  {
    "sampling_id": 2,
    "fingerprint": [{ "RSSI": -37, "BSSID": "74:ea:3a:cc:8e:3d" }, { "RSSI": -80, "BSSID": "2c:08:8c:62:c4:cc" },
      { "RSSI": -85, "BSSID": "cc:4e:ec:d7:97:68" }, { "RSSI": -89, "BSSID": "38:d5:47:bd:26:88" },
      { "RSSI": -91, "BSSID": "d4:7b:b0:b8:01:c0" }, { "RSSI": -92, "BSSID": "e0:88:5d:90:1b:72" } ]
  }
]
```

Figure 14 The example of fingerprint data of an Adaptive Indoor Localization System for Large-Scale Area from Ref. 13

Figure 14 shows the example of fingerprint data of an Adaptive Indoor Localization System for a Large-Scale Area from Ref. 13. As shown in the figure, each sample of fingerprint data is a list of access points that can be scanned from each location labeled by a Basic Service Set Identifier (BSSID) and sorted by the value of the Received Signal Strength Indicator (RSSI). The fingerprint characteristics of each fingerprint-based technique can be various due to the technology e.g., sensor, Bluetooth, geo-magnetic. The fingerprint format that we used in the framework design and in the experiments was not fixed or relied on any specific fingerprint-based localization algorithm. We only considered the size of the fingerprint data and the number of fingerprint data impact on the localization processing time. The different fingerprint format does not affect the performance of our proposed framework.

5.3. Discussions on Historical Usage Quantity Data

The historical usage quantity data reflects the user workload at each location. In our proposed framework, we use the data of the number of requests served by each

node representing point-of-interest data. In the placement strategy module, we used this data to calculate the least product to greedily pair unserved and selected edge nodes.

Collecting point-of-interest data to be used as historical data needs to be well-planned to capture all the characteristics of service usage quantity in possible scenarios. Some places might have different usage quantities depending on the incident. For example, in a shopping mall, the historical data should include the workload on normal days and peak time on the event day or the weekend, which could be higher.

With improper data preparation, some servers could be overloaded resulting in longer response time than expected. Our proposed framework performs static multi-objective optimization to find a service placement solution. To illustrate, the optimization process is done once before the edge deployment process takes place.

Our proposed framework can handle this situation by performing the optimization process again with the latest point-of-interest data corresponding to the workload that covers the peak time situation to achieve a suitable placement solution and resource allocation.

5.4. Discussions on Performance Trade-Off

The objective function of our proposed framework is shown in Equation 3.1.1, which is to minimize the total cost, the number of edge nodes, and the average response time. As a result, the framework's performance involves a tradeoff between the total cost, the number of edge nodes, and the average response time depending on the weight of each factor. Each weight factor can be adjusted and tailored to the target application. For example, the application that requires a real-time response with a strict limit of delay like Video conferencing and Emergency services, the weight of the average service response time should be set with the highest value to provide the least average service response time. When the deployment budget is limited, e.g., the scenario with plenty of Internet of Things (IoT) devices and small-scale businesses, the weight of the total cost should be set with the highest value to achieve the placement solution with the lowest deployment cost.

5.4.1. Parameter Tuning

In the performance evaluation, we included the EdgeOn framework as one of the benchmarks. Since the weights of our proposed framework can be adjusted arbitrarily, we needed suitable weights for the experiment to compare the experimental results to the EdgeOn framework fairly. The weights that we used for our proposed framework were required to provide results close to that of the EdgeOn framework in the aspects of the number of edge nodes required for edge deployment and the total cost. Therefore, we performed parameter tuning to find the weights.

The weights that we focused on were w_1 (for the total cost), w_2 (for the number of edge nodes), and w_3 (for the average service response time). A placement solution was acquired from each framework run which is a space solution exploration process of simulated annealing and solution generation of placement strategy. The goal of the parameter tuning is to find the weights that can provide the placement solution result close to that of EdgeOn in those two aspects. At first, a placement solution was retrieved by using EdgeOn framework and then was evaluated in terms of the total cost and the number of edge nodes to be used in the comparison.

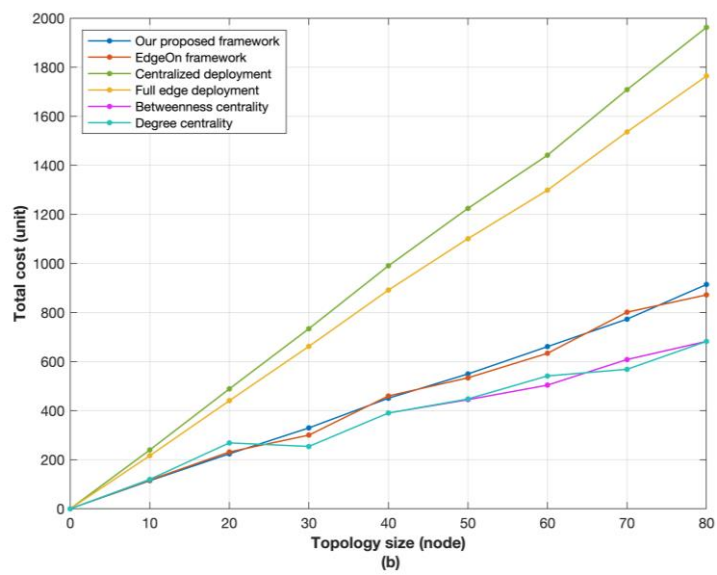
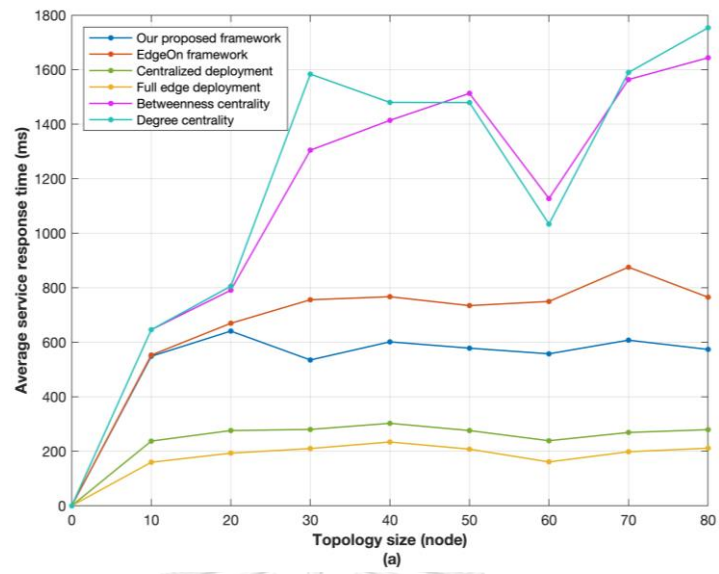
The list of the w_1 parameter was [0.0625, 0.125, 0.3125, 0.625, 0.75, 0.875, 1]. The list of the w_2 parameter was [0.0625, 0.125, 0.3125, 0.625, 0.75, 0.875, 1]. The list of the w_3 parameter was [0.0625, 0.125, 0.3125, 0.625, 0.75, 0.875, 1]. In each run, the result was recorded and calculated. To measure the closeness of our proposed framework's results with given weights and that of the EdgeOn framework in terms of the total cost and the number of edge nodes, we computed the Root Mean Square Error (RMSE) for each aspect separately, comparing that of the EdgeOn framework values. Then sum up the individual RMSE values to obtain an overall measure of the closeness between results. By minimizing the sum of RMSE values, you can find the combination of weights (w_1 , w_2 , w_3) that brings results closer to the results of the reference framework in terms of aspects of the total cost and the number of edge nodes. The time used in each run of our proposed framework was approximately 120 seconds when using 37 buildings in Chulalongkorn University data [13] or around 11 hours in total. The results show that the weights including w_1 , w_2 , and w_3 that

provide the least sum of RMSE are 0.1875, 0.625, and 0.1875, respectively. Therefore, we used these weights for our proposed framework in the experiments for performance evaluation.

5.5. Discussions on Centralized Deployment

In the experiments, we defined the centralized deployment as one of the benchmarks. The centralized deployment is an approach in which there is only one node in that the service is implemented. In the parameter configuration, we set the number of equal to 1 server, the network cost to 10 times the local network cost and the computational capability to 10 times the server used in other approaches. The experimental results on simulated data from Figure 11 shows that at the large topology size, the total cost of the centralized deployment is lower than that of our proposed framework while the average service response time is higher. The results cannot be interpreted directly and fairly because the computational capability of the server used in the centralized deployment approach was fixed.

To compare the results of our proposed framework to the centralized deployment with fairness, we configured the computational capability of the server used in the centralized deployment to be M times of the server used in other approaches where M is the number of nodes of input topology size. To clarify, if the number of input nodes is 20 nodes, the computational capability of the server is 20 times that of the normal server. The server cost was also M times the cost of the normal server. Then, the experiment with new centralized deployment was carried out.



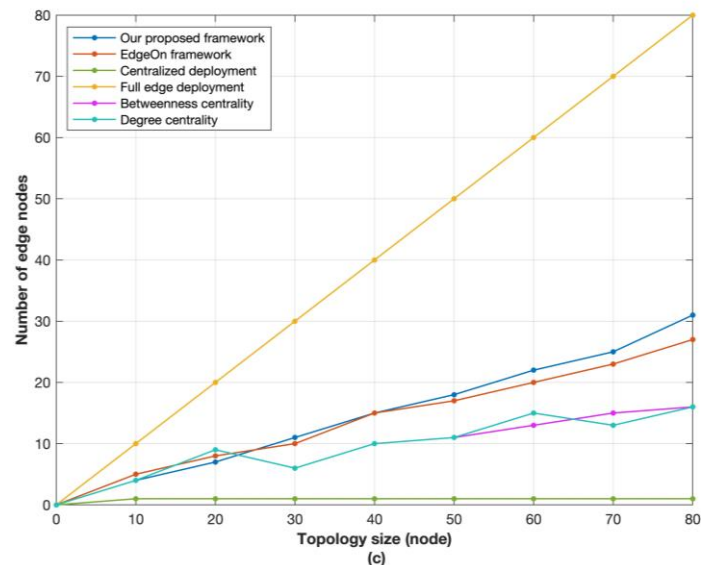


Figure 15 The impact of topology sizes of simulated data using the new centralized deployment parameter on (a) Average service response time; (b) Total cost; (c) Number of edge nodes

Figure 15(a) shows the impact of topology sizes of simulated data using the new centralized deployment parameter on the average service response time. The average service response time of the centralized deployment approach is lower than that of our proposed framework at any topology size up to 338.68 ms at the topology size of 70 nodes. On the other hand, in terms of the total cost, Figure 15(b) shows that the centralized deployment provides the worst total cost compared to all approaches at any topology size. The total cost of the centralized deployment is more than 114.63% when compared to that of our proposed framework. Figure 15(c) shows that our proposed framework requires less than 50% number of the edge nodes of the input topology size at any topology size. As our proposed framework uses the local network traffic which is cheaper than the cloud network traffic and also uses a smaller number of servers, the total cost of our proposed framework is lower than the centralized deployment. Even though the centralized deployment can achieve better average service response time by scaling up the resources, this approach still requires much more cost than other approaches.

5.6. Discussions on Workload Imbalance

From the result discussion 5.4.1, the workload imbalance presents due to the use of the average value of response time in the optimization goal of our framework. In the experiments, we assumed that every edge node holds sufficient computational resources for any number of requests and that every link also holds adequate bandwidth for all network traffic loads.

However, in the real scenario, the heavy workload could affect the server performance. As a result, some edge servers face overload requests. Using the same specification of the edge server can cause server failure, heavy network traffic, and bottleneck problems in case of service overuse due to workload imbalance. To avoid these problems, the limitation of the number of requests each edge server needs to serve and the network traffic that each link and node can handle need to be defined as constraints.

With constraints on the number of requests of each edge node and the network traffic, our proposed framework can manage and divide the workload of each node within the constraints to achieve the optimal placement solution by leveraging the total cost and the number of edge nodes to prevent such problems. As a result, the number of edge nodes is scaled up to serve all requests, also the total cost increases. Despite the increase in the number of edge nodes and the total cost, our proposed framework can still optimize the average service response time.

5.7. Discussions on Application

The experimental results show that our proposed framework can achieve less than 30.50% average service response time compared with the existing work using the simulated data and less than 63.25% or 1,128.11 ms average service response time compared with the existing work using the real-world large-scale data. For such a lower response time, the performance of many applications could be improved.

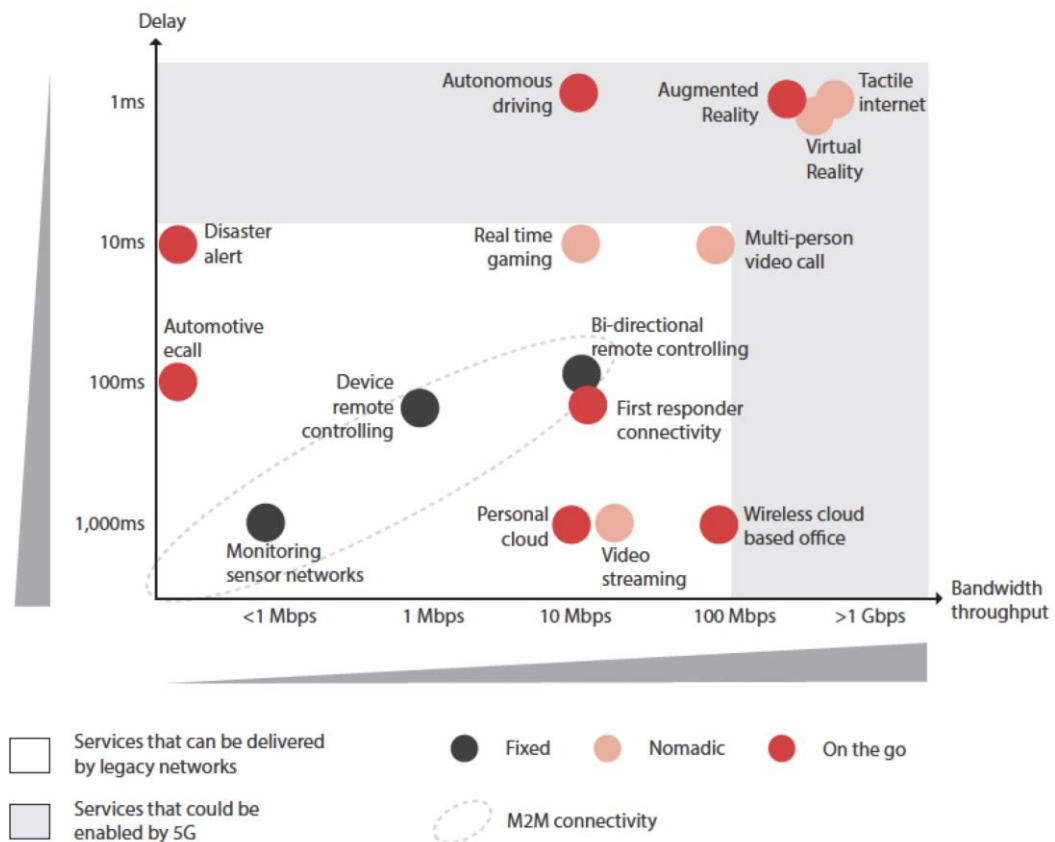


Figure 16 Bandwidth and latency requirements for different application [22]

Figure 16 shows the bandwidth and latency requirements for different applications [22]. Many location-based services such as Augmented Reality, Virtual Reality, Disaster alert, and Real-time gaming are also included. These applications require less than 100 ms of delay. It can be concluded that our proposed framework is able to improve those application performances which are location-based services with better average response time.

5.8. Discussions on Limitations and Future Works

Despite several benefits, there are limitations that should be mentioned.

Our proposed framework can provide a better average service response time compared with the existing work. However, by using the average value in the optimization process, the framework still faces a workload imbalance problem. The workload distribution needs to be studied more and included in optimization or constraints.

Another topic is that the problem of the proposed framework was formulated as a static multi-objective optimization problem. That is the optimization process is done once before the edge deployment process takes place. As a result, the data used in the optimization could be obsolete when time passes due to the change in the network environment and user behavior. Optimizing the solution dynamically could be important in the future when the data changes rapidly.

Although the experiments were designed considering any possible factors, the real data was not derived directly completely. Some of them were adapted to use in the experiments, for example, the propagation delay of each link is calculated by using geographical distance. Data collecting design and realistic network topology construction could proceed in the future to provide more accuracy in the framework performance evaluation.



REFERENCES

1. Ahmad, I., et al. *Current technologies and location based services*. in *2017 Internet Technologies and Applications (ITA)*. 2017.
2. Zhang, M., et al., *Pedestrian Dead-Reckoning Indoor Localization Based on OS-ELM*. IEEE Access, 2018. **6**: p. 6116-6129.
3. Zhou, C., et al., *Exploiting Fingerprint Correlation for Fingerprint-Based Indoor Localization: A Deep Learning Based Approach*. IEEE Transactions on Vehicular Technology, 2021. **70**(6): p. 5762-5774.
4. Yang, Z., C. Wu, and Y. Liu, *Locating in fingerprint space*, in *Proceedings of the 18th annual international conference on Mobile computing and networking*. 2012. p. 269-280.
5. He, S., J. Tan, and S.H.G. Chan, *Towards area classification for large-scale fingerprint-based system*, in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 2016. p. 232-243.
6. Douch, S., et al., *Edge Computing Technology Enablers: A Systematic Lecture Study*. IEEE Access, 2022. **10**: p. 69264-69302.
7. Cao, B., et al., *Large-Scale Many-Objective Deployment Optimization of Edge Servers*. IEEE Transactions on Intelligent Transportation Systems, 2021. **22**(6): p. 3841-3849.
8. Salaht, F.A., F. Desprez, and A. Lebre, *An Overview of Service Placement Problem in Fog and Edge Computing*. ACM Computing Surveys, 2020. **53**(3): p. 1-35.
9. Ma, S.-P. and D.-Y. Yan, *Location-Based Web Service Delivery: A Data-Mining-Based Approach*, in *2012 International Symposium on Computer, Consumer and Control*. 2012. p. 666-669.
10. Huang, H. and S. Gao, *Location-Based Services*. Geographic Information Science & Technology Body of Knowledge, 2018. **2018**(Q1).
11. Khaoampai, K., K. Na Nakorn, and K. Rojviboonchai, *FloorLoc-SL: Floor Localization System with Fingerprint Self-Learning Mechanism*. International Journal of Distributed Sensor Networks, 2015. **11**(11).

12. Min, Z., P. Ling, and D. Xiaotie, *GraphSLAM-based Crowdsourcing framework for indoor Wi-Fi fingerprinting*, in *2016 Fourth International Conference on Ubiquitous Positioning, Indoor Navigation and Location Based Services (UPINLBS)*. 2016. p. 61-67.
13. Vongsuteera, T. and K. Rojviboonchai, *Adaptive Indoor Localization System for Large-Scale Area*. IEEE Access, 2021. **9**: p. 8847-8865.
14. Madamori, O., et al., *A Latency-Defined Edge Node Placement Scheme for Opportunistic Smart Cities*, in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. 2021. p. 142-147.
15. Santoyo-Gonzalez, A. and C. Cervello-Pastor, *Network-Aware Placement Optimization for Edge Computing Infrastructure Under 5G*. IEEE Access, 2020. **8**: p. 56015-56028.
16. Farahani, R.Z., et al., *Covering problems in facility location: A review*. Computers & Industrial Engineering, 2012. **62**(1): p. 368-407.
17. Wu, L.-Y., X.-S. Zhang, and J.-L. Zhang, *Capacitated facility location problem with general setup cost*. Computers & Operations Research, 2006. **33**(5): p. 1226-1241.
18. Dowsland, K.A. and J.M. Thompson, *Simulated Annealing*, in *Handbook of Natural Computing*. 2012. p. 1623-1655.
19. *Central Pattana, Shopping Centers*. [cited January 11, 2023]; Available from: <https://www.centralpattana.co.th/th/our-business/shopping-center>.
20. *K6, Load testing for engineering teams | Grafana k6* [cited July 3, 2023]; Available from: <https://k6.io/docs/>.
21. Newman, M., *Networks: An Introduction*. 2010: Oxford University Press.
22. Xu, L., *Context aware traffic identification kit (TriCK) for network selection in future HetNets/5G networks*, in *2017 International Symposium on Networks, Computers and Communications (ISNCC)*. 2017. p. 1-5.



จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

VITA

NAME Karnkitti Kittikamron

DATE OF BIRTH 25 July 1999

PLACE OF BIRTH Bangkok

INSTITUTIONS ATTENDED Chulalongkorn University

HOME ADDRESS 55/72 Ban Klang Mueang The Paris Ratchawipha,
Khampangphet6 Rd., Lat Yao, Chatuchak, Bangkok, 10900



จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY