Privacy preserving internet of things recommender systems for smart cities

Yasir Saleem Shaikh

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAS001

Thèse de doctorat

Privacy Preserving Internet of Things Recommender Systems for Smart Cities

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 Institut Polytechnique de Paris (IP Paris)
Spécialité de doctorat : Informatique et Réseau

Thèse présentée et soutenue à Évry, le 21/01/2020, par

YASIR SALEEM SHAIKH

Composition du Jury :

Maria Potop-Butucaru Professeure, Sorbonne University, LIP6 - France	Président
Payam Barnaghi Professeur, University of Surrey - ROYAUME-UNI	Rapporteur
JaeSeung Song Professeur associé, Sejong University - COREE, REPUBLIQUE DE	Rapporteur
Maria Potop-Butucaru Professeure, Sorbonne University, LIP6 - FRANCE	Examinatrice
Luigi Atzori Professeur, University of Cagliari - ITALIE	Examinateur
Martin Bauer Chercheur Principal, NEC Europe Ltd - ALLEMAGNE	Examinateur
Noel Crespi Professor, Télécom SudParis - FRANCE	Directeur de thèse
Roberto Minerva Maître de Conférences, Télécom SudParis - FRANCE	Co-directeur de thèse

Titre : Protection de la Confidentialité des Services de Recommandation pour les Villes Intelligentes

Mots clés : Internet des objets, recommandation IoT, villes intelligentes, stationnement intelligent, ski intelligent, préservation de la vie privée

Résumé : Au cours de la dernière décennie, la technologie Internet des objets (IoT) a révolutionné presque tous les domaines de la vie quotidienne et a dynamisé les villes intelligentes. Les villes intelligentes utilisent la technologie IoT pour collecter divers types de données de capteurs (trafic, stationnement, météo et environnement), puis les utilisent pour offrir diverses applications, telles que les systèmes de transport intelligent, le stationnement intelligent, le réseau intelligent et le ski intelligent, pour n'en nommer que quelques-uns. Les villes intelligentes ont pour objectif d'améliorer la qualité des services gouvernementaux et le bien-être des citoyens. Comme les applications des villes intelligentes sont utilisées par les citoyens, donc leur fournir des services de recommandation personnalisés en fonction de leurs préférences, de leurs localisations et de leurs profils ainsi que l'exploitation des données IoT (par exemple, la congestion du trafic et l'occupation du parking) est d'une grande importance qui pourrait être fournie par un recommandateur IoT. Cependant, comme le recommandateur IoT utilise les données privées des citoyens (profils, préférences et habitudes, par exemple), il viole la vie privée des utilisateurs car il pourrait suivre les routines et les habitudes des utilisateurs en analysant la base de données historique ou en analysant les services de recommandation réguliers qu'il offre. Par conséquent, il est important de préserver la confidentialité des utilisateurs du programme de recommandation IoT.

Dans cette thèse, nous proposons un nouveau système de recommandation IoT préservant la confidentialité pour les villes intelligentes, qui fournit des recommandations en exploitant les données IoT des capteurs et en tenant compte de diverses métriques. Notre approche est organisée en trois parties. Tout d'abord, nous développons un système de recommandation IoT conforme au règlement européen sur la protection des données (GDPR) pour les systèmes de stationnement intelligent. Ces systèmes fournissent des recommandations sur les emplacements et les itinéraires de stationnement en exploitant les données des capteurs de stationnement et de circulation. Par conséquence, nous proposons d'abord une approche pour la cartographie des capteurs de trafic avec les coordonnées d'itinéraires afin d'analyser les conditions de trafic (par exemple le niveau de congestion) sur les routes Ensuite, nous avons mis en place un dispositif de recommandation IoT. Ce dispositif offre quatre fonctions. Premièrement, il aide l'utilisateur à trouver une place de parking gratuite en fonction de différentes mesures (par exemple, la place de stationnement fiable ou la plus proche). Deuxièmement, il recommande un itinéraire (l'itinéraire le moins fréquenté ou l'itinéraire le plus court) menant à l'emplacement de stationnement recommandé à partir de l'emplacement actuel de l'utilisateur. Troisièmement, il fournit la disponibilité en temps réel des zones de stationnement prévues (comprenant des places de stationnement organisées en groupes) de manière conviviale. Enfin, il fournit une implémentation conforme au GDPR pour fonctionner dans un environnement sensible à la confidentialité. Le recommandateur IoT a été intégré au scénario d'utilisation du stationnement intelligent d'un projet H2020 EU-KR WISE-IoT et a été évalué par les citoyens de Santander, en Espagne, à l'aide d'un prototype. Il a également été démontré à trois reprises. De plus, nous développons un recommendateur IoT pour le ski intelligent qui fournit des itinéraires de ski comprenant des types de pistes spécifiques, ainsi que la piste la plus proche. Pour les itinéraires de ski, il n'existe aucun moteur de calcul stable. Par conséquent, un nouveau moteur de routage pour les itinéraires de ski a été développé. Ce travail a également été intégré dans le cas d'utilisation du ski intelligent du projet WISE-IoT.

Deuxièmement, bien que le recommandateur IoT développé pour le stationnement intelligent soit conforme au GDPR, il ne protège toutefois pas totalement la vie privée des utilisateurs. En effet, le partage sans discernement des données des utilisateurs avec un système tiers de recommandation de stationnement IoT non approuvé ou semi-fiable provoque une violation de la vie privée. En effet, le comportement et les schémas de mobilité des utilisateurs pouvant être déduits en analysant l'historique de leurs déplacements. Par conséquent, nous préservons la confidentialité des utilisateurs contre le système de recommandation de stationnement tout en analysant leur historique de stationnement en utilisant des techniques de k-anonymat (anonymisation) et de confidentialité différentielle (perturbation).

Enfin, étant donné que les applications de villes intelligentes sont développées de manière verticale et ne se parlent pas, c'est-à-dire que chaque application est développée pour un certain scénario qui ne partage généralement pas les données avec d'autres applications de villes intelligentes. Par conséquent, nous avons proposé deux cadres pour les services de recommandation parmi les applications de villes intelligentes utilisant l'IdO social. Tout d'abord, sur la manière dont l'IdO social peut être utilisé pour les services de recommandation entre applications de villes intelligentes. Deuxièmement, nous proposons un autre type de communication de l'IdO social au niveau mondial, à savoir les communications inter-domaines sociales qui permettent aux applications de villes intelligentes de communiquer entre elles et établir des relations sociales entre elles. Ces frameworks sont les blocs de construction des recommandations inter-domaines dans les applications de villes intelligentes.

Title : Privacy Preserving Internet of Things Recommender Systems for Smart Cities

Keywords : Internet of Things, IoT Recommender, Smart Cities, Smart Parking, Smart Skiing, Privacy Preservation

Abstract : During the past decade, the Internet of Things (IoT) technology has revolutionized almost all the fields of daily life and has boosted smart cities. Smart cities use IoT technology to collect various types of sensors' data (e.g., traffic, parking, weather and environmental) and then use such data to offer a variety of applications, such as intelligent transportation system, smart parking, smart grid and smart skiing, to name a few. The objective of smart cities is to improve the quality of governmental services and citizens welfare. Since the smart cities' applications are used by the citizens, therefore providing the customized recommendation services to the citizens based on their preferences, locations and profiles, as well as by exploiting the IoT data (e.g., traffic congestion and parking occupancy) is of great importance which could be provided by an IoT recommender. However, since the IoT recommender utilizes the private data of citizens (e.g., profiles, preferences and habits), it breaches the privacy of the users because the IoT recommender could track the routines and habits of the users by analyzing the historical database or by analyzing the regular recommendation services it offers. Therefore, it is important to preserve the privacy of the users from the IoT recommender.

In this thesis, we propose a novel privacy preserving IoT recommender system for smart cities that provides recommendations by exploiting the IoT data of sensors and by considering various metrics. Our approach is organized in three parts. Firstly, we develop an EU General Data Protection Regulation (GDPR)-compliant IoT recommender system for smart parking system that provides recommendations of parking spots and routes by exploiting the data of parking and traffic sensors. For this, we first propose an approach for the mapping of traffic sensors with route coordinates in order to analyze the traffic conditions (e.g., the level of congestion) on the roadways and then developed an IoT recommender. The IoT recommender provides four-fold functions. Firstly, it helps a user to find a free parking spot based on different metrics (e.g., nearest or nearest trusted parking spot). Secondly, it recommends a route (the least crowded or the shortest route) leading to the recommended parking spot from the user's current location. Thirdly, it provides the real-time provisioning of expected availability of parking areas (comprised of parking spots organized into groups) in a user-friendly manner. Finally, it provides a GDPR-compliant implementation for operating in a privacy-aware environment. The IoT recommender has been integrated into the smart parking use case of an H2020 EU-KR WISE-IoT project and has been evaluated by the citizens of Santander, Spain through a prototype. It has also been demonstrated at three occasions. Additionally, we develop an IoT recommender for smart skiing that provides skiing routes comprised of specific types of slopes, as well as the nearest slope. For skiing routes, there does not exist any stable routing engine. Therefore, a novel routing engine for skiing routes was developed. This work has also been integrated into the smart skiing use case of WISE-IoT project.

The developed IoT recommender for smart parking is GDPR-compliant. However, it does not fully protect the privacy of the users. Because, an indiscriminately sharing of users' data with an untrusted or semi-trusted third-party IoT parking recommender system causes a breach of privacy, as user's behavior and mobility patterns can be inferred by analyzing the past travelling history of the users. Therefore, we preserve the privacy of users against parking recommender system while analyzing their past parking history using k-anonymity (anonymization) and differential privacy (perturbation) techniques.

Lastly, since the smart cities applications are developed in a vertical manner and do not talk/communicate with each other, i.e., each application is developed for a certain scenario which generally does not share data with other smart cities applications. Therefore, we proposed two frameworks for the recommendation services across smart cities applications using social IoT. Firstly, on how social IoT can be used for the recommendation services across smart cities applications, and secondly, we propose another type of communication of social IoT at a global level, i.e., social cross-domain application-to-application communications, that enables smart cities applications to communicate with each other and establish social relationships between them. These frameworks are the building blocks for cross-domain recommendations in smart cities applications.

Institut Polytechnique de Paris
91120 Palaiseau, France

Doctor of Philosophy (PhD) Thesis
Institut-Mines Télécom, Télécom SudParis
& Institut Polytechnique de Paris (IP Paris)

Specialization

Computer Science and Networks

presented by

Yasir Saleem Shaikh

Privacy Preserving Internet of Things Recommender Systems for Smart Cities

Commitee:

Payam Barnaghi	Reviewer	Professor, University of Surrey - United Kingdom
JaeSeung Song	Reviewer	Associate Professor, Sejong University - South Korea
Maria Potop-Butucaru	Examiner	Professor, Sorbonne University, LIP6 - France
Luigi Atzori	Examiner	Professor, University of Cagliari - Italy
Martin Bauer	Examiner	Senior Researcher, NEC Europe Ltd - Germany
Noel Crespi	Advisor	Professor, Institut-Mines Telecom, Telecom SudParis - France
Roberto Minerva	Co-advisor	Assistant Professor, Institut-Mines Telecom, Telecom SudParis - France

Thèse de Doctorat (PhD) de
Institut-Mines Télécom, Télécom SudParis
et l'Institut Polytechnique de Paris (IP Paris)

Spécialité

Informatique et Réseaux

présentée par

Yasir Saleem Shaikh

Protection de la Confidentialité des Services de Recommandation pour les Villes Intelligentes

Jury composé de :

Payam Barnaghi	Rapporteur	Professor, University of Surrey - United Kingdom
JaeSeung Song	Rapporteur	Associate Professor, Sejong University - South Korea
Maria Potop-Butucaru	Examinateur	Professor, Sorbonne University, LIP6 - France
Luigi Atzori	Examinateur	Professor, University of Cagliari - Italy
Martin Bauer	Examinateur	Senior Researcher, NEC Europe Ltd - Germany
Noel Crespi	Directeur de thèse	Professor, Institut-Mines Telecom, Telecom SudParis - France
Roberto Minerva	Co-encadrant	Maître de Conférences, Institut-Mines Telecom, Telecom SudParis - France

Dedication

To My Family

Acknowledgements

First and foremost, I am thankful to Almighty ALLAH for each and every blessing. By His bounty and blessing, I am able to conduct this research and write this thesis.

A deep and special thanks goes to my thesis director Prof. Noel Crespi who gave me the opportunity to do this research and provided me all the support and freedom. I always enjoyed working and talking to him and it increased my exposure and knowledge everytime I talked to him. I would like to thank my co-supervisor Dr. Roberto Minerva for very fruitful and enjoyable discussions with him all the times. He always gave me very useful suggestions which helped me a lot to come up with new ideas, as well as improved my research.

I wish to express my since gratitude to my thesis reviewers, Prof. Payam Barnaghi and Prof. JaeSeung Song, for their useful reviews and suggestions, which helped me to improve the quality of my thesis. A special thank to Prof. Maria Potop-Butucaru, Prof. Luigi Atzori and Dr. Martin Bauer for being the part of my jury as examiners for my thesis defense.

A very special thanks goes to Mubashir bhai who has been mentoring me like a brother since 2011. He always motivated me a lot and I really appreciate his suggestions and advices. I am also very grateful to my elder brother Farrukh bhai. They both encouraged me, supported me, provided advices on every entangled situation, as well as continuous motivation whenever I lacked inspiration. Their motivations gave me a new zeal all the times.

I wish to dedicate a very special thanks to Mufti Naeem Memon Sahib for his prayers, guidance and moral support. I always used to discuss with him whenever I lacked motivation, faced difficulties or had confusions and he always helped me to get rid of them. I always had a fresh mind and high morale after talking to him.

My special thanks to all my team members, specially Praboda, Samin, Ibrahim, Faraz, Reza, Marzieh, Hamza, Komal, Shanay, Amir, Koosha, Ehsan and Shohreh. Most of my team members were also my neighbors and I alway felt like a family with them. Praboda was always my closest friend with whom I used to share everything and she is my secret keeper. I will always miss the coffee/tea parties and hangouts with colleagues and friends, specially with Praboda, Samin, Ibrahim and Marzieh. Special thanks goes to Valerie Mateus, the secretary of RS2M Department. She was always very kind and generous in solving tedious administrative tasks. A deep thanks to Veronique Guy, the administrative responsible of PhD program, who always helped me a lot in dealing with PhD administrative tasks.

Finally, my profound love, respect and thanks goes to my family members : Abbu, Ammi, Rida, Tariq bhai, Farrukh bhai, Mehwish Aapi and Tooba. They always prayed for me and supported me with encouragement to achieve this new and hard milestone of my life. A special thanks to my wife Rida who stood with me during the hard time of my PhD, tolerated all the hardships and provided her continuous support. I hope they find here the expression of my deep gratitude and appreciation.

Yasir Saleem Shaikh
21^{st} January 2020

Abstract

During the past decade, the Internet of Things (IoT) has revolutionized almost all the fields of daily life and has boosted the development of smart city applications. Smart cities use IoT technologies to collect various types of sensors' data (e.g., traffic, parking, weather and environmental) and then use such data to offer a variety of applications, such as intelligent transportation systems, smart parking and smart grid, to name a few. The objective of smart cities is to improve the quality of governmental services and citizens' welfare. Since the smart cities' applications are used by the citizens, therefore providing the customized recommendation services to the citizens based on their preferences, locations and profiles, as well as by exploiting IoT data (e.g., traffic congestion and parking occupancy) is of great importance which could be provided by an IoT recommender. However, since an IoT recommender utilizes the private data of citizens (e.g., profiles, preferences and habits), it breaches the privacy of the users because IoT recommenders could track the routines and habits of the users by analyzing the historical database or by analyzing the regular recommendation services it offers. Therefore, it is important to preserve the privacy of the users while using IoT recommender systems.

In this thesis, we propose a novel privacy preserving IoT recommender system for smart cities that provides recommendations by exploiting the IoT data of sensors and by considering various metrics. Our approach is organized in three parts. Firstly, we develop an EU General Data Protection Regulation (GDPR)-compliant IoT recommender system for smart parking system that provides recommendations of parking spots and routes by exploiting the data of parking and traffic sensors. For this, we first propose an approach for the mapping of traffic sensors with route coordinates in order to analyze the traffic conditions (e.g., the level of congestion) on the roadways and then developed an IoT recommender. The mapping algorithm of traffic sensors coordinates on the routes has been evaluated using simulations in terms of correct detection, missed detection and false detection. The IoT recommender provides four-fold functions. Firstly, it helps a user to find a free parking spot based on different metrics (e.g., nearest or nearest trusted parking spot). Secondly, it recommends a route (the least crowded or the shortest route) leading to the recommended parking spot from the user's current location. Thirdly, it provides the real-time provisioning of expected availability of parking areas (comprised of parking spots organized into groups) in a user-friendly manner. Finally, it provides a GDPR-compliant implementation for operating in a privacy-aware environment. The IoT recommender has been integrated into the smart parking use case of the H2020 EU-KR WISE-IoT project and has been evaluated by the citizens of Santander in Spain through a prototype. We have developed an IoT recommender for smart skiing that provides skiing routes comprised of specific types of slopes. For skiing routes, there are not any stable routing engines. Therefore, a novel routing engine for skiing routes was developed. This work has also been integrated into the smart skiing use case in the WISE-IoT project and has been evaluated by comparing it with OpenSnowMap for different types of slopes with different expertise

levels.

The developed IoT recommender for smart parking is GDPR-compliant. However, it does not fully protect the privacy of the users. Because, an indiscriminately sharing of users' data with an untrusted or semi-trusted third-party IoT parking recommender system causes a breach of privacy, as user's behavior and mobility patterns can be inferred by analyzing the past travelling history of the users. Therefore, we preserve the privacy of users against parking recommender system while analyzing their past parking history using k-anonymity (anonymization) and differential privacy (perturbation) techniques. The main novelty in this contribution is that k-anonymity and differential privacy have not previously been applied in smart parking, specifically for preserving the users privacy in parking database. k-anonymity has been evaluated for various values of k and for different combinations of attributes in terms of average group size, total number of groups, generalization height, number of suppressed records, discernibility cost and execution time. Differential privacy has been evaluated for various values of privacy budget and sensitivity values in terms of mean absolute error and root mean square error.

Lastly, since the smart cities applications are developed in a vertical manner and do not talk/communicate with each other, i.e., each application is developed for a certain scenario which generally does not share data with other smart cities applications. Therefore, we proposed two frameworks for the recommendation services across smart cities applications using social IoT. Firstly, on how social IoT can be used for the recommendation services across smart cities applications, and secondly, we propose another type of communication of social IoT at a global level, i.e., social cross-domain application-to-application communications, that enables smart cities applications to communicate with each other and establish social relationships between them. These frameworks are the building blocks for cross-domain recommendations in smart cities applications.

Keywords

Internet of Things, IoT Recommender, Smart Cities, Smart Parking, Smart Skiing, Privacy Preservation, k-anonymity, Differential Privacy, Cross-Domain.

Résumé

Au cours de la dernière décennie, la technologie Internet des objets (IoT) a révolutionné presque tous les domaines de la vie quotidienne et a dynamisé les villes intelligentes. Les villes intelligentes utilisent la technologie IoT pour collecter divers types de données de capteurs (trafic, stationnement, météo et environnement), puis les utilisent pour offrir diverses applications, telles que les systèmes de transport intelligent, le stationnement intelligent, le réseau intelligent et le ski intelligent, pour n'en nommer que quelques-uns. Les villes intelligentes ont pour objectif d'améliorer la qualité des services gouvernementaux et le bien-être des citoyens. Comme les applications des villes intelligentes sont utilisées par les citoyens, donc leur fournir des services de recommandation personnalisés en fonction de leurs préférences, de leurs localisations et de leurs profils ainsi que l'exploitation des données IoT (par exemple, la congestion du trafic et l'occupation du parking) est d'une grande importance qui pourrait être fournie par un recommandateur IoT. Cependant, comme le recommandateur IoT utilise les données privées des citoyens (profils, préférences et habitudes, par exemple), il viole la vie privée des utilisateurs car il pourrait suivre les routines et les habitudes des utilisateurs en analysant la base de données historique ou en analysant les services de recommandation réguliers qu'il offre. Par conséquent, il est important de préserver la confidentialité des utilisateurs du programme de recommandation IoT.

Dans cette thèse, nous proposons un nouveau système de recommandation IoT préservant la confidentialité pour les villes intelligentes, qui fournit des recommandations en exploitant les données IoT des capteurs et en tenant compte de diverses métriques. Notre approche est organisée en trois parties. Tout d'abord, nous développons un système de recommandation IoT conforme au européen règlement sur la protection des données (RGPD) pour les systèmes de stationnement intelligent. Ces systèmes fournissent des recommandations sur les emplacements et les itinéraires de stationnement en exploitant les données des capteurs de stationnement et de circulation. Par conséquence, nous proposons d'abord une approche pour la cartographie des capteurs de trafic avec les coordonnées d'itinéraires afin d'analyser les conditions de trafic (par exemple le niveau de congestion) sur les routes Ensuite, nous avons mis en place un dispositif de recommandation IoT. L'algorithme de cartographie des coordonnées des capteurs de trafic sur les itinéraires a été évalué à l'aide de simulations en termes de détection correcte, de détection manquée et de fausse détection. Ce dispositif offre quatre fonctions. Premièrement, il aide l'utilisateur à trouver une place de parking gratuite en fonction de différentes mesures (par exemple, la place de stationnement fiable ou la plus proche). Deuxièmement, il recommande un itinéraire (l'itinéraire le moins fréquenté ou l'itinéraire le plus court) menant à l'emplacement de stationnement recommandé à partir de l'emplacement actuel de l'utilisateur. Troisièmement, il fournit la disponibilité en temps réel des zones de stationnement prévues (comprenant des places de stationnement organisées en groupes) de manière conviviale. Enfin, il fournit une implémentation conforme au RGPD pour fonctionner dans un environnement sensible à la confidentialité. Le recommandateur IoT a été intégré au scénario d'utilisation du stationnement intelligent d'un projet

H2020 EU-KR WISE-IoT et a été évalué par les citoyens de Santander, en Espagne, à l'aide d'un prototype. Il a également été démontré à trois reprises. De plus, nous développons un recommendateur IoT pour le ski intelligent qui fournit des itinéraires de ski comprenant des types de pistes spécifiques. Pour les itinéraires de ski, il n'existe aucun moteur de calcul stable. Par conséquent, un nouveau moteur de routage pour les itinéraires de ski a été développé. Ce travail a également été intégré dans le cas d'utilisation du ski intelligent du projet WISE-IoT et a été évalué en le comparant avec OpenSnowMap pour différents types de pistes avec différents niveaux d'expertise.

Deuxièmement, bien que le recommandateur IoT développé pour le stationnement intelligent soit conforme au RGPD, il ne protège toutefois pas totalement la vie privée des utilisateurs. En effet, le partage sans discernement des données des utilisateurs avec un système tiers de recommandation de stationnement IoT non approuvé ou semi-fiable provoque une violation de la vie privée. En effet, le comportement et les schémas de mobilité des utilisateurs pouvant être déduits en analysant l'historique de leurs déplacements. Par conséquent, nous préservons la confidentialité des utilisateurs contre le système de recommandation de stationnement tout en analysant leur historique de stationnement en utilisant des techniques de k-anonymity (anonymisation) et de confidentialité différentielle (perturbation). La principale nouveauté de cette contribution est que k-anonymity et la confidentialité différentielle n'ont pas été appliqués auparavant dans le stationnement intelligent, en particulier pour préserver la confidentialité des utilisateurs dans la base de données de stationnement. k-anonymity a été évalué pour différentes valeurs de k et pour différentes combinaisons d'attributs en termes de taille moyenne de groupe, nombre total de groupes, hauteur de généralisation, nombre d'enregistrements supprimés, coût de discernibilité et temps d'exécution. La confidentialité différentielle a été évaluée pour diverses valeurs du budget de confidentialité et des valeurs de sensibilité en termes d'erreur absolue moyenne et d'erreur quadratique moyenne.

Enfin, étant donné que les applications de villes intelligentes sont développées de manière verticale et ne se parlent pas, c'est-à-dire que chaque application est développée pour un certain scénario qui ne partage généralement pas les données avec d'autres applications de villes intelligentes. Par conséquent, nous avons proposé deux cadres pour les services de recommandation parmi les applications de villes intelligentes utilisant l'IoT social. Tout d'abord, sur la manière dont l'IoT social peut être utilisé pour les services de recommandation entre applications de villes intelligentes. Deuxièmement, nous proposons un autre type de communication de l'IoT social au niveau mondial, à savoir les communications inter-domaines sociales qui permettent aux applications de villes intelligentes de communiquer entre elles et établir des relations sociales entre elles. Ces frameworks sont les blocs de construction des recommandations inter-domaines dans les applications de villes intelligentes.

Mots-clés

Internet des objets, recommandation IoT, villes intelligentes, stationnement intelligent, ski intelligent, préservation de la vie privée, k-anonymat, confidentialité différentielle, inter-domaines.

Table of contents

Chapter 1

Introduction

Contents

1.1 Motivation

During the past decade, Internet of Things (IoT) has revolutionized almost all the fields of daily life and has boosted smart cities. Smart cities use IoT technologies to collect various types of sensors' data (e.g., traffic, parking, weather and environmental) and then use such data to offer a variety of applications, such as intelligent transportation systems, smart parking, smart grid and smart skiing, to name a few. The objective of smart cities is to improve the quality of governmental services and citizens welfare. Since the smart cities' applications are used by the citizens, city authorities and urban planners, therefore providing the customized recommendation services to them based on their preferences, locations and profiles, as well as by exploiting the IoT data (e.g., traffic congestion, parking occupancy) is of great importance.

Smart parking and smart skiing are two major examples of smart cities. A smart parking system provides the recommendation of available parking spots to the drivers looking for them, thereby minimizing the time spent on finding available parking spots, as well as minimizing the cost associated with hiring humans for manual parking management. In addition, in the recommendation of parking spots, it is equally important to consider the traffic on the route leading to the recommended parking spots and to recommend the least congested route. With the enforcement of the EU General Data Protection Regulation (GDPR), protecting the privacy of EU citizens throughout the data collection, data storage and data processing of a user's personal data is now a basic requirement. Parking systems gather a lot of contextual data and it is quite possible that the users' personal data can be collected indirectly. GDPR affects also smart parking applications and hence, the smart parking systems should therefore be designed in a way that protects user's privacy and thus be GDPR-compliant. Additionally, such smart parking systems could also be applied to other parts of the world having similar privacy preservation concerns. A smart skiing system recommends slopes to the skiers based on the level of their expertise (e.g., novice, beginner, intermediate or advance) as well as recommends routes between two points on ski resort passing through ski slopes and ski lifts. The route recommendation in skiing is quite different from the route recommendation on roads because of different nature of medium. Hence, a mechanism is needed that can recommend the specific types of slopes and routes between two points for skiing.

For recommendations in smart parking, the parking database comprised of users' past history is shared with a recommender system for efficient and personalized recommendations. However, since the parking database contains the private data of users (e.g., parking history, profiles, preferences and habits), it breaches the privacy of the users because a recommender system could track the routines and habits of the users by analyzing the his-

torical database or by analyzing the regular offered recommendation services. Therefore, privacy of the users must be protected.

Additionally, many of the smart cities applications available today have been developed in a vertical manner by focusing on a specific scenario or use case without considering data exchange and reuse with other smart cities applications [1]. This very specific focus results in poor service because of the lack of integration of different data and hence the interoperability in the smart cities data and systems. However, if smart cities applications could collaborate by exchanging and reusing each other's data, opportunities for new value-added and more efficient recommendation services could be generated.

The main goal of this thesis is to provide privacy preserving recommendation services in smart cities and frameworks for recommendation services across smart cities applications. To this end, we consider two applications of smart cities: smart parking and smart skiing for recommendation services.

1.2 Objectives of the Thesis

In this section, we present the main objectives of this thesis. We address each objective with one contribution. This thesis aims to design a privacy preserving recommendation services for smart cities. The main objectives to achieve this aim are as follows:

- To design an IoT recommender for smart cities. The aim is to provide the IoT recommenders for smart cities, as well as to study the required component in order to achieve this aim.
- To preserve the privacy of IoT recommender that is designed in the first objective.
- To provide frameworks for cross-domain recommendation services in smart cities.

1.3 Contributions of the Thesis

Our approach to achieve the above research objectives is organized into three parts as three contributions, each corresponding to each research objective. We discuss each contribution as follows:

C.1 The first contribution is on developing IoT recommender systems for smart cities that offer recommendations based on IoT data. Firstly, it proposes an algorithm for the mapping of sensor and route coordinates that is used for analyzing the sensors data (e.g., traffic sensors data) on the routes to infer and analyze traffic conditions on the roads. Secondly, it provides two IoT recommender systems for smart parking and

smart skiing. From a recommendation perspective, the IoT recommender is different from a general recommender system in that IoT recommender mainly considers the actual IoT data from parking and traffic sensors, rather than the data shared or acquired by users' terminals. Secondly, the IoT recommender also considers the trust by interacting with the Trust Monitoring of WISE-IoT project to offer trusted recommendations. Thirdly, it provides GDPR-compliant implementation that works in a privacy-aware environemnt. Fourthly, it offers the expected availability (occupancy statistics) of parking areas to users and enables them to analyze the weekly, monthly and yearly statistics statistics by themselves. More specifically, the first contribution provides three sub-contributions, as follows:

C.1.1 Firstly, it proposes an algorithm for the mapping of sensors and route coordinates by introducing a deviation margin. It presents an algorithm and two illustrative examples that cover all the possible scenarios. It evaluates the performance of mapping algorithm by considering four different routes and measures the correct detection, missed detection and false detection of traffic sensors on the routes.

C.1.2 Secondly, it presents an IoT recommender for smart parking that uses the mapping algorithm proposed in the first part and provides four-fold functions: recommendation of parking spots based on different metrics (e.g., nearest or nearest trusted), recommendation of routes leading to the recommended parking spots (the least crowded or the shortest route), real-time provisioning of expected availability of parking spots, and a GDPR-compliant implementation for operating in a privacy-aware environment. It offers its services using REST APIs and has been integrated into and evaluated in an H2020 EU-KR project, as well as been demonstrated in three occasions.

C.1.3 Thirdly, it presents an IoT recommender for smart skiing that provides the recommendations of ski routes between two points on a ski resort, passing through ski slopes and ski lifts by allowing to specify specific types of slopes based on the level of expertise (e.g., novice, easy, intermediate or advance). It offers its service through REST APIs and has also been integrated in an H2020 EU-KR project.

C.2 The second contribution is about the privacy preservation of IoT recommender for smart parking, that is presented in the first contribution. Although the IoT recommender for smart parking is GDPR-compliant, however, it does not fully protect the privacy of the users. Because, an indiscriminately sharing of users' data with an IoT parking recommender system causes a breach of privacy as user's behavior and

mobility patterns can be inferred by analyzing the past travelling history of users. Therefore, we preserve the privacy of users against parking recommender system while analyzing their past parking history using k-anonymity (anonymization) and differential privacy (perturbation) techniques. It also extensively evaluates the performance of both privacy preservation techniques in terms of privacy and utility.

C.3 The third contribution is in the application domain in which we proposed two frameworks for recommendations across smart cities applications: one on how social IoT can be used for recommendation services, and second on the social cross-domain application-to-application communications. Since smart cities applications are developed in a vertical manner and do not talk / communicate with each, i.e., each application is developed for a certain scenario which generally does not share data with other smart cities applications, therefore, these frameworks are the building blocks for cross-domain recommendations in smart cities application. More specifically, the third contribution provides two sub-contributions, as follows:

C.3.1 Firstly, it proposes a framework on the exploitation of social IoT for recommendation services across smart cities applications. It presents a sample application scenario as well as implementation challenges for the realization of this conceptual framework.

C.3.2 Secondly, it proposes a framework on a new type of communication of social IoT at global level, i.e., social cross-domain IoT application-to-application communications. It presents the conceptual framework, some use case scenarios and challenges to realize this concept.

1.4 Project Contribution

The work of Chapter 3 in this thesis has been performed as part of an European Union's Horizon 2020 research and innovation collaborative programme between Europe and South Korea (H2020 EU-KR), titled as "Worldwide Interoperability for Semantic IoT (WISE-IoT)". The work is performed under the grant agreement No. 723156, the Swiss State Secretariat for Education, Research and Innovation (SERI) and the South-Korean Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MISP) (No. R7115-16-0002) with a consortium of 19 partners (9 from EU and 10 from South Korea) and includes 3 industries, 6 SMEs, 6 universities, 3 leading research institutes and 1 city municipality. The partners include EGM, NEC, SJU, KAIST, IMT-TSP, CEA, UC, LJMU, SAN, FHNW, PIQ, KNU, KETI, SKT, SAMSUNG SDS, GBC, SOLUM, IReIS and GSPA.

WISE-IoT addressed the fragmentation between different IoT standards and their ecosystems. It offered gateways and morphing procedures that bridged heterogeneous IoT deployments and translated data expressed by using one ontology into another. These innovations facilitated the global interoperability and mobility of IoT applications and devices. The main objective of WISE-IoT project it to deepen the interoperability and interworking of IoT existing systems. It was a use-case-driven project which exploited the experience and expertise available in the consortium to build a comprehensive mediation framework that can be used across various IoT systems. Another objective was to build up federated and interoperable platforms by ensuring end-to-end security and trust for reliable business environments with a multiplicity of IoT applications. Building synergies with national and international initiatives in both EU and KR, the project addressed standardization requirements, fostering IoT development and interoperability.

The IoT recommender of this thesis in Chapter 3 was developed in the WISE-IoT project and has been integrated and deployed into two use cases of WISE-IoT of smart parking and smart skiing.

1.5 Publications List

Journal Papers

- Yasir Saleem, Noel Crespi, Mubashir Husain Rehmani and Rebecca Copeland. "Internet of Things-Aided Smart Grid: Technologies, Architectures, Applications, Prototypes, and Future Research Directions", *IEEE Access*, 2019, vol. 7, pp. 62962-63003.

- Ayesha Naeem, Mubashir Husain Rehmani, Yasir Saleem, Imran Rashid and Noel Crespi. "Network Coding in Cognitive Radio Networks: A Comprehensive Survey", *IEEE Communications Surveys & Tutorials*, 2017, vol. 19, no. 3, pp. 1945-1973.

- Faraz Malik Awan, Yasir Saleem, Roberto Minerva and Noel Crespi, "A Comparative Analysis of Machine/Deep Learning Models for Prediction of Parking Space Availability", *Sensors*, vol. 20, no. 1, 322.

Conference Papers

- Yasir Saleem and Noel Crespi. "Mapping of Sensor and Route Coordinates for Smart Cities", *In 42nd IEEE International Conference on Computers, Software & Applications: Staying Smarter in a Smartening World (COMPSAC)*, 23-27 July 2018, Tokyo, Japan, pp. 570-576.

- Yasir Saleem, Noel Crespi, Mubashir Husain Rehmani, Rebecca Copeland, Dina Hussein and Emmanuel Bertin. "Exploitation of Social IoT for Recommendation Services", *In IEEE World Forum on Internet of Things (WF-IoT)*, 12-14 Dec 2016, Reston, VA, USA, pp. 359-364.

- Yasir Saleem, Noel Crespi and Pasquale Pace. "SCDIoT: Social Cross-Domain IoT enabling Application-to-Application Communications", *In IEEE International Conference on Cloud Engineering (IC2E)*, 17-20 April 2018, Orlando, FL, USA, pp. 346-350.

Under Review

- Yasir Saleem, Pablo Sotres, Samuel Fricker, Carmen Lopez de la Torre, Noel Crespi, Gyu Myoung Lee, Roberto Minerva and Luis Sanchez. "IoTRec: The IoT Recommender for Smart Parking System", *IEEE Transactions on Emerging Topics in Computing (TETC).*

- Yasir Saleem, Mubashir Husain Rehmani, Noel Crespi and Roberto Minerva, "Privacy Preservation of Parking Recommender System through Anonymization and Differential Privacy", *Engineering Reports.*

1.6 Relationship of Publications with Contributions

In this section, we provide the relationships of publications with contributions.

- The publication '*Mapping of Sensor and Route Coordinates for Smart Cities*' corresponds to Contribution C.1.1 in Section 3.2.

- The publication '*IoTRec: The IoT Recommender for Smart Parking System*' corresponds to Contribution C.1.2 in Section 3.3.

- The publication '*Privacy Preservation of Parking Recommender System through Anonymization and Differential Privacy*' corresponds to Contribution C.2 in Chapter 4.

- The publication '*Exploitation of Social IoT for Recommendation Services*' corresponds to Contribution C.3.1 in Section 5.3.

- The publication '*SCDIoT: Social Cross-Domain IoT enabling Application-to-Application Communications*' corresponds to Contribution C.3.2 in Section 5.4.

1.7 Outline of the Thesis

The thesis is structured into six chapters.

- **Chapter 1** describes the background of the research topics, motivation, contributions of this thesis, project contribution, summary of each chapter and the outline of the thesis.
- **Chapter 2** presents the background and related technologies relevant to the main topics of this thesis, i.e., IoT, smart cities, smart parking, smart skiing, recommendation services in smart cities, privacy preservation, anonymization, differential privacy, social IoT, cross-domain recommendation services and semantic web.
- **Chapter 3** presents the IoT recommender which is divided into three parts: i) the mapping of sensors and route coordinates, ii) the IoT recommender for smart parking and iii) the IoT recommender for smart skiing.
- **Chapter 4** presents the privacy preservation of parking recommender system through k-anonymity and differential privacy and extensively studies the performance of both techniques.
- **Chapter 5** presents two frameworks for the recommendations across smart cities applications.
- **Chapter 6** summarizes the thesis and discusses possible future directions for the advancements of this thesis.

Chapter 2

Background and Related Technologies

Contents

2.1 Overview

The background and related technologies presented in this chapter give a general overview relevant to the main topics of the thesis. Later on, a separate and detailed overview of the related work will be discussed for each study in this thesis.

2.2 IoT Recommender for Smart Cities

In this section, we provide the background and related technologies that are required for the IoT recommender system for smart cities.

2.2.1 Internet of Things

Initially, the Internet provided connectivity for people-to-people and people-to-things. By 2008, the number of things connected to the Internet exceeded the number of people in the world, which has increased the impact of the Internet of Things (IoT) because more and more devices are connected to the Internet. IoT is a network of physical objects or things that contains embedded technology to interact with their internal and external environments. These objects are often connected to the Internet and can sense, control, analyze and decide in an autonomous, distributed and collaborated manner with other objects. Some of the objectives of the IoT applications are tracking, location identification, monitoring and management. The application design in the IoT is based on three main concepts: things-oriented, Internet-oriented and semantic-oriented. The things-oriented concept deals with smart objects such as sensors and actuators, and RFIDs [2]. The Internet-oriented concept enables smart objects to communicate with other objects using a number of telecommunication technologies, such as cellular communications and ZigBee [3], and connects them to the Internet. The semantic-oriented concept deals with applications that are built using smart objects or devices.

The IoT has attained considerable attention over the past few years in a number of applications. It has enabled the interconnection of network-embedded objects used in our daily life to the Internet, as well as enabled the automation of many systems in parking management, leisure, power grids, agriculture and health care [4].

2.2.2 Smart Cities

Most modern cities have faced the problems of traffic congestion, public safety and shortage of resources, in addition to other urban problems. Smart cities make an attempt to solve these problems with the help of IoT technologies by leveraging the infrastructures and resources of cities to become integrated and automated systems. Smart cities use IoT

technologies to collect various types of sensors' data (e.g., traffic, parking, weather and environmental) and then use such data to offer a variety of applications, such as intelligent transportation system, smart parking, smart grid and smart skiing, to name a few. The objective of smart cities is to improve the quality of government services and citizens welfare with the goal of turning the applications and systems into smart environments [5]. Many initiatives around the world have been made to the study and development of smart cities that target various sectors, such as parking management, traffic management, waste management, energy management, urban mobility and healthcare. Some examples of such initiatives include SmartSantander [6] (Santander, Spain), Digital Heidelberg [7] (Heidelberg, Germany), Global Smart City [8] (Busan, South Korea), Amsterdam Smart City [9] (Amsterdam, Netherlands), ConnectingCopenhagen [10] (Copenhagen, Denmark), MiNT Madrid Inteligente [11] (Madrid, Spain) and DubaiNow [12] (Dubai, UAE), to name a few. In this thesis, our main focus of study is on SmartSantander [6].

2.2.3 Smart Parking

Smart parking is one of the major example of smart cities. Traditionally, drivers try to find available parking spots on the streets by driving around, only locating a parking spot empirically due to their local knowledge and luck. This practice wastes a significant amount of both time and fuel, and sometimes it is impossible to find an available parking spot during high vehicle traffic times. One solution would be to find a parking area with high capacity of free parking spots, increasing the chances of getting a parking spot. However, this parking area could be very far from the user's destination. Another solution is to design a system that shows free parking spots to the driver and lets the driver chooses a free parking spot manually. However, this is not an optimal solution because firstly, it is an extra task for the drivers to select a parking spot by themselves. Secondly, the path leading to the selected parking spot could be very congested, causing the parking spot to be occupied when the driver arrives. In additional to finding the available parking spots, it is equally important to consider the traffic on the route to each available parking spot and to recommend the least congested route leading to the recommended parking spot to avoid frustrations to the drivers by stucking in the traffic.

Smart parking has solved this problem of finding the available parking spots to the drivers looking for them, as well as it provides routes leading to the parking spots, thereby minimizing the time spent on finding free parking spots and stucking on the crowded roads. It also minimizes the cost associated with hiring humans for manual parking management [13–15]. Smart parking has been widely considered by many smart cities initiatives because of its high importance by the citizens in their daily life.

2.2.4 Smart Skiing

Leisure activities are important for the physical well-being and mental wellness and for the health of citizens. Skiing is one of the leisure activity for physical and mental wellness. Therefore, for the comfort of citizens, the smart city enriches the skiing by turning it into smart skiing. Smart skiing offers various services, such as slope and route recommendations, gamification, accident detection and coordination, and injury prediction and avoidance. In this thesis, our focus of study is on slope and route recommendations for smart skiing.

2.2.5 Recommendation Services in Smart Cities

Since the smart cities' applications are tailored to improve the lifestyle and welfare of the citizens, therefore providing the customized recommendation services to the citizens based on their preferences, locations and profiles, as well as by exploiting the IoT data (e.g., traffic congestion, parking occupancy, leisure) is of great importance which demand the need of IoT recommender systems.

The IoT recommender system should provide the recommendations based on the users preferences. For instance, for smart parking, the IoT recommender should provide the recommendations of parking spots with which the users have good experience in the past, as well as the least crowded routes leading to the recommended parking spots. For smart skiing, generally, in ski resorts, there are different types of slopes for different expertise levels (e.g., novice, beginner, intermediate and advance) which are one-way, i.e., the skiers can go from top to down but not vice versa. To reach the top of the slope, the skiers need to take ski lifts. Such constraint should be consider by the IoT recommender system for slope and route recommendations in smart skiing. Firstly, the IoT recommender for smart skiing should be able to recommend a slope based on the level of expertise of the skier. Secondly, while recommending a route (either to reach to the recommended slope or between two points in a ski resort), it should be able to recommend a route that should pass through ski lifts and the specific type of the slope in accordance with the level of expertise of the skier.

2.2.5.1 Requirements

Here we list the requirements that the IoT recommender needs to consider in its design.

Requirements for IoT Recommender for Smart Parking

- Consideration of IoT data of parking sensors for the recommendation of parking spots.
- Consideration of IoT data of traffic sensors for the recommendation of least congested route leading to the recommended parking spot.

- Consideration of user preference (e.g., trusted or nearest parking spot, least congested or shortest route.
- Interaction with the trust monitoring component to obtain trust score based on users' experience and sensors quality.
- Provide user the parking statistics to analyse in a user-friendly interface and allow him to choose a parking area by himself as well.
- Offer RESTful APIs to enable the reuse of the service offered by the IoT recommender for smart parking.

Requirements for IoT Recommender for Smart Skiing

- Consideration of expertise level of skier in ski route recommendation.
- Consideration of the slope to be one way.
- Consideration of ski lifts to have fixed starting and ending points (or stations).
- Offer RESTful APIs to enable the reuse of the services offered by the IoT recommender for smart skiing.

2.3 Privacy Preservation

The IoT recommender discussed above accesses the database comprised of users' past history in order to provide efficient and personalized recommendations. However, since the database contains the private data of users (e.g., history, profiles, preferences and habits), it breaches the privacy of the users because a recommender system could track the routines and habits of the users by analyzing the historical database or by analyzing the regular recommendation services it offers. Therefore, the privacy of users must be protected. In this section, we present the widely adopted privacy preservation models.

2.3.1 Anonymization

In this section, we discuss three anonymization techniques for privacy preservation: k-anonymity, ℓ-diversity and t-closeness. The anonymization technique preserves privacy by anonymizing the data and is applied on the microdata. The microdata is raw data that contains the information of the users, comprised of multiple attributes (or columns) [16]. The attributes in microdata are categorized into three types: i) *explicit identifiers* that can identify a user uniquely, e.g., *social security number*, ii) *quasi-identifiers* that can identify a

user when they are combined together, e.g., *age, gender, zipcode*, iii) *sensitive attributes* are the attributes that need to be protected, e.g., *salary* [17]. The first step in anonymization is to remove the explicit identifer.

2.3.1.1 k-anonymity

$k-$anonymity [18] is the earliest work on privacy preservation that anonymizes a dataset in such as a way that with respect to the set of quasi-identifier attributes, each record (or row) is indistinguishable from at least $k-1$ other records. It achieves anonymization using generalization and suppression. The main purpose of k-anonymity is to counter against the linking attacks in which an adversary could not be able to uniquely identify a user by linking the quasi-identifier attributes (such as birthdate, zip code and gender) with external data. k-anonymity is suitable for non-interactive data publishing when there is no sensitive attribute or the distribution of sensitive attribute is sparse. In this approach, the data publisher (i.e., curator) does not want to get involve in answering all the queries and instead, releases an anonymized dataset that will be queried by the recommender systems. k-anonymity is discussed in Section 4.4.1 formally and in more details.

2.3.1.2 ℓ-diversity

k-anonymity protects from linking attacks (i.e., privacy against identifying the records), however it is susceptible to two other types of attacks of homogeneity and background knowledge attacks. In homogeneity attack, if all the sensitive attributes are same in a group of k records, the value of sensitive attribute can be identified by an adversary. In background attack, an adversary uses background knowledge to identify the individuals. To address the limitation of k-anonymity, Machanavajjhala et al., [19] extended k-anonymity by proposing ℓ-diversity that requires each record in a group to have at least 'ℓ' diverse values for the sensitive attribute. ℓ-diversity is also suitable for non-interactive data publishing when the data publisher wants to release an anonymized dataset and does not want to get involved in answering each query. However, unlike k-anonymity, ℓ-diversity is used when the anonymized dataset should contain each record in a group to have at least 'ℓ' diverse values for the sensitive attribute. It is formally defined as:

"An equivalence group fulfills ℓ-diversity if it has at least 'ℓ' well-represented values for the sensitive attribute. A dataset having equivalence groups, all of which are ℓ-diverse, is said to be an ℓ-diverse dataset."

In brief, ℓ-diversity ensures intra-group heterogeneity of sensitive attributes by at least 'ℓ' different values. If k=ℓ, ℓ-diversity automatically satisfies k-anonymity.

2.3.1.3 t-closeness

Although ℓ-diversity was proposed to solve the limitations of k-anonymity, however, Li et al., [20] proved that ℓ-diversity does not completely counter against the homogeneity attack. They used two types of attacks: skewness attack and similarity attack to demonstrate the limitation of ℓ-diversity. In skewness attack, the anonymized dataset has skewed distribution of sensitive attribute in equivalence groups and ℓ-diversity failed to prevent the attack because the distribution of the sensitive attribute is different from the dataset. In similarity attack, the anonymized dataset has distinct values of sensitive attribute in equivalence groups but they are semantically similar. ℓ-diversity also failed to prevent the attack because an adversary can estimate the value of a sensitive attribute by linking it to another sensitive attribute. These limitations of ℓ-diversity are overcome by Li et al. [20] by proposing t-closeness. t-closeness is also suitable for non-interactive data publishing and is used when a dataset that needs to be anonymized has sensitive attributes. It is suitable when the sensitive attribute has skewed distribution or distinct values in the equivalence groups of anonymized dataset. t-closeness is fomally defined as:

"An equivalence group fulfills t-closeness if the distance between the distribution of a sensitive attribute in this group and that in the whole dataset is no more than a threshold t. A dataset fulfills t-closeness if all the equivalence groups have t-closeness."

2.3.2 Differential Privacy

Differential privacy was first coined by Dwork [21] with the definition that the output of a differentially private mechanism is not highly affected by the addition or the removal of a single record of dataset. It can protect the privacy of users while sharing the database with the untrusted entity by perturbing the data. It overcomes the limitation of anonymization techniques, specifically the curse of dimensionality [22]. Differential privacy uses interactive data publishing and is suitable when the curator wants to answer each query of the recommender systems by adding the noise. Differential privacy is discussed in Section 4.4.2 formally and in more details.

2.4 Cross-Domain Recommendation Services in Smart Cities

2.4.1 Social Internet of Things

Social IoT is the application of social networking concepts to the IoT which is initially proposed by Atzori et al. [23]. It is attracting much attention these days, as it establishes social relationships among smart objects (or things) so that they can collaborate with each other autonomously without human intervention. The motivation behind the Social IoT is

to enhance the selection, discovery and composition of resources through social relationships and circles among objects in the same manner as in social relationships among humans in a social network [23], [24].

In the social IoT, there may be five types of relationships for defining relationship profiles within social IoT network: social object relationship (SOR), ownership object relationship (OOR), co-work object relationship (CWOR), co-location object relationship (CLOR) and parental object relationship (POR) [23], [25]. An SOR is established when objects are in direct contact with each other. This direct contact can either be continuous or sporadic and is among the objects' owners (such as devices associated with friends). An OOR is established between heterogenous objects having the same owner. A CWOR is established among objects collaborating with each other to achieve some common goals. A CLOR is established among objects (can be either homogeneous or heterogeneous) that operate in the same environment (such as smart cities, smart buildings and smart homes). CWOR and CLOR are built among objects in a fashion similar to how humans share their public or personal experiences. A POR is built among heterogeneous objects belonging to the same owner/manufacturer having the same period in which the production batch acts like a family.

These relationships are established and updated according to the characteristics of objects (such as battery life, computational power, type and brand) and activities, and are used by resource discovery components to find the objects that can offer the required services (similar to how humans look for friendships and information). Additionally, to manage the relationships, a relationship management component is required by the Social IoT architecture to enable cognition and intelligence into the Social IoT which can allow the objects to establish, maintain and terminate the relationships as needed. These relationships may be built based on several parameters, such as required services, providing connectivity to disconnected objects, a publish-subscribe model and the distance between objects. In this context, Nitti et al., [25] presented a scheme of friendship selection in the social IoT to improve information diffusion.

2.4.2 Cross-domain Recommendation Services

A smart city is comprised of various applications, such as smart parking, smart skiing, intelligent transportation system, smart grid, healthcare and vehicle-to-vehicle communications. Such applications are developed in a vertical manner by focusing on a specific scenario or use case without considering data exchange and reuse with other smart cities applications. This very specific focus results in poor service because of the lack of integration of different data and hence the interoperability across smart cities applications. However, if these applications could collaborate by exchanging and reusing each other's

data, opportunities for new value-added and more efficient cross-domain recommendation services could be generated. Additionally, the social IoT can enable social relationship between the applications, thus allowing them to interact in a similar manner as humans interact with each other using their social networks. For instance, let us consider two smart cities applications of smart parking and smart skiing discussed above. As smart parking also considers traffic information to recommend the least congested route, hence if both smart parking and smart skiing can collaborate with each other, new recommendation services could be generated. For example, a user of social IoT is going for skiing. His social IoT system could interact with his health devices to know about his health conditions in order to recommend the slopes suitable for him, could recommend him the least congested route from his home to the ski resort, as well as could plan his other activities (e.g., hangout with his friends or movie) by considering his expected return time from skiing.

2.4.3 Semantic Technologies in IoT

The role of social IoT in cross-domain recommendation services is to enable social relationships among smart cities applications and to provide context awareness. However, it could not enable the interoperability between them to allow them to communicate with each other. Since each application is developed in a vertical manner using different semantics, therefore there is a need of some mechanism for this purpose. Here, semantic web technologies help to achieve this objective. Semantic web technologies [26] are getting popular and have been widely adopted by industries, such as Google for search engine. Semantic web technologies enable interoperability between smart cities applications [27] and hence enable cross-domain recommendations services across IoT applications using social IoT. The semantic web technologies enables interoperability and data exchange by explicitly defining the description of each application's data in an structured manner, hence allowing machines to read and understand the applications data. Subsequently, they enable data integration by converting heterogeneous applications' data into the same vocabulary [28]. Once interoperability is achieved among smart cities applications, the social IoT enables social relationships among applications and provides context-awareness, hence providing application-to-application communications and recommendations services.

2.4.4 Integration of IoT Recommender into Smart City Architecture

The IoT recommender has been integrated into a smart city architecture of WISE-IoT project. Figure 2.1 presents the WISE-IoT architecture [29] that is comprised of various components. The oneM2M platform is used to store the data of IoT sensor devices because it has strong support for the functionalities of IoT devices, e.g., device registration and management, data management and repository, etc. oneM2M platform has a con-

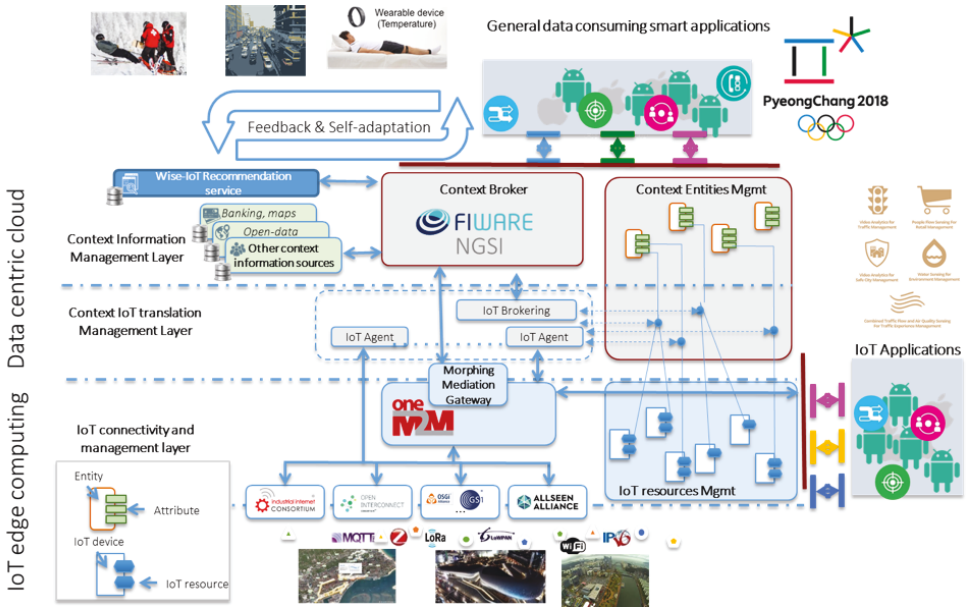


Figure 2.1 – WISE-IoT architecture [29].

tainer resource for semantic annotation containing sensor readings as content instances. The Adaptive Semantic Module of the Morphing Mediation Gateway [30] discovers the semantically annotated data in oneM2M platform by subscribing to the sensors readings. On triggering the availability of a new sensor reading, the Adaptive Semantic Module creates an NGSI data structure to update FIRWARE Orion Context Broker [31]. One of the component of WISE-IoT architecture is 'Self-Adaptive Recommender (SAR)' system that is responsible for 'Wise-IoT Recommendation Service' in the architecture and the IoT recommender is a part of SAR system. The SAR and IoT recommender then access FIWARE Orion Context Broker to access the parking and traffic sensors data. Figure 2.2 presents the SAR architecture that is comprised of various components. The IoT Recommender is integrated into it as one of its component. The IoT Recommender interacts with other SAR components (e.g., Adherence Monitor, QoI Monitor and Trust Monitor) to provide more efficient recommendation services, as well as with the SAR component to interact with use case applications.

2.5 Summary and Conclusion

This chapter presented a general overview of the major topics relevant to this thesis. To summarize, it covered three major areas in three parts. In the first part, it discussed IoT recommender for smart cities and presented the IoT, smart cities, smart parking, smart

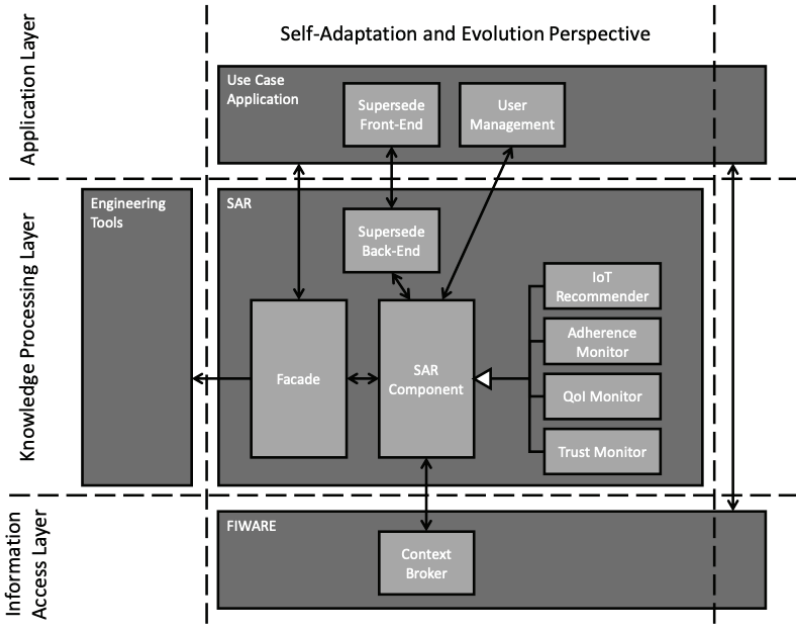


Figure 2.2 – Self-Adaptive Recommender (SAR) system architecture [32].

skiing and recommendation services in smart cities. In the second part, it discussed the privacy preservation and presented four privacy preservation techniques of k-anonymity, ℓ-diversity, t-closeness and differential privacy. In the third part, it discussed the cross-domain recommendation services in smart cities and presented social IoT, cross-domain recommendation services and semantic web. In addition to this chapter, for each study in this thesis, a separate related work will be discussed focusing on the main relevant works to the specific study.

In the next chapter, we will discuss the IoT recommenders for smart cities that provide recommendation based on IoT data.

Chapter 3

IoT Recommender

Contents

3.1 Introduction

The main focus of this chapter is to propose IoT recommender for smart cities to provide recommendations based on IoT data. To this end, an algorithm is proposed for the mapping of sensors and route coordinates (to be used in IoT recommender) and two IoT recommenders are proposed for smart parking and skiing use cases.

A high-level diagram of interfaces of IoT recommender is presented in Figure 3.1. More specifically, this chapter is organized into three parts. The first part in Section 3.2 proposes an algorithm for the mapping of sensors and route coordinates by introducing a deviation margin. Because we found that in Santander, some traffic sensors coordinates were deviated from the coordinates of the main routes, hence making it impractical to analyze the road traffic from the data of traffic sensors. The performance of the proposed algorithm is evaluated using correct detection, missed detection and false detection. The second part in Section 3.3 presents an IoT recommender for smart parking that is developed in and EU-KR H2020 WISE-IoT project [33]. It recommends parking spots and routes using the mapping algorithm proposed in the first part. It also provides the real-time expected availability (occupancy statistics) of parking areas. It offers its services through REST APIs and has been integrated and deployed in WISE-IoT project. It is evaluated by the citizens of Santander through a prototype. The third part in Section 3.4 presents an IoT recommender for smart skiing that is also developed in WISE-IoT project [33]. It provides the recommendations of ski routes from one point to another on a ski resort (recommending a route on ski resort is different from recommending a route on roads). It also allows to specify the specific type of slopes, such as novice, easy, intermediate or advance. It offers its services through REST APIs and has also been integrated into WISE-IoT project.

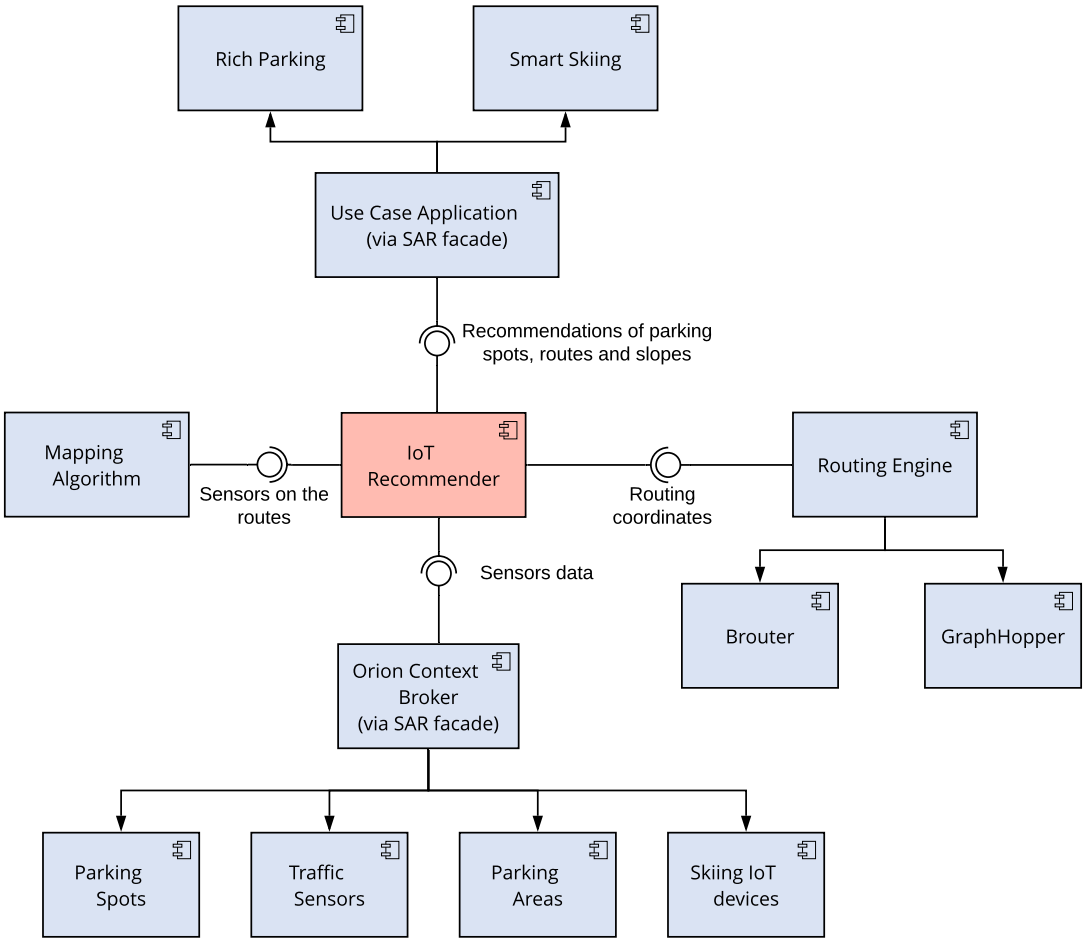


Figure 3.1 – A high-level diagram of interfaces of IoT recommender.

3.2 Mapping of Sensors and Route Coordinates

In the past few years, the Internet of Things (IoT) has attracted increasing interest in its potential application in a number of domains, including transportation, healthcare, smart cities and the smart grid. The exponential growth in urban populations has resulted in a higher number of cars in cities than ever before, causing traffic congestion and higher pollution levels. A number of IoT solutions to address this problem have been developed for intelligent transportation systems (ITS) through various smart city projects [34–37].

In smart cities, several types of sensors are installed throughout a city, such as traffic, parking and weather sensors. Traffic sensors, deployed on the roads/streets, monitor the traffic conditions, such as the road load, the number of vehicles and their speed. Such

traffic sensors are useful in recommending the least congested route towards a destination point, as well as in distributing the traffic to different routes to reduce the traffic congestion. According to one study, 30% of traffic congestion is the result of drivers looking for parking spots [38]. Therefore, by managing the traffic crowd, we can also efficiently manage parking spots, e.g., the parking system can take the traffic congestion into account through traffic sensors and subsequently can recommend different routes leading to parking spots for those seeking them, thereby minimizing the scenario of high traffic congestion on some routes and low traffic congestion on the other routes. In this manner, we can achieve a more balanced distribution of traffic on all the routes.

However, there is a challenge to achieve the above objective. Generally, most solutions focus on the development of applications assuming that they have readily available information about the mapping of traffic sensors on the routes. These solutions do not focus on the mapping of sensor coordinates into the routes. Inspired by this lacunae, this study presents a new approach for the mapping of sensor coordinates into the routes. We propose an algorithm for this mapping, utilizing real traffic sensors deployed in the city of Santander, Spain to demonstrate their mapping abilities with four random routes between two points. Since traffic sensors are deployed in large numbers, some traffic sensors may reside outside of the edges of streets (i.e., a small distance from the routes). In order to best incorporate this constraint, our proposed approach considers a deviation margin that allows some flexibility; the main novelty and contribution of this study. We evaluate the performance of our proposed approach in terms of correct detection, missed detection (non-detection) and false detection of sensors on the routes, showing the effective and significant advantage of our proposed approach. We believe that our proposed approach will be helpful in the development of various smart city applications, such as traffic management and smart parking. The main contribution of this study is proposing an algorithm that considers a deviation margin for the mapping of sensor coordinates with the route coordinates. Our proposed system has been exploited in a smart parking use-case of an H2020 EU-KR WISE-IoT project [33] as well.

3.2.1 Related Work

While much work is being done in IoT for the development of smart cities, that work is mainly focused on the applications, such as smart parking, ITS, traffic management etc. Those studies do not consider the mapping of sensor coordinates into the routes, to the best of our knowledge. For instance, Lau [36] designed a framework that considers user-contributed posts about traffic and road conditions and analyzes driving navigation information from the archived data on online social media. Subsequently, it transmits the collected data to ITS for its dissemination to other drivers. However, this framework

considers that raw data collection from sensors and their mapping to the routes are not really within the objectives of those cities. Another work [35] studied the joint prediction of road traffic and parking occupancy in a city by using machine learning techniques. However, it does not work on the mapping of traffic sensors with the routes, instead this technique is mainly focused on applying machine learning for predicting road traffic and parking occupancy by using real-time data. Fernandez et al., [34] studied real traffic and mobility scenarios for a smart city by using real traffic and mobility data gathered in the city of Granada, Spain. Their main purpose was to analyze the collected data using Big Data techniques and subsequently derive useful information. However, they also do not focus on the mapping of sensor coordinates with route coordinates and simply assume the availability of such mapping.

One important observation is that traffic prediction is a very old problem; but we are not working on traffic prediction. We have proposed a novel approach for the mapping of sensor coordinates with route coordinates, which is the basic pre-requisite of traffic prediction.

3.2.2 Mapping of Sensor Coordinates with Route Coordinates

This section presents our proposed algorithm for the mapping of sensor coordinates with route coordinates. The traffic sensors used in Santander, Spain are magneto-resistive sensors that detect the movement and presence of vehicles. They operate at 2.4 GHz frequency band and 250Kbps data rate. They are located at the main entrances of the Santander city and are buried under the asphalt. They measure the main traffic parameters, e.g., road occupancy, vehicle speed, traffic volumes and queue length [6]. For simplicity, we assumed that the incoming and outgoing traffic on two-way roads is similar. Therefore, the identified traffic sensors on either side of the road represent similar traffic conditions. This assumption is reasonable for our current system as we deployed it for testing purposes in Santander, Spain where the traffic sensors are deployed on one-way streets.

Algorithm 1 presents the mapping of traffic sensors' coordinates with a route's coordinates. The inputs of this algorithm are route coordinates (longitudes and latitudes) which are comprised of starting coordinates (route start longitude $R_{strt,lon}$ and route start latitude $R_{strt,lat}$) and ending coordinates (route end longitude $R_{end,lon}$ and route end latitude $R_{end,lat}$), traffic sensor coordinates (traffic sensor longitude TS_{lon} and traffic sensor latitude TS_{lat}) and finally the deviation margin D. The deviation margin D is a tunable parameter which can be set based on the scenario (i.e., how much is the deviation of the traffic sensors' coordinates from the main route, in general). We will explain deviation margin D in detail in the following discussion.

Part I of the algorithm checks whether the route start longitude $R_{strt,lon}$ is less than or equal to the route end longitude $R_{end,lon}$; a necessary step in order to determine the

Algorithm 1 Mapping of traffic sensor coordinates with route coordinates.

Input: Route start longitude ($R_{strt,lon}$), route end longitude ($R_{end,lon}$), route start latitude ($R_{strt,lat}$), route end latitude ($R_{end,lat}$), traffic sensor longitude (TS_{lon}), traffic sensor latitude (TS_{lat}), deviation margin (D)
/ Part I */*
if $R_{strt,lon} <= R_{end,lon}$ **then**
    / Part I(a) */*
    if $TS_{lon} >= R_{strt,lon}$ & $TS_{lon} <= R_{end,lon}$ & $TS_{lat} >= R_{strt,lat}$ & $TS_{lat} <= R_{end,lat}$ **then**
        isMatched = true;
    else if $TS_{lon} >= R_{strt,lon}$ & $TS_{lon} <= R_{end,lon}$ & $TS_{lat} <= R_{strt,lat}$ & $TS_{lat} >= R_{end,lat}$ **then**
        isMatched = true;
    end if
    / Part I(b) */*
    if (not isMatched) **then**
        if $(TS_{lon} + D >= R_{strt,lon}$ & $TS_{lon} - D <= R_{end,lon})$ & $(TS_{lat} + D >= R_{strt,lat}$ & $TS_{lat} - D <= R_{end,lat})$ **then**
            isMatched = true;
        else if $(TS_{lon} + D >= R_{strt,lon}$ & $TS_{lon} - D <= R_{end,lon})$ & $(TS_{lat} - D <= R_{strt,lat}$ & $TS_{lat} + D >= R_{end,lat})$ **then**
            isMatched = true;
        end if
    end if
    / Part II */*
else if $R_{strt,lon} > R_{end,lon}$ **then**
    / Part II(a) */*
    if $TS_{lon} <= R_{strt,lon}$ & $TS_{lon} >= R_{end,lon}$ & $TS_{lat} <= R_{strt,lat}$ & $TS_{lat} >= R_{end,lat}$ **then**
        isMatched = true;
    else if $TS_{lon} <= R_{strt,lon}$ & $TS_{lon} >= R_{end,lon}$ & $TS_{lat} >= R_{strt,lat}$ & $TS_{lat} <= R_{end,lat}$ **then**
        isMatched = true;
    end if
    / Part II(b) */*
    if (not isMatched) **then**
        if $(TS_{lon} - D <= R_{strt,lon}$ & $TS_{lon} + D >= R_{end,lon})$ & $(TS_{lat} - D <= R_{strt,lat}$ & $TS_{lat} + D >= R_{end,lat})$ **then**
            isMatched = true;
        else if $(TS_{lon} - D <= R_{strt,lon}$ & $TS_{lon} + D >= R_{end,lon})$ & $(TS_{lat} + D >= R_{strt,lat}$ & $TS_{lat} - D <= R_{end,lat})$ **then**
            isMatched = true;
        end if
    end if
end if
return isMatched;

Figure 3.2 – Irregular deployment of traffic sensors at Santander, Spain.

direction of the route. This part consists of two sub-parts: Part I(a) checks the exact location of traffic sensors on the route, while Part I(b) uses a deviation margin D which gives some flexibility to traffic sensors that are deviated from the main route.

In Part I(a), there are two possible scenarios for the existence of traffic sensor coordinates on the route. Firstly, the traffic sensor's longitude TS_{lon} is greater than or equal to the route's start longitude $R_{strt,lon}$ (i.e., $TS_{lon} \geq R_{strt,lon}$) and less than or equal to the route's end longitude $R_{end,lon}$ (i.e., $TS_{lon} \leq R_{end,lon}$), and the traffic sensor's latitude TS_{lat} is greater than or equal to the route's start latitude $R_{strt,lat}$ (i.e., $TS_{lat} \geq R_{strt,lat}$) and less than or equal to the route's end latitude $R_{end,lat}$ (i.e., $TS_{lat} \leq R_{end,lat}$). Secondly, the traffic sensor's longitude TS_{lon} is greater than or equal to route's start longitude $R_{strt,lon}$ (i.e., $TS_{lon} \geq R_{strt,lon}$) and less than or equal to the route's end longitude $R_{end,lon}$ (i.e., $TS_{lon} \leq R_{end,lon}$), and the traffic sensor's latitude TS_{lat} is less than or equal to the route's start latitude $R_{strt,lat}$ (i.e., $TS_{lat} \leq R_{strt,lat}$) and greater than or equal to the route's end latitude $R_{end,lat}$ (i.e., $TS_{lat} \geq R_{end,lat}$).

However, as presented in Figure 3.2, given the irregular deployment of traffic sensors in Santander, Spain, it might be possible that a traffic sensor is slightly deviated from the main route. For this situation, we propose a deviation margin D in order to give some flexibility in the detection of traffic sensors on the route. Hence, in Part I(b) of the algorithm, we

have added the deviation margin D to the traffic sensor coordinates when comparing them with the route start coordinates (i.e., while checking whether a traffic sensor's coordinates are greater than the route start coordinates) and subtracted the deviation margin D from traffic sensor coordinates when comparing with route end coordinates (i.e., while checking whether a traffic sensor's coordinates are less than the route's end coordinates). This helps to give some flexibility for the traffic sensor coordinates that are deviated from the straight path of the route. Subsequently, after adding and subtracting the deviation margin D to and from the traffic sensor coordinates when comparing them with start and end route coordinates, respectively, they are compared in a similar manner as presented in Part I(a).

If the route start longitude is greater than route end longitude of the route, Part II will be executed. Part II also consists of two sub-parts: Part II(a) checks the exact existence of traffic sensor on the route, while Part II(b) uses a deviation margin D, which gives some flexibility to traffic sensors which are deviated from the main route.

In Part II(a), there are again two possible scenarios for the existence of traffic sensor coordinates on the route. Firstly, the traffic sensor's longitude TS_{lon} is less than or equal to the route's start longitude $R_{strt,lon}$ (i.e., $TS_{lon} \leq R_{strt,lon}$) and greater than or equal to the route's end longitude $R_{end,lon}$ (i.e., $TS_{lon} \geq R_{end,lon}$), and the traffic sensor's latitude TS_{lat} is less than or equal to the route's start latitude $R_{strt,lat}$ (i.e., $TS_{lat} \leq R_{strt,lat}$) and greater than or equal to the route's end latitude $R_{end,lat}$ (i.e., $TS_{lat} \geq R_{end,lat}$). Secondly, the traffic sensor's longitude TS_{lon} is less than or equal to the route's start longitude $R_{strt,lon}$ (i.e., $TS_{lon} \leq R_{strt,lon}$) and greater than or equal to the route's end longitude $R_{end,lon}$ (i.e., $TS_{lon} \geq R_{end,lon}$), and the traffic sensor's latitude TS_{lat} is greater than or equal to the route's start latitude $R_{strt,lat}$ (i.e., $TS_{lat} \geq R_{strt,lat}$) and less than or equal to the route's end latitude $R_{end,lat}$ (i.e., $TS_{lat} \leq R_{end,lat}$).

If the traffic sensor is not identified on the route in Part I, Part II uses a deviation margin D to check the existence of traffic sensors on the route. The reason for using a deviation margin is explained before in the description of Part I(b) of the algorithm. However, in contrast to Part I(b), in Part II(b), the deviation margin D is subtracted from the traffic sensor's coordinates when comparing them with the route's start coordinates (i.e., while checking whether a traffic sensor's coordinates are less than the route's start coordinates), and added to the traffic sensor's coordinates when comparing with the route's end coordinates (i.e., while checking whether a traffic sensor's coordinates are greater than the route's end coordinates). Subsequently, after subtracting and adding the deviation margin D from and to the traffic sensor's coordinates when comparing them with the start and end route coordinates, respectively, this part is compared in a similar procedure in Part II(a).

Note that although we can combine the two '*if*' conditions into one within each sub-

part, we have presented them separately for a better understanding, as well as a better differentiation using the examples (presented in next Section 3.2.3). We have combined them in our implementation in order to reduce the processing load.

3.2.3 Examples

For better understanding of the algorithm, we present two illustrative examples to demonstrate the operation of the algorithm for the mapping of traffic sensor coordinates with route coordinates. The first example demonstrates the scenario in which the traffic sensor lies exactly within the routes (i.e., there is no deviation), while the second example demonstrates the scenario in which traffic sensors are slightly deviated from the route, and hence we will use deviation margin D to detect such traffic sensors on the route.

3.2.3.1 Example 1 (no deviation)

In this section, we demonstrate the first scenario in which traffic sensor lies exactly within the route without any deviation.

Figure 3.3 presents four different routes covering all the possible scenarios of coordinates mapping with different colors and line patterns. The traffic sensor which is our main point of interest to be detected on the route is located at the center (i.e., the red map marker with black dot in the middle). For all the coordinates of map markers in the figure, the first part represents the longitude and the second part represents the latitude. For example, TS (-3.8085, 43.4730) shows the traffic sensor having longitude = -3.8085 and latitude = 43.4730. Similarly, $R_{1,1}$(-3.8085, 43.4721) represents point 1 (can be either starting or ending point) of route 1 having longitude = -3.8085 and latitude = 43.4721.

Let us start with the route in blue color $R_{1,1} \rightarrow R_{1,2}$ by considering $R_{1,1}$ (-3.8085, 43.4721) as starting point and $R_{1,2}$ (-3.8085, 43.4741) as ending point. Here, the route start longitude (-3.8085) is equal to route end longitude (-3.8085), so it matches Part I of Algorithm 1. Within Part I, it satisfies the first condition of Part I(a), i.e., traffic sensor longitude (-3.8085) is equal to the route start longitude (-3.8085) and is also equal to the route end longitude (-3.8085), and traffic sensor latitude (43.4730) is greater than route start latitude (43.4721) and is less than route end latitude (43.4741). Hence, the traffic sensor is identified within the route. In a similar manner, the green route $R_{2,1} \rightarrow R_{2,2}$ with $R_{2,1}$ (-3.8096, 43.4723) as starting point and $R_{2,2}$ (-3.8075, 43.4737) as ending point also fulfills the first condition of Part I(a). For the route in red color $R_{3,1} \rightarrow R_{3,2}$ with $R_{3,1}$ (-3.8102, 43.4730) as starting point and $R_{3,2}$ (-3.8071, 43.4730) as ending point, it meets both conditions defined in Part I(a) of the algorithm. Hence, the traffic sensor is identified within the route by fulfilling both conditions of Part I(a). The purple route $R_{4,1} \rightarrow R_{4,2}$

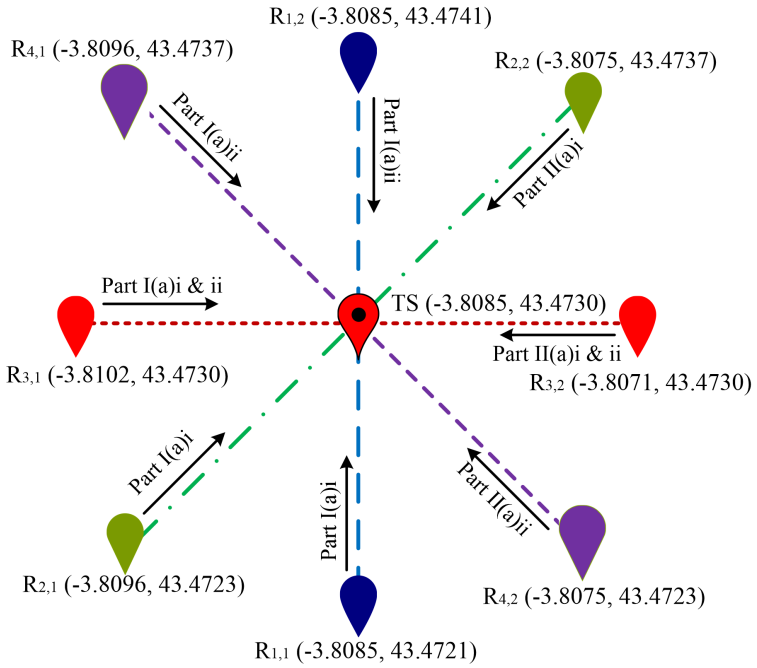


Figure 3.3 – An example of coordinates mapping having no deviation. The lines originating from and terminating at map markers, represent the routes. The arrowhead lines represent the direction of the route (i.e., starting and ending points) and the "Part xyz" above the arrowhead lines corresponds to the matching parts mentioned in Algorithm 1.

with $R_{4,1}$ (-3.8096, 43.4737) as starting point and $R_{4,2}$ (-3.8075, 43.4723) as ending point, as well as blue route $R_{1,2} \rightarrow R_{1,1}$ with $R_{1,2}$ (-3.8085, 43.4741) as starting point and $R_{1,1}$ (-3.8085, 43.4721) as ending point, fulfill the second condition of Part I(a).

The above described routes match Part I of the algorithm, while the remaining routes match Part II of the algorithm which we are going to present next. The green route $R_{2,2} \rightarrow R_{2,1}$ with $R_{2,2}$ (-3.8075, 43.4737) as starting point and $R_{2,1}$ (-3.8096, 43.4723) as ending point matches Part II of the algorithm because start route longitude (-3.8075) is greater than end route longitude (-3.8096). Within Part II, it matches the first condition of Part II(a), i.e., traffic sensor longitude (-3.8085) is less than route start longitude (-3.8075) and greater than route end longitude (-3.8096), and traffic sensor latitude (43.4730) is less than route start latitude (43.4737) and greater than route end latitude (43.4723). Similar to the red route in Part I(a), the red route $R_{3,2} \rightarrow R_{3,1}$ with $R_{3,2}$ (-3.8071, 43.4730) as starting point and $R_{3,1}$ (-3.8102, 43.4730) as ending point fulfills both conditions defined in Part II(a) of the algorithm. Finally, the purple route $R_{4,2} \rightarrow R_{4,1}$ with $R_{4,2}$ (-3.8075, 43.4723) as starting point and $R_{4,1}$ (-3.8096, 43.4737) as ending point matches the second condition in Part II(a).

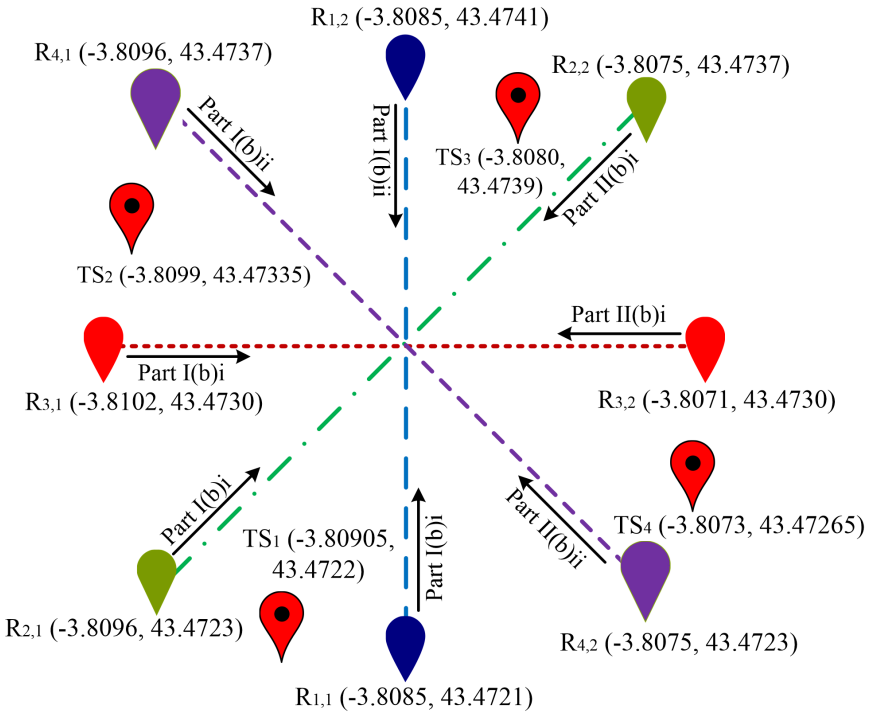


Figure 3.4 – An example of coordinates mapping with deviation. The lines originating from and terminating at map markers, represent the routes. The arrowhead lines represent the direction of the route (i.e., starting and ending points) and the "Part xyz" above the arrowhead lines corresponds to the matching parts mentioned in Algorithm 1.

Hence, the traffic sensor is successfully identified within all the routes $R_{1,1} \rightarrow R_{1,2}$, $R_{2,1} \rightarrow R_{2,2}$, $R_{3,1} \rightarrow R_{3,2}$, $R_{4,1} \rightarrow R_{4,2}$, $R_{1,2} \rightarrow R_{1,1}$, $R_{2,2} \rightarrow R_{2,1}$, $R_{3,2} \rightarrow R_{3,1}$ and $R_{4,2} \rightarrow R_{4,1}$. This example also verifies the correct operation of Algorithm 1 without deviation margin D.

3.2.3.2 Example 2 (with deviation)

In this section, we demonstrate the second scenario in which traffic sensors do not lie within the exact routes, rather they are deviated from the routes.

Similar to previous figure, Figure 3.4 presents four different routes covering all the possible scenarios with different colors and line patterns. Unlike Figure 3.3, due to deviation, we cannot have a single traffic sensor which can cover all the possible scenarios, therefore, we have presented four traffic sensors (i.e., TS_1, TS_2, TS_3 and TS_4) to be detected. Each traffic sensor, located between the two routes, is used for both routes separately to check its existence on the route by using deviation margin D. For instance, traffic sensor TS_1 is used for routes $R_{1,1} \rightarrow R_{1,2}$ and $R_{2,1} \rightarrow R_{2,2}$, TS_2 is used for routes $R_{3,1} \rightarrow R_{3,2}$ and $R_{4,1}$

$\rightarrow R_{4,2}$, TS_3 is used for routes $R_{1,2} \rightarrow R_{1,1}$ and $R_{2,2} \rightarrow R_{2,1}$, and TS_4 is used for routes $R_{3,1} \rightarrow R_{3,2}$ and $R_{4,1} \rightarrow R_{4,2}$. For all the coordinates of map markers in the figure, the first part represents the longitude and the second part represents the latitude. For example, TS_1 (-3.80905, 43.4722) shows the traffic sensor 1, having longitude = -3.80905 and latitude = 43.4722. Similarly, $R_{1,1}$(-3.8085, 43.4721) shows point 1 (can be either starting or ending point) of route 1 ($R_{1,1} \rightarrow R_{1,2}$) having longitude = -3.8085 and latitude = 43.4721. Note that in this example, we have set deviation margin $D = 0.0006$.

Let us start with the route in blue color $R_{1,1} \rightarrow R_{1,2}$ by considering $R_{1,1}$ (-3.8085, 43.4721) as starting point and $R_{1,2}$ (-3.8085, 43.4741) as ending point. The traffic sensor TS_1 (-3.80905, 43.4722) will be checked using deviation margin D whether it exists or not in the route $R_{1,1} \rightarrow R_{1,2}$. Here, the route start longitude (-3.8085) is equal to route end longitude (-3.8085), so it matches the Part I of Algorithm 1. Within Part I, it does not satisfy any of the condition in Part I(a). For instance, the traffic sensor longitude (-3.80905) is less than route start longitude (-3.8085) and is also less than route end longitude (-3.8085). This condition is not fulfilled which is the preliminary part of both conditions within Part I(a), therefore the status 'isMatched' is still 'false' and so, it goes to Part I(b). Here, the traffic sensor TS_1 longitude (-3.80905) plus deviation margin $D = 0.0006$ (-3.80905 + 0.0006 = -3.80845) is greater than route start longitude (-3.8085), and traffic sensor longitude (-3.80905) minus deviation margin $D = 0.0006$ (-3.80905 − 0.0006 = -3.80965) is less than route end longitude (-3.8085). The traffic sensor TS_1 latitude (43.4722) plus deviation margin $D = 0.0006$ (43.4722 + 0.0006 = 43.4728) is greater than route start latitude (43.4721), and traffic sensor TS_1 latitude (43.4722) minus deviation margin $D = 0.0006$ (43.4722 − 0.0006 = 43.4716) is less than route end latitude (43.4741), therefore it matches the first condition in Part I(b) of Algorithm 1. Hence, the traffic sensor TS_1 is identified within the route $R_{1,1} \rightarrow R_{1,2}$ by using deviation margin D.

In a similar manner, for traffic sensor TS_1 (-3.80905, 43.4722), the green route $R_{2,1} \rightarrow R_{2,2}$ with $R_{2,1}$ (-3.8096, 43.4723) as starting point and $R_{2,2}$ (-3.8075, 43.4737) as ending point, as well as for traffic sensor TS_2 (-3.8099, 43.47335), the red route $R_{3,1} \rightarrow R_{3,2}$ with $R_{3,1}$ (-3.8102, 43.4730) as starting point and $R_{3,2}$ (-3.8071, 43.4730) as ending point, both also fulfill the first condition of Part I(b). For traffic sensor TS_2 (-3.8099, 43.47335), the purple route $R_{4,1} \rightarrow R_{4,2}$ with $R_{4,1}$ (-3.8096, 43.4737) as starting point and $R_{4,2}$ (-3.8075, 43.4723) as ending point, as well as for traffic sensor TS_3 (-3.8080, 43.4739), the blue route $R_{1,2} \rightarrow R_{1,1}$ with $R_{1,2}$ (-3.8085, 43.4741) as starting point and $R_{1,1}$ (-3.8085, 43.4721) as ending point, both fulfill the second condition of Part I(b). Hence, the traffic sensor TS_1, TS_2, TS_2 and TS_3 are identified within the routes $R_{2,1} \rightarrow R_{2,2}$, $R_{3,1} \rightarrow R_{3,2}$, $R_{4,1} \rightarrow R_{4,2}$ and $R_{1,2} \rightarrow R_{1,1}$, respectively, using deviation margin D.

The above routes match Part I of the algorithm, while the remaining routes fulfill Part

II of the algorithm which we are going to present next. For traffic sensor TS_3 (-3.8080, 43.4739), the green route $R_{2,2} \rightarrow R_{2,1}$ with $R_{2,2}$ (-3.8075, 43.4737) as starting point and $R_{2,1}$ (-3.8096, 43.4723) as ending point matches Part II of the algorithm because route start longitude (-3.8075) is greater than route end longitude (-3.8096). Within Part II, it does not match any condition within Part II(a), rather it matches the first condition of Part II(b), i.e., traffic sensor TS_3 longitude (-3.8080) minus deviation margin $D = 0.0006$ (-3.8085 − 0.0006 = -3.8086) is less than route start longitude (-3.8075), and traffic sensor TS_3 longitude (-3.8080) plus deviation margin $D = 0.0006$ (-3.8085 + 0.0006 = -3.8074) is greater than route end longitude (-3.8096). The traffic sensor TS_3 latitude (43.4739) minus deviation margin $D = 0.0006$ (43.4739 − 0.0006 = 43.4733) is less than route start latitude (43.4737) and traffic sensor TS_3 latitude (43.4739) plus deviation margin $D = 0.0006$ (43.4739 + 0.0006 = 43.4745) is greater than route end latitude (43.4723). Hence, the traffic sensor TS_3 is identified within the route using the deviation margin D. Finally, for traffic sensor TS_4 (-3.8073, 43.47265), the red route $R_{3,2} \rightarrow R_{3,1}$ with $R_{3,2}$ (-3.8071, 43.4730) as starting point and $R_{3,1}$ (-3.8102, 43.4730) as ending point fulfills the first condition of Part II(b), while for the same traffic sensor TS_4 (-3.8073, 43.47265), the purple route $R_{4,2} \rightarrow R_{4,1}$ with $R_{4,2}$ (-3.8075, 43.4723) as starting point and $R_{4,1}$ (-3.8096, 43.4737) as ending point fulfills the second condition of Part II(b).

In summary, traffic sensors: TS_1 in routes $R_{1,1} \rightarrow R_{1,2}$ and $R_{2,1} \rightarrow R_{2,2}$, TS_2 in routes $R_{3,1} \rightarrow R_{3,2}$ and $R_{4,1} \rightarrow R_{4,2}$, TS_3 in routes $R_{1,2} \rightarrow R_{1,1}$ and $R_{2,2} \rightarrow R_{2,1}$, and TS_4 in routes $R_{3,2} \rightarrow R_{3,1}$ and $R_{4,2} \rightarrow R_{4,1}$ have been successfully detected which complies and verifies the successful operation of Algorithm 1 for identifying deviated traffic sensors on the routes using deviation margin D.

3.2.4 Performance Evaluation

In this section, we evaluate the performance of our proposed approach for the mapping of sensors coordinates into route coordinates.

3.2.4.1 Evaluation Setup

Figure 3.5 shows the deployment of traffic sensors at Santander, Spain. For performance evaluation, we randomly selected two locations which serve as starting and destination points. Subsequently, we plotted four possible routes using different colors between these two points by using Brouter offline routing engine [39] (a third-party application) and applied our proposed algorithm to map traffic sensors into these routes which is presented in Figure 3.6. This figure provides an overview of mapping of traffic sensor coordinates into the route coordinates. We will provide the detailed analysis of mapping using and without using deviation margin D in this section.

Figure 3.5 – Deployment of traffic sensors at Santander, Spain.

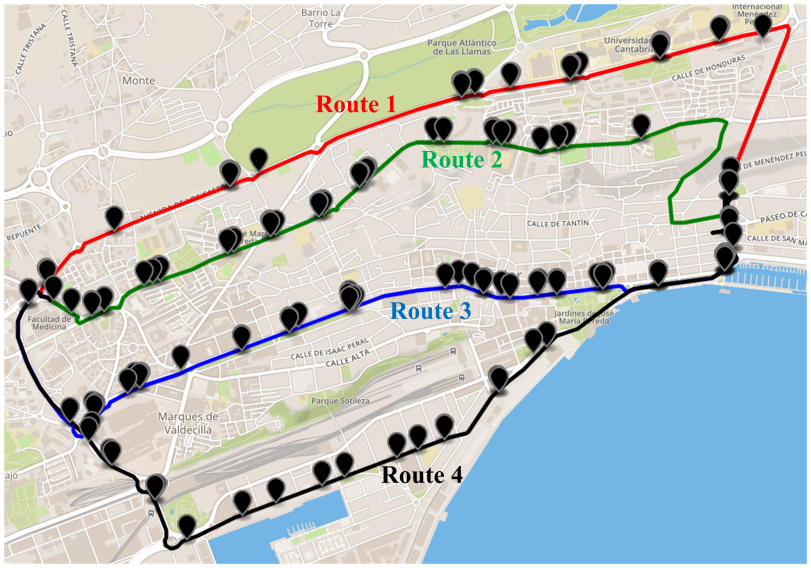


Figure 3.6 – Mapping of traffic sensors coordinates into routing coordinates.

In our performance evaluation, we use the value of deviation margin $D = 0.00006$ which is selected after doing some experiments, however it can similarly be set to different value in a different place based on how much is the deviation of sensors coordinates

from the route coordinates. We used Brouter offline routing engine [39]. Brouter is an offline and online routing engine which is built upon Open Street Maps (OSM) [40]. It calculates routes using OSM and elevation data. It is available as offline engine, Android application as well as a web service. Its unique features include freely configurable routing profiles, completely offline operation, advanced routing algorithm with elevation consideration, alternative route calculations, support of nogo and via points, and consideration of long distance cycle routes. We created a script that takes traffic sensors data (including coordinates) deployed at Santander, Spain from NGSI context broker [41] using REST API. These traffic sensors are part of an EU-KR H2020 WISE-IoT project [33]. NGSI is a protocol developed by the Open Mobile Alliance (OMA) for managing the contextual information. In WISE-IoT project, the FIWARE version of the OMA NGSI interface is used to exchange the contextual information of traffic sensors via RESTful APIs. The traffic sensors were queried through a GET REST call to `https://mu.tlmat.unican.es:8443/v2/entities?limit=1000&type=TrafficFlowObserved`.

The script takes the list of route coordinates in JSON format as input which are generated using Brouter routing engine. Finally, the script uses our mapping algorithm (presented in Algorithm 1) to map traffic sensors' coordinates into the list of route coordinates and provides the output in GeoJSON format that can be directly imported into http://geojson.io to see the mapping into a user-friendly visual interface (as can be seen in Figure 3.6) and then we analyze the mapping of sensors on the routes manually.

3.2.4.2 Performance Metrics

We manually proposed and used three performance metrics for the evaluation.

- *Correct detection* is the percentage of correctly detected traffic sensors on the routes.
- *Missed detection (non-detection)* is the percentage of missed detections (or non-detection) of traffic sensors on the routes, i.e., traffic sensors that exist on the route but are not detected by the algorithm.
- *False detection* is the percentage of false detection of traffic sensors on the route, i.e., the traffic sensors, that lie outside the route.

3.2.4.3 Performance Evaluation

Figure 3.7 presents the percentage of correct detection of traffic sensors into the four routes (which are shown in Figure 3.6) by using and without using the deviation margin D. This figure shows that by using deviation margin D, we achieve almost 100% detection of traffic sensors on all the four routes. However, on the other hand, without using deviation margin

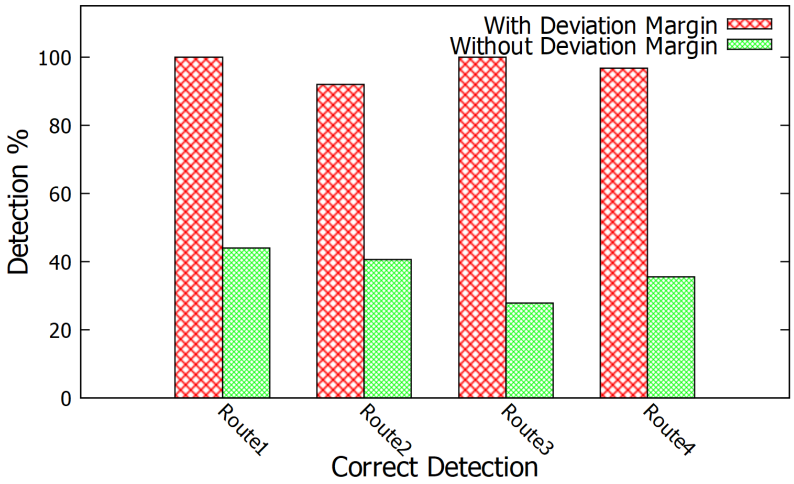


Figure 3.7 – Percentage of correct detection of sensors on the routes.

D, the detection of traffic sensors on the four routes is very low, i.e., even lower than 50%. This proves the effectiveness and significant advantage of using deviation margin D for the mapping of sensors coordinates into the routes.

Figure 3.8 presents the percentage of missed detection (non-detection) of traffic sensors into the four routes by using and without using the deviation margin D. Similar to previous figure, this figure also shows the effectiveness of using deviation margin D which causes very low missed detection (or non-detection) as compared to the case without using deviation margin D. By using deviation margin D, the percentage of missed detection is very low, i.e., lower than 10%, while without using deviation margin D, the percentage of missed detection is very high, i.e., between 50% to 70%.

Finally, Figure 3.9 presents the percentage of false detection of traffic sensors into the four routes by using and without using the deviation margin D. An interesting point to note here is that we have exactly the same results by using and without using deviation margin D. This is because some streets in Santander, Spain are very closed to each other and hence the algorithm considers some sensors to be in the same street due to closeness. It also proves that deviation margin D does not cause more false detection.

In summary, the results related to correct detection, missed detection and false detection of traffic sensors into route coordinates proves the effectiveness of our proposed algorithm which takes into account the deviation margin D. Our proposed algorithm will be beneficial for new developments in smart cities in order to map the deployed sensors with the routes.

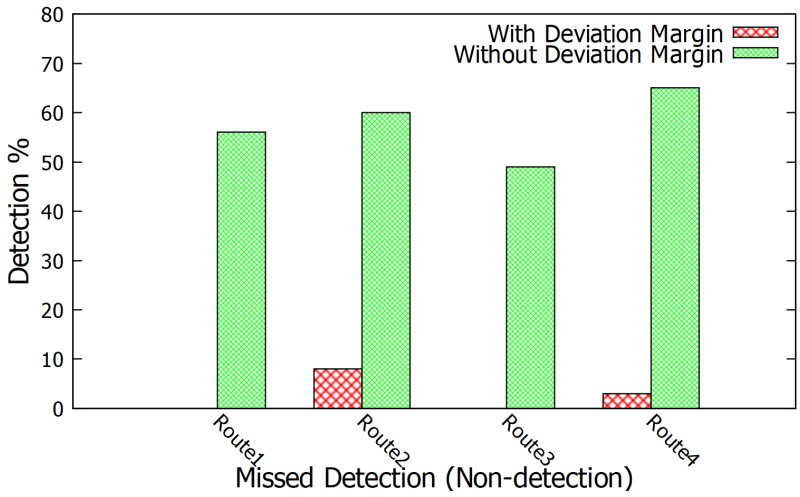


Figure 3.8 – Percentage of missed detection (non-detection) of sensors on the routes.

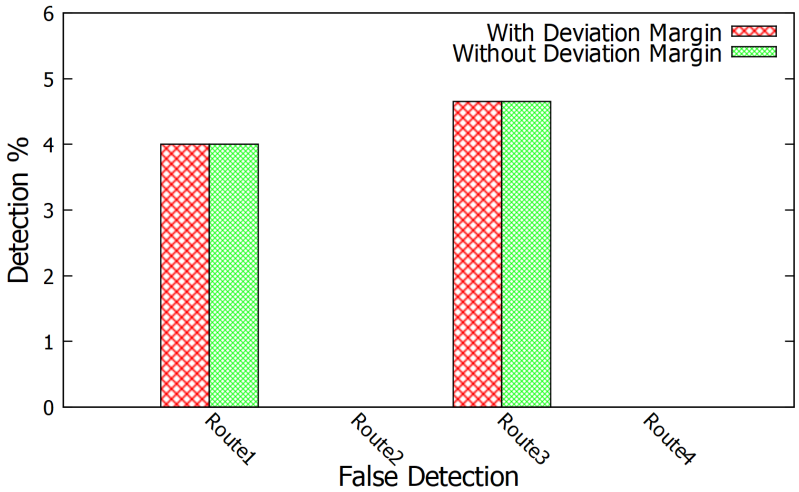


Figure 3.9 – Percentage of false detection of sensors on the routes.

3.2.5 Summary and Discussion

In this study, we proposed an approach for the mapping of sensor coordinates into route coordinates by introducing a deviation margin to provide sensors with the flexibility to deviate from the main route. We presented an algorithm along with two illustrative examples that cover all of the scenarios of mapping coordinates. We evaluated the performance of our proposed approach in terms of correct detection, missed detection and false detection. The results prove the efficiency and efficacy of our deviation margin feature. Our proposed approach will certainly be advantageous for new developments in smart cities as they map the deployed sensors along their routes.

The study in the next section presents the second part on designing an IoT Recommender for Smart Parking.

3.3 GDPR-compliant IoT Recommender for Smart Parking Supporting Semantics

During the past decade, there is a significant number of cars circulating in the cities which makes the parking and the traffic, two serious issues. Customarily, drivers try to find available parking spots on the streets by driving around, only locating a parking spot empirically due to their local knowledge and luck. This practice wastes a significant amount of both time and fuel, and sometimes it is impossible to find a free parking spot during high vehicle traffic times. One solution would be to find a parking area with high capacity of free parking spots, increasing the chances of getting a parking spot. However, this parking area could be very far from the user's destination. Another solution is to design a system that shows free parking spots to the driver and lets the driver chooses a free parking spot manually. However, this is not an optimal solution because firstly, it is an extra task for the drivers to select a parking spot by themselves. Secondly, the path leading to the selected parking spot could be very congested, causing the parking spot to be occupied when the driver arrives.

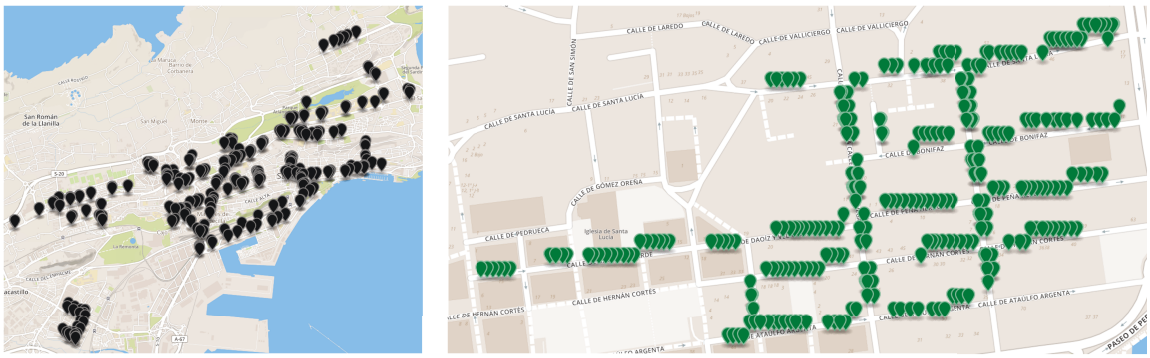
IoT technology has revolutionized almost all fields of daily life, including parking systems, by exploiting the immense developments in technology. Inspired by these new possibilities, a smart parking system has been designed to automate the recommendation of free parking spots to the drivers looking for them, thereby minimizing the time spent on finding free parking spots, as well as minimizing the cost associated with hiring humans for manual parking management [13–15]. Such a solution is based on a parking spot reservation system that utilizes various wireless networking technologies, such as ZigBee [3], Radio Frequency Identification (RFID) [2] and the Internet. The system provides information about nearby free parking spots and allows drivers to reserve the parking spots through their devices by using the ID that univocally identifies each vehicle in a parking spots reservation system [13]. However, it is not always possible to reserve parking spots in advance because of the regulations in some cities (e.g, Santander, Spain). Therefore, there is still a need for a system that can recommend the best possible parking spots based on certain metrics.

In additional, in the recommendation of parking spots, it is very important to consider the traffic on the route to each available parking spot and to recommend the least congested route leading to the recommended parking spot. To better understand the importance of traffic congestion, let us consider a rush hour scenario when the traffic in the city center is at its peak. In this scenario, the streets will be very congested, and many people will be looking for parking spots. When a driver finds an available parking spot and heads towards it, there is a high likelihood that when the driver reaches the parking spot, it will already be taken because multiple drivers are looking for similar parking spots, and

the traffic congestion caused a delay in reaching the parking spot. We solve this problem by selecting a parking spot that is the nearest to the user and by providing the expected availability (occupancy statistics) of parking areas to the users.

Many of the IoT applications available today have been developed in a vertical manner by focusing on a specific scenario or use case without considering data exchange and reuse with other IoT applications. This very specific focus results in poor service because of the lack of integration of different data and hence the interoperability in the IoT data and systems. However, if IoT applications could collaborate by exchanging and reusing each other's data, opportunities for new value-added and more efficient services could be generated. The semantic web is a promising technology with which to achieve the needed interoperability [42]. Hence, newer IoT applications, including smart parking system should be semantics-enabled adopting semantic data modeling and semantic web technologies for supporting the interoperability in IoT.

With the enforcement of the EU General Data Protection Regulation (GDPR), protecting the privacy of EU citizens throughout the data collection, data storage and data processing of a user's personal data is now a basic requirement [43, 44]. Parking systems gather a lot of contextual data and it is quite possible that the users' personal data can be collected indirectly. GDPR affects also smart parking applications and hence, the smart parking systems should therefore be designed in a way that protects user's privacy and thus be GDPR-compliant.



(a) (b)

Figure 3.10 – Deployment of traffic and parking sensors in Santander, Spain (a) traffic sensor deployment, (b) parking sensor deployment.

A smart city infrastructure was deployed in Santander, Spain in 2010−2013 as part of an EU project, SmartSantander [6, 45–47]. Quite a number of sensors were deployed, including traffic, parking, bus stop, bus line, irrigation, environmental and mobile sensors [48]. Figure 3.10 presents the deployment of traffic and parking sensors in the city of Santander.

In this study, we propose and develop the IoT Recommender (IoTRec), a smart parking system that is GDPR-compliant and considers the road traffic conditions by following the requirements listed in Section 2.2.5.1. We utilized the semantic IoT data of the deployed parking and traffic sensors as presented in Figure 3.10 for the recommendation of available parking spots and the best routes based on some metrics. The envisaged metrics for parking spots are the nearest, or nearest trusted parking spot, while the metrics for a route are the least congested or the shortest route. A trusted parking spot is defined as a parking spot that is trusted by the user based on his past experience, and that is also trusted based on the quality assessment of the parking sensor. For the nearest trusted parking spot, the IoTRec collaborates with a Trust Monitoring component [49], a component of WISE-IoT project [33], that calculates the trust scores of parking spots by analyzing the user's experience through a feedback mechanism [50] (another component of WISE-IoT project [33]) and sensor quality assessment. Such trust scores are utilized by the IoTRec in the calculation of the nearest trusted parking spot. The Trust Monitoring component is not the main scope of this study; the readers are referred to [49] for details. In this study, we first present the semantic data modeling of parking and traffic sensors data utilized by the IoTRec in the recommendations of parking spots and routes and we then present an overview of our proposed IoTRec. To allow the drivers to analyze the expected availability (occupancy statistics) of parking areas (parking spots organized into groups) in a user-friendly interface and to select a parking area, we develop and present the statistics of Santander's parking areas. Our proposed IoTRec is integrated into the real world in Santander in a prototype of the H2020 EU-KR WISE-IoT project [33, 51] called the Rich Parking application, making use of REST APIs. REST APIs make the integration with the smart parking application easy and simple, and also ease the interoperability and reusability by allowing other IoT applications to integrate these APIs and offer new efficient services. It is important to note that although the IoTRec is developed for Santander, it can be applied to other cities that have similar infrastructure.

The main contributions of this study are summarized as follows:

- Development of a parking spot (nearest or nearest trusted) and route (least congested or shortest) recommendation system by exploiting the semantic IoT data of traffic and parking sensors;

- The real-time provision of expected availability (occupancy statistics) of parking areas based on historical IoT data; and

- The development of a GDPR-compliant implementation that can work on a privacy-aware environment.

3.3.1 Related Work

Much work has been done on smart parking systems since the emergence of smart cities. However, our proposed system is significantly different in terms of recommendations of nearest parking spots, as well as routes by considering the traffic congestion on the streets and the trust scores of parking spots, expected availability (occupancy statistics) of parking areas, a prototype and evaluation. We present the state-of-the-art and explore the need for the features we offer in IoTRec.

Pham et al. [13] developed a cloud-based smart parking system using the Global Positioning System (GPS) coordinate data of vehicles, the number of free parking spots in parking areas, and the distance between parking areas to calculate the costs of a parking request made by a driver. They also developed a prototype using an open-source platform based on Arduino, RFIDs [2] and smartphones. Mainetti et al. [14] designed a smart parking system by integrating RFID, Ultra-High Frequency (UHF) and Wireless Sensor Network (WSN) technologies. This system comprises software features to collect parking spot occupancy and was developed into an application to navigate drivers to the nearest free parking spot. The application also enables users to pay their parking fee through an Near-Field Communication (NFC)-based e-wallet system. It uses Java REST APIs and Google cloud messaging, installed on a central server for managing alerts (such as the expiration of purchased time and the abuse of the reserved space) which promptly alerts the traffic police. This system was demonstrated with a proof-of-concept prototype. However, the authors mainly implemented the prototype and did not evaluate the performance of the parking system.

Hsu et al. [15] proposed SmartValet, a parking guidance system in which drivers can make parking spot reservation by smartphone thirty minutes in advance. SmartValet was developed for outdoor as well as indoor parking. It reserves a parking spot using a vehicle ID. The location of the reserved parking spot is passed to the driver on the map using Dedicated Short-Range Communication (DSRC) technology at the entrance to the parking area. SmartValet implements a navigation system, called the inertial navigation system, to guide the vehicle to the reserved parking spot. The status of the parking spot is updated periodically, ensuring system's accuracy. The authors used the accuracy of the inertial navigation system as a parameter for evaluating the system performance, and GPS accuracy as a parameter for evaluating the system implementation. Similarly, Barone et al. [52] designed an architecture called Intelligent Parking Assistant (IPA) for parking management in Smart Cities. IPA provides information about parking spot availability to drivers and allows them to reserve the most suitable parking spot before their arrival for their destination by using RFIDs and magnetic loops. When a vehicle parks or leaves the parking spot, the magnetic loop and RFID reader identifies such an action and informs

the unit controller. The unit controller subsequently updates the status of the parking spot. However, it is not always possible to reserve parking spots in advance because of the regulations in some cities. Therefore, there is a need for a system that can recommend the best possible parking spot instead of reserving parking spots.

Shiyao et al. [53] also proposed and implemented a smart parking system. Their proposed system is based on ZigBee technology [3] which forwards the information at the server through a gateway, and the server subsequently updates the database. For people looking for free parking spots, the application layer of this system obtains the parking spot availability information through the Internet, gathers all the scattered parking spots' information using web services and passes the information to drivers. However, it is a simple application that does not consider complex problems, such as traffic congestion, navigation and expected availability of parking spots. Furthermore, Shiyao et al. did not evaluate the performance of their system.

Lambrinos et al. [54] designed a parking management system for disabled people called DisAssist. This system is built upon the IoT and smart cities' capabilities by integrating smartphones, sensors and mobile/wireless communications. DisAssist offers real-time availability information about disabled parking spots in the area of interest to disabled drivers (or clients) and allows them to reserve parking spots. However, similar to other existing work, DisAssist considers the reservation of parking spots, which is not always possible.

The works mentioned above either consider the reservation of parking spots or their proposed architecture has been implemented without real-time planned routes, or are designed specifically for disabled persons. To the best of our knowledge, none of the systems recommends the nearest parking spots and routes by considering the traffic congestion on the streets and the trust scores of parking spots, offers the expected availability (occupancy statistics) of parking areas and evaluates the system. Additionally, none of the systems uses semantic data models for their parking systems which is nowadays a very important requirement to an IoT system for the interoperability with other IoT applications and systems.

3.3.2 Semantic Data Modeling

The purpose of semantic data modeling in the IoT is to facilitate data interoperability, data sharing and data reuse across cross-domain IoT applications. The IoTRec is designed for the smart parking system of a WISE-IoT project [33] that is mainly focused on the: i) interoperability between two IoT platforms: FIWARE [55] and oneM2M [56]; ii) interoperability between two continents: Europe and Asia (South Korea); and iii) interoperability between cross-domain IoT applications. Therefore, the parking and traffic sensors' data are semantically modeled for interoperability and easy integration with other IoT applications

and platforms.

One of the best practices of the semantic web is to reuse existing ontologies and data models instead of creating new ones from the scratch. Therefore, we have reused the data models provided by FIWARE. FIWARE [55] is an open-source IoT platform that offers various data models [57] for the IoT. For our IoTRec, we use FIWARE Data Models that provide a semantic representation of our required entities (e.g., parking spot, parking area and traffic information).

Table 3.1 presents a consolidated data model of the parking sensors (parking spot and parking area) and traffic sensors, and Figure 3.11 illustrates the ontology of the parking spot, OnStreetParking (parking area) and TrafficFlowObserved (traffic information) entities. The rest of this section provides the details about their semantic data modeling.

3.3.2.1 Semantic Data Modeling of Parking Sensors (Parking Spots and Parking Areas)

There are hundreds of parking sensors deployed in the city of Santander to identify the status of parking spots (i.e., free or occupied). The parking sensor data is structured using an NGSI [58] model in JSON format. A parking sensor represents a parking spot, and parking spots are grouped into various parking areas to provide additional valuable information. The IoTRec offers the expected availability (occupancy statistics) of parking areas using the parking sensor data.

The FIWARE Data Model offers a list of attributes for defining the characteristics of a parking spot entity [59]. We have selected some parking sensor attributes that are mainly required for our case. Similarly, for defining the characteristics of a parking area entity, FIWARE offers a data model, called *OnStreetParking* [59] that provides the attributes required to describe parking areas. Both of these groups of attributes (as entities and their properties) are presented in Table 3.1 and in the ontology in Figure 3.11.

3.3.2.2 Semantic Data Modeling of Traffic Sensors

Traffic sensors provide traffic information, such as occupancy, intensity and load. The IoTRec utilizes traffic information to calculate the least congested route leading to the recommended parking spot. FIWARE provides a data model of *TrafficFlowObserved* [60] entity for them that offers the required attributes to describe the traffic flow information in a city. We have selected some of the attributes from the *TrafficFlowObserved* data model to define the characteristics of our traffic information entity, and add a new attribute *roadLoad* that is not offered by the *TrafficFlowObserved* FIWARE data model. *roadLoad* estimates the level of traffic congestion calculated by intensity and occupancy parameters. These

Table 3.1 – Properties of parking spot, parking area (OnStreetParking) and traffic information (TrafficFlowObserved) entities.

Entity	Property	Type	Description
Parking spot, parking area & traffic information	`id`	String	Unique identifier of a parking spot, parking area and traffic flow
	`type`	String	Type of entity. Must be either a *ParkingSpot, OnStreetParking* or *TrafficFlowObserved*
	`dateModified`	DateTime (ISO8601)	The last modified time of the entity in ISO8601 format (e.g., 1900-12-31T23:59:59.000Z))
	`location`	geo:json	The location coordinates of the entity in geo:json format
Parking spot & parking area	`name`	String	The name of the entity for identification and distinguishing
	`category`	String	The category of the entity (e.g., *onStreet* or *offStreet* for parking spot. *free, forElectricCharging, feeCharged* or *forDisabled* for parking area)
Parking spot	`status`	String	The occupancy status of the parking spot, e.g., *free* or *occupied*
	`refParkingSite`	String	A reference to *OnStreetParking* or *OffStreetParking* based on the value of the *category* property
Parking area	`chargeType`	List<Text>	The type of charges for the parking site, such as free, monthlyPayment, annualPayment and flat rate
	`requiredPermit`	List<Text>	The permit required to park in that area, such as residentPermit, governmentPermit, emergencyVehiclePermit and noPermitNeeded
	`permitActiveHours`	List<Text>	Hours/days during which the permit is required
	`allowedVehicleType`	String	The type of vehicle that is allowed, such as a car, bicycle, motorcycle, bus, small truck, minivan
	`areBordersMarked`	Boolean	Indicates whether parking spots are separated with borders (painted lines) or not
	`totalSpotNumber`	Integer	Number of spots in the parking area
	`occupancyDetectionType`	List<String>	Technique of identifying the occupancy of a parking spot, such as modelBased, singleSpaceDetection, balancing and none
Traffic information	`dateObserved`	DateTime (ISO8601)	The observed date and time of the traffic sensor in ISO8601 format (e.g., 1900-12-31T23:59:59.000Z))
	`intensity`	Integer	The number of vehicles detected during the observation period (e.g., twenty vehicles in one minute)
	`occupancy`	Integer	The fraction of observation time in which vehicles occupy the road
	`roadLoad`	Integer	Estimation of the traffic congestion calculated by the intensity and occupancy

attributes (as entities and their properties) are presented in Table 3.1 and in the ontology presented in Figure 3.11.

3.3.3 Overview of the IoT Recommender (IoTRec)

This section presents an overview of the IoTRec for a smart parking system. The IoTRec exploits the semantic IoT data of parking and traffic sensors (see Section 3.3.2) to offer recommendations of available parking spots and optimal routes leading to the recommended parking spots based on different metrics. The metrics include the nearest or nearest trusted parking spot and the least crowded or shortest route. By default (i.e., when no preference

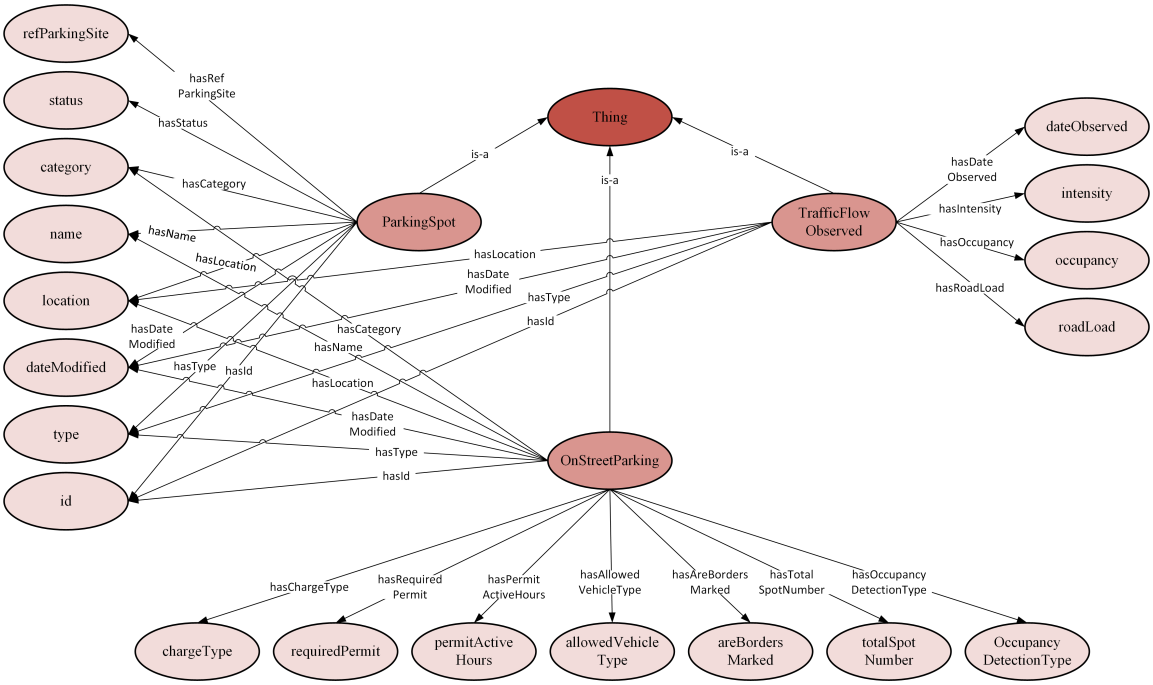


Figure 3.11 – Ontology of the parking spot, OnStreetParking (parking area) and TrafficFlowObserved (traffic information) entities.

is provided), the IoTRec selects the nearest trusted parking spot and the least crowded route. From a recommendation perspective, the IoTRec is different from non-IoT based recommendation systems in that IoTRec mainly considers the actual IoT data from parking and traffic sensors, rather than the data shared or acquired by users' terminals. Google Maps[1], for example, provides a routing path from one point to another by showing the real-time Google Traffic and recommends a least congested route. However, Google Traffic is based on crowdsourced GPS-based locations collected from a large number of users through their smartphones [61]. Google analyzes Google Traffic by calculating the speed of users along the road to calculate the congestion on the streets. The IoTRec considers and analyses traffic sensor data to estimate the level of congestion on the streets to recommend the least crowded route. Hence, if there is a lot of traffic on the streets but very few (or none) of the users has GPS location enabled on their smartphones, Google Traffic will not be able to estimate the congestion, while this is not the case for the IoTRec because it considers IoT data from real traffic sensors. Secondly, the IoTRec is integrated with Trust Monitoring [49], a component of the WISE-IoT project. The Trust Monitoring component calculates trust scores by analyzing users' experience through a feedback mechanism [50]

[1]https://www.google.com/maps

and sensor quality assessment, forwarding the trust scores to the IoTRec which are then utilized by the IoTRec in the calculation of its recommendations. To the best of our knowledge, this feature does not exist elsewhere in the literature. Thirdly, the IoTRec provides GDPR-compliant implementation that works in a privacy-aware environment. Fourthly, the IoTRec offers the expected availability (occupancy statistics) of parking areas to users and enables the users to analyze the weekly, monthly and yearly statistics of their preferred parking areas in a user-friendly interface. Hence, in addition to the automatic recommendation of parking spots, the IoTRec also allows users to select a parking area themselves by analyzing the occupancy statistics.

To better understand how this system functions, we summarize the main mechanisms in the operation of the IoTRec below. The IoTRec:

- exploits the semantic IoT data of parking sensors and the user's current location to identify the nearest parking spot to the user;
- uses trust scores from the Trust Monitoring component in the recommendation of the parking spot to determine the nearest trusted parking spot;
- utilizes the semantic IoT data of traffic sensors for analyzing the road congestion;
- provides GDPR-compliant implementation for privacy-aware environments;
- exploits the historical semantic IoT data of parking sensors to generate the real-time expected availability (occupancy statistics) of parking areas and shows the statistics in a user-friendly manner to allow a user to analyze parking areas; and
- uses the extracted congestion data and selected parking spot/area to recommend the least congested route to the user.

Figure 3.12 presents an overview of the interfaces of the IoTRec using UML component diagram. The IoTRec first obtains a request (containing the user's current location for GDPR-compliant implementation, or the user's current location and user ID for normal implementation) from the parking application to find a parking spot and a route. The IoTRec then accesses the semantic IoT data of parking sensors from the FIWARE Orion Context Broker. The occupancy data of parking spots is identified from the underlying parking sensors and is updated to the FIWARE Context Broker every two minutes. For each request, the IoTRec fetches the latest data of parking spots from FIWARE Orion Context Broker and therefore, the status of parking spots (i.e., *occupied* or *free*) is most likely up-to-date. Orion Context Broker is an NGSIv2 server implementation which is mainly used for the management of context information and its availability. It allows users to create context elements and to then use updates and queries to manage them.

The readers interested in Orion Context Broker are referred to [62] for complete details. Subsequently, the IoTRec interacts with the Trust Monitoring component [49] to obtain the trust scores which are used to find the nearest trusted parking spot. Once the IoTRec finds the most suitable parking spot (i.e., the nearest trusted parking spot), it interacts with a BRouter routing engine [39] to obtain various routes from the user's current location to the selected parking spot. BRouter is a routing engine that offers both online and offline versions and is built on the top of Open Street Maps (OSMs) [40]. It calculates the routes using elevation data and OSMs. After obtaining the routes, the IoTRec accesses the semantic IoT data from the appropriate traffic sensors and maps the traffic sensors into the routes obtained from the BRouter routing engine using the algorithm proposed in the first study of this chapter [63] on the mapping of sensor and route coordinates. The IoTRec then selects the least congested route and recommends the parking spot and the route to the smart parking application using REST API. The details of the functionalities of recommendations of parking spots and routes are provided in Section 3.3.5.

Additionally, the IoTRec also provides the expected availability (occupancy statistics) of parking areas using historical IoT data to allow a user to manually select a parking area. The complete details about the expected availability (occupancy statistics) of parking areas are provided in Section 3.3.6.

3.3.4 Operation

Figure 3.13 presents the operation of the IoTRec through sequence diagram. Initially, the parking application initiates a request (containing user's current location in case of non-GDPR compliant implementation, or user's current location and trust scores of all parking spots in case of GDPR-compliant implementation) to the IoTRec through REST API, asking for recommendation. Subsequently, the IoTRec obtains the list of free parking spots from FIWARE Orion Context Broker. Subsequently, in order to find the nearest parking spots, the IoTRec interacts with BRouter routing engine [39] and obtains the shortest paths from user current location to each parking spot. Then it finds and selects the nearest parking spot and obtains its trust score from Trust Monitoring component in case of non-GDPR-compliant implementation or obtains its trust score from its request body (in which it received trust scores from the parking application) for the selected parking spot by following the process presented in Sections 3.3.5.1 and 3.3.5.2, respectively, and finds the nearest trusted parking spot. After deciding the parking spot, the IoTRec fetches the semantic IoT data of traffic sensors from FIWARE Orion Context Broker and obtains all the available routes from user's current location to the selected parking spot using BRouter routing engine. It then maps the traffic sensor coordinates into route coordinates provided by BRouter routing engine using our proposed algorithm in [63] and identifies the least

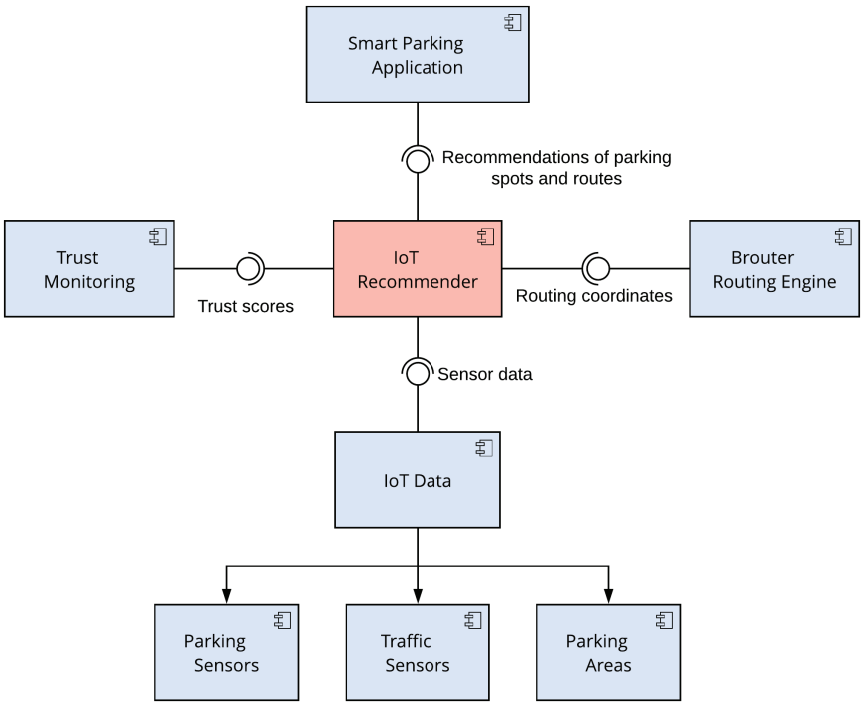


Figure 3.12 – UML component diagram of the IoTRec interfaces.

congested route (see Section 3.3.5.3) leading to the selected parking spot. Finally, it sends the recommendation of nearest trusted parking spot and the least crowded route back to the parking application.

3.3.5 Recommendation of Parking Spots and Routes

This section presents the mechanisms for parking spot and route recommendation. We provide two parking spot recommendation systems, a normal implementation for scenarios that do not operate in a privacy-aware environment and one for GDPR-compliant implementation in a privacy-aware environments.

3.3.5.1 Normal Implementation of Nearest and Nearest Trusted Parking Spot Recommendation

In a normal implementation that does not operate in a privacy-aware environment, when asking for a recommendation, the application provides the user's current location and user ID to the IoTRec. Next, the IoTRec obtains the list of available parking spots and calculates the distance from the user's current location to the available parking spots using

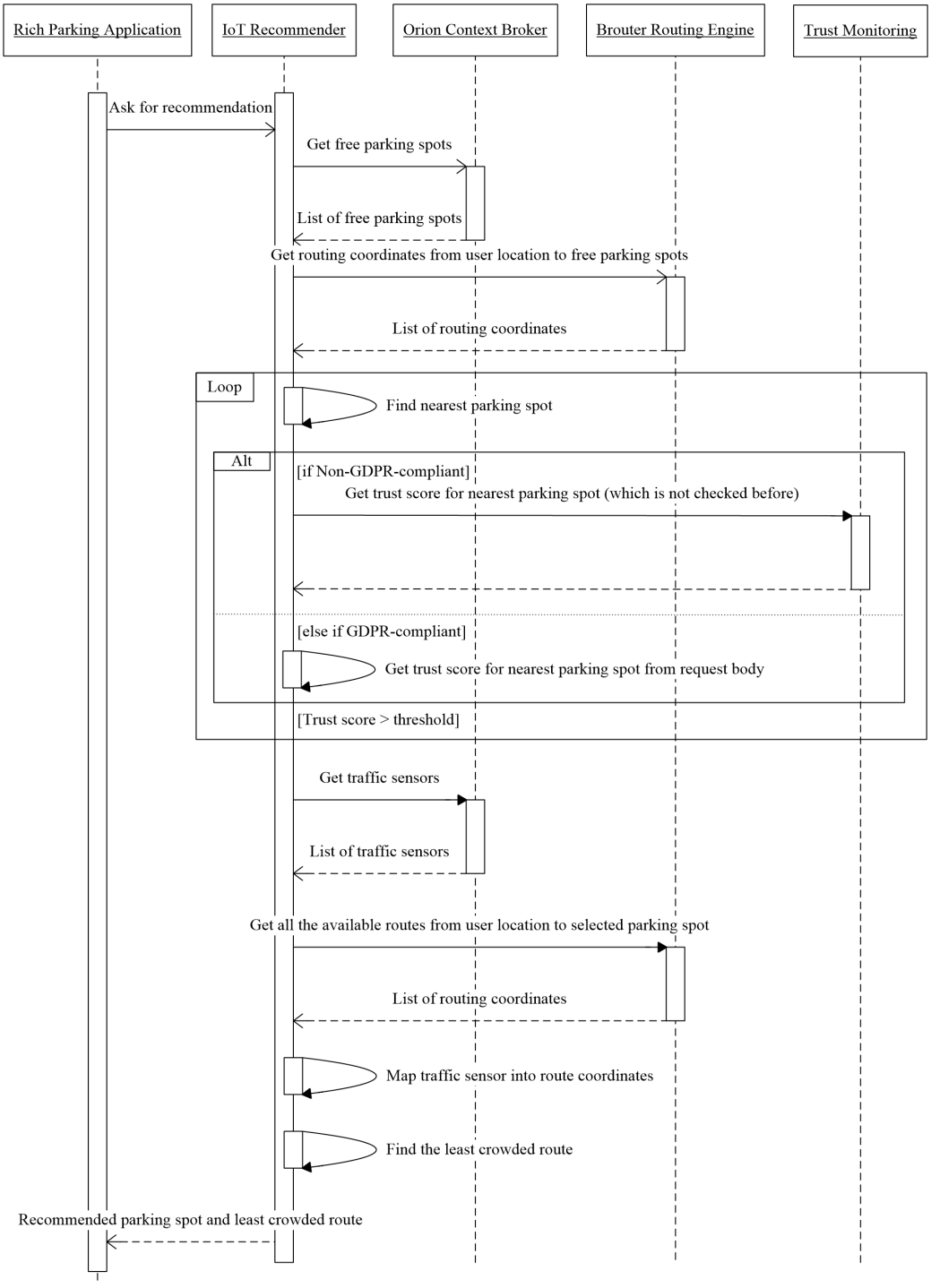


Figure 3.13 – Operation of the IoTRec.

the BRouter routing engine [39]. It then sorts the available parking spots based on their distance from the user and selects the nearest parking spot for the nearest parking spot recommendation. For the nearest trusted parking spot recommendation, the IoTRec obtains the trust score of the selected nearest parking spot from the Trust Monitoring component associated to the user's id. This trust score is dependent upon the user and on sensor quality. For example, the trust score for the same parking spot for two different users may vary. Similarly, the trust scores for two different parking spots for the same user will also vary. If the obtained trust score is below the defined threshold (the threshold value lies between 0 and 1, and was here set to 0.5 based on experimentations and to find a balance), it indicates that the selected parking spot is not appropriate for the user due to some specific reasons. For example, the user had a bad experience with the selected parking spot in the past, or the selected parking spot has some technical problems (e.g., the sensor is not working properly). After finding an available parking spot that does not have a trust score above the threshold value, the IoTRec selects the second nearest parking spot and obtains its trust score from the Trust Monitoring component. If the trust score is again below the threshold value, the IoTRec will keep checking the parking spots one by one until it finds a parking spot with a trust score that meets the threshold value. When the trust score for the selected parking spot meets the threshold value, the IoTRec selects the parking spot as the recommended parking spot. Subsequently, it finds a route (the least congested or the shortest) from the user's current location to the recommended parking spot (see Section 3.3.5.3) and sends the recommended parking spot and route back to the smart parking application.

3.3.5.2 GDPR-compliant Implementation of Nearest Trusted Parking Spot Recommendation

We also provide GDPR-compliant implementation for the recommendation of parking spot and route for scenarios that require users' privacy protection and that are operated in a privacy-aware environments. GDPR is an EU regulation on data protection and privacy of all the EU citizens [64]. Its goal is to help align the existing data protection protocols while increasing the data protection levels of the EU citizens. The main requirements to be GDPR-compliant include: obtaining explicit consent from users for data collection and freely withdrawn anytime the user wants, notification of timely breach, right to access data, right to be forgotten, data portability, privacy by design, and potential data protection officers [65]. In our GDPR-compliant implementation of IoT recommender for smart parking, we ask explicit consent of the users to store the data, store data in the user terminals instead of our own servers, and we design our IoT recommender software ensuring privacy by design, as discussed next in this section.

In a normal implementation of IoT recommender, the application provides the user's current location and user ID to the IoTRec and the IoTRec then passes the user ID to the Trust Monitoring component to obtain the trust scores of the parking spots based on the user's past experience and on the sensor quality. However, since the IoTRec receives user IDs from the application, it breaches the user's privacy because, through the `user ID`, the IoTRec could acquire much information about users, including tracking their movements, and perhaps deducing their routines. This knowledge is a breach of user privacy if the user is not willing to share his information. To cope with this problem, we offer privacy-protected and GDPR-compliant implementation in which the IoTRec does not receive `user IDs` from the application and does not directly interact with the Trust Monitoring component. Instead, the application first obtains the trust scores of all the parking spots for a specific user from the Trust Monitoring component, then passes the trust scores (comprised of `trusteeId, score` and `timestamp`) to the IoTRec. Subsequently, the IoTRec utilizes these trust scores in a similar way as discussed above to select the nearest trusted parking spot. In this manner, the IoTRec does not have any direct knowledge of the users.

The selection of the normal or the GDPR-compliant implementation depends upon the scenario. For instance, if the application/service needs to operate in privacy-aware environment, it should use the GDPR-compliant implementation. Otherwise the application/service that do not have privacy issues can choose the normal implementation, which reduces the overhead on the application. For example, since the IoTRec is reusable by other applications/services through REST APIs, applications that do not have privacy issues would generally not be willing to undertake the complications of first obtaining the trust scores for the specific user from the Trust Monitoring component at each request and then passing the user's current location and trust scores to the IoTRec. Such applications/services would most likely prefer making the IoTRec responsible for interacting with the Trust Monitoring component and thus should utilize the normal implementation.

3.3.5.3 Least Congested and Shortest Route Recommendation

After identifying the recommended parking spot, the IoTRec selects the route (the least congested or the shortest route based on the user's preference). The least congested route is identified by exploiting the semantic IoT data of traffic sensors that provide the measurements of road load, occupancy and traffic intensity. To identify the route possibilities, the IoTRec uses a third-party BRouter routing engine [39] to obtain the coordinates of various available routes from the user's current location to the recommended parking spot. Next, the IoTRec applies our previously proposed algorithm [63] to map the traffic sensor coordinates into the route coordinates. After mapping the coordinates, in the case of the shortest route, it selects the shortest available route from user's current location to the

Figure 3.14 – Aggregation of parking spots into parking areas.

recommended parking spot. For the case of the least congested route, it utilizes the road load measurements provided by traffic sensors to find the least congested route. Road load measurements provide the estimated level of congestion on the streets by interpreting the traffic intensity (i.e., the number of vehicles per hour). Since a route is comprised of multiple traffic sensors and hence multiple measurements of the road load, the IoTRec calculates the average of the road load measurements provided by the traffic sensors on each route. Subsequently, it selects the route with the the least average road load (i.e., the least bottlenecked road load) as the least congested route. In the future, we plan to study the bottleneck road loads of routes and develop a route selection mechanism by applying machine learning techniques.

3.3.6 Expected Availability (Occupancy Statistics) of Parking Areas

The IoTRec also offers the real-time expected availability of parking areas by calculating the occupancy statistics to allow users to select a parking area themselves by analyzing the statistics presented in a user-friendly manner. In this section, we explain the mechanisms and the calculation of parking area occupancy statistics. First, we provide an overview of parking area occupancy statistics, then we discuss their calculation and finally present the algorithms utilized to store the information used to calculate the statistics.

Table 3.2 – Example of Status-wise Storage of Parking Sensor Data.

ID	Status	Start Time	End Time	Duration (sec)
3601	*occupied*	2018-07-26 08:00:00	2018-07-26 09:30:00	5400
3602	*occupied*	2018-07-26 08:00:00	2018-07-26 10:00:00	7200
3601	*free*	2018-07-26 09:30:00	2018-07-26 10:00:00	1800
3602	*free*	2018-07-26 10:00:00	2018-07-26 10:30:00	1800
3601	*occupied*	2018-07-26 10:00:00	2018-07-26 10:15:00	900
3601	*free*	2018-07-26 10:15:00	2018-07-26 10:30:00	900

3.3.6.1 Overview

To better organize the expected availability (occupancy statistics) of parking areas, parking spots are aggregated into parking areas as presented in Figure 3.14. Each line in a different color represents a parking area that comprises multiple nearby parking spots. For parking area occupancy statistics, storing the occupancy data of parking sensors is the first step. The parking sensors' data is updated at a FIWARE Orion Context Broker [59] every $T_{upd} = 120$ seconds. For expected availability (occupancy statistics) of parking areas, the parking sensors' data is collected for the duration of nine months, i.e., from October 2017 to June 2018. For the storage of parking sensors' occupancy data to be used in calculating parking areas occupancy statistics, instead of storing a new record of parking sensor data every T_{upd} duration, we store the status-wise data. For example, if a record R_i for a parking sensor PS_i data having status $PS_{i,status} = free$ is stored at time instant T_n, then at the next time instant T_{n+1}, if the status remains the same (i.e., $PS_{i,status} = free$ at T_{n+1}), we update the duration of the record R_i rather than adding a new record R_{i+1}. Otherwise, if the status of parking spot PS_i is changed (i.e., $PS_{i,status} = occupied$ at T_{n+1}), we add a new record R_{i+1}. This approach helps to avoid one step of the processing to aggregate all the consecutive records having the same status while calculating the duration of the *free* and *occupied* status. Table 3.2 presents an example of status-wise storage of two parking sensors' data which shows the ID of the parking sensor (i.e., 3601 and 3602), the status (*free* or *occupied*), start time, end time and the duration.

3.3.6.2 Calculation of Parking Areas' Occupancy Statistics

The status-wise storage of parking sensor data is used in calculating the occupancy statistics of parking areas. At the beginning of each day, we calculate and store the statistics for the previous day in the database. We calculate the parking areas' occupancy statistics for a timing window of every hour to provide more accurate statistical information. We offer three levels of statistic granularity: weekly (past 1 week), monthly (past 4 weeks) and yearly (the last 52 weeks) through a REST API and present them in a user-friendly

Table 3.3 – Example of Storage of Parking Spot Occupancy Statistics for 52 weeks.

Parking Area	**Parking Spot ID**	**Day**	**Start Time**	**End Time**	**Week 1**	**Week 2**	**Week 3**	**...**	**Week 52**
HernanCortesCentre	3601	Monday	09:00:00	10:00:00	1500	1800	1500	...	1200
HernanCortesCentre	3601	Monday	10:00:00	11:00:00	1800	1200	1500	...	1800
HernanCortesCentre	3602	Monday	09:00:00	10:00:00	1200	2400	1800	...	1200
HernanCortesCentre	3602	Monday	10:00:00	11:00:00	1800	2700	1800	...	1500
DaoizVelardeEast	3620	Monday	09:00:00	10:00:00	2700	2100	3000	...	2700
DaoizVelardeEast	3620	Monday	10:00:00	11:00:00	3000	2700	2400	...	2700
DaoizVelardeEast	3623	Monday	09:00:00	10:00:00	2100	2700	1800	...	3000
DaoizVelardeEast	3623	Monday	10:00:00	11:00:00	2400	2100	2400	...	2400

interface to be analyzed by the end user.

For better understanding of this process, we present a scenario in which a user looking for a parking spot wants to analyze the parking area occupancy statistics by himself. The smartphone parking application shows the parking area statistics to the user. These are shown to the user in terms of occupancy percentage, i.e., parking area A was occupied 65% on Monday from 13:00 to 14:00. The user clicks on a parking area on Monday at 13:10 to see the statistics. The user will receive three types of statistics: the statistics of last Monday from 13:00 to 14:00 (i.e., weekly statistics); the statistics of the average value of the last four Mondays from 13:00 to 14:00 (i.e., monthly statistics); the statistics of the average value of last fifty-two Mondays from 13:00 to 14:00 (i.e., yearly statistics). All three levels of statistics are shown to the user in terms of occupancy percentage.

Table 3.4 – Example of Storage of Parking Area Occupancy Statistics.

Parking Area	**Day**	**Start Time**	**End Time**	**Weekly Stats**	**Monthly Stats**	**Yearly Stats**
HernanCortesCentre	Monday	09:00:00	10:00:00	1500	1650	1560
HernanCortesCentre	Monday	10:00:00	11:00:00	1650	1762	1740
DaoizVelardeEast	Monday	09:00:00	10:00:00	2850	2512	2580
DaoizVelardeEast	Monday	10:00:00	11:00:00	3600	2775	2730

The calculation of parking areas occupancy statistics is divided into two parts. In the first part, we calculate the hourly occupancy duration of each parking spot for each day for fifty-two weeks (one year) and store them in our database. In the second part, we calculate the weekly, monthly and yearly occupancy statistics of parking areas for each hour of the day by aggregating and averaging the hourly occupancy duration of parking spots belonging to parking areas, and store them in another database.

For the first part, to store the parking spot occupancy statistics of 52 weeks for each hour of a day, we present the structure of the database in Table 3.3. In this table, the

parking area is the area where the parking spot is situated. There are two parking areas in this table, i.e., HernanCortesCentre and DaoizVelardeEast. Parking spot ids are the identifiers of the parking spots, which are grouped into parking areas. There are four parking spots in this table, i.e., parking spots 3601 and 3602 belonging to parking area HernanCortesCentre, and parking spots 3620 and 3623 belonging to parking area DaoizVelardeEast. The day indicated in the Day column is the day for which the statistics are being calculated, which is Monday in our example. The start time and the end time delineate the one-hour timing window. In our example table, we considered two timing windows of one hour each: 09:00:00−10:00:00 and 10:00:00−11:00:00. Week 1, Week 2, ... Week 52 show the occupancy duration in seconds of the parking spot during the considered hourly timing window for the specific day. For example, in the first row of Table 3.3, 1500 in the column of Week1 represents the occupancy duration in seconds of parking spot 3601 which is in the HernanCortesCentre parking area from 09:00:00 to 10:00:00 on Monday for the first week of the year.

The second part generates the occupancy statistics of parking areas by aggregating and averaging the hourly occupancy statistics of parking spots within each area. Table 3.4 presents the structure of the database table that stores the final occupancy statistics of parking spots. Let us assume we are currently in week 5. Then in the first row of Table 3.4, 1500 in the Weekly Stats column shows the average occupancy duration of all the parking spots in the HernanCortesCentre parking area (e.g., 3601 and 3602 in Table 3.3) for Week 4. The value of 1650 in Monthly Stats shows the average occupancy duration of all the parking spots in parking area HernanCortesCentre for Week 4, Week 3, Week 2 and Week 1. Similarly, 1560 in the Yearly Stats column shows the average occupancy duration of all the parking spots in the HernanCortesCentre parking area for the last fifty-two weeks.

For a detailed discussion and complete understanding, we present the algorithms in the next subsection.

3.3.6.3 Algorithms

We present the algorithms for the status-wise storage of parking spots, hourly parking spot occupancy statistics and for the calculation of parking area occupancy statistics.

Status-wise Storage of Parking Sensor Data Algorithm 2 presents the mechanism for status-wise storage of parking spot data. This algorithm runs periodically every $T_{upd} = 120$ seconds and collects parking spot data from an Orion Context Broker and stores the data in the database based on their status (i.e., *free* or *occupied*). In each cycle, this algorithm starts by fetching the parking sensors' data PS_* from an Orion Context Broker. Next, it processes each parking sensor's data PS_i by first fetching the latest record R_i

Algorithm 2 Status-wise storage of parking sensor data.

Fetch parking sensor data PS_* from Orion Context Broker every T_{upd} duration;
for PS_i in PS_* **do**
    $R_i \leftarrow$ latest record corresponding to PS_i in DB;
    / *Part I: The first entry. Add a new record.* */*
    if $R_i = \emptyset$ **then**
        / *It is the first entry* */*
        Add a new record R_i for PS_i;
        $R_{i,parkingSpotId} \leftarrow PS_{i,id}$;
        $R_{i,status} \leftarrow PS_{i,status}$;
        $R_{i,startTime} \leftarrow$ `currentTime`;
        $R_{i,endTime} \leftarrow$ `currentTime`;
        $R_{i,duration} \leftarrow 0$;
        / *Part II: The status stays the same. Update the existing record.* */*
    else if $R_{i,status} = PS_{i,status}$ **then**
        Update record R_i;
        $R_{i,duration} \leftarrow$ (`currentTime`$-R_{i,startTime}$);
        $R_{i,endTime} \leftarrow$ `currentTime`;
        / *Part III: The status changes. Update the existing record and add a new record.* */*
    else if $R_{i,status} \neq PS_{i,status}$ **then**
        Update record R_i;
        $R_{i,duration} \leftarrow$ (`currentTime`$-R_{i,startTime}$);
        $R_{i,endTime} \leftarrow$ `currentTime`;
        Add a new record R_{i+1};
        $R_{i+1,parkingSpotId} \leftarrow PS_{i,id}$;
        $R_{i+1,startTime} \leftarrow$ `currentTime`;
        $R_{i+1,endTime} \leftarrow$ `currentTime`;
        $R_{i+1,status} \leftarrow PS_{i,status}$;
        $R_{i+1,duration} \leftarrow 0$;
    end if
end for

from the status-wise parking spot database corresponding to the parking spot PS_i. As presented in Part I of Algorithm 2, if there is no record in the database corresponding to PS_i (i.e., $R_i = \emptyset$), it shows that this is the first entry of the PS_i. Therefore, it adds a new record R_i for PS_i in the database by setting the parking spot Id $R_{i,parkingSpotId}$ and status $R_{i,status}$ to be same as those of the parking spot PS_i (i.e., $PS_{i,id}$ and $PS_{i,status}$). Since it is the first entry, it sets the start time $R_{i,startTime}$ and end time $R_{i,endTime}$ as the current time (`currentTime`), and sets the duration $R_{i,duration}$ as zero, a value which will be updated in the next round. Otherwise, as presented in Part II, if the current status of the parking spot did not change from its previous status and there is a matching record R_i in the database for PS_i with the same status (i.e., $R_{i,status} = PS_{i,status}$), it updates the record R_i by calculating the new duration (i.e., `currentTime`$-R_{i,startTime}$) and updating the end time $R_{i,endTime}$ to the `currentTime`. Finally, as presented in Part III, if the status

Algorithm 3 Calculation of hourly parking spot statistics.

Input: `dateToCalculate`, database of status-wise statistics of parking spots;
/ Part II: Add hourly parking spot statistics from status-wise statistics */*
$PS_* \leftarrow$ parking spots data from status-wise DB having *status = occupied*;
`dayOfWeek` $\leftarrow$ `GetDayOfWeek(dateToCalculate)`;
`weekOfYear` $\leftarrow$ `GetWeekOfYear(dateToCalculate)`;
`numHoursOfDay` $\leftarrow 24$;
for PS_i in PS_* **do**
    `parkingArea` $\leftarrow$ `Mapping`($PS_{i,id}$, parkingAreas);
    / Part I: Loop on 24 hours of the day for hourly statistics */*
    for h in `numHoursOfDay` **do**
        `startHourWindow` $= h$;
        `endHourWindow` $= h + 1$;
        `excessDuration` $\leftarrow 0$;
        `currentDuration` $\leftarrow 0$;
        `actualDuration` $\leftarrow 0$;
        / Part II: Add an hourly entry R_i of parking spot in hourly stats DB */*
        if no R_i for PS_i in hourly stats DB **then**
            $R_{i,parkingArea} \leftarrow$ `parkingArea`;
            $R_{i,parkingSpotId} \leftarrow PS_{i,id}$;
            $R_{i,day} \leftarrow$ `dayOfWeek`;
            $R_{i,startTime} \leftarrow$ `startHourWindow`;
            $R_{i,endTime} \leftarrow$ `endHourWindow`;
        end if
        / Part III: Calculate the occupancy duration */*
        if `startTimeWindow` $> PS_{i,startTime}$ **then**
            `excessDuration`$\leftarrow$ `startTimeWindow`$-PS_{i,startTime}$;
        end if
        if $PS_{i,endTime} >$ `endTimeWindow` **then**
            `excessDuration`.append($PS_{i,endTime}-$ `endTimeWindow`);
        end if
        `currentDuration` $= PS_{i,duration}-$ `excessDuration`;
        Fetch R_i corresponding to PS_i;
        `occupancyValue` $\leftarrow R_{i,weekOfYear}$;
        `actualDuration` $\leftarrow$ `occupancyValue` + `currentDuration`;
        Update $R_i | R_{i,weekOfYear} \leftarrow$ `actualDuration`;
    end for
end for

of the latest record $R_{i,status}$ is different from the current status $PS_{i,status}$, it first updates the duration $R_{i,duration}$ and end time $R_{i,endTime}$ as explained in Part II, and then it adds a new record R_{i+1} as explained in Part I.

The same process continues for all the parking spots. At the end of each cycle, the status-wise database is maintained as presented in Table 3.2.

Calculation of Hourly Parking Spot Occupancy Statistics Algorithm 3 presents the mechanism for calculating hourly parking spot occupancy statistics. This algorithm takes as input the `dateToCalculate` (date of the previous day because this algorithm starts at the beginning of each day) and the status-wise data of parking spots. The algorithm starts by fetching the status-wise occupied parking sensors' data PS_* and initializing the `dayOfWeek, weekOfYear` and `numHoursOfDay`. Next, it processes each record of parking spot PS_i in PS_*. It first extracts the `parkingArea` from the mapping of the parking spot id and parking areas (i.e., `Mapping`($PS_{i,id}$, parkingAreas)) and runs a loop h for each hour of the day in `numHoursOfDay`, as presented in Part I of Algorithm 3. To store the hourly occupancy statistics and calculate the occupancy duration in each hour, it then sets the `startHourWindow` as h, `endHourWindow` as $h+1$, and initializes `excessDuration`, `currentDuration` and `actualDuration` to zero. `excessDuration` is the duration outside of the pre-set window time. For example, let us consider the first row of Table 3.2 with `totalDuration` = 5400. If we want to calculate hourly statistics from 08:00:00−09:00:00, the `excessDuration` is 1800 seconds (i.e., 09:00:00−09:30:00) which we need to exclude in our calculation, and hence, the `currentDuration` is `totalDuration - excessDuration` (i.e., 5400−1800=3600 seconds). Finally, the `actualDuration` is the sum of the previous occupancy duration `occupancyValue` and the `currentDuration`.

Part II of Algorithm 3 adds an hourly entry R_i of parking spots in the hourly statistics database without a duration. The duration will be calculated in Part III of the algorithm. Part III first checks whether the `startTimeWindow` is higher than the parking sensor's start time $PS_{i,startTime}$, and if so, it calculates the `excessDuration` by taking the difference of the `startTimeWindow` from the parking sensor start time $PS_{i,startTime}$. Part III also checks whether the parking sensor end time $PS_{i,endTime}$ is higher than the `endTimeWindow`, which is the case in our considered example, e.g., `endTimeWindow`=9:00:00, while the parking sensor end time $PS_{i,endTime}$ =9:30:00. Hence, it updates the `excessDuration` by appending the difference of the parking sensor end time $PS_{i,endTime}$ to the `endTimeWindow` and calculates the `currentDuration` by subtracting the `excessDuration` from the total duration $PS_{i,duration}$. Finally, to calculate the actual duration for the current week of the year for the current day, it first fetches the record R_i corresponding to PS_i, obtains the `occupancyValue` (i.e., the sum of previous occupancy duration) and calculates the `actualDuration` by taking the sum of the `currentDuration` and the `occupancyValue`. Subsequently, it updates the occupancy statistics of the current week of the year of the current day $R_{i,weekOfYear}$ with the `actualDuration`. The same process follows for all the parking spots. In the end, this algorithm creates a database as shown in Table 3.3.

Algorithm 4 Calculation of parking areas occupancy statistics.

1: */* Part I: Create arrays of weekly, monthly and yearly occupancy statistics for each hour of the parking spots associated to parking areas.*/*
2: `parkingAreasJson` ← `new JSON`;
3: `currentWeek` ← `GetWeekOfYear()`;
4: RS ← Occupancy stats of parking spots from hourly stats DB;
5: **for** PS_i in RS **do**
6: `parkingAreaId` ← $PS_{i,parkingAreaId}$;
7: `day` ← $PS_{i,day}$;
8: `startTime` ← $PS_{i,startTime}$;
9: `endTime` ← $PS_{i,endTime}$;
10: */* Part I(a): Define nested JSON objects/arrays for parking area, daily, hourly, weekly, monthly and yearly stats for those not already defined*/*
11: `parkingAreasJson.parkingArea` ← `new JSON`;
12: `parkingAreasJson.parkingArea.day` ← `new JSON`;
13: `parkingAreasJson.parkingArea.day.hour` ← `new JSON`;
14: `parkingAreasJson.parkingArea.day.hour. weeklyStatArray` ← `new JSON`;
15: `parkingAreasJson.parkingArea.day.hour. monthlyStatArray` ← `new JSON`;
16: `parkingAreasJson.parkingArea.day.hour. yearlyStatArray` ← `new JSON`;
17: */* Part I(b): Calculate weekly, monthly and yearly average stats*/*
18: j ← `currentWeek`-1;
19: `weeklyStatArray.append`$(PS_{i,week_j})$;
20: `monthlyStatArray.append(AVG`$(\sum_{k=j-4}^{j} PS_{i,week_k}))$;
21: `yearlyStatArray.append(AVG`$(\sum_{k=j-52}^{j} PS_{i,week_k}))$;
22: **end for**
23: */* Part II: Calculate the accumulated weekly, monthly and yearly occupancy statistics of parking areas for each hour of the day and each day of the week and store the statistics in the database*/*
24: **for** `parkingAreaJson in parkingAreasJson` → `dayJson in parkingAreaJson` → `hourJson in dayJson` **do**
25: */* Part II(a): Calculate the accumulated weekly, monthly and yearly occupancy statistics of parking areas*/*
26: `parkingArea` ← `key(parkingAreaJson)`;
27: `day` ← `key(dayJson)`;
28: `hour` ← `key(hourJson)`;
29: `weeklyStatArray` ← $\texttt{hourJson}_{weeklyStatsArray}$;
30: `monthlyStatArray` ← $\texttt{hourJson}_{monthlyStatsArray}$;
31: `yearlyStatArray` ← $\texttt{hourJson}_{yearlyStatsArray}$;
32: $\texttt{length}_w$ ← `length(weeklyStatArray)`;
33: $\texttt{length}_m$ ← `length(monthlyStatArray)`;
34: $\texttt{length}_y$ ← `length(yearlyStatArray)`;
35: `avgWeeklyStat` ← `AVG`$(\sum_{k=1}^{\text{length}_\text{w}}$`weeklyStatArray)`;
36: `avgMonthlyStat` ← `AVG`$(\sum_{k=1}^{\text{length}_\text{m}}$`monthlyStatArray)`;
37: `avgYearlyStat` ← `AVG`$(\sum_{k=1}^{\text{length}_\text{y}}$`yearlyStatArray)`;
38: */* Part II(b): Store the statistics in the database*/*
39: Add a new record R_l in parking areas stats DB;
40: $R_{l,parkingArea}$ ← `parkingArea`;
41: $R_{l,day}$ ← `day`;
42: $R_{l,startTime}$ ← `startTime` (fetched from `hour`);
43: $R_{l,endTime}$ ← `endTime` (fetched from `hour`);
44: $R_{l,weeklyStats}$ ← `avgWeeklyStat`;
45: $R_{l,monthlyStats}$ ← `avgMonthlyStat`;
46: $R_{l,yearlyStats}$ ← `avgYearlyStat`;
47: **end for**

Calculation of Parking Areas' Occupancy Statistics Algorithm 4 presents the mechanism of calculation of parking areas' occupancy statistics. This algorithm is comprised of two main parts. In the first part, it creates arrays of weekly, monthly and yearly occupancy statistics for each hour of the parking spots associated with the parking areas. Each element within an array corresponds to the occupancy duration of all the parking spots within a parking area. For example, if parking area A has five parking spots, then in this part, the weekly, monthly and yearly arrays will have five elements each. In the second part, this algorithm calculates the average weekly, monthly, and yearly occupancy statistics of parking areas for each hour of the day and each day of the week and stores the occupancy statistics in the database.

The algorithm starts by initializing the variables `parkingAreaJson, currentWeek` and the result set RS of occupancy statistics of parking spots from the hourly statistics database. As presented in Part I(a) of Algorithm 4, the algorithm processes each parking spot PS_i in the result set RS and defines nested JSON objects/arrays (for those not already defined) of `parkingArea, day, hour, weeklyStatArray, monthlyStatArray` and `yearlyStatArray` statistics. Part I(b) calculates the average weekly, monthly and yearly statistics. For weekly statistics, since that is comprised of just one past week which does not require an average, it appends the occupancy statistics of the past week. For monthly statistics, it first takes the sum of the occupancy statistics of last four weeks, then takes their average and appends that value into a monthly statistics array. The yearly statistics are calculated similar to the monthly statistics, but for fifty-two weeks instead of four weeks. Similarly, the weekly, monthly and yearly occupancy statistics for other parking spots belonging to the same parking area are also appended into the same weekly, monthly, and yearly arrays.

Part II(a) of Algorithm 4 calculates the accumulated weekly, monthly and yearly occupancy statistics of parking areas. It iterates on each JSON object in `parkingAreasJson` (i.e., `parkingAreaJson, dayJson` and `hourJson`), and obtains the ids of `parkingAreas, days` and `hours`, respectively, from their `keys`. Next, it obtains the `weeklyStatArray, monthlyStatArray` and `yearlyStatArray` from `hourJson` and calculates their lengths. Then for weekly statistics, it first takes the sum of all the weekly statistics in the array and takes their average, which is the accumulated weekly occupancy statistics of a `parkingArea` for the particular `hour` of the specific `day`. The monthly and yearly accumulated statistics are calculated in a similar fashion. Finally, as presented in Part II(b) of the algorithm, the accumulated weekly, monthly and yearly statistics are stored in the database of parking area occupancy statistics, which is used to generate the REST API for parking area occupancy statistics. Table 3.4 presents a snapshot of the final parking area occupancy statistics database where the weekly, monthly and yearly statistics are presented for each hour of the day and each day of the month.

3.3.7 Scalability Analysis

The IoT recommender for smart parking is currently developed as a proof-of-concept prototype for the smart city of Santander. Two types of scalabilities are envisioned. Firstly, the area covering the deployment of parking and traffic sensors is varied. In this case, the IoT recommender currently serves well in Santander (as can be seen next in Section 3.3.9 of evaluation). Hence, if there is another city similar to Santander, we believe that the IoT recommender would not have any issue. However, if the IoT recommender has to operate in a much larger city (e.g., New York or Tokyo), then the IoT recommender (that is currently developed as a proof-of-concept prototype) needs to be extended to be a fully-fledge system in a way that it only loads the parking and traffic sensors of the concerned areas (e.g., areas pertaining to user's current and destination locations) instead of the parking and traffic sensor deployed all over the city.

The second case of scalability concerns the involvement of higher number of concurrent users making requests for recommendations in parallel, e.g., a million users. In this case, the IoT recommender would be needing to have multiple instances and a data center to fulfill such load of user requests. Furthermore, it is also an interesting direction to analyse the complexity and the overhead of the IoT recommender. We envisage to analyse them in a similar way as analysed by Betterstetter & Stefan [66] and Er & Seah [67] for wireless networks.

3.3.8 REST APIs

The IoTRec offers REST APIs for the recommendation of a parking spot and route coordinates (discussed in Section 3.3.5), and parking areas occupancy statistics (discussed in Section 3.3.6). The purpose of these REST APIs is to make the IoTRec reusable by other IoT applications and platforms through these REST APIs.

3.3.8.1 Normal Implementation-based REST APIs for Parking Spot and Route Recommendation

The IoTRec offers various REST APIs for the recommendation of a parking spot and a route. These APIs are the combination of the nearest parking spot, trusted nearest parking spot, shortest route and the least crowded route.

The APIs for the parking spot and route recommendation take as input the GPS coordinates of the current location of the user (`double lon & double lat`) as a query string. The APIs which provide trusted parking spot take an additional input of user ID (`String userId`) as a query string. The current location of the user is used to find the parking spot nearest to the user, as well as to select a route from the user's current location to the

selected parking spot. The `userId` (used by the APIs for the trusted parking spot) is used to obtain the trust scores of parking spots by passing it in the REST API of the Trust Monitoring component [49].

These APIs provide two-fold functions: i) the recommendation of a parking spot (trusted or nearest), and ii) the recommendation of a route (least crowded or nearest). The recommendation of parking spot could be either the nearest trusted or only the nearest parking spot which depends upon the choice of the user. The nearest parking spot is selected by finding the nearest available parking spot from the user's current location. The trusted parking spot is selected by following the procedure described in Section 3.3.5.1. After finding the parking spot (nearest or nearest trusted), the next function and the step is route recommendation. The route recommendation could be either the shortest route (regardless of consideration of traffic congestion) or the least congested route which also depends upon the choice of user. The shortest route is directly obtained from BRouter routing engine [39] from the user's current location to the selected parking spot. The least crowded route is calculated by following the procedure described in Section 3.3.5.3.

The REST APIs generate the response in JSON format containing JSON objects of `parking-lot` and `routing-path`. Listing 3.1 describes the APIs for the recommendation of a parking spot and the route. Listing 3.2 presents an example response of JSON objects of `parking-lot` and `routing-path`.

3.3.8.2 GDPR-compliant REST APIs for the Parking Spot and Route Recommendation

In GDPR-compliant REST APIs, the IoTRec does not receive `user IDs` from the parking application and does not directly interact with the Trust Monitoring component. Instead, the parking application first obtains the trust scores of parking spots for specific user from Trust Monitoring component, then passes the trust scores (comprised of `trusteeId, score and timestamp`) into the body of REST APIs of the IoTRec with the POST method. Subsequently, the IoTRec utilizes these trust scores in a similar way (as discussed before) for recommending the nearest trusted parking spot. In this manner, the IoTRec does not have any knowledge of the users.

Listing 3.3 presents the REST APIs related to the nearest trusted parking spot with the HTTP POST method and sample trust scores in the body of the method. The response is the same as presented in Listing 3.2.

3.3.8.3 Parking Areas Occupancy Statistics

The mechanism of parking areas occupancy statistics is discussed in detail in Section 3.3.6. In this section, we describe the REST API for the parking areas occupancy statistics of-

```
GET
Nearest Trusted Parking Spot with Least Crowded Route
http://serverIP:port/IoTRecommender/v1/
    nearestTrustedParkingSpotWithLeastCrowdedRoute?
    lon={longitudeCoordinates}&
    lat={latitudeCoordinates}&
    userId={userId}

Nearest Trusted Parking Spot with Shortest Route
http://serverIP:port/IoTRecommender/v1/
    nearestTrustedParkingSpotWithShortestRoute?
    lon={longitudeCoordinates}&
    lat={latitudeCoordinates}&
    userId={userId}

Nearest Parking Spot with Least Crowded Route
http://serverIP:port/IoTRecommender/v1/
    nearestParkingSpotWithLeastCrowdedRoute?
    lon={longitudeCoordinates}&
    lat={latitudeCoordinates}

Nearest Parking Spot with Shortest Route
http://serverIP:port/IoTRecommender/v1/nearestParkingSpotWithShortestRoute?
    lon={longitudeCoordinates}&
    lat={latitudeCoordinates}

Query Parameters:
    double lon: the longitude coordinates of user's current location
    double lat: the latitude coordinates of user's current location
    String userId: the id of the current user. It is used to get the trust
        scores for respective user from Trust Monitoring component.
Headers:
   Content-Type: application/json
   Accept: application/json
Response: application/json (See Listing 3.2 for an example)
```

Listing 3.1 – APIs for the recommendation of a parking spot and route

fered by the IoTRec. This REST API is mainly called by the application which shows the statistics to the user in a user-friendly manner on his smartphone application. Listing 3.4 describes the API of parking areas occupancy statistics with an example response and the statistics are updated at the beginning of each day for the last day. The example response presents weekly, monthly and yearly parking areas occupancy statistics for each day of the week. In the example response, `id` presents the ID of the parking area, `refParkingSite` presents the reference parking site, e.g., Sta. Lucia East `StaLuciaEast`. Then it provides weekly, monthly and yearly statistics for each day. For instance, let us consider `mondayParkingStatistics` that provides weekly, monthly and yearly parking areas occupancy statistics for each hour of the day. `[w1,m1,y1]` represents the weekly, monthly and

```
Example Response:
"parking-lot": {
   "id": "urn:entity:santander:parking:parkingSpot:3601",
   "type": "ParkingSpot",
   "status": {
      "type": "Text",
      "value": "free",
      "metadata": { }
   },
   "name": {
      "type": "Text",
      "value": "parkingSpot3601",
      "metadata": { }
   },
   "dateModified": {
      "type": "ISO8601",
      "value": "2018-07-26 08:00:00",
      "metadata": { }
   },
   "location": {
      "type": "geo:json",
      "value": {
         "coordinates": [-3.80076,43.4627],
         "type": "Point",
         "metadata": {}
      }
   },
   "refParkingSite": {
      "type": "Text",
      "value": "HernanCortesCentre",
       "metadata": { }
   },
   "category": {
      "metadata": {},
      "type": "StructuredValue",
      "value": ["onstreet"]
   }
},
"routing-path": {
   "type": "FeatureCollection",
   "features": [
      {
         "type": "Feature",
         "geometry": {
            "type": "LineString",
            "coordinates": [
               [-3.821461, 43.463945, 65],
               [-3.821067, 43.464059, 65.75],
               .....
            ]
         },
         "properties": {
            "track-length": "2399"
         }
      }
    ]
}
```

Listing 3.2 – An example response of `parking-lot` and `routing-path` JSON objects

```
POST
Nearest Trusted Parking Spot with Least Crowded Route
http://serverIP:port/IoTRecommender/v1/
    nearestTrustedParkingSpotWithLeastCrowdedRoute?
     lon={longitudeCoordinates}&
     lat={latitudeCoordinates}

Nearest Trusted Parking Spot with Shortest Route
http://serverIP:port/IoTRecommender/v1/
    nearestTrustedParkingSpotWithShortestRoute?
     lon={longitudeCoordinates}&
     lat={latitudeCoordinates}

Query Parameters:
     double lon: the longitude coordinates of user's current location
     double lat: the latitude coordinates of user's current location
Headers:
   Content-Type: application/json
   Accept: application/json
Body:
[
   {
      "trusteeId":"urn:entity:santander:parking:parkingSpot:3637",
      "score":0.1,
      "timestamp":1499003993933
   },
   {
      "trusteeId":"urn:entity:santander:parking:parkingSpot:3641",
      "score":3.0,
      "timestamp":1499003993938
   },
   ...
]
Response: application/json
```

Listing 3.3 – APIs of privacy protected recommendation of parking spot and route

yearly parking areas occupancy statistics, respectively from 00:00:00−01:00:00. Similarly `[w2,m2,y3],[w3,m3,y3],...,[w24,m24,y24]` represent statistics for rest of the hours of the day.

3.3.8.4 Walking Route to the Parked Vehicle

After the user parks his car at the parking spot recommended by the IoTRec, he can allow the application to save the location where he has parked the car so, when he wants to reach back to his parked car, the IoTRec offers a REST API to recommend a walking route from user's current location to his parked car. This REST API takes as input the GPS coordinates of the current location of the user `(double start_lon & double`

```
GET
http://serverIP:port/IoTRecommender/v1/getParkingAreasStatistics

Response: application/json
Example
{
    "id": "urn:entity:santander:parking:parkingAreaStatistics:StaLuciaEast
        ",
    "type": "parkingStatistics",
    "dateModified": {
      "type": "ISO8601",
      "value": "2017-07-20T11:56:00.00Z",
      "metadata": {}
    }
    "refParkingSite": {
      "type": "Text",
      "value": "urn:entity:santander:parking:onStreet:StaLuciaEast",
      "metadata": {}
    },
    "mondayParkingStatistics": {
      "type": "StructuredValue",
      "value":[[w1,m1,y1],[w2,m2,y2],[w3,m3,y3],..., [w24,m24,y24]],
      "metadata": {}
    },
    "tuesdayParkingStatistics": {
      "type": "StructuredValue",
      "value":[[w1,m1,y1],[w2,m2,y2],[w3,m3,y3],..., [w24,m24,y24]],
      "metadata": {}
    },
    "wednesdayParkingStatistics": {
      "type": "StructuredValue",
      "value":[[w1,m1,y1],[w2,m2,y2],[w3,m3,y3],..., [w24,m24,y24]],
      "metadata": {}
    },
    "thursdayParkingStatistics": {
      "type": "StructuredValue",
      "value":[[w1,m1,y1],[w2,m2,y2],[w3,m3,y3],..., [w24,m24,y24]],
      "metadata": {}
    },
    "fridayParkingStatistics": {
      "type": "StructuredValue",
      "value":[[w1,m1,y1],[w2,m2,y2],[w3,m3,y3],..., [w24,m24,y24]],
      "metadata": {}
    },
    "saturdayParkingStatistics": {
      "type": "StructuredValue",
      "value":[[w1,m1,y1],[w2,m2,y2],[w3,m3,y3],..., [w24,m24,y24]],
      "metadata": {}
    },
    "sundayParkingStatistics": {
      "type": "StructuredValue",
      "value":[[w1,m1,y1],[w2,m2,y2],[w3,m3,y3],..., [w24,m24,y24]],
      "metadata": {}
    }
}
```

Listing 3.4 – API of Parking Areas Occupancy Statistics

```
GET
http://serverIP:port/IoTRecommender/v1/walkingRouteToParkedCar?
    start_lon={startLongitudeCoordinates}&
    start_lat={startLatitudeCoordinates}&
    end_lon={endLongitudeCoordinates}&
    end_lat={endLatitudeCoordinates}

Query Parameters:
    double start_lon: the start longitude coordinates of user's current
        location
    double start_lat: the end latitude coordinates of user's current
        location
    double end_lon: the start longitude coordinates of parked car (parking
        spot)
    double end_lat: the end latitude coordinates of parked car (parking spot
        )
Headers:
   Content-Type: application/json
   Accept: application/json
Response: application/json (See "routing-path" in Listing 3.1)
```

Listing 3.5 – API of walking route to the parked vehicle

`start_lon`) and GPS coordinates of the parked car (`double end_lon & double end_lat`). Subsequently, it recommends a walking route from the user's current location to his parked car. Listing 3.5 describes the API for the recommendation of a walking route to the parked car. The response generated by this REST API is the same as `"routing-path"` JSON presented in Listing 3.2.

3.3.8.5 Parking Spots

The IoTRec offers REST APIs to obtain a list of all, free or occupied parking spots in JSON format. The REST API for the free parking spots could be used by the IoTRec itself while finding the available parking spots. Additionally, the REST APIs for all, free and occupied parking spots could be used by the application to show graphically all, free or occupied parking spots to the user. Listing 3.6 presents the REST APIs.

3.3.9 The Prototype and Evaluation

3.3.9.1 The Prototype

In order to evaluate IoTRec, an Android application, called Rich Parking [46] was developed by the smart parking use case owner [68] in WISE-IoT project and IoTRec was integrated into it. It was tested and evaluated as a prototype by the citizens of Santander, Spain. The prototype serves as the first step towards developing a fully-fledged application to improve

```
GET
All Parking Spots
http://serverIP:port/IoTRecommender/v1/allParkingLots

Free Parking Spots
http://serverIP:port/IoTRecommender/v1/freeParkingLots

Occupied Parking Spots
http://serverIP:port/IoTRecommender/v1/occupiedParkingLots

Headers:
   Content-Type: application/json
   Accept: application/json
Response: application/json
```

Listing 3.6 – APIs of parking spots (all, free and occupied)

the parking experience of users in the city. The prototype provides various functionalities to the users using the services offered by the IoTRec through REST APIs. The screenshots of these functionalities are depicted in Figure 3.15. We present the features of the prototype in this section.

Show Parking Spots The prototype allows users to see the status of parking spots. To request the status of parking spots, a user clicks the corresponding button. The parking application calls the REST APIs of the IoTRec for free and occupied parking spots and obtains the list of free and occupied parking spots. It subsequently shows the free parking spots as green markers and occupied parking spots as grey markers, as presented in Figure 3.15(a).

Recommendation of Parking Spot and Route A user can request a parking spot recommendation and that of a route from their current location by specifying their preferences (e.g., for a parking spot: nearest or nearest trusted parking spot, and for a route: least crowded or shortest) in the application. The application then calls the required REST API of the IoTRec of the parking spot and route recommendation and presents the recommended parking spot and route to the user, as presented in Figure 3.15(b).

Walking Route to the Parked Car After a user has parked his car in the recommended parking spot and saved the location of the car in the application, he can later request a walking route from his current location to his parked car. The application then calls the REST API of the IoTRec for the walking route to the parked car and shows the walking route to the parked car on the application screen, as presented in Figure 3.15(c).

Figure 3.15 – Screenshots of the prototype Rich Parking application [68]: (a) view of free and occupied parking spots; (b) recommendation of a parking spot and a route; (c) walking route to the parked car; and (d) an example of parking areas occupancy statistics.

Parking Area Occupancy Statistics The prototype shows the parking areas to the user if the user wants to analyze the statistics manually, as presented in Figure 3.15(d). When the user clicks on any parking area, the application calls the REST API of the IoTRec for parking area occupancy statistics, obtains the parking area statistics for the current

Table 3.5 – Questionnaire for the evaluation of the IoTRec functionalities through the prototype.

No.	Question	Possible Answers
1	Quality of the routes and parking information?	1-star (very bad) to 5-stars (very good)
2	Functionality of show parking spots?	1-star (little useful) to 5-stars (very useful)
3	Functionality of route calculation?	1-star (little useful) to 5-stars (very useful)
4	Functionality of walking route to the parked car?	1-star (little useful) to 5-stars (very useful)
5	Functionality of parking statistics?	1-star (little useful) to 5-stars (very useful)
6	Easy to navigate?	1-star (very difficult) to 5-stars (very easy)
7	Easy to calculate a route?	1-star (very difficult) to 5-stars (very easy)
8	Easy to analyze a route?	1-star (very difficult) to 5-stars (very easy)
9	Provided data of parking spots and statistics are reliable?	No, Probably No, I do not know, Probably Yes, Yes

day and current hour for the selected parking area, calculates the occupancy statistics in percentage, and presents the results to the user. For example, in Figure 3.15(d), when the user clicks on the area `Sta. Lucia east`, the application shows a popup to the user that indicates there is a 54% chance of finding a free parking spot in this parking area.

3.3.9.2 Evaluation Overview

To evaluate our IoTRec system, the developed prototype was shared and tested with the citizens of Santander. For expected availability (occupancy statistics) of parking areas, the parking sensors' data is collected for the duration of nine months, i.e., from October 2017 to June 2018. A higher number of Santander's citizens were approached through various meetups requesting volunteers for the evaluation of the prototype application. A meeting was then conducted with the volunteered citizens willing to participate in the evaluation and were explained how to use and evaluate the application. A total of 41 citizens of Santander committed to being engaged in the evaluation, and 30 of them installed the application from the Google Play store. To conduct the evaluation, a questionnaire related to the functionalities offered by the IoTRec was designed that participants completed at the end of evaluating the application. Out of the 30 participants, 27 evaluated the application by answering to questionnaire. Table 3.5 presents the questionnaire with its possible answers to these questions for the evaluation of the IoTRec functionalities in the prototype.

3.3.9.3 Evaluation Results

The evaluations results are based on the questionnaires completed by the 27 citizen participants. We present the evaluation results for each question in the questionnaire.

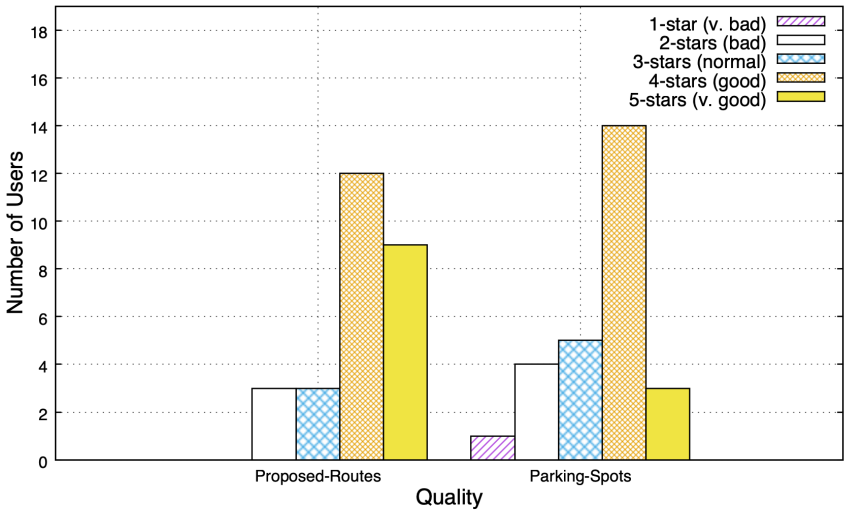


Figure 3.16 – Evaluation results of the quality of recommended routes and parking spots.

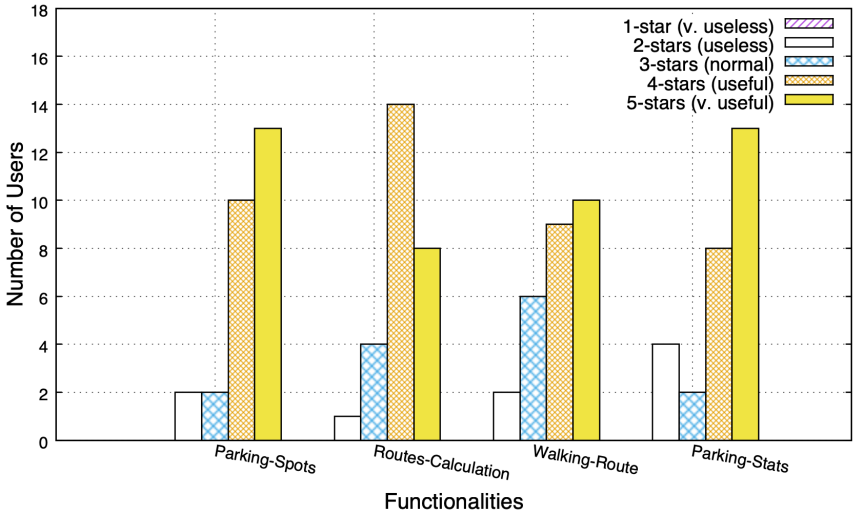


Figure 3.17 – Evaluation results of the functionalities of the recommendations for parking spot availability, route calculation, walking route to a parked car and parking area statistics.

Quality Figure 3.16 presents the evaluation results of the quality of the recommended routes and parking spots. It shows a good response as 89% and 81% of the involved citizens were satisfied (identified by their positive ratings of 3, 4 and 5-stars) with the quality of the recommended routes and parking spots, respectively. 78% and 63% of the participants found the quality of recommended routes and parking spots, respectively to be good/very good (4 and 5-stars). Overall, these evaluation results give us a good indication of the high quality of the recommended routes and parking spots and the satisfaction of the users.

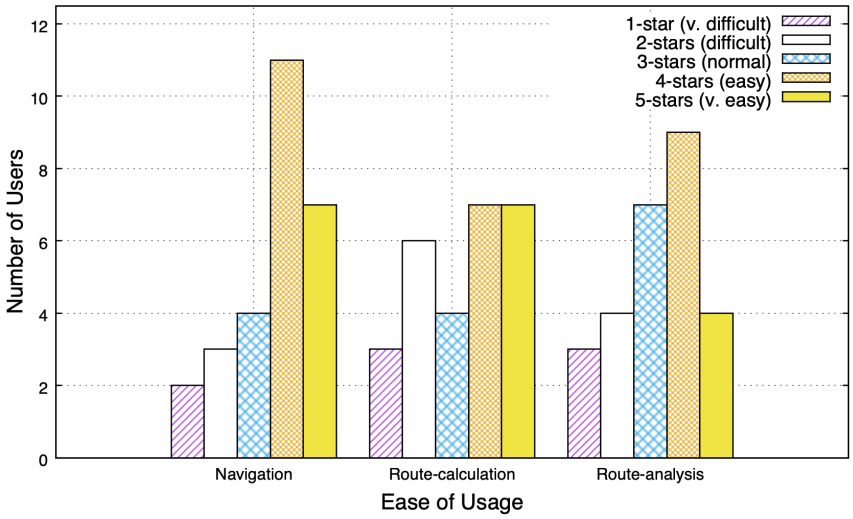


Figure 3.18 – Evaluation results of the ease of use of the navigation, route calculation and route analysis.

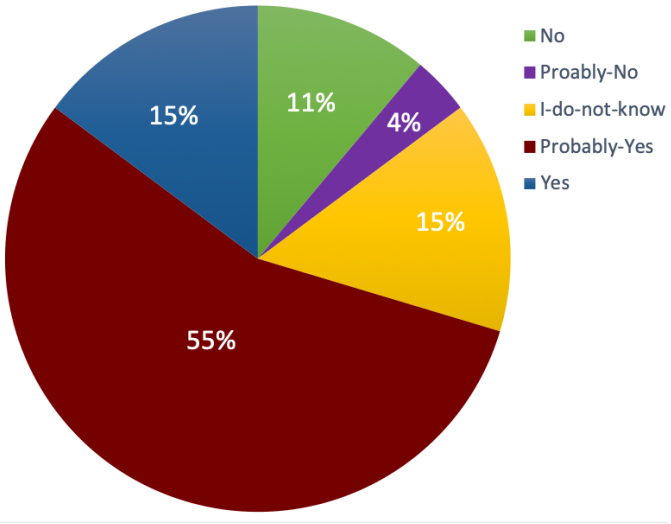


Figure 3.19 – Evaluation results of the reliability of the provided parking spot data and parking area occupancy statistics.

Functionality Figure 3.17 shows the evaluation results of the recommendation of parking spots, routes calculation, walking route and parking area occupancy statistics in terms of usefulness, with a scale of 1-star (very useless) to 5-stars (very useful). For the functionalities of the recommendations of parking spots, route calculation, walking routes and parking area occupancy statistics, around 93%, 96%, 93% and 85%, respectively, of the participants found these functionalities to be useful (identified by positive ratings of 3, 4

and 5-stars), and 85%, 81%, 70% and 78% of them found these respective functionalities to be useful/very useful (ratings of 4 and 5-stars).

Ease of Use Figure 3.18 presents the evaluation results of the ease of use of the navigation, route calculation and route analysis in the application. For the ease of use of the navigation, route calculation and route analysis, around 81%, 67% and 74% of the participants gave positive ratings. More specifically, 67%, 52% and 48% of the participants found it easy or very easy to use the navigation, route calculation and route analysis, respectively. This result indicates that we need to focus more on improving the application interface for route calculation and route analysis.

Reliability The results for the reliability question about whether the participants believe that the provided data of parking spot and parking area occupancy statistics are reliable are shown in Figure 3.19. Based on the interaction of the involved citizens with the application, close to 85% of them believe that the data provided about parking spot and parking area occupancy statistics are reliable, which is a good indication.

3.3.10 Demonstrations

The IoT recommender for smart parking system that is developed in WISE-IoT has been demonstrated in three occasions. In this section, we provide the details of these demonstrations.

3.3.10.1 Setup and Configuration

The IoT recommender for smart parking system is developed in WISE-IoT project and is one of the four components of Self-Adaptive Recommender (SAR) system [32]. For the demonstrations setup of the WISE-IoT project, the WISE-IoT components were deployed as Docker containers. The data of Santander parking sensors was stored in a oneM2M platform. The Morphing Mediation Gateway (MMG) [30] translates the data and create NGSI data structure based on the semantic information. Subsequently, it updates the information to the FIWARE Orion Context Broker. Then the IoT recommender accessed the sensors data from FIWARE Orion Context Broker, and interacted with other SAR components to receive the trust score from Trust Monitoring component [49]. Subsequently, the IoT recommender identified the parking spot and the route, and finally sent the output result of recommended parking spot and route to the smart parking application [46] and Adherence Monitoring [50] component.

3.3.10.2 IoT Week 2017 Geneva Demonstration

The first demonstration of IoT recommender was in the IoT Week 2017 Geneva from 6-9 June 2017 which was collocated with the Global IoT Summit 2017. The event was comprised of around 800 participants, 200 sessions and activities, and over 300 speakers. This first demonstration helped us to get the opinions and suggestions of the participants and experts, as well as helped us to identify some issues of IoT recommender, such as high processing time for some recommendation requests and some sensors not behaving well due to some physical problem with them. Overall, it was a good chance to identify the limitations of IoT recommender and improve its performance and functionality.

3.3.10.3 WISE-IoT First Review Meeting Demonstration

The second demonstration of IoT recommender was in the first review meeting of WISE-IoT in Brussels on 6 July 2017 and was comprised of all the WISE-IoT partners. This was the fully functioned demonstration of IoT recommender, however, it was not GDPR-compliant, rather was a normal implementation (as discussed in Section 3.3.5.1) because it was the first fully functioning version of IoT recommender. Hence, the outcome of this demonstration was to enhance IoT recommender to be GDPR-compliant (as discussed in Section 3.3.5.2).

3.3.10.4 IoT Week 2017 Korea Demonstration

The third and the last demonstration of IoT recommender was in the IoT Week 2017 South Korea from 10-11 October 2017 at COEX mall, Seoul, South Korea. The IoT recommender together with all SAR components was demonstrated with compliance to GDPR and it was the fully functioning GDPR-compliant demonstration of IoT recommender. However, later on, some enhancements were still made, such as walking route to the car and some sensors not working due to constructions in Santander, that were identified later at the time of field trials, hence IoT recommender was updated accordingly to address these changes.

3.3.11 Summary and Discussion

This study has presented the development, implementation and evaluation of a IoT Recommender (IoTRec) for a smart parking system. The main purpose of this system is to provide a better experience to both users and managers in terms of vehicular mobility in a city by relying on IoT technologies. The IoTRec mainly provided the GDPR-compliant recommendations of a parking spot (nearest or nearest trusted parking spot) and a route (least congested or shortest route) leading to the recommended parking spot, as well as the real-time provision of the expected occupancy of parking areas based on the historical IoT

data. Finally, the proposed system was evaluated through a prototype by the citizens of Santander.

The study in the next section presents the third part on designing an IoT Recommender for Smart Skiing.

3.4 IoT Recommender for Smart Skiing

The IoT recommender for smart skiing is developed for the smart skiing use case [69] of WISE-IoT project [33]. The main objective of smart skiing use case is to set up a testbed in Chamrousse, a ski resort in France that gathers skiing performance data of skiers using the deployed sensors and utilizes such data to provide recommendations to skiers for improving their skiing performance, compare their performance with other skiers, provide location coordinates and routes passing through ski slopes[2] and ski lifts to friends/family or ski devices. The IoT recommender for smart skiing is designed to provide route recommendations from user's current location to a point of Interest (POI) that can be either location of friends/family or ski devices for gamification.

To the best of our knowledge, for skiing routes, there does not exist any stable routing engine. Although a routing engine for skiing route is provided by OpenSnowMap [70], however, it is not accurate and has a number of limitations. For instance, firstly, it does not consider slopes to be one-way, rather it considers them to be two-way which is not realistic, secondly, it does not consider ski lifts to have fixed start and end stations, and therefore, it recommends to take a ski lift in the middle which is again not realistic, and thirdly, it considers all types of slopes in skiing route recommendations and does not provide any means to specify the type of slope a user is interested in. For example, it can only work for advanced users who can do ski on all types of slopes, but for novice, beginner and intermediate users, it is not possible to take such advance slopes. Therefore, there is a need of a novel routing engine for skiing routes that overcomes such limitations.

Inspired from this lacunae, in this study, we designed IoT recommender by following the requirements listed in Section 2.2.5.1 that offers a novel routing engine for skiing routes which is implemented in an open-source and widely-adopted routing engine, GraphHopper [71]. Our solution overcomes the limitations of OpenSnowMap [70] and considers slopes to be one-way, ski lifts can only be taken from the ski station instead of anywhere in the middle, provides skiing routes by considering different types of slopes (e.g., novice, easy, intermediate and advance), and provides multiple routes which gives more options to users to choose from.

3.4.1 Functionality

The IoT recommender has been deployed and integrated into smart skiing use case of WISE-IoT project. It offers the recommendations of ski routes for different level of expertise of skiers. In this section, we discuss the operation of IoT recommender and compare visually the routes offered by the IoT recommender and OpenSnowMap [70]. The routes by IoT

[2]We use slope and piste interchangeably

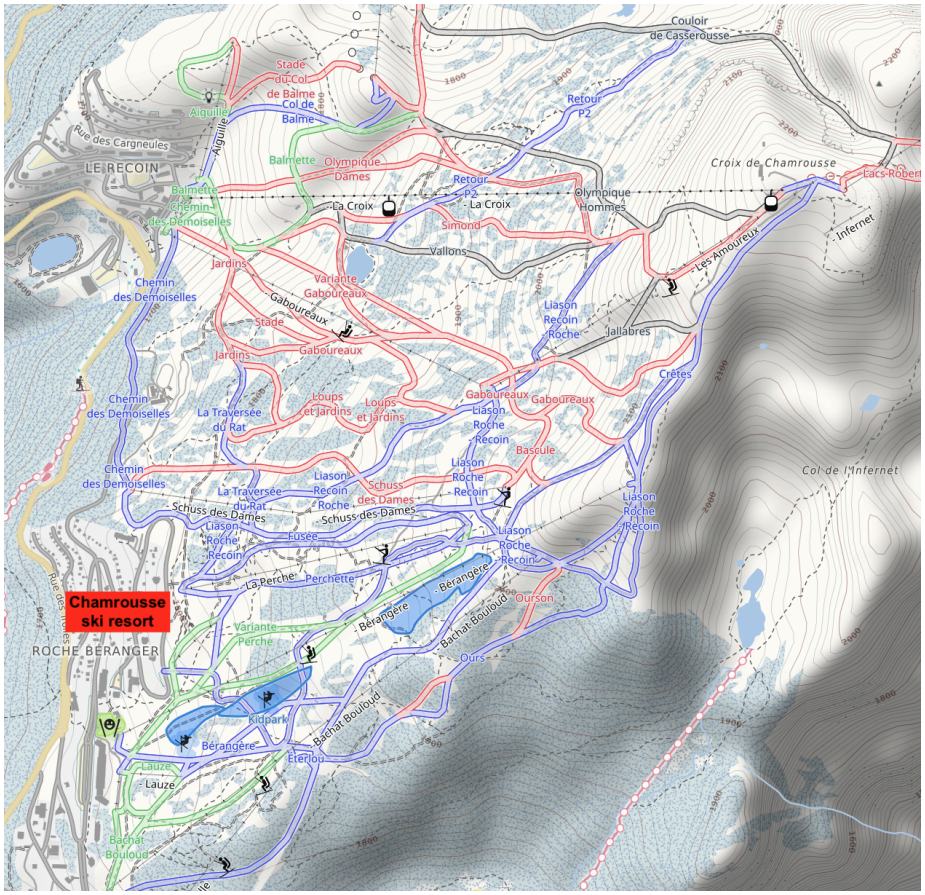
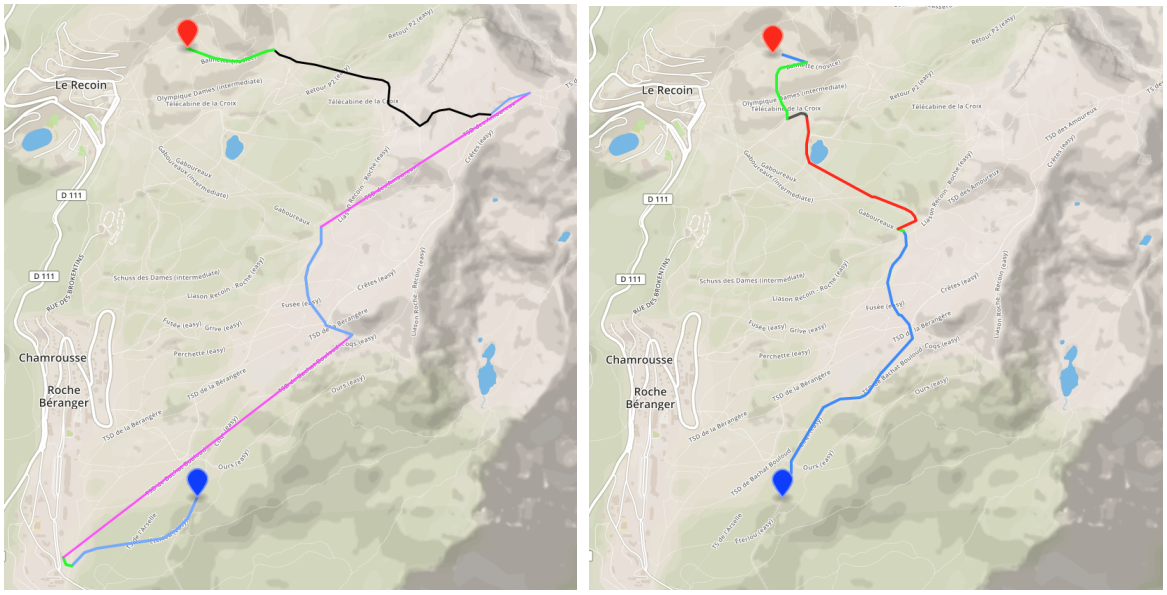


Figure 3.20 – An overview of Chamrousse ski resort [70].

recommender and OpenSnowMap are obtained in GeoJSON format that are imported into http://geojson.io to see the routes on the map in a user-friendly visual interface (presented in Figures 3.21-3.24).

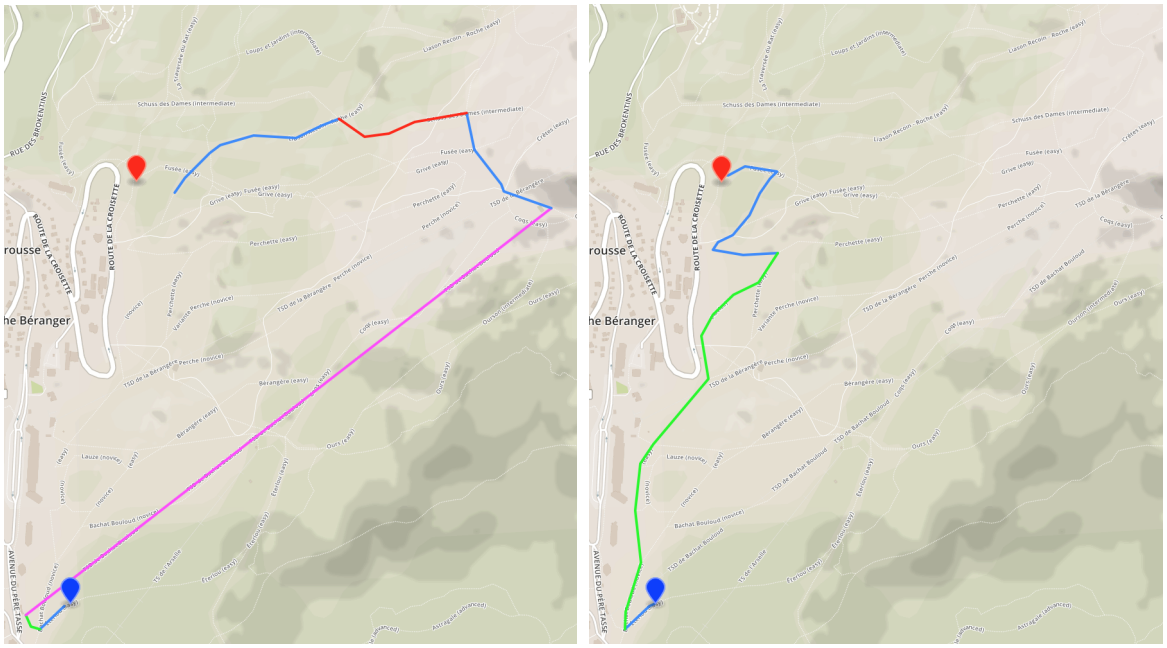
Figure 3.20 presents an overview of Chamrousse ski resort in France taken from OpenSnowMap [70]. There are various types of slopes/pistes in different colors. The green, blue, red and black colored slopes represents novice, easy, intermediate and advance slopes, respectively. The slopes are one-way, i.e., skiers can take slopes from top to bottom. There are also ski lifts that are two-way. The ski resort is situated at the bottom left of Figure 3.20. The skiers take ski lifts from there to reach the top of the resort and then take slopes. We have tested IoT recommender for smart skiing in Chamrousse, however, it can similarly be applied to anywhere.

There are four levels of expertise of skiers: advance, intermediate, beginner and novice skiers. Advance skiers are the most expert skiers and they can take any type of slope,



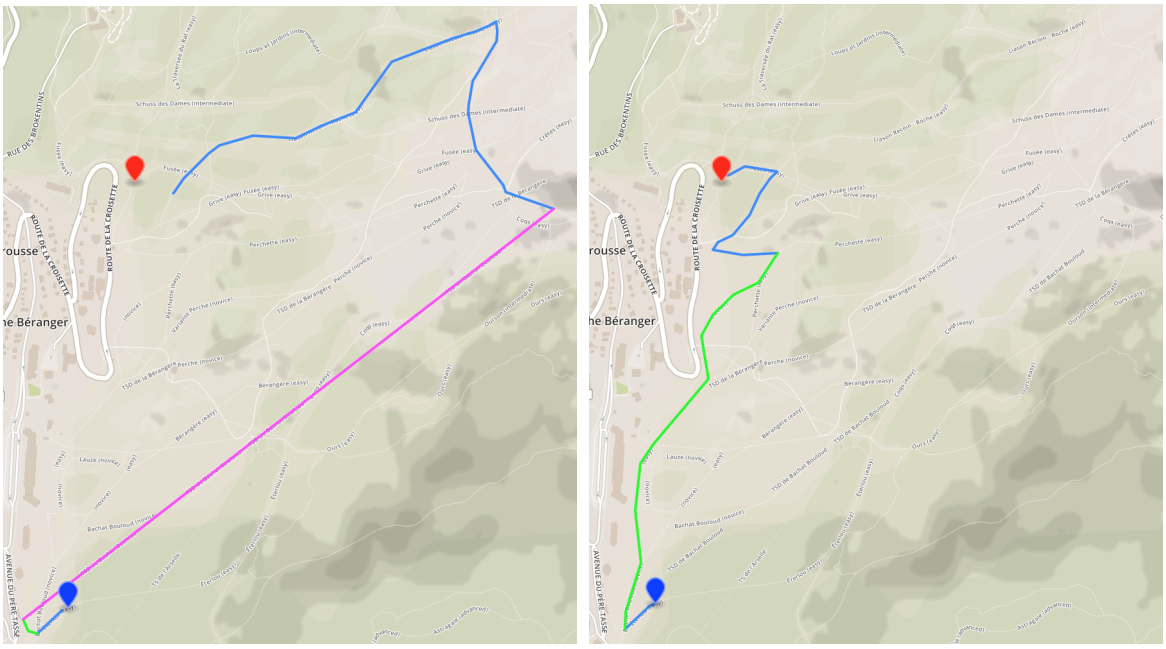
(a) Route offered by IoT recommender. (b) Route offered by Open Snow Map

Figure 3.21 – Skiing route for advanced users by IoT recommender and Open Snow Map.



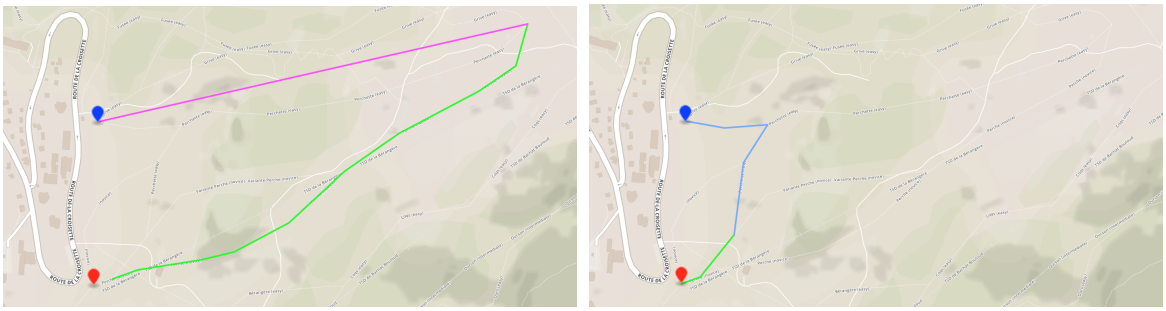
(a) Route offered by IoT recommender. (b) Route offered by Open Snow Map

Figure 3.22 – Skiing route for intermediate users by IoT recommender and Open Snow Map.



(a) Route offered by IoT recommender. (b) Route offered by Open Snow Map

Figure 3.23 – Skiing route for beginner users by IoT recommender and Open Snow Map.



(a) Route offered by IoT recommender. (b) Route offered by Open Snow Map

Figure 3.24 – Skiing route for novice users by IoT recommender and Open Snow Map.

e.g., advance (black), intermediate (red), easy (blue) and novice (green). The intermediate skiers can take intermediate (red), easy (blue) and novice (green) slopes. The beginner skiers can take easy (blue) and novice (green) slopes. Finally, the novice skiers can only take novice (green) slopes.

Figures 3.21, 3.22, 3.23, 3.24 compare the ski routes offered by IoT recommender and OpenSnowMap for advance, intermediate, beginner and novice users, respectively. The blue marker shows the starting location and the red marker shows the destination location. The pink colored line shows the ski lift. The figures show the limitation of OpenSnowMap and

the usefulness of IoT recommender. For instance, a skier can only take slopes from top to down, rather than from down to top which is not realistic. The OpenSnowMap recommends ski routes that require climbing the slopes instead of taking ski lifts to reach on top and then take the slope. This is achieved by the ski routes offered by IoT recommender. Additionally, the start and destination locations of ski routes for intermediate skier (in Figure 3.22) and beginner skier (in Figure 3.23) are same, yet the ski routes offered by the IoT recommender are different while they are same in the case of OpenSnowMap. This is because of although the ski route for intermediate skier (in Figure 3.22) in shorter but it passes through a slope of intermediate (red color) type. Therefore, it could not be recommended for beginner skier and hence, the IoT recommender recommends a slightly longer route passing through easy (blue colored) slopes for beginner skier in Figure 3.23. Finally, for novice skier in Figure 3.24, the ski route offered by the IoT recommender is comprised on only the novice (green colored) slopes and ski lift (pink colored) because they are in accordance with the expertise of novice skier while is not the case for OpenSnowMap because of not considering different types of slopes.

Note that in the ski route examples above, there is not any case in which OpenSnowMap takes the ski lift, but we found several cases when OpenSnowMap recommends to take a ski lift in the middle or leave the ski lift in the middle.

In summary, we exhibit the usefulness of IoT recommender and the limitations of OpenSnowMap. The IoT recommender has been integrated in real world into the smart skiing use case of WISE-IoT project [33].

3.4.2 Interfaces and Operation

Figure 3.25 presents the interfaces of the IoT recommender for smart skiing use case using UML component diagram. The IoT recommender first obtains a request of ski routes from smart skiing application. The request contains as query param: slope type, number of alternate routes, session id, and either user's current and destination locations, or start and end ski devices IDs equipped on skier body or buried under snow for gamification. There are two cases. Firstly, the IoT recommender receives location coordinates. In this case, it uses its novel skiing routing engine, developed in GraphHopper [71], to generate the ski routes (equal to the number of alternate routes specified in the request) for the specific type of slope and sends them to the smart skiing application. Secondly, the IoT recommender receives start and end ski devices. In this case, it first interacts with the SAR facade [32], a component of WISE-IoT project, to obtain the URL of the relevant Orion Context Broker containing the coordinates of ski devices. Then it interacts with the Orion Context Broker to obtain the location coordinates of the start and end ski devices and finally, it generates the ski routes for the specific type of ski slopes and sends them to

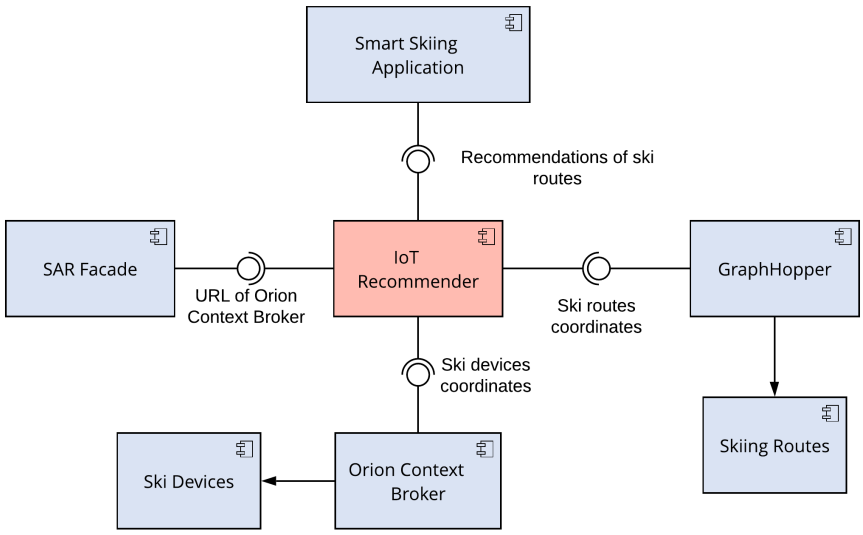


Figure 3.25 – UML component diagram of the interfaces of IoT recommender for smart skiing.

the smart skiing application.

3.4.3 REST APIs

In this section, we present the REST APIs offered by the IoT recommender for ski routes.

The REST API of IoT recommender for skiing routes, presented in Listing 3.7, takes an input the the session id (`long sessionId`), the slope type (`String type`), number of alternate routes (`int max_routes`), current location coordinates (`double start_lon & double start_lat`), and destination location coordinates (`double end_lon & double end_lat`). Alternative to the location and destination coordinates, this REST API can take as input a start device id (`String start_device_id`) and a destination device id (`String end_device_id`). The session id (`sessionId`) is used to retrieve the URL of the Orion Context Broker for skiing IoT devices to retrieve their GPS coordinates. The IoT recommender calculates and returns the number of alternate routes ('`max_routes`' passed as query param) for a specific type of slope ('`type`' passed as query param) in JSON format. If the start and destination coordinates are provided, the IoT Recommender considers them. Otherwise, if start and end device ids are provided instead of start and destination coordinates, the IoT Recommender fetches the respective GPS coordinates from the skiing Orion Context Broker via the SAR facade. The allowed slopes types are `piste_novice`, `piste_easy`, `piste_intermediate` and `piste_advance`.

Listing 3.7 presents the API and Listing 3.8 presents an example response.

```
GET
http://serverIP:port/IoTRecommender/v1/skiingRoute?
    sessionId={sessionId}&
    type={slopeType}&
    max_routes={maxAlternateRoutes}&
    start_lon={startLongitudeCoordinates}&
    start_lat={startLatitudeCoordinates}&
    end_lon={endLongitudeCoordinates}&
    end_lat={endLatitudeCoordinates}&
    start_device_id={startDeviceId}&
    end_device_id={endDeviceId}

Query Parameters:
    long sessionId: the session id of SAR facade to retrieve URL of skiing
        Orion Context Broker
    String type: type of slope (e.g., piste_novice, piste_easy,
        piste_intermediate, piste_advance)
    int max_routes:maximum number of alternate routes
    double start_lon: the start longitude coordinates of user's current
        location
    double start_lat: the start latitude coordinates of user's current
        location
    double end_lon: the end longitude coordinates of destination location
    double end_lat: the end latitude coordinates of destination location
    String start_device_id: the start skiing device id (equipped on the
        skier)
    String end_device_id: the end skiing device id (could be a ski
        gamification device or friends/family ski device)
Headers:
   Content-Type: application/json
   Accept: application/json
Response: application/json (See Listing 3.8 for an example)
```

Listing 3.7 – API for the recommendation of a skiing route for start and destination location coordinates

3.5 Summary and Discussion

To summarize and conclude, this chapter presented the first contribution of this thesis of IoT recommender for smart cities into three parts.

In the first part, it proposed the mapping of sensors and route coordinates by introducing a deviation margin. It presented an algorithm and two illustrative examples that cover all the possible scenarios. It evaluated the performance of mapping algorithm by considering four different routes and measured the correct detection, missed detection and false detection of traffic sensors on the routes.

In the second part, this chapter presented an IoT recommender for smart parking that utilized the mapping algorithm proposed in the first part and provided four-fold func-

```
Example Response:
"skiing-route": {
   "features": [
      {
         "geometry": {
            "coordinates": [
               [5.896236879294445, 45.11720601731254, 0]
               .........
            ],
            "type": "LineString"
         },
         "type": "Feature",
         "properties": {
            "instructions": [
            {
               "distance": 491.889,
               "sign": 7,
               "interval": [27, 35],
               "text": "Keep right onto Olympique Hommes",
               "time": 118053,
               "street_name": "Olympique Hommes"
            },
            {
               "distance": 329.068,
               "sign": 1,
               "interval": [46, 52],
               "text": "Turn slight right onto Balmette",
               "time": 78975,
               "street_name": "Balmette"
            },
            {
               "distance": 0,
               "sign": 4,
               "interval": [53, 53],
               "text": "Arrive at destination",
               "time": 0,
               "street_name": ""
            },
            "pistes-names": ["Olympique Hommes", "Balmette"],
            "track-length": 6252.60
         }
      }
   ],
   "type": "FeatureCollection",
   "errors": ""
}
```

Listing 3.8 – An example response of `skiing-route` JSON object

tions: recommendation of parking spots based on different metrics (e.g., nearest or nearest trusted), recommendation of routes leading to the recommended parking spots (the least crowded or the shortest route), real-time provisioning of expected availability of parking spots, and a GDPR-compliant implementation for operating in a privacy-aware environment. It offered its services using REST APIs and has been integrated and deployed into smart parking use case of WISE-IoT project. It was evaluated with the citizens of Santander and has been demonstrated at various occasions.

In the third part, this chapter presented an IoT recommender for smart skiing that offered the recommendations of ski routes between two points on a ski resort, passing through ski slopes and ski lifts by allowing to specify specific types of slopes (e.g., novice, easy, intermediate or advance). It offered its service through REST APIs and has also been integrated into smart skiing use case of WISE-IoT project.

In the next chapter, we will discuss the privacy preservation of users in the smart parking system, specifically in the users parking database by applying two well-known privacy preservation techniques of k-anonymity and differential privacy.

Chapter 4

Privacy Preservation for Smart Parking System

Contents

The main focus of this chapter is to preserve the privacy of IoT recommender system in smart parking system. More specifically, the historic parking dataset is shared with the parking recommender system for efficient and personalized recommendations based on the users past parking experience. However, since the parking dataset contains the past history of parking information of users, it breaches the privacy of users because the parking recommender (that is an adversary in our system) can track the routine and mobility patterns of users by analyzing such parking dataset. This chapter preserves the privacy of users against the parking recommender (an adversary). It preserves privacy using two well-known privacy preservation techniques of anonymization and perturbation: k-anonymity and differential privacy and evaluates the privacy and utility through extensive experimentations.

4.1 Introduction

Recent advancements in the Internet of Things (IoT) have revolutionized our daily lives and have transformed traditional applications into smart applications. Smart parking is one example of this transition. Generally, two types of implementations are considered in smart parking systems. In the first type, the smart parking application is responsible for receiving user requests and finding the available parking spots for the user by itself. This is a widely-adopted implementation. In the second type, the smart parking application receives user requests and forwards them to the third-party recommender systems which are responsible for recommending the parking spots based on various metrics, such as traffic conditions on the roads, distance, quality of parking spots, and users' past experience. The consideration of such diverse metrics into recommendations of parking spots is difficult for a smart parking application because of lack of access to diverse services and hence it is better to exploit the services of third-party recommender systems dedicated for this purpose. This implementation is currently less widely-adopted, however, is gaining attention with the horizontal and vertical emergence of IoT and smart cities applications, as well as with the interoperability in IoT that interconnects various applications/deployments, hence also interconnects third-party recommender systems with the smart parking system. For instance, there is a recent EU-KR H2020 WISE-IoT project [33], that enabled the interoperability between two IoT platforms: FIWARE and oneM2M which are widely used in Europe and South Korea, respectively. It demonstrated such interoperability through a smart parking use case by adopting the second type of implementation. In this demonstration, a smart parking application operates both in Europe and South Korea. When in Europe, it connects to the FIWARE platform and recommender system to obtain the recommendations of parking spots. While when in South Korea, it connects to oneM2M infrastructure and recommender system to obtain the recommendation of parking spots [68].

In this study, our focus is on the second type of implementation. Both types of implementation considers the smart parking application to be trustworthy that is responsible for receiving user requests and maintaining a parking database. However, the second type of implementation has an additional third-party parking recommender system. Since we do not know much about the third-party parking recommender system, therefore one cannot identify its trustworthiness and it could be either trusted or semi-trusted or untrusted.

In our considered scenario, a user, registered on a smart parking application, makes a request, comprised of user ID (e.g., registration id) and user's current location. On each user's request, the smart parking application (trusted entity) performs two fold functions. Firstly, it forwards the request to the third-party parking recommender system (semi-trusted or untrusted entity), obtains the recommended parking spot, sends it back to the user and collects the rating from user after completing the parking. Secondly, it maintains a parking database that contains user ID and user's current location (obtained from user's request), parking spot (obtained from recommender system), user rating (obtained from user after completing the parking) and current timestamp (the time of the user's request). The sample database is presented in Table 4.1. This database needs to be shared with the parking recommender system for personalized recommendations of parking spots based on user's past experience. For instance, by tracking the user's parking behavior and rating, it is possible to recommend those parking spots, the users have good experience with (e.g., frequently used and highly rated). However, the parking recommender system could identify an individual and infer user's routine and mobility patterns by analyzing the user's location and parking behavior, therefore the indiscriminate sharing of user's data with parking recommender system violates the privacy of the users. The parking recommender system can easily identify a user uniquely and trace his habits, behaviours and mobility patterns by analyzing the parking database. For example, as presented in Table 4.1, even if we remove the user ID (the unique identifier), the recommender system could easily guess the routine of user 1, as he leaves daily in the morning from the same place (*his home*) between 8:30am to 9:00am only on weekdays (*for the work*) and parks in the similar area (*his work place*), as parking spots 3601, 3602 and 3603 are very close to each other. Hence, the parking recommender system could exploit this routine to do malicous activity, e.g., plan stealing at user's home in his absence. Therefore, the user's request and the database contain user private information, and sharing them in their current form with the recommender system seriously violates the privacy of users. Hence, there is a need of preserving the privacy of users. One solution is that the parking application does not share such historical database and the recommender system recommends the parkings spots only based on the real-time information. However, this will cause lack of personalized and efficient recommendations of parking spots. Another solution is that application shares the

Table 4.1 – An example snapshot of data table for parking recommendation.

User ID	**User Location (lon, lat)**	**Parking ID**	**User Rating**	**Timestamp**
1	-3.80944, 43.4659	3601	5	2019-07-26 08:30:00
1	-3.80944, 43.4659	3601	5	2019-07-29 08:35:00
1	-3.80944, 43.4659	3602	5	2019-07-30 08:25:00
1	-3.80944, 43.4659	3603	4	2019-07-31 08:55:00
2	-3.80431, 43.4643	3605	1	2019-08-01 09:00:00
2	-3.79092, 43.4635	3872	5	2019-08-01 12:00:00
2	-3.80659, 43.4627	3610	3	2019-08-01 15:30:00
2	-3.79888, 43.4622	3625	4	2019-08-01 18:00:00
2	-3.79927, 43.4661	3901	4	2019-08-01 19:00:00

historical database by removing the user ID, however, the parking recommender system can still easily identify an individual by analyzing the other quasi-identifer attributes (e.g., user current location, parking spot and timestamp) as we discussed above. Hence, the preferred solution is to apply the privacy preservation techniques so that the parking recommender system could not be able to identify the private information of the individuals.

We focus on preserving privacy within the parking database containing users' parking history that could lead to infer users' behavior and mobility patterns. We assume that when the application sends user's current location to the parking recommender to obtain parking spot, it sends the perturbed user location by applying differential privacy (e.g., Geo-indistinguishabilty [72]), hence the parking recommender does not get the actual location of the user and the privacy is already preserved in the case of user's request. To preserve the privacy of statistical databases, there is an emerging interest in $k-$anonymity and differential privacy techniques that preserve privacy through anonymization and perturbation, respectively [18, 21]. $k-$anonymity [18] is the earliest work on privacy preservation that anonymizes a dataset in such as a way that with respect to the set of quasi-identifier attributes (i.e., attributes that can identify the individuals when combined together), each record (or row) is indistinguishable from at least $k-1$ other records. Differential privacy, instead, operates on the principle of data perturbation by adding noise to the query result [21]. Therefore, the parking recommender system would not be able to differentiate among multiple records (in $k-$anonymity), as well as would not be able to find the actual query result (in differential privacy), *hence making the users unidentifiable and indistinguishable in both cases.* Both $k-$anonymity and differential privacy are formally defined and discussed in detail in Section 4.4.1 and 4.4.2, respectively.

4.2 Related Work

For privacy preservation in current smart parking systems, the major focus of existing works is on protecting real-time user's location and navigation information, cryptography, pseudonymity, encryption and consortium blockchain. The protection of historic parking database, which is the focus of our study, is not investigated much.

For instance, Ni et al., [73, 74] preserve the privacy of parking navigation using Bloom filters by enabling a user (or vehicle) to receive the navigation results, even the user is moved out of range of the queried roadside unit. They preserve the privacy using pseudonymity in which the users make queries to the cloud server which handles the parking information for available parking spots in an anonymous manner. The cloud server enables the vehicle to receive the navigation query results even if the vehicle has moved out of the range of the queried roadside unit.

Chatzigiannakis et al., [75] preserves the privacy of a smart parking system by using public key cryptography scheme, known as elliptic curve cryptography that is suitable for resource constraint devices and is platform independent. The authors used zero knowledge proofs that avoids the exchange of confidential information, hence achieving the privacy. The authors evaluate the performance by studying the execution time and system overhead. However, the authors did not evaluate the privacy and utility of their proposed system.

Huang et al., [76] worked on automated valet parking system for which the parking reservation is a prerequisite in order to achieve automated parking. The authors worked on preserving the private information of drivers (e.g., identity and locations) that are revealed by the reservation requests by removing the user identity and making it anonymous. However, making the users anonymous cause security problem, e.g., double-reservation attack. The authors address this security issue by allowing each anonymous user to posses only one reservation token that can be used to reserve one available parking spot. In this way, the authors claimed to preserve the privacy of user's identity and location, as well as avoid double-reservation attack using zero knowledge proofs and proxy re-signature. However, the authors mainly preserve the privacy by using pseudonymity (making the user anonymous) and did not evaluate the privacy and utility of their system.

Lu et al., [77, 78] designed a smart parking system for large parking spots using vehicular communications that offers real-time navigation and anti-theft protection. The authors preserve the privacy of users by keeping the identity of users secret, i.e., by using pseudonymity. However, only protecting the explicit identifier is not sufficient because an adversary could still identify the users uniquely by linking and disclosure attacks.

Yan et al., [79] designed a privacy preserving parking system that relys on wireless network and sensor communications and allows users to reserve parking spots. The authors

protects the privacy by using encryption technique.

Alqazzaz et al., [80] proposed a privacy preserving and secure smart parking framework based on publish/subscribe mechanism. It provides two fold functions. Firstly, it offers the parking services, e.g., parking availability, navigation and parking reservation. Secondly, it offers security on application and network layers, as well as preserves privacy. The authors protect the privacy using encryption technique which is basically a security mechanism.

Garra et al., [81] implemented an anonymous e-coin system that protects the privacy of a parking system and offers payment by phone without disclosing start and end time.

Hu et al., [82] proposed a blockchain-based parking system using smart contracts that preserves privacy through a consortium blockchain in which the transactions are controlled by the legitimate nodes and are not disclosed to external entities.

The above mentioned works on privacy preservation for smart parking are firstly focused on the real-time user's location and navigation information. Secondly, they preserve privacy using pseudonimity, cryptography and encyption techniques which are prone to privacy leakage using linking and disclosing attacks, as proved in the literature. We, on the other hand, focus on privacy preservation using two well-known privacy preservation techniques of k-anonymity and differential privacy and we focus on preserving privacy within the historic parking database.

4.3 System and Adversary Models

4.3.1 System Model

Our system model is organized into two systems: smart parking and parking recommender systems. The smart parking system is comprised of users, smart parking application front-end, a service logic, a users parking database, an anonymized database (for privacy through k-anonymity), and a perturbation mechanism (for privacy through differential privacy). The parking recommender system is a third-party recommender system that uses various metrics (such as parking and traffic information, and sensors quality) to provide recommendations. The system architecture is presented in Fig. 4.1. A user, registered on a smart parking application, makes a request for a nearby parking spot comprised of his user id (e.g., registration number) and current location. The smart parking application is a trusted entity that receives requests from users, forwards user's request to service logic entity, obtains recommended parking spots from the parking recommender and provides them to the user, as well as collects ratings from users after they have completed their parking and forward them to the service logic entity. The service logic entity is also a trusted entity and it maintains a users parking database comprised of user ids and current locations, parking spots, ratings, and timestamp attributes. The parking recommender is

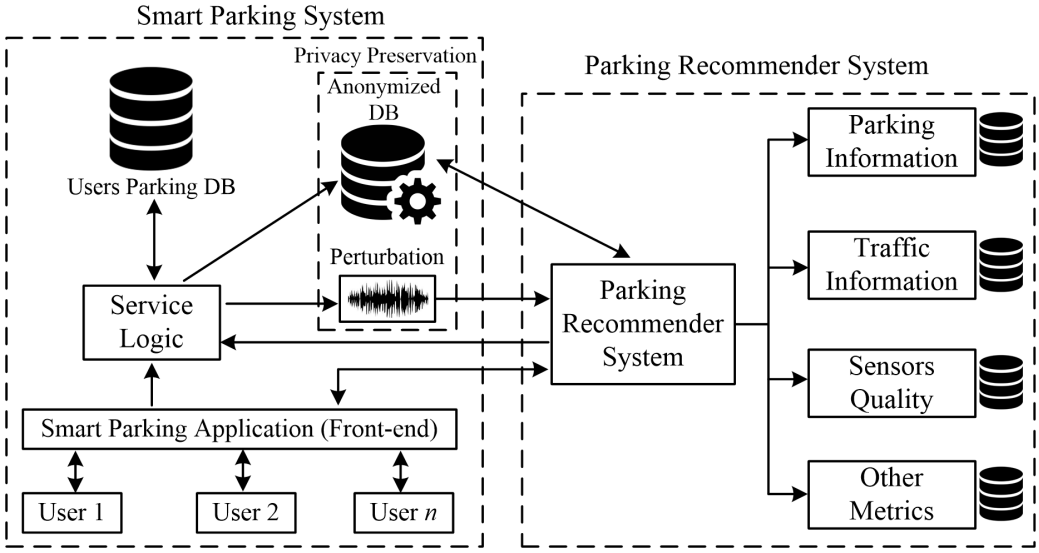


Figure 4.1 – System architecture of privacy preserving parking system.

either an untrusted or a semi-trusted entity that receives users' current locations, analyzes their past parking history and provides parking spots recommendations. To protect the privacy of users, the parking recommender should not be able to uniquely identify the users from the users parking database. We achieve this by using two well-known privacy preservation techniques: k-anonymity and differential privacy. In k-anonymity, the service logic entity generates an anonymized version of the users parking database and releases it to the parking recommender for analysis. In differential privacy, the parking recommender makes numeric queries to the service logic entity, but instead of receiving the actual responses, the service logic entity sends the perturbed responses to the parking recommender with noise added by the Laplace mechanism.

4.3.2 Adversary Model

The primary adversary[1] in our system is an untrusted (or semi-trusted) parking recommender system that needs access to the historical parking database for its recommendations of personalized and efficient parking spots. This system is susceptible to a disclosure attack, in which an adversary (i.e., the parking recommender) can recognize the behavior and mobility patterns of the users by observing the historical parking database. The adversary can track the behavior and mobility patterns of the users and uniquely identify them by analyzing the past history of users in the parking database. Such tracking could lead to the discovery of the users' private information and unique identification. For example,

[1]We use parking recommender system and adversary interchangeably

from past parking history, a user can be identified when he is at work, when he returns home, as well as other personal information, e.g., when and which hospitals or clinics he visits etc. Therefore, the parking database must preserve the privacy of the users such that an adversary could not be able to uniquely identify a user. We assume that the adversary is curious but not malicious.

4.4 Privacy Preservation

We preserve privacy using two techniques; one uses non-interactive data publishing through k-anonymity [18] and the other uses interactive data publishing through differential privacy [21]. Out of the four privacy preservation techniques discussed in Section 2.3, we adopted these two techniques of k-anonymity and differential privacy. We do not use ℓ-diversity and t-closeness because they are used when the distribution of sensitive attribute is homogeneous or skewed, respectively, however, we do not have sensitive attribute in our parking dataset.

4.4.1 Privacy Preservation through k-anonymity

$k-$anonymity [18] is the earliest work on privacy preservation that anonymizes a dataset in such as a way that with respect to the set of quasi-identifier attributes, each record (or row) is indistinguishable from at least $k-1$ other records. The main purpose of k-anonymity is to counter linking attacks, so that an adversary cannot uniquely identify a user by linking the quasi-identifier attributes (such as birthdate, zip code and gender) with external data. However, the quasi-identifier attributes in our scenario (e.g., user current location, parking spot and timestamp), while they cannot by their nature be used to uniquely identify users by linking to the external data, they can be combined together to track a user's behavior and mobility pattern (e.g., *a disclosure attack*). Therefore, we apply k-anonymity for indistinguishability among multiple users, thus preventing an adversary from identifying a user uniquely. k-anonymity is formally defined as:

Definition 1 (k-anonymity). *A dataset $D(A_1, A_2, ..., A_n)$ having attributes $(A_1, A_2, ..., A_n)$, where n is the number of attributes, QI_D be the quasi-identifier attributes associated with this dataset and $D[A_1]$ is the value of A_1 attribute in dataset D. Then the dataset D satisfies k-anonymity if each sequence of values of quasi-identifier attributes in dataset D (i.e., $D[QI_D]$) appears atleast k times* [18].

The higher is the value of k, the stronger is the privacy. However, a trade-off exists between privacy and utility, the stronger is the privacy (e.g., higher value of k), the lesser will be the utility. Hence, a balance between privacy and utility is required.

k-anonymity achieves anonymization using generalization and suppression [18]. In this

study, we consider *single dimensional global recoding* (i.e., mapping a value to the same level of generalization in all the records for each attribute individually). In anonymization process, removing the explicit identifiers is the first step, hence we first remove *user ID* and apply anonymization on quasi-identifier attributes of *user location (e.g., latitude and longitude)*, *parking spot* and *timestamp*, while constructing the anonymized dataset. The implementation and experimentation details are provided in Section 4.5.2

4.4.2 Privacy Preservation through Differential Privacy

Dwork [21] coined the term differential privacy, with the definition that the outcome of a differentially private mechanism does not get highly affected by adding or removing a single record in the dataset. This mechanism can protect the privacy of users while sharing a database with an untrusted recommender system by perturbing the data. Differential privacy thus overcomes the limitations of k-anonymity, specifically the curse of dimensionality [22]. We adopt the interactive differentially private data publishing approach for numeric queries by adding noise created by the Laplace mechanism, thereby answering each numeric query f as it reaches the smart parking system (e.g., the curator) without revealing any individual record [83].

We next define differential privacy and some important notations.

Definition 2 (ϵ-Differential Privacy).

A randomized process X adheres to ϵ-differential privacy if it fulfills the following two conditions: i) two adjacent datasets D_1 and D_2 differ only in one element, and ii) all outputs $S \in range(X)$ where $range(X)$ is the range of outputs of the process X [84]. Formally,

$$P_r[X(D_1) \in S] \leq e^{\epsilon} P_r[X(D_2)] \in S \tag{4.1}$$

where $X(D_1)$ and $X(D_2)$ are the randomized processes applied to datasets D_1 and D_2, and ϵ is a parameter of privacy, known as the privacy budget. The smaller the value of ϵ, the stronger the privacy.

Definition 3 (Sensitivity). The sensitivity defines the amount of the required perturbation. Assuming a query function $f(.)$ in a given dataset, the sensitivity $\triangle f$ is defined as:

$$\triangle f = \max_{D_1, D_2} ||f(D_1) - f(D_2)||_1 \tag{4.2}$$

Definition 4 (Laplace Mechanism). Differential privacy uses the Laplace mechanism to perturb the results for numeric queries. It adds Laplace noise to the query result sampled from the Laplace distribution that is centered at 0 with scaling b. The Laplace

noise is represented by $Lap(b)$. The higher the value of b, the higher the noise. The probability density function (pdf) of the Laplace distribution is given as $Lap(x) = \frac{1}{2b}\epsilon^{-(|x|/b)}$. The Laplace mechanism for differential privacy is formally defined as: Given a function $f : D \rightarrow \mathbb{R}$, the randomize process X adheres to ϵ-differential privacy if:

$$X(D) = f(D) + Lap(\frac{\triangle f}{\epsilon}) \tag{4.3}$$

Equation 4.3 shows that the amount of noise is dependent upon the privacy budget ϵ and sensitivity $\triangle f$. A lower privacy budget ϵ and higher sensitivity $\triangle f$ generate higher amount of noise.

The parking recommender makes the numeric query f to analyze the parking history, e.g., the rating of a selected parking spot belonging to the user's current location because it may be possible that users of a certain location did not like certain parking spots due to various reasons, e.g., too far away, crowded or a narrow or poorly-maintained road. A sample query is:

f: How many users from a specific location (user current location, e.g., 43.3905, -3.8896) gave a rating (e.g., 5-stars) for a specific parking spot (e.g., 3601) between a specific timestamp (e.g., 2019-08-01 08:00−2019-08-02 08:00)?

This type of query is used with different parameters to evaluate differential privacy in the next section.

4.5 Experiments

4.5.1 Experimental Setup

We used a real parking dataset of Santander, Spain that is comprised of the occupancy time of parking spots for the month of December 2017. Real locations within Santander were then used to generate a synthetic parking dataset by assigning the user locations and ratings randomly for each record of the real parking occupancy dataset in order to evaluate the privacy preservation of k-anonymity and of differential privacy techniques. Hence, although our dataset is synthetic, it is generated from a real parking occupancy dataset and real locations, and therefore it reflects a real dataset. The size of the dataset is 15306 records composed of 500 distinct real locations as users' current locations (latitude and longitude), 265 parking spots, 6242 timestamps and ratings of between 1 to 5 for the duration of December 2017. The experiments were performed using Python 3.7.3 with NumPy v1.16.4 and Pandas v0.24.2 libraries on a macOS Catalina v.10.15 with an Intel Core i7 2.7 GHz processor with a 16 GB LPDDR3 RAM.

Table 4.2 – Description of the experimentation database for anonymization.

Attribute	**Distinct Values**	**Generalization type**	**Height**
User latitude	500	25 intervals between 0.0001 and 0.05	26
User longitude	500	25 intervals between 0.0001 and 0.05	26
Timestamp	6242	Intervals of 1, 2, 3, 4, 5 and 6 hours	7
Parking spot	265	13 intervals between 10 and 300	14

4.5.2 Evaluation of k-anonymity

We used the four attributes (user latitude, user longitude, timestamp and parking id) presented in Table 4.2 of the parking dataset as Quasi-Identifier Attributes (QIA) and evaluated k-anonymity using different values of k from 2 to 750. We analyzed the performance of k-anonymity by individually studying different QIA sizes to have a complete and detailed analysis. In Section 4.5.2.2, we analyze QIA size = 1 by selecting *user latitude* as QIA. In Section 4.5.2.3, we analyze QIA size = 2 by selecting *user latitude* and *user longitude* as QIA. In Section 4.5.2.4, we analyze QIA size = 3 by selecting *user latitude*, *user longitude* and *parking id* as QIA. In Section 4.5.2.5, we analyze another case of QIA size = 3 by selecting *user latitude*, *user longitude* and *timestamp* as QIA. In Section 4.5.2.6, we analyze QIA size = 4 by selecting all the attributes: *user latitude*, *user longitude*, *parking id* and *timestamp* as QIA. Finally in Section 4.5.2.7, we present all the QIA sizes together that are discussed above to see the consolidated effect.

4.5.2.1 Performance Metrics

We evaluate the performance of k-anonymity in terms of privacy and utility using six widely-adopted metrics:

1. *Average groups size* is the average size of the anonymized blocks/groups generated by the anonymization technique. It is used to measure the privacy and utility of the anonymized algorithm and has been widely used in the literature [19,20]. If the groups sizes are smaller, an adversary/analyst would be able to infer more information that enhances the utility but it weakens the privacy because due to smaller groups size, it is relatively easier to uniquely identify the users. On the other hand, when the groups sizes are larger, the privacy is stronger because it is difficult to identify the users but it reduces the utility. So, the smaller average groups size is favourable for utility while higher average groups size is favourable for privacy.

2. *Total number of groups* is the number of groups generated by the anonymization algorithm. A higher number of groups causes smaller groups size which enhances

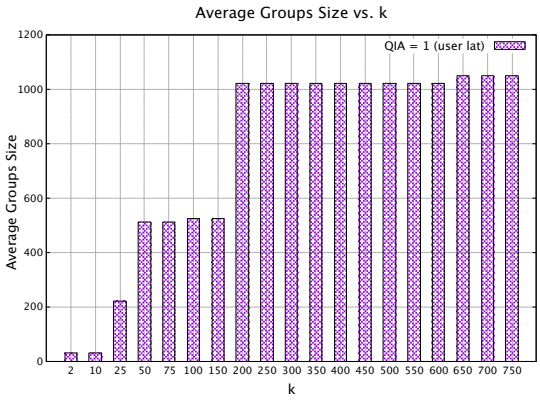
the utility but weakens the privacy. On the other hand, a lower number of groups makes the privacy stronger but it reduces the utility. So, the higher number of groups is favourable for utility while lower number of groups (or higher groups size) is favourable for privacy.

3. *Generalization height* is the height of an anonymized database, i.e., the number of generalization levels applied. It has been widely used in the literature for measuring the privacy and utility of anonymization technique [16,19,85]. With lower generalization height, the values of records are closer to their actual values, hence it enhances the utility but it weakens the privacy because an adversary/analyst could identify the users uniquely. On the other hand, when the generalization height is higher, the values of records are in the much generalized form, making it difficult for an adversary/analyst to infer useful information from the anonymized dataset which results in stronger privacy but lower utility. So, a lower generalization height is favourable for utility while higher generalization height is favourable for privacy.

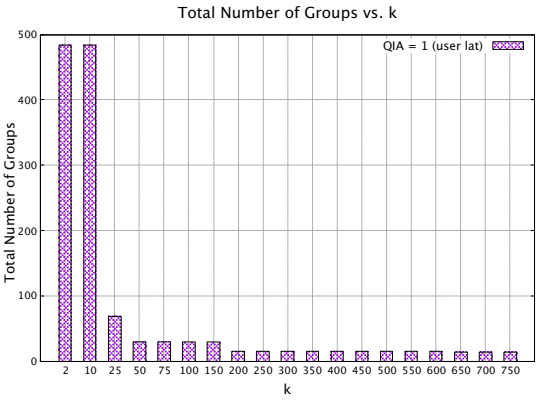
4. *Number of suppressed records* is the number of records that are suppressed because they could not fit into any anonymized block/group (because of not fulfilling the requirement of k) while privacy preservation. Suppressed records enhance privacy because if they do not get suppressed, an adversary could identify the users because of not fulfilling the requirement of k (i.e., not fulfilling the indistinguishability of records). However, they reduce the utility because the suppressed records reduce the size of the dataset, making the inference of useful information lower. So, a lower number of suppressed records is favourable for utility.

5. *Discernibility cost* is the metric to measure the indistinguishability of records with each other. The discernibility metric penalizes each record based on their indistinguishability from each other. Each unsuppressed record in a group of size j incurs a cost j, while each suppressed record incurs a cost $|D|$, i.e., the size of the original dataset D. This metric is used to measure the utility and privacy of anonymization algorithm and it has also been widely-adopted in the literature [19, 20, 86]. A lower discernibility cost if favourable for utility while higher discernibility cost if favourable for privacy.

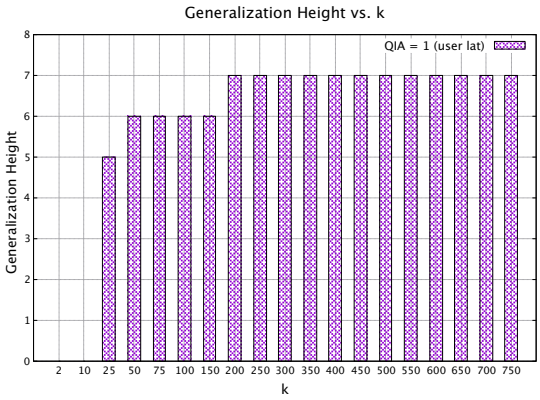
6. *Execution time* is the time required to generate an anonymized database of the original database [85]. A lower execution time is favourable.



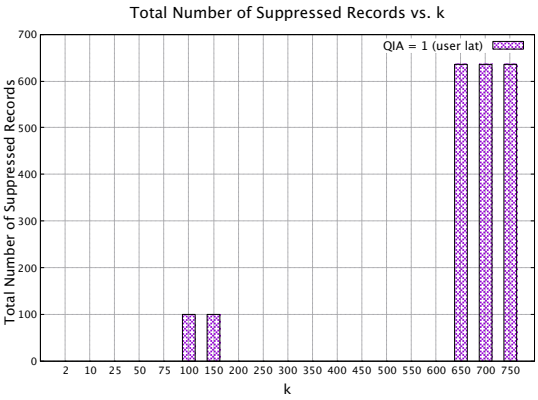
(a) Average group size



(b) Total number of groups

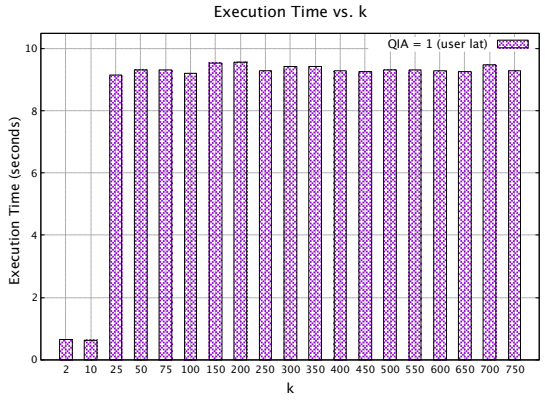


(c) Generalization height



(d) Number of suppressed records

(e) Discernibility cost



(f) Execution time

Figure 4.2 – Performance evaluation of k-anonymity when QIA = 1 (user latitude)

4.5.2.2 Analysis of One Quasi-Identifier Attribute

In this section, we analyze the performance of k-anonymity when one attribute is selected as QIA, i.e., QIA = 1 (user latitude).

Figure 4.2a presents the average groups size generated by the anonymization algorithm from k=2 to k=750. For k=2 and k=10, the average groups size is around 30. This is because no anymization is required as there is a total of 500 locations (i.e., num_loc=500) that are randomly assigned to the parking dataset D of size $|D|$=15306. Therefore, each user latitude appears around 30 times on average (i.e., $avg\_appearance = \frac{|D|}{num\_loc} \approx 30$), hence it already fulfils the requirement of k=2 and k=10, by default. When k=25, the average groups size is around 200. Although, apparently it seems that there should also be no need of anonymization of user latitude at k=25 because the expected repetition of each user latitude is 30 times (as discussed above, i.e., k=25 < $avg\_appearance$=30), however, firstly that is an average repetition appearance, and secondly, since the user locations are assigned randomly to the parking dataset, therefore the user latitude repetitions vary (i.e., from 17 to 71 times). Hence, the anonymization needs to be performed at k=25 and it generates the average groups size of around 200 records. From k=50 to k=150, the average groups size is around 500, and finally from k=200 to k=750, the average groups size is around 1000. This result shows that the average groups size increases with the increasing values of k. Higher is the value of k, higher is the group size, and hence lower is the utility.

Figure 4.2b presents the total number of groups generated by the anonymization algorithm from k=2 to k=750. For k=2 and k=10, the total number of groups is very high and around 500. This is because no anonymization is required (as shown in Figure 4.2a). The total number of groups keeps decreasing from k=25 to k=750. This is because for each increasing value of k, the anonymization algorithm has to maintain indistinguishable groups of records that fulfills the requirements of k, hence causes larger groups and hence smaller number of total groups. This result shows that the total number of groups reduces with the increasing values of k. Higher is the value of k, lower is the total number of groups, and hence lower is the utility.

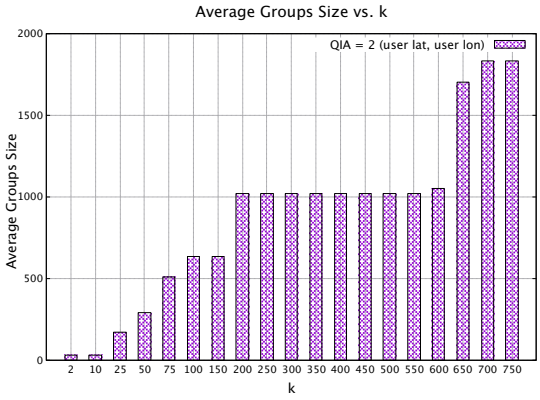
Figure 4.2c presents the generalization height applied by the anonymization algorithm from k=2 to k=750. For k=2 and k=10, the generalization height is zero because no anonymization is required (as discussed above). The generalization height for k=25 is 5 because the anonymization algorithm is able to generate anonymized parking database at this height. The generalization height for k=50 to k=150 is same and is 6 because as we analyzed in Figure 4.2a, since the average groups size are same from k=50 to k=150, therefore they are achieved by applying the same height of generalization. Similarly, the generalization height from k=200 to k=750 is also same and is 7. This result shows that the generalization height increases with the increasing values of k. Higher is the value of

k, higher is the generalization height, and hence lower is the utility.

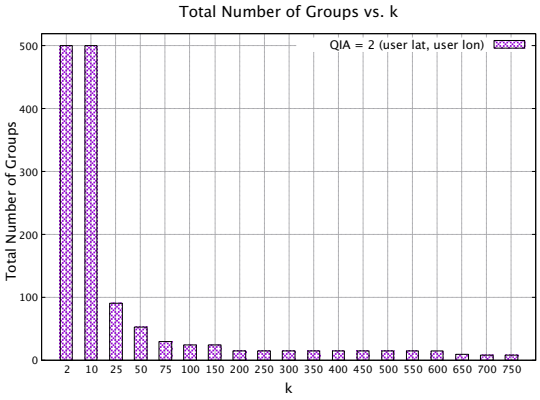
Figure 4.2d presents the number of suppressed records by the anonymization algorithm to generate an anonymized parking database from k=2 to k=750. The records are suppressed only when k=100,150 and when k=650,700,750. This is because in order to maximize the utility, the anonymization algorithm tries to apply as minimal generalization height as possible. Therefore, while applying a new generalization level, it first checks the number of records that are not k-anonymous (i.e., N_{non_anon}). If $N_{non\_anon} > k$, it goes for another level of generalization, otherwise if $N_{non\_anon} < k$, it suppresses these N_{non_anon} records for maximizing the utility. This is why, the number of records that are not k-anonymous (N_{non_anon}) at k=100,150 and k=650,700,750 are suppressed. This result shows that on the one hand, the number of suppressed records reduces the utility by reducing size of the dataset, but on the other hand, it actually enhances the utility by avoiding another level of generalization. Because the generalization affects the whole dataset and may reduce the utility drastically by making the records more generalized and hence more difficulty in analysis, as compared to the suppression of a small number of records (i.e., less than k).

Figure 4.2e presents the discernibility cost from k=2 to k=750. For k=2 and k=10, the discernibility costs are very low because no anonymization is required (as discussed in the description of Figure 4.2a). At k=25, the discernibility cost is around 5×10^6. An important point to note here is that from k=50 to k=150, the discernibility cost for k=100,150 is higher than that of k=50,75, while they all apply the same generalization height, have the same average group size and total number of groups, however, they differ in the number of suppressed tuples (as shown in previous Figure 4.2d) and this is the reason of higher discernibility cost at k=100,150 as compared to k=50,75. The similar explanation applies to the higher discernibility cost at k=650,700,750 as compared to k=200-600. This result is a very significant metric of utility and it shows that the discernibility cost increases with the increasing values of k because of higher groups size and number of suppressed records. Higher is the value of k, higher is the discernibility cost, and hence lower is the utility.

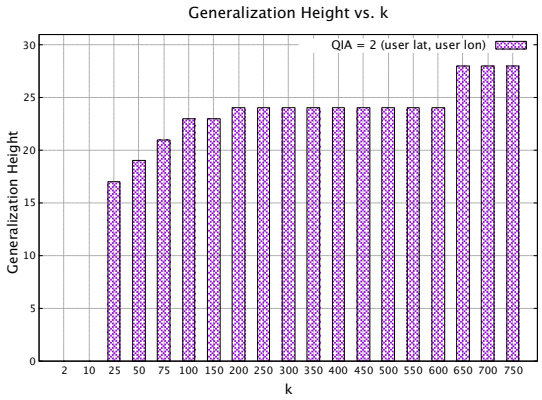
Finally, Figure 4.2f presents the execution time from k=2 to k=750. The execution time for k=2 and k=10 is very negligible because no anonymization is required (as discussed above). While for k=25 to k=750, the execution time is almost similar because the main execution time is consumed in making the generalizations of the records. Since, there is no much difference in the generalization heights of k=25 to k=750 (as presented in Figure 4.2c), therefore the execution time is similar.



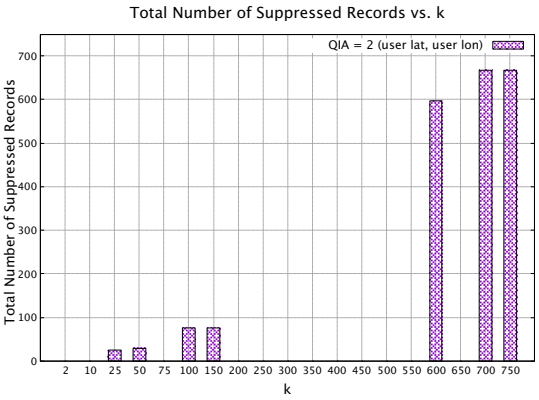
(a) Average group size



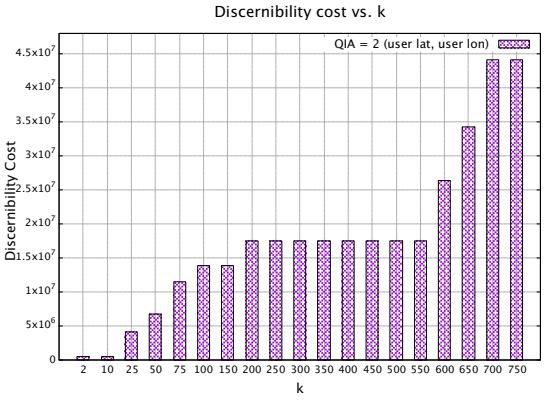
(b) Total number of groups



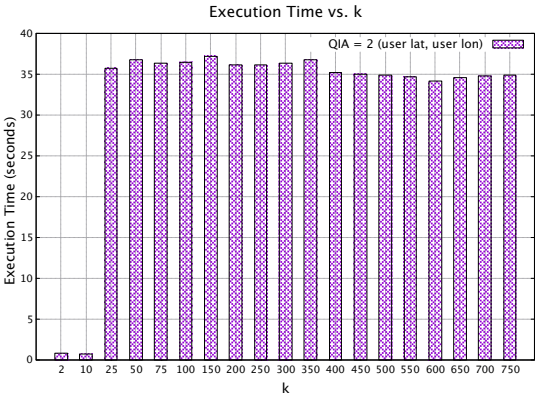
(c) Generalization height



(d) Number of suppressed records



(e) Discernibility cost



(f) Execution time

Figure 4.3 – Performance evaluation of k-anonymity when QIA = 2 (user latitude, user longitude)

4.5.2.3 Analysis of Two Quasi-Identifier Attributes

In this section, we analyze the performance of k-anonymity when two attribute are selected as QIA, i.e., QIA = 2 (user latitude, user longitude).

Figure 4.3a presents the average groups size generated by the anonymization algorithm from k=2 to k=750 when QIA size = 2 (user latitude, user longitude). For k=2 and k=10, the average groups size are around 30. Since a location is a combination of latitude and longitude, therefore the same reason discussed in the description of Figure 4.2a in Section 4.5.2.2 applies here as well, i.e., no anonymization is required because the original parking dataset already fulfills the requirement of k=2 and k=10 by default. In other words, the average appearance of each location ($avg\_appearance$=30) is greater than k=2 and k=10. When k increases from 25 to 150, the average groups size also keeps increasing to fulfill the indistinguishability requirement of k. However, from k=200 to k=600, the average groups size is similar and is around 1000. This is because at k=200, the minimim group size at k=200 is 600, therefore the higher level of anonymization (or generalization) is required above k=600. Finally, the average groups size from k=650 to k=750 are much higher and around 1700-1800. This result shows that the average groups size increases with the increasing values of k. Higher is the value of k, higher is the group size, and hence lower is the utility.

Figure 4.3b presents the total number of groups generated by the anonymization algorithm from k=2 to k=750 when QIA size = 2 (user latitude, user longitude). For k=2 and k=10, the total number of groups are very high and around 500. This is because no anonymization is required (as discussed before). The total number of groups keeps reducing from k=25 to k=750. This is because for each increasing value of k, the anonymization algorithm has to maintain indistinguishable groups of records that fulfills the requirements of k, hence causes larger groups and hence smaller number of total groups. This result is very similar to the result of total number of groups when QIA size = 1 (user latitude) in Figure 4.2b because a location is comprised of a pair of latitude and longitude. Hence, when we apply either one part of location (e.g., latitude) or both parts (e.g., latitude and longitude), we exhibit the similar trend. It shows that the total number of groups reduces with the increasing values of k. Higher is the value of k, lower is the total number of groups, and hence lower is the utility.

Figure 4.3c presents the generalization height applied by the anonymization algorithm from k=2 to k=750 when QIA size = 2 (user latitude, user longitude). For k=2 and k=10, the generalization height is zero because no anonymization is required (as discussed above). The generalization height for k=25 is around 17 because the anonymization algorithm is able to generate anonymized parking database at this height when anonymizing two QIA of user latitude and user longitude. The generalization height for k=50 to k=150 keeps

increasing, but remains same from k=200 to k=600. The reason is similar as explain in the result of average groups size, i.e., the minimum groups size at k=200 is 600, hence no more generalization (or anonymization) is required until k=600. Finally, the generalization height from k=650 to k=750 is around 28 and is the highest. This result shows that the generalization height increases with the increasing values of k. Higher is the value of k, higher is the generalization height, and hence lower is the utility.

Figure 4.3d presents the number of suppressed records by the anonymization algorithm to generate an anonymized parking database from k=2 to k=750 when QIA size = 2 (user latitude, user longitude). The records are suppressed when k=25,50,100,150,600,700,750. This is because in order to maximize the utility, the anonymization algorithm tries to apply as minimal generalization height as possible. Therefore, while applying a new generalization level, it first checks the number of records that are not k-anonymous (i.e., N_{non_anon}). If $N_{non\_anon} > k$, it goes for another level of generalization, otherwise if $N_{non\_anon} < k$, it suppresses these N_{non_anon} records for maximizing the utility. This is why, the number of records that are not k-anonymous (N_{non_anon}) at k=25,50,100,150,600,700,750 are suppressed. This result shows that on the one hand, the number of suppressed records reduces the utility by reducing the size of the dataset, but on the other hand, it actually enhances the utility by avoiding another level of generalization. Because the generalization affects the whole dataset and may reduce the utility drastically by making the records more generalized and hence more difficulty in analysis, as compared to the suppression of a small number of records (i.e., less than k).

Figure 4.3e presents the discernibility cost from k=2 to k=750 when QIA size = 2 (user latitude, user longitude). For k=2 and k=10, the discernibility cost is very low and almost negligible because no anonymization is required (as discussed before). The discernibility cost keeps increasing from k=25 to k=150 because of having varying groups sizes. However from k=200 to k=550, the discernibility cost is same because of having the similar groups size. Although, the average group size at k=600 is also same (in Figure 4.3a) but it incurs higher discernibility cost because of suppressing the records. Finally, the discernibility cost increases at k=650 and stays constant from k=700 to k=750. This result shows that the discernibility cost increases with the increasing values of k because of higher groups size and number of suppressed records. When the groups size and number of suppressed records are similar, the discernibility cost is also similar (e.g., from k=200 to k=550). Higher is the value of k, higher is the discernibility cost, and hence lower is the utility.

Finally, Figure 4.3f presents the execution time from k=2 to k=750 when QIA size = 2 (user latitude, user longitude). The execution time for k=2 and k=10 is very negligible because no anonymization is required (as discussed above). While for k=25 to k=750, the execution time is almost similar because the main execution time is consumed in making

the generalizations of the records. Since, there is no much difference in the generalization heights of k=25 to k=750 (as presented in Figure 4.2c), therefore the execution time is similar.

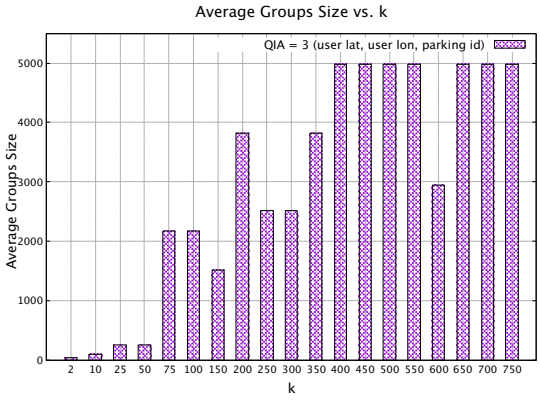
4.5.2.4 Analysis of Three Quasi-Identifier Attributes (Case 1)

We have two cases when three attributes are selected as QIA. In both case, the first two QIA are *user location* and *user longitude*. In the first case, the *parking id* is selected as the third QIA, while in the second case, *timestamp* is selected as the third QIA. In this section, we consider the first case and analyze the performance of k-anonymity when three attributes are selected as QIA, i.e., QIA = 3 (user latitude, user longitude, parking id).

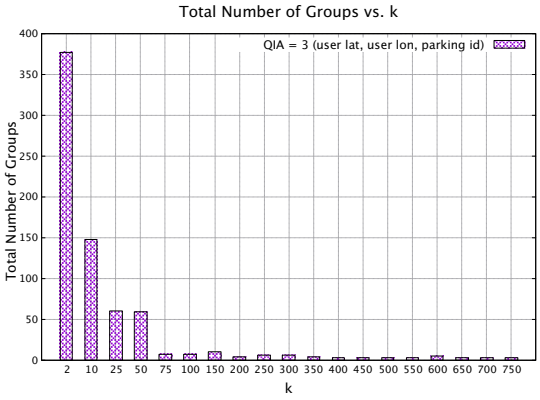
Figure 4.4a presents the average groups size generated by the anonymization algorithm from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, parking id). The average groups size from k=2 to k=50 are very small as compared to others because due to the repeated locations (user latitude and longitude) and parking spots, the anonymization algorithm was able to make smaller groups causing lower groups sizes. However, from k=75 to k=750, the average groups size keep increasing because since the anonymization algorithm has to fulfill the indistinguishability of records equal to k, it ended up making bigger groups. However, at k=150,250,300,600, the average groups sizes are smaller as compared to their neighbors. The reason is that at these values of k, the anonymization algorithm was able to enhance the utility by applying slightly lower generalization height (discussed next in Figure 4.4c) by the suppression of records (discussed next in Figure 4.4d). Overall, this result shows that the average groups size increases with the increasing values of k. Higher is the value of k, higher is the group size, and hence lower is the utility.

Figure 4.4b presents the total number of groups generated by the anonymization algorithm from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, parking id). In this result, the total number of groups are very high for k=2,10,25,50 having total number of groups around 400, 150, 60 and 60 respectively. However, at k=75, the total number of groups reduces drastically with having a total of 7 groups. From k=75 to k=750, the total number of groups are very low and between 3 to 10. The pattern is very obvious and self-explanatory. For lower values of k, the anonymization algorithm has to ensure low indistinguishability of records, and hence it can make smaller groups sizes resulting in a higher number of total groups. But as the value of k gets higher, the anonymization algorithm has to ensure high indistinguishability of records, resulting in larger groups sizes and lower number of groups. This result shows that the total number of groups reduces with the increasing values of k. Higher is the value of k, lower is the total number of groups, and hence lower is the utility.

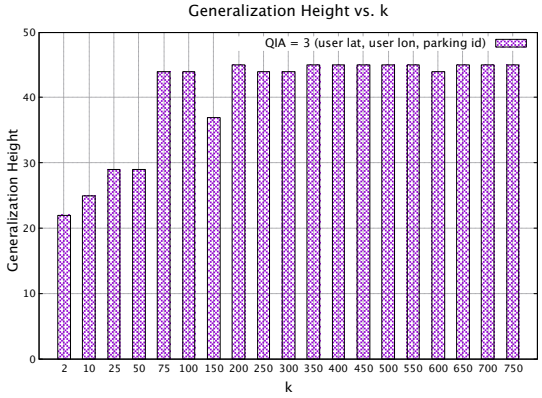
Figure 4.4c presents the generalization height applied by the anonymization algorithm



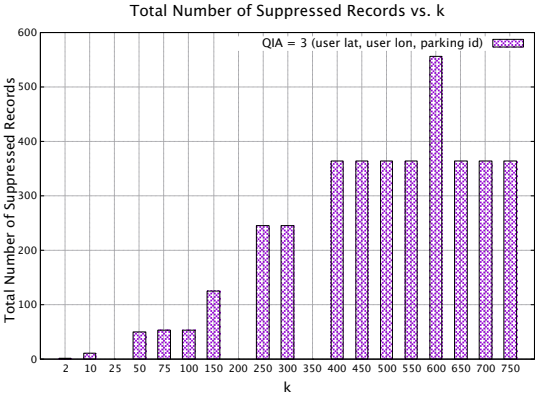
(a) Average group size



(b) Total number of groups

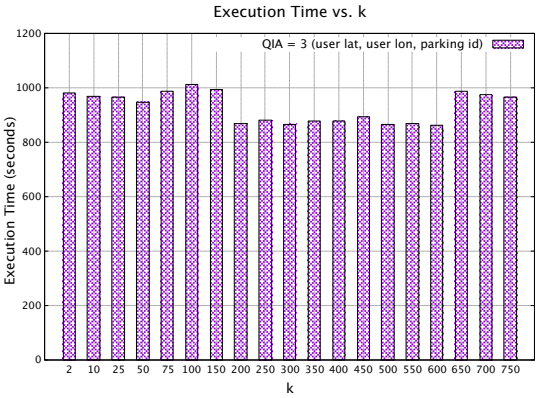


(c) Generalization height



(d) Number of suppressed records

(e) Discernibility cost



(f) Execution time

Figure 4.4 – Performance evaluation of k-anonymity when QIA = 3 (user latitude, user longitude, parking id)

from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, parking id). The generalization heights from k=2 to k=50 are much lower as compared to others because due to the repeated locations (user latitude and longitude) and parking spots, the anonymization algorithm was able to achieve indistinguishable records satifying the requirement of k at lower generalization heights. However, from k=75 to k=750, the generalization heights keep increasing because since the anonymization algorithm has to fulfill the indistinguishable of records equal to higher values of k, it achieved it by applying higher generalization heights. However, at k=150,250,300,600, the generalization heights are slightly lower as compared to their neighbors. This is because at these values of k, the anonymization algorithm was able to enhance the utility by applying slightly lower generalization height at the cost of suppressing the non anonymized records (N_{non_anon}) lower than k (i.e., $N_{non\_anon} < k$)(discussed next in Figure 4.4d). Overall, this result shows that the generalization height increases with the increasing values of k. Higher is the value of k, higher is the generalization height, and hence lower is the utility.

Figure 4.4d presents the number of suppressed records by the anonymization algorithm to generate an anonymized parking database from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, parking id). The records are suppressed to maximize the utility by applying as minimal generalization height as possible. The number of suppressed records are highest at k=600, i.e., around 200 more suppressed records than its neighbors, e.g., k=400-750. This is because the anonymization algorithm was able to apply one less generalization height than its neighbors, hence enhancing the utility.

Figure 4.4e presents the discernibility cost from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, parking id). The discernibility costs from k=2 to k=50 are comparably quite low because the anonymization algorithm was able to make clusters of smaller groups having lower groups sizes due to the repeated locations (user latitude and longitude) and parking spots. However, from k=75 to k=750, the discernibility cost keeps increasing because since the anonymization algorithm has to fulfill the indistinguishable of records equal to k, it ended up making bigger groups and suppressing the more number of records. However, at k=150,250,300,600, the discernibility costs are comparable lower in the neighborhood because at these values of k, the anonymization algorithm generated smaller groups sizes by applying slightly lower generalization height. Hence, smaller groups sizes results in lower discernibility cost. Overall, this result shows that the discernibility cost increases with the increasing values of k. Higher is the value of k, higher is the discernibility cost, and hence lower is the utility.

Finally, Figure 4.4f presents the execution time from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, parking id). For all values of k, the execution time is almost similar because the main execution time is consumed in making the generalizations

of the records. Since, there is no much difference in the generalization heights, therefore the execution time is similar.

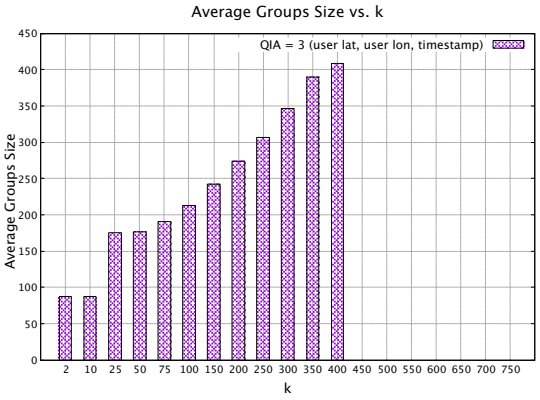
4.5.2.5 Analysis of Three Quasi-Identifier Attributes (Case 2)

In this section, we consider the second case and analyze the performance of k-anonymity when three attributes are selected as QIA, i.e., QIA = 3 (user latitude, user longitude, timestamp).

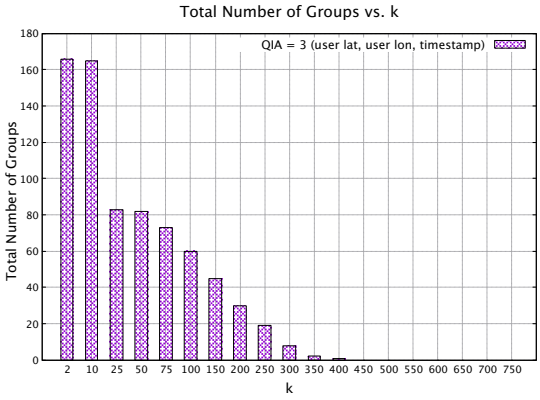
Figure 4.5a presents the average groups size for the second case generated by the anonymization algorithm from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, timestamp). When k=2 and k=10, the average groups size is similar and is around 90 because the anonymization algorithm was able to make smaller groups to fulfill the indistinguishability of records for lower values of k. The average groups size keeps increasing with the higher values of k because for the higher values of k, the anonymization algorithm has to maintain groups having sizes equal to or greater than the higher values of k. Another point to note here is that from k=450 to k=750, the average groups size is zero, this is because at this point (i.e., when $k >= 450$), the anonymization algorithm is unable to make an anonymized dataset from the original dataset. This result shows that the average groups size increases with the increasing values of k and the anonymization is not possible beyond $k >=$450. Higher is the value of k, higher is the group size, and hence lower is the utility.

Figure 4.5b presents the total number of groups generated by the anonymization algorithm from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, timestamp). In this result, the total number of groups are very high for k=2 and k=10 having a total number of groups around 160. This is because the anonymization algorithm has to maintain very smaller groups of indistinguished records, i.e., 2 and 10, hence it makes higher number of groups by creating smaller groups sizes. However, from k=25 to k=400, the total number of groups reduces drastically low because as the value of k gets higher, the anonymization algorithm has to ensure high indistinguishability of records, resulting in larger groups sizes and lower number of groups. Finally, from k=450 to k=750, the total number of groups is zero because the anonymization algorithm is unable to make an anonymized parking dataset from the original parking dataset by satisfying the requirement of k=450-700. This result shows that the total number of groups reduces with the increasing values of k. Higher is the value of k, lower is the total number of groups, and hence lower is the utility.

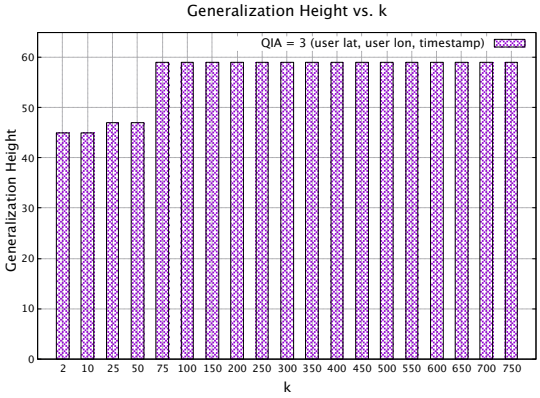
Figure 4.5c presents the generalization height applied by the anonymization algorithm from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, timestamp). The generalization height for k=2 to k=50 is almost similar, i.e., around 45 because the anonymization algorithm is able to make an anonymized parking dataset at this generalization height.



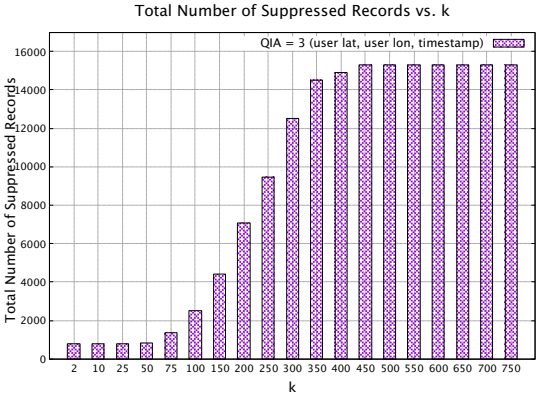
(a) Average group size



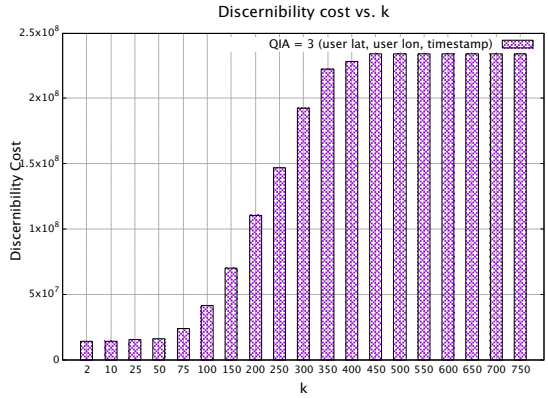
(b) Total number of groups



(c) Generalization height



(d) Number of suppressed records



(e) Discernibility cost

(f) Execution time

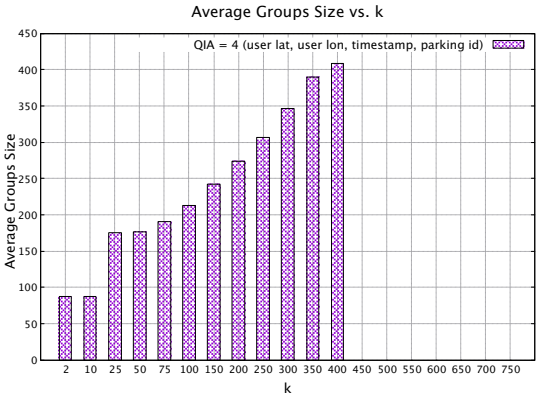
Figure 4.5 – Performance evaluation of k-anonymity when QIA = 3 (user latitude, user longitude, timestamp)

However, from k=75 to k=450, the generalization height is around 60 and is same because this is the highest generalization height possible. At this point, the anonymization algorithm has to suppress the records (presented in next Figure 4.5d) because there is no more higher generalization available.

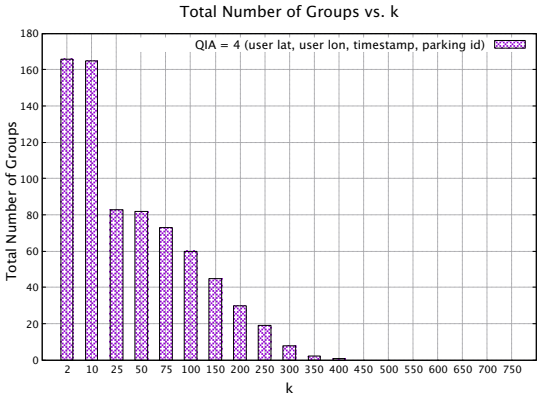
Figure 4.5d presents the number of suppressed records by the anonymization algorithm to generate an anonymized parking dataset from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, timestamp). The records are suppressed to maximize the utility by applying as minimal generalization height as possible. The anonymization algorithm tries to enhance the utility by applying the minimum possible generalizations and suppressing the records. The number of suppressed records from k=2 to k=50 are same because the anonymization algorithm is able to make anonymized parking dataset at the same generalization height. However, the number of suppressed records keep increasing from k=75. This is because at this point, the anonymization algorithm has applied the maximum possible generalizations (as discussed in previous figure). Hence, the only possibility is to suppress the records to achieve the k-anonymity. Here, note that from k=450 to k=750, the number of suppressed records is 15306 that is equal to the size of our original dataset. This is because the anonymization algorithm could not find an anonymization that fulfills the requirement of k=450 to k=750, hence it dropped all the records.

Figure 4.5e presents the discernibility cost from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, timestamp). The discernibility cost is another measure of utility that is dependent upon number of groups, size of groups and number of suppressed tables. For k=2 to k=50, the discernibility cost is very low and is same (results in a higher utility) because of similar number of suppressed records, and the similar ratio of number of groups to groups sizes. However, the discernibility cost keeps increasing from 75 to 450 because of the phenomena described in the results of number of suppressed records, i.e., it already has applied the maximum generalizations available, hence it suppressed the records (the only possible solution) and therefore incurs higher discernibility cost (and lower utility). The discernibility cost from k=450 to k=750 is the highest possible discernibility cost (and the worst utility) because the anonymization algorithm is unable to make anonymized dataset and suppressed all the records. Overall, this result shows that the discernibility cost increases with the increasing values of k. Higher is the value of k, higher is the discernibility cost, and hence lower is the utility.

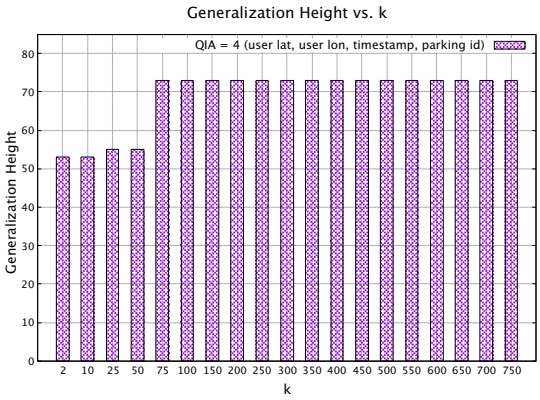
Finally, Figure 4.5f presents the execution time from k=2 to k=750 when QIA size = 3 (user latitude, user longitude, timestamp). For all values of k, the execution time is almost similar because the main execution time is consumed in making the generalizations of the records. Since, there is no much difference in the generalization heights, therefore the execution time is similar.



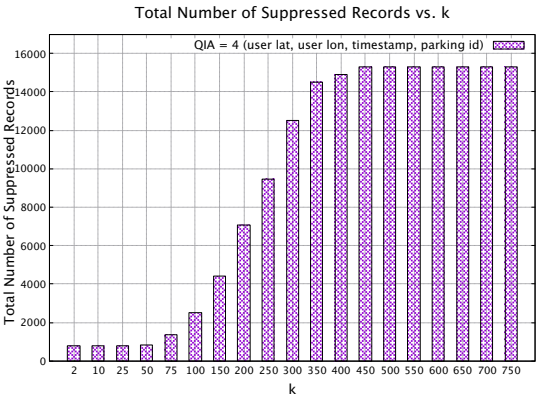
(a) Average group size



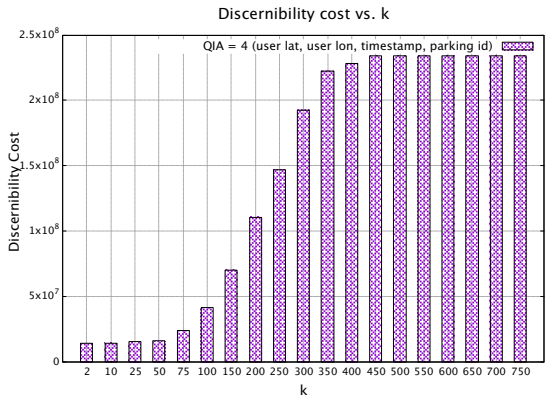
(b) Total number of groups



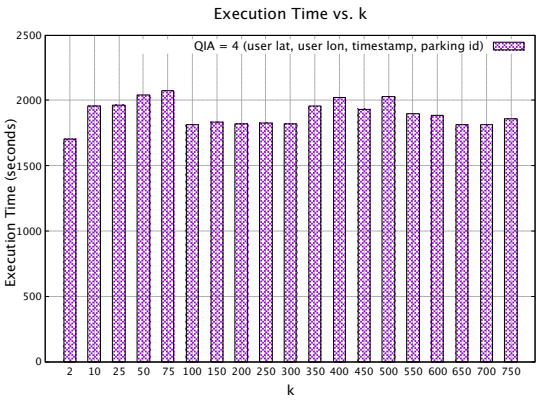
(c) Generalization height



(d) Number of suppressed records



(e) Discernibility cost



(f) Execution time

Figure 4.6 – Performance evaluation of k-anonymity when QIA = 4 (user latitude, user longitude, timestamp, parking id)

4.5.2.6 Analysis of Four Quasi-Identifier Attributes

In this section, we analyze the performance of k-anonymity when all the four attributes are selected as QIA, i.e., QIA = 4 (user latitude, user longitude, timestamp and parking id).

Figure 4.6a presents the average groups size generated by the anonymization algorithm from k=2 to k=750 when QIA size = 4 (user latitude, user longitude, timestamp, parking id). The average groups size in this figure is very similar to the average groups size in Figure 4.5a (the second case of QIA=3). This is because the most heterogeneous attribute is *timestamp* having 6242 distinct values, while the *parking id* attribute is not much heterogeneous as it has 265 distinct values that is much less diverse than the *timestamp* attribute. This is why, we learned in this result that if *timestamp* and *parking id* attributes are both selected as QIA, then *parking id* attribute does not have much significance. In other words, we can say that when we select *timestamp* as QIA in anonymization, it also covers *parking id* attribute by default. To summarize the result in Figure 4.6a, the average groups size is very small and same at k=2 and k=10, however, it keeps increasing from k=25 to k=400 because of fulfilling the requirement of higher indistinguishability of records to satisfy higher values of k. For $k >=$400, the average groups size is zero because no anonymization exists at these points. Overall, it shows that the average groups size increases with the increasing values of k and the anonymization is not possible beyond $k >=$450. Higher is the value of k, higher is the group size, and hence lower is the utility.

Figure 4.6b presents the total number of groups generated by the anonymization algorithm from k=2 to k=750 when QIA size = 4 (user latitude, user longitude, timestamp, parking id). The total number of groups in this figure is very similar to the total number of groups in Figure 4.5b (the second case of QIA=3). The reason is same as described above, i.e., the *timestamp* attribute is much more diverse than *parking id* attribute and hence, it already covers the *parking id* attribute in anonymization. To summarize the Figure 4.6b, the total number of groups are very high at k=2 and k=10 because of having lower requirement of indistinguishability of records. The total number of groups then keep reducing from k=25 to k=400 in order to fulfill the requirement of higher indistinguishability of records. From k=450 to k=750, there is no group because anonymization is not possible. Overall, the total number of groups reduces with the increasing values of k. Higher is the value of k, lower is the total number of groups, and hence lower is the utility.

Figure 4.6c presents the generalization height applied by the anonymization algorithm from k=2 to k=750 when QIA size = 4 (user latitude, user longitude, timestamp, parking id). The trend in this figure is similar to the trend in Figure 4.5c (the second case of QIA=3) but the values are different. The reason of similar trend is the same as discussed above, i.e., the *timestamp* attribute is much more diverse than *parking id* attribute and hence, it already covers the *parking id* attribute in anonymization. While, the reason of different

values of generalization height is that *parking id* attribute still has to be generalized to make it indistinguishable. Otherwise, it would not be possible to generate an anonymize dataset without generalizing the *parking id* attribute. To summarize the Figure 4.6c, the generalization height for k=2 to k=50 is almost similar because the anonymization algorithm is able to make an anonymized parking dataset at this generalization height. However, from k=75 to k=450, the generalization height is higher and same because this is the highest generalization height possible. At this point, the anonymization algorithm has to suppress the records (presented in next Figure 4.6d) because there is no more higher generalization available.

Figure 4.6d presents the number of suppressed records by the anonymization algorithm to generate an anonymized parking dataset from k=2 to k=750 when QIA size = 4 (user latitude, user longitude, timestamp, parking id). The total number of suppressed records in this figure is very similar to the total number of suppressed records in Figure 4.5d (the second case of QIA=3). The reason is similar as described above, i.e., the *timestamp* attribute is much more diverse than *parking id* attribute and hence, it already covers the *parking id* attribute in anonymization. To summarize, the number of suppressed records from k=2 to k=50 are same because the anonymization algorithm is able to make anonymized parking dataset at the same generalization height. However, the number of suppressed records keep increasing from k=75 because at this point, the anonymization algorithm has applied the maximum possible generalizations and follow the only possible solution of suppressing the records to achieve the k-anonymity. From k=450 to k=750, the number of suppressed records is 15306 that is equal to the size of our original dataset because the anonymization algorithm could not find an anonymization that fulfills the requirement of k=450 to k=750, hence it dropped all the records.

Figure 4.6e presents the discernibility cost from k=2 to k=750 when QIA size = 4 (user latitude, user longitude, timestamp, parking id). Similar to previous results, the discernibility cost in this figure is very similar to the discernibility cost in Figure 4.5e (the second case of QIA=3). The same explanation applies here as well. To summarize, for k=2 to k=50, the discernibility cost is very low and constant (results in a higher utility) because of similar number of suppressed records, and the similar ratio of number of groups to groups sizes. However, the discernibility cost keeps increasing from k=75 to k=450 because of the phenomena described in the results of number of suppressed records, i.e., it already has applied the maximum generalizations available, hence it suppressed the records (the only possible solution) and therefore incurs higher discernibility cost (and lower utility). Also, the discernibility cost increases with the increasing values of k. Higher is the value of k, higher is the discernibility cost, and hence lower is the utility.

Finally, Figure 4.6f presents the execution time from k=2 to k=750 when QIA size =

4 (user latitude, user longitude, timestamp, parking id). For all values of k, the execution time is almost similar and is around 2000 seconds because the main execution time is consumed in making the generalizations of the records. Since, there is no much difference in the generalization heights, therefore the execution time is similar.

4.5.2.7 Consolidated Analysis

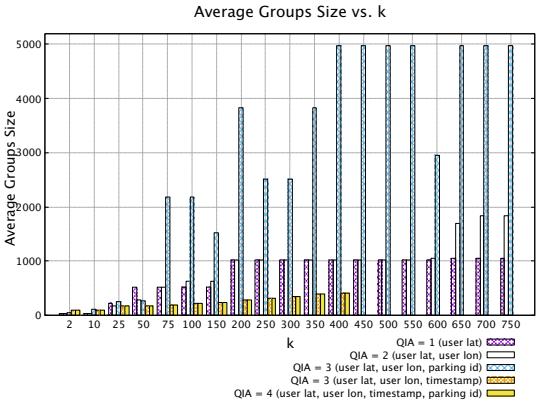
In this section, we present the consolidated results of all the previous analysis of k-anonymity with varying QIA sizes (i.e., QIA =1, 2, 3 (case 1 and case 2) and 4) in order to have a complete and consolidated view.

Figure 4.7a presents the average groups size generated by the anonymization algorithm from k=2 to k=750 for all the QIA sizes presented before. The result shows that the average groups size increases with the higher values of k. However, there is a surprising behaviour of the first case of QIA = 3 (user latitude, user longitude, parking id). It makes much average higher groups sizes as compared to other QIA sizes (i.e., QIA = 1, 2, 3 (case 2) and 4). This is because of non suppression of records, i.e., QIA = 3 (case 1) does not suppress the records and hence, it results in larger groups sizes, while QIA = 3 (case 2) and QIA = 4 suppress the records, causing smaller groups sizes. The main insight is that the groups sizes increases with the increasing values of k, as well as with the increasing QIA sizes at a limit when no suppression is made. Overall, the average groups size increases with the increasing values of k. Higher is the value of k, higher is the groups size, and hence lower is the utility.

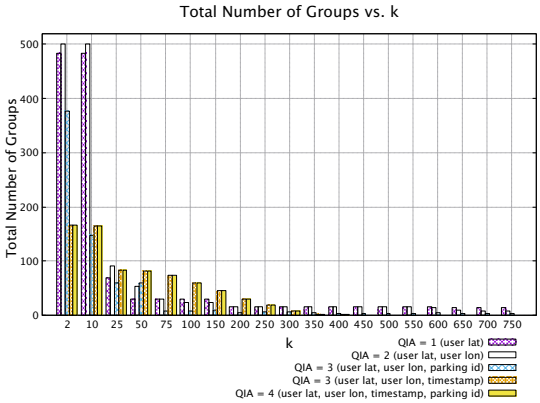
Figure 4.7b presents the total numbers of groups generated by the anonymization algorithm from k=2 to k=750 for all the QIA sizes presented before. There are two insights gained from this result. Firstly, the total number of groups reduces with the increasing values of k. Secondly, the total number of groups also reduces with the increasing QIA sizes. Higher is the value of k and QIA sizes, lower is the number of groups, and hence lower is the utility.

Figure 4.7c presents the generalization heights applied by the anonymization algorithm from k=2 to k=750 for all the QIA sizes presented before. There are three insights gained from this result. Firstly, the generalization height increases with the higher values of k. Secondly, the generalization height also increases with the higher QIA sizes. Thirdly, the generalization heights increases until k=75. After k=75, the generalization height is same because no more higher generalization is available. Higher is the value of k and QIA sizes, higher is the generalization height, and hence lower is the utility.

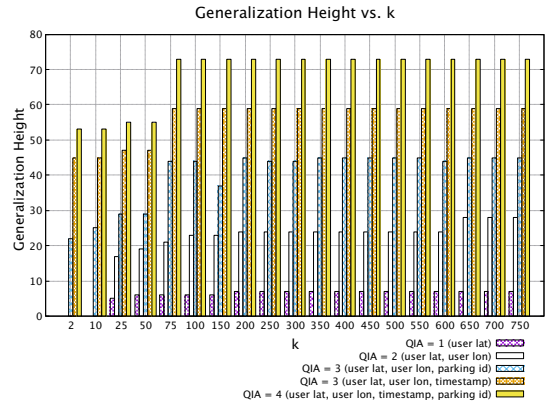
Figure 4.7d presents the number of suppressed records by the anonymization algorithm from k=2 to k=750 for all the QIA sizes presented before. There are four insights gained from this result. Firstly, the number of suppressed records increases with the higher values



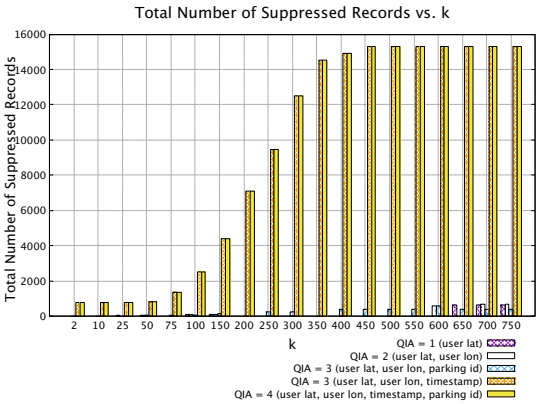
(a) Average group size



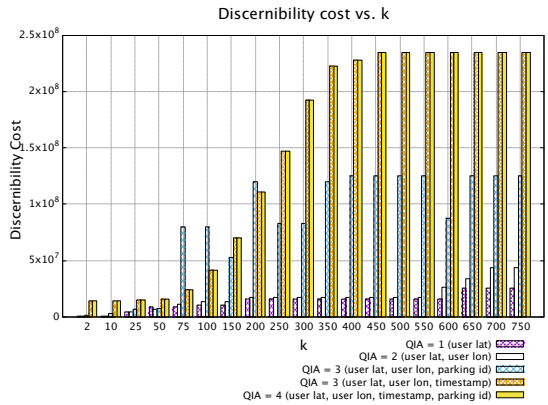
(b) Total number of groups



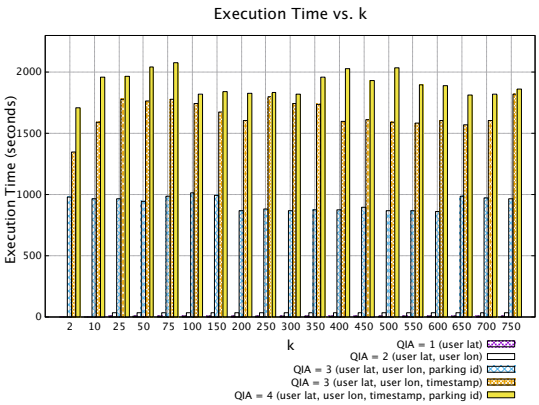
(c) Generalization height



(d) Number of suppressed records



(e) Discernibility cost



(f) Execution time

Figure 4.7 – Performance evaluation of k-anonymity for all QIA = 1, 2, 3, 4

of k. Secondly, the number of suppressed records also increases with the higher QIA sizes. Thirdly, the number of suppressed records for QIA = 3 (case 2) and QIA = 4 increases until k=400. From k >=450, the number of suppressed records is equal to the size of the original dataset, i.e., no anonymization is made. Fourthly, the number of suppressed records by QIA = 3 (case 2) and QIA = 4 is same, it means that they exhibit the same behaviour. Higher is the value of k and QIA sizes, higher is the number of suppressed records, and hence lower is the utility.

Figure 4.7e presents the number of discernibility cost incurred by the anonymization algorithm from k=2 to k=750 for all the QIA sizes presented before. There are four insights gained from this result. Firstly, the discernibility cost increases with the higher values of k. Secondly, the discernibility cost also increases with the higher QIA sizes. Thirdly, the discernibility cost for QIA = 3 (case 2) and QIA = 4 increases until k=400. From k >=450, the discernibility is equal and is the highest possible discernibility cost because all the records in the datatset are suppressed. Fourthly, the number of suppressed records by QIA = 3 (case 2) and QIA = 4 is same and they exhibit the same behaviour. Higher is the value of k and QIA sizes, higher is the number of discernibility cost, and hence lower is the utility.

Finally, figure 4.7f presents the execution time by the anonymization algorithm from k=2 to k=750 for all the QIA sizes presented before. There are two insights gained from this result. Firstly, the execution time increases with the higher values of k. Secondly, for each QIA size, the execution time is almost similar. It means that the execution time is mainly dependent upon the size of QIA, or in other words, it depends upon the level of generalizations that need to be applied to construct an anonymized parking dataset.

4.5.3 Evaluation of Differential Privacy

We evaluated differential privacy for the numeric query type as discussed in Section 4.4.2 by generating 1000 random queries. For each query, the user' current location, timestamp, parking spot and rating are randomly selected from the parking dataset. Subsequently, a time range is randomly selected between 1 to 30 days and for each location, we consider nearby locations within 5km radius. We evaluate differential privacy using different values of privacy budget ϵ from ϵ=0.1 to ϵ=1.0. The sensitivity $\triangle f$ defines the number of records that gets affected with the addition or removal of a user. As in our parking dataset, a user may appear multiple times but we do not know the exact number of appearences, therefore, we analyze the effects of different values of sensitivity $\triangle f$ values from 1 to 5, i.e., the appearence or removal of a user in the dataset affects 1 to 5 records, respectively.

4.5.3.1 Performance Metrics

We evaluate the accuracy and privacy of differential privacy by using two widely-adopted metrics:

- *Mean Absolute Error (MAE)* measures the average amount of errors. It is an average over the number of queries of the absolute differences between the actual query result and noisy result. It measures the privacy and the utility. If the MAE is high, the difference between the actual and noisy query results is high that makes the privacy stronger but reduces the utility. While if the MAE is low, the difference between the actual and noisy query results is low that enhances the utility but weakens the privacy. MAE has been widely-adopted in the literature for evaluating differential privacy [87–89]. It is defined as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|r_{a,i} - r_{n,i}| \tag{4.4}$$

  where N is the total number of queries, $r_{a,i}$ is the actual response of query i, $r_{n,i}$ is the noisy response of query i.

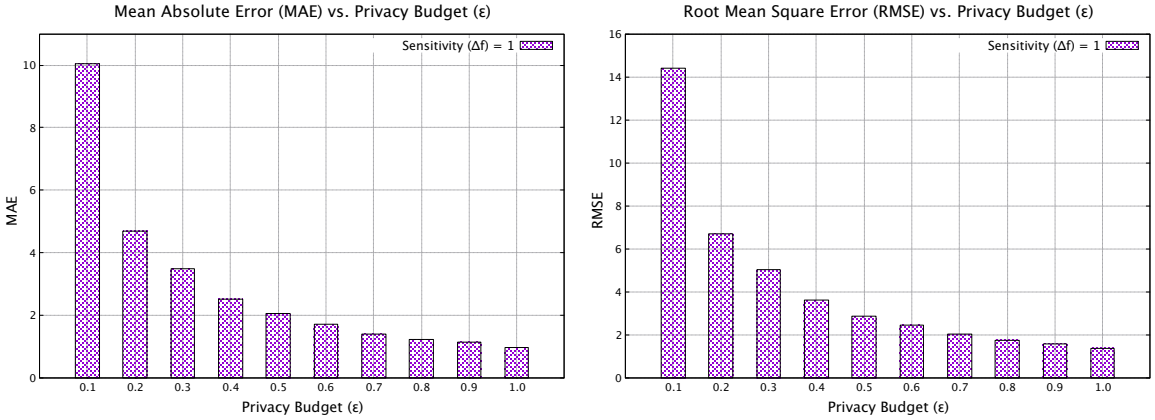
- *Root Mean Square Error (RMSE)* is a quadratic scoring function and it also measures the average amount of errors. It is the square root of the average of squared differences between the actual query result and noisy result. It measures the privacy and the utility. Similar to MAE, if the RMSE is high, the difference between the actual and noisy query results is high that makes the privacy stronger but reduces the utility. While if the RMSE is low, the difference between the actual and noisy query results is low that enhances the utility but weakens the privacy. It has been widely-adopted in the literature for evaluating differential privacy [87–90]. It is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(r_{a,i} - r_{n,i})^2}{N}} \tag{4.5}$$

  where N is the total number of queries, $r_{a,i}$ is the actual response of query i, $r_{n,i}$ is the noisy response of query i.

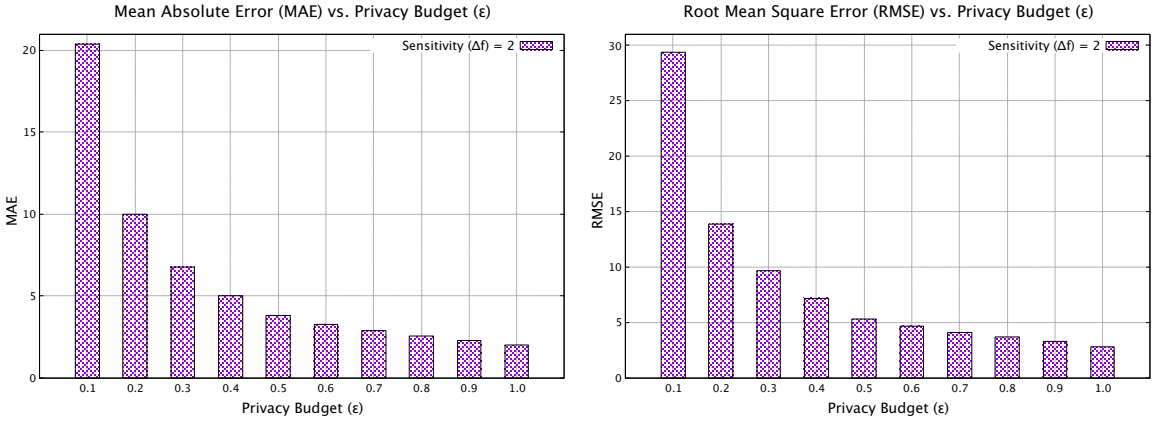
4.5.3.2 Analysis of Individual Sensitivities

In this section, we analyze the performance of differential privacy in terms of accuracy and privacy using MAE and RMSE for privacy budget ϵ=0.1 to ϵ=1.0 by analyzing each sensitivity ($\triangle f$) individually.



(a) MAE (b) RMSE

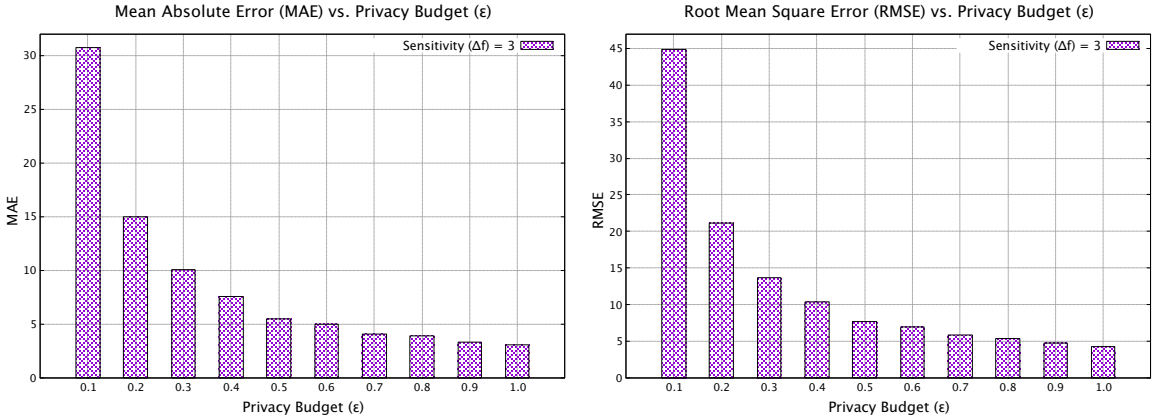
Figure 4.8 – MAE and RMSE for varying privacy budget ϵ when sensitivity ($\triangle f$)=1



(a) MAE (b) RMSE

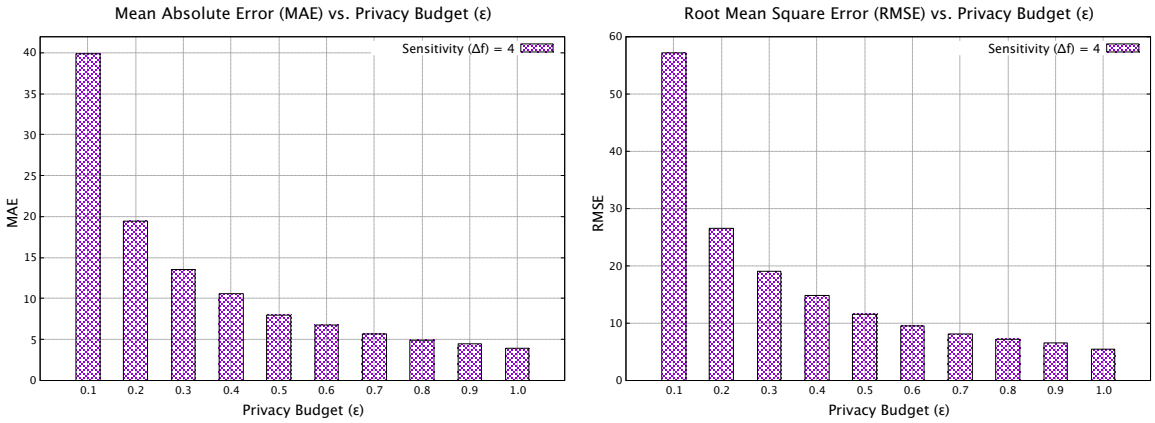
Figure 4.9 – MAE and RMSE for varying privacy budget ϵ when sensitivity ($\triangle f$)=2

Figure 4.8 presents MAE and RMSE for privacy budget ϵ=0.1 to ϵ=1.0 when sensitivity $\triangle f$=1 (i.e., the addition or removal of a user affects one record in the parking dataset). It shows that when ϵ=0.1, the MAE and RMSE are very high, i.e., 10 and 14, respectively because ϵ=0.1 guarantees the highest privacy, however at the cost of the worst utility. However, as ϵ keeps increasing, the MAE and RMSE keeps reducing drastically and at ϵ=1.0, both MAE and RMSE are close to zero. It means that we have the highest utility at this point, however there is no privacy because the noisy results are very similar to the actual results. Overall, the MAE and RMSE reduces with the increasing values of privacy budget ϵ. Higher is the privacy budget ϵ, higher is the privacy but lower is the utility. Inversely, lower is the privacy budget ϵ, lower is the privacy but higher is the utility.



(a) MAE (b) RMSE

Figure 4.10 – MAE and RMSE for varying privacy budget ϵ when sensitivity ($\triangle f$)=3

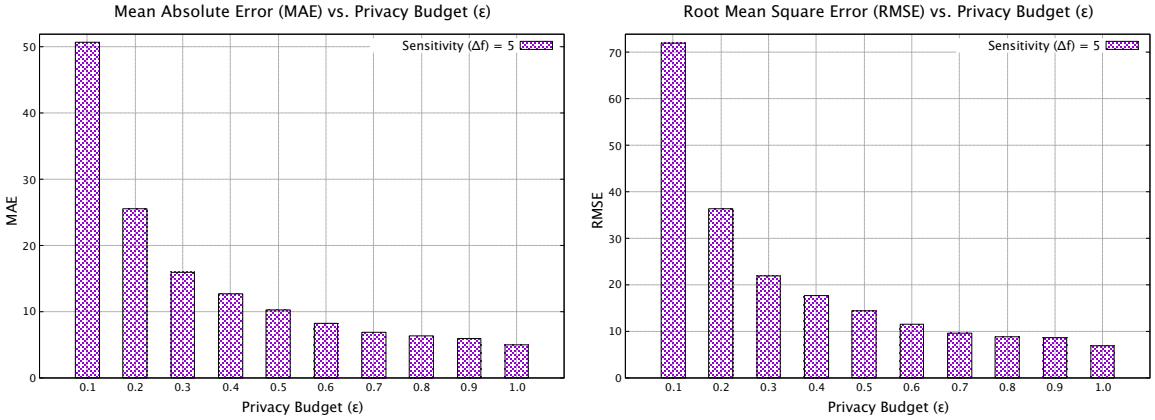


(a) MAE (b) RMSE

Figure 4.11 – MAE and RMSE for varying privacy budget ϵ when sensitivity ($\triangle f$)=4

Figure 4.9 presents MAE and RMSE for privacy budget ϵ=0.1 to ϵ=1.0 when sensitivity $\triangle f$=2 (i.e., the addition or removal of a user affects two records in the parking dataset). The trend is very similar to Figure 4.8 when $\triangle f$=1, however, at ϵ=0.1 the MAE and RMSE are almost double. This is because when $\triangle f$=2, since an additional or removal of a user affects at least two records, therefore we have two times (i.e., 2×) MAE and RMSE than at $\triangle f$=1. However, the MAE and RMSE at $\triangle f$=2 keeps reducing with the increasing values of privacy budget ϵ and at ϵ=1.0, the MAE and RMSE are almost similar to those at $\triangle f$=1.

Similarly, figures 4.10, 4.11 and 4.12 present MAE and RMSE for privacy budget ϵ=0.1 to ϵ=1.0 when sensitivity $\triangle f$=3, $\triangle f$=4 and $\triangle f$=5 (i.e., the addition or removal of a user



(a) MAE (b) RMSE

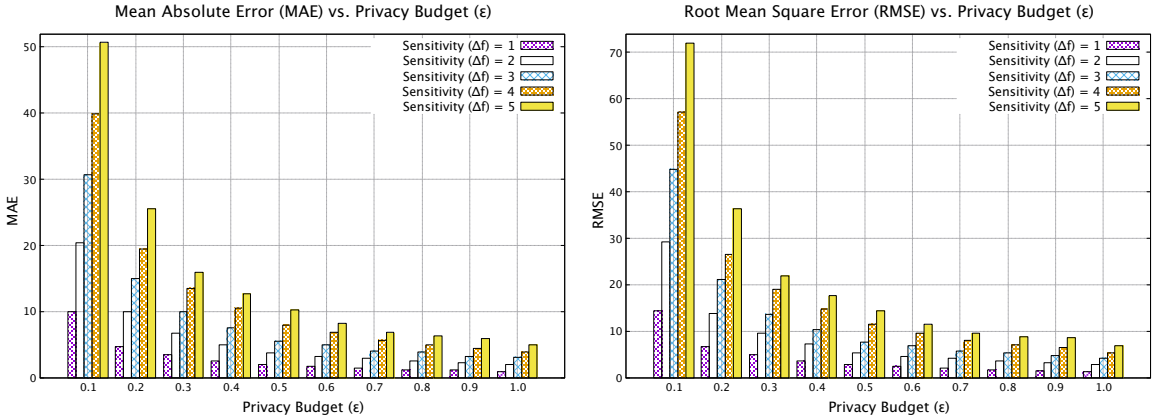
Figure 4.12 – MAE and RMSE for varying privacy budget ϵ when sensitivity ($\triangle f$)=5

affects three, four and five records in the parking dataset), respectively. The trends are similar as discussed before. Initially, at ϵ=0.1, the MAE and RMSE are three, four and five times (3×, 4× and 5×) for $\triangle f$=3, 4, 5, respectively, as compared to MAE and RMSE for $\triangle f$=1. However, as ϵ increases and reaches towards 1.0, the MAE and RMSE get close to zero. Overall, the MAE and RMSE are very high at privacy budget ϵ=0.1, which give very strong privacy, however at the cost of the worst utility. The MAE and RMSE keep reducing with the increasing values of privacy budget ϵ and at ϵ=1.0, the MAE and RMSE are very similar for all sensitivities $\triangle f$=3, 4, 5.

4.5.3.3 Consolidated Results

In this section, we present the consolidated results of all the previous analysis of differential privacy with all previously discussed sensitivity $\triangle f$ values (i.e., $\triangle f$=1, 2, 3, 4, 5) in order to have a complete and consolidated view.

Figure 4.13 presents the consolidated results of MAE and RMSE for all sensitivity values $\triangle f$=1, 2, 3, 4, 5 for privacy budget ϵ=0.1 to ϵ=1.0. These results provide two insights. Firstly, initially at ϵ=0.1, the MAE and RMSE are very high for all sensitivity $\triangle f$ values (that provides very strong privacy but no utility), however, as privacy budget ϵ keeps getting closer to 1.0, the MAE and RMSE are becoming similar and getting close to zero (that provides very high utility but no privacy). Secondly, as the sensitivity $\triangle f$ value increases, the MAE and RMSE also gets higher to many folds but then they get closer to other sensitivity $\triangle f$ values when privacy budget ϵ values gets higher. Overall, the MAE and RMSE reduces with the increasing values of privacy budget ϵ. Higher is the privacy budget ϵ, higher is the privacy but lower is the utility. Inversely, lower is the privacy budget



(a) MAE (b) RMSE

Figure 4.13 – MAE and RMSE for varying privacy budget ϵ when sensitivity ($\triangle f$)=1, 2, 3, 4, 5

ϵ, lower is the privacy but higher is the utility.

4.6 Summary and Discussion

To summarize and conclude, this chapter presented the second contribution of this thesis of privacy preservation for smart parking system. It protects the privacy of users in the historic parking dataset against third-party parking recommender that is untrusted or semi-trusted or one could not identify its trustworthiness. The parking dataset is shared with the parking recommender system to provide efficient and personalized recommendations of parking spots based on users' past experience. However, since it contains the past history of parking information of users, it breaches the privacy of the user because the parking recommender (or an adversary) can track the routine and mobility patterns of users by analyzing such parking dataset. In this study, we preserve the privacy of users while sharing their parking information (that contains their private behavior and mobility pattern) with a parking recommender system through two well-known privacy preservation techniques of anonymization and perturbation: k-anonymity and differential privacy. We discuss the system and adversary models, discussion and applicability of k-anonymity and differential privacy on parking dataset and then performed extensive experiments to study the privacy and utility of k-anonymity and differential privacy on our parking dataset. The proposed implementation enables users to receive personalized and efficient recommendations of parking spots based on their past parking experience while protecting their privacy. Experimental results evaluated the utility and privacy of both privacy preservation

techniques.

From our experiments, we found that k-anonymity is suitable for smaller values of k and for lower QIA sizes. When k is much higher, the utility is very low. Specifically, when $k >=450$ and QIA = 3 (user latitude, user longitude, timestamp) or QIA size = 4 (user latitude, user longitude, timestamp, parking id), the k-anonymity is unable to generate an anonymized parking dataset because the requirement of k does not get fulfilled. Also, the behaviour of QIA = 3 (user latitude, user longitude, timestamp) and QIA = 4 is very similar because *timestamp* attribute is much more diverse than *parking id* attribute and therefore, it covers *parking id* in anonymization by default. For differential privacy, we found that when the privacy budget ϵ is very low (e.g., ϵ=0.1), the privacy is very strong, however the utility is worse. As the privacy budget ϵ keeps getting higher, the utility starts improving, however at the cost of weakening the privacy. Additionally, the sensitivity $\triangle f$ also affects the privacy and utility. The higher is the sensitivity $\triangle f$ value, stronger is the privacy but lower is the utility.

In the next chapter, we presents two frameworks for cross-domain recommendation services and application-to-application communications using social IoT across smart cities/IoT applications.

Chapter 5

Frameworks of Cross-Domain Recommendation Services using Social IoT

Contents

5.1 Introduction

The main focus of this chapter is to propose two frameworks for cross-domain recommendation services and communications across smart cities applications using Social IoT (SIoT). More specifically, many of the smart cities applications available today have been developed in a vertical manner by focusing on a specific scenario or use case without considering data exchange and reuse with other IoT applications. This very specific focus results in poor service because of the lack of integration of different data and hence the interoperability in the smart cities data and systems. However, if smart cities applications could collaborate by communicating, exchanging and reusing each other's data, opportunities for new value-added and more efficient recommendation services could be generated. In this chapter, we provide two frameworks for the recommendation services and application-to-application communications using SIoT in smart cities. The first framework (in Section 5.3) proposes to use SIoT for cross-domain recommendation services across smart cities applications, while the second framework (in Section 5.4) proposes another type of communication for the SIoT at a global level that enables application-to-application communication, known as social cross-domain IoT (SCDIoT).

5.2 Related Works

In the last few years, a large number of research studies have been focused on the SIoT, however, to the best of our knowledge, none of them has focused neither on SIoT-based recommendation services across IoT/smart cities application nor on social cross-domain IoT. We present the state-of-the-art in this section.

One of the early proposals for establishing social relationships among objects is presented in [91]. This work focused on establishing temporary relationships using wireless devices, specifically wireless sensor nodes, and on how the owners of sensors can control this relationship establishment. However, this work was performed in 2001 and at that time, the IoT, smart cities and social networks were still in their infancy.

The authors in [92] distinguished the things connected to the Internet with the things involved in social networks, which they termed as the neologism Blogject (i.e., objects that blog). Another concept, Embodied Microblogging (EM), was presented in [93]. EM introduces two new roles that augment daily life objects rather than focusing on people-to-thing or thing-to-thing paradigms. These two roles are mediating people-to-people communication and supporting new procedures for considering the noticing and noticeable activities in daily life. The authors in [94] proposed a concept in which objects are able to participate in conversations previously reserved for humans. These objects are context-aware, and hence are able to create a networking infrastructure based on the dissemination of information,

rather than information on the objects themselves.

Recently, integrating the two worlds of the IoT and social networks is proposed in the literature [95–97]. The authors in [95] visualize the future of the Internet as ubiquitous IoT architecture, which is similar to a social organization framework (SOF) model and provides an overview of future IoT network structure. However, this work does not exploit social network features into the IoT [23]. The authors in [96] suggest that as things are involved together with humans in the network, the social network can be more meaningful if it is built on the IoT by investigating the relationships of IoT objects. The main convergence of social networks and the IoT is also introduced in [97], in which the social network is a social network of humans that is used by things as an infrastructure for service discovery, access and advertisement. In this work, a person can share the services offered by his smart objects with his friends as well as sharing their things (or devices).

Another work on the SIoT investigates the integration of social networks and the IoT with some sample applications [98]. However, it neither discusses how social relationships can be established by objects nor provides any solution for the required protocols and architectures. In [99], the authors investigate the social attributes or relations among mobile nodes by considering two parameters, i.e., an interaction factor and a discount factor, as well as investigating the behavior of mobile nodes by applying social networks. However, their approach assumes a one-to-one relation between objects and humans, whereas in the IoT, many objects are associated with a single human, and hence a large number of objects would not be considered in this work [23]. Some work has been done on recommendation services in smart homes to allow smart assistance [100, 101]. The humans' current situation, needs, preferences and habits are stored in repositories which are used by recommendation systems. The recommendation systems adapt themselves according to these humans' preferences. However, [100] does not consider context-awareness (i.e., humans long-term and short terms goals and preferences, events, localization information) from social networks to identify the current context of the user and provide intelligent recommendations. The work in [101] does address this issue, but it is specifically designed for task-oriented recommendations in smart homes.

Kim et al., [102] propose "Socialite", an end-user programming tool for the SIoT, by exploiting semantic technologies. The authors identified eight desired features of the SIoT through an online survey and clustered them into four rule categories which can be programmed by end users. These rules were then used to reason about devices and people in their social circles to support automated decisions at runtime. Socialite uses semantic technologies for knowledge representation and for encapsulating the heterogeneity of devices belonging to different manufacturers. Additionally, Socialite's rules allow for social relationships and collaboration by sharing information and configurations among social circles

(e.g., friends).

Girau et al. [103] propose "Lysis", a cloud-based platform for the IoT using SIoT. Lysis offers three main features: objects have social relationships and they behave like autonomous social agents, it exploits PaaS (Platform as a Service) and considers reusability at various layers, and it allows users to have full control over their data. In Lysis, the SIoT is mainly exploited for its first feature, which enables objects to build social relationships in an autonomous manner, offering the advantages of enhancing both network scalability and information discovery.

Colom et al., [104] propose an IoT framework for the collaborative building of behavioral models by using the SIoT. The SIoT is used to support collaborative applications and to build social dimensions by allowing the addition of computing resources by the user without affecting other ongoing activities offered by IoT devices.

Lee et al., [105] propose a game theory-based vulnerability quantification method by using an attack tree for SIoT. This is consisted of three steps: game strategy modeling, cost-impact analysis and payoff calculation. They also present a case study of an SIoT-based network environment. Their approach can serve as a reference for developing a safer SIoT system.

Other works on SIoT focus on modeling and optimization of features selection in Big Data-based SIoT [106], routing protocols based on source location protection [107], robustness management for customization manufacturing [108], SWARM-based data delivery [109], general overviews of 5G-enabled devices [110], IoT platforms for SIoT [111] and recommendation services [1].

Neither of these works considers recommendation services across smart cities/IoT applications that are developed in a vertical and standalone manner nor on social cross-domain application-to-application communication.

5.3 Social IoT for Recommendation Services across IoT Applications

In this study, we propose a framework on the exploitation of SIoT for recommendation services across smart cities/IoT applications, as well as present an application scenario and implementation challenges.

5.3.1 Introduction

The smart cities applications are generally developed in a standalone and vertical manner, i.e., each application is developed for a certain scenario, such as smart parking, traffic monitoring, smart grid, building automation and e-health [112]. Such smart cities applications generally do not share and use other system's data for recommendation services, leading to an inefficient exploitation of the services offered by other applications. Such recommendation services could be achieved with the help of the SIoT by using the data from multiple smart cities applications, thereby enhancing the services and performance of each IoT application.

The convergence of the IoT with social networks is becoming a reality due to the increasing awareness that the SIoT can realize many of the future implications of intelligent devices used in our daily life. The SIoT has witnessed a shift in the IoT from a network of connected smart objects to a network of social objects. The application of social networking to the IoT (i.e., SIoT) helps to guarantee network navigability by shaping the structures as required for the effective discovery of objects and services, helps to establish trust for interactions among things (as with friends), and reuses the models designed for social networks for solving IoT issues related to networks of interconnected objects [23].

In the SIoT, objects can have social relationships between people-and-things and between things-and-things that behave like social circles. It builds profiles on the basis of various IoT applications' data. Such profiles are exchanged within a SIoT network that can be accessible to other IoT applications. In this manner, SIoT networks provide recommendation services for improving the performance of IoT applications by sharing and using other IoT applications' data. Additionally, the profiles built by SIoT networks can also help a single IoT application by looking for similar conditions that have been addressed in the past for the same IoT application.

The SIoT differs from social networks and from the IoT in three main aspects. Firstly, the SIoT establishes and exploits social relationships among things, rather than only among owners or humans. Humans can be involved for mediation, but the key roles are performed by things. Secondly, things can discover resources and services themselves through social relationships to the IoT, which provides a distributed solution and reduces human efforts.

Thirdly, the SIoT is a platform for social networking services (SNSs) which enables social networking between objects [113].

In this study, we propose a concept for the exploitation of the SIoT for recommendation services across smart cities and IoT applications. We present a sample application scenario which highlights how the SIoT can have social relationships between people-and-things and between things-and-things in order to offer recommendation services. Some implementation challenges for this concept are also presented.

5.3.2 Proposed Framework

In this section, we present the proposed framework of exploiting SIoT for recommendation services across smart cities and IoT applications. The architecture of this concept is presented in Figure 5.1 which is comprised of perception layer, network layer, interoperability layer, SIoT recommendation system and IoT applications. The SIoT perception layer is responsible for sensing and collecting information from IoT devices. It consists of various heterogeneous devices, such as sensors and actuators, RFIDs, smartphones and cameras. After collecting the information, the IoT devices establish social relationships and friendship circles among themselves using SIoT technique. Subsequently, the collected sensing and friendship circles information are forwarded to network layer in order to utilize this information by IoT applications. The network layer is composed of various telecommunication networks (e.g., private wireless networks, public mobile networks and satellite networks) and the Internet. It maps IoT devices' data received from the perception layer to the telecommunication protocols, and forwards it to the upper layer for processing and to be converted into useful information for the realization of various IoT applications.

Generally, IoT applications are developed in a vertical manner with different structures and semantics. The SIoT recommendation system requires data sharing among IoT applications; providing recommendation services based on this shared data. Interoperability is required for data sharing among various IoT applications due to the different semantics of each IoT application. Nowadays, there are two widely used IoT interoperability platforms: oneM2M [56] and FIWARE [55]. IoT applications can thus be developed using either oneM2M or FIWARE or both, or semantic technologies can be applied to achieve interoperability and data sharing among IoT applications. In [114], the authors worked on creating a semantic service for the SIoT. Once interoperability is achieved, IoT applications' data can easily be shared with an SIoT recommendation system. The SIoT recommendation system can receive data from the interoperability layer, as well as applications data from IoT applications. Since IoT applications contain SIoT data (friendship circles) of IoT devices received from SIoT perception layer, the social IoT recommendation system uses this SIoT data to build and maintain social relationships and profiles between people-and-

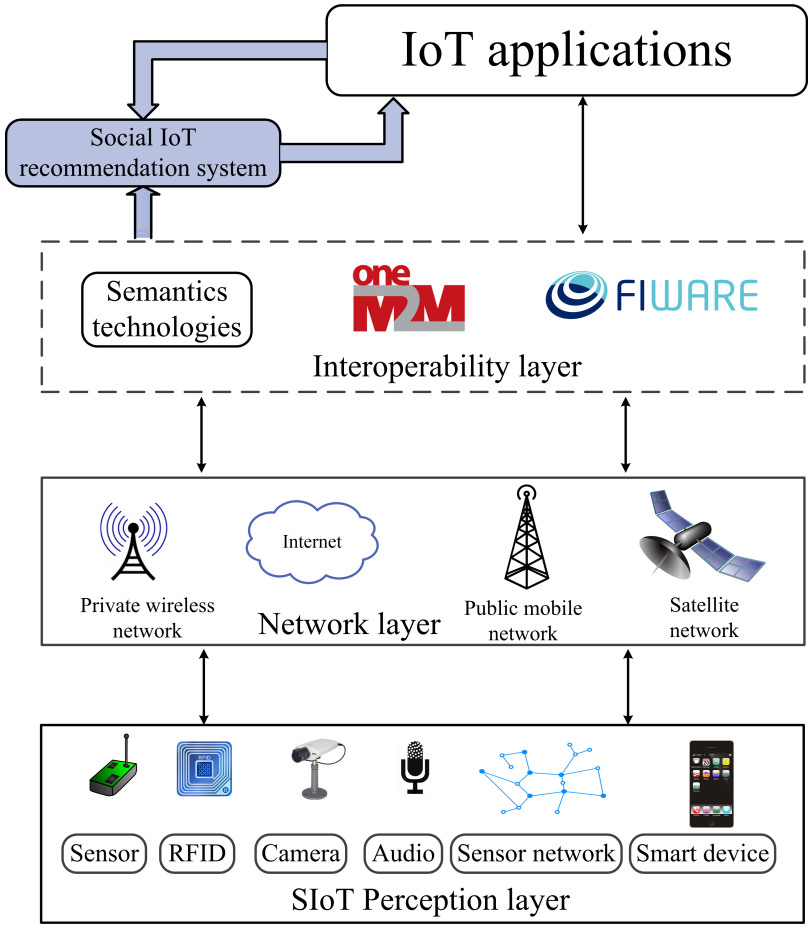


Figure 5.1 – Proposed concept of exploiting the SIoT for recommendation services across IoT applications.

things, and between things-and-things that behave like social circles. These profiles are used for recommendation services among various IoT applications.

A sample application scenario of how the SIoT can provide recommendation services among various IoT applications based on their shared data is presented next.

5.3.3 A Sample Application Scenario

The availability of social relationships between things-and-things and between people-and-things interconnected through a SIoT can help several IoT applications to benefit from other IoT applications' data. The main benefit of SIoT over traditional IoT is that smart objects can establish social relationships among themselves in an autonomous and ad hoc manner. This helps smart objects to learn about other (homogeneous and heterogeneous) objects in a distributed way, and subsequently take decisions/actions based on this informa-

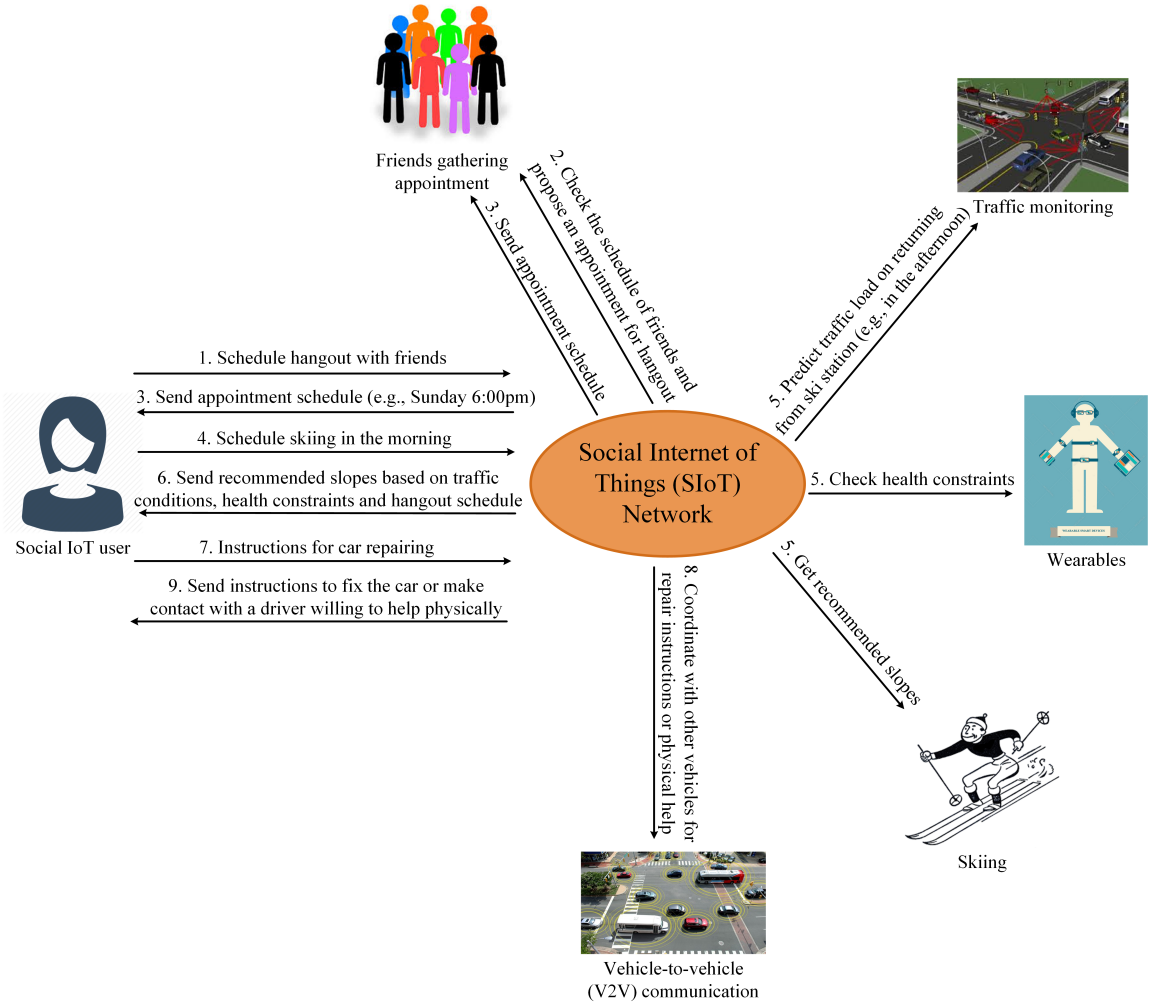


Figure 5.2 – A sample application scenario in which different applications benefit from the SIoT by using other application's data.

tion. Additionally, SIoT improves the scalability when the network is composed of a large number of objects. Figure 5.2 presents a sample application scenario in which different applications benefit from the SIoT by using other applications' data. In this figure, there are various IoT applications (i.e., skiing, friendship gathering appointment, traffic monitoring, wearables and vehicle-to-vehicle communication). The interoperability layer (shown in Figure 5.1) provides interoperability between them so that social IoT recommendation system can use the data of these heterogeneous IoT applications and subsequently provides recommendation services to the user.

Let's consider a girl named Maria, who is a user of a SIoT network and a SIoT recommendation system. Maria is planning to get together with her friends over the weekend

and she wants to set an appointment which should be feasible for all her friends based on their availability. For this purpose, she initiates an appointment using her system which is based on a SIoT network that contains the profiles of Maria and her friends. It is important to note that since Maria is using SIoT network, her IoT devices already have maintained social relationships with other IoT devices using SIoT perception layer as shown in Figure 5.1. Her system coordinates with her friends' scheduling systems and proposes an appointment time to her and the other friends based on their availability. When Maria and her friends confirm this appointment, the system sets this appointment and sends invitation to all the friends. Suppose the appointment is set on Sunday at 6:00pm. On the same day, Maria also wants to go for skiing in the morning but she needs to return before 6:00pm to hang out with her friends. Accordingly, her system interacts with a traffic monitoring system to predict the traffic load during her return in the afternoon, and based on this prediction, interacts with the smart skiing system by means of a SIoT network to suggest recommended slopes, so that Maria can enjoy skiing and still return on time. Moreover, Maria has some health problems and her health profile already exists on her SIoT network through wearable devices attached to her body. In this manner, a SIoT network can recommend ski slopes to Maria based on her health condition. Based on these recommendations by SIoT networks, Maria is able to do ski according to her health conditions and hang out with her friends at the scheduled time.

On her return from skiing, Maria's car got a problem which appears to be difficult for her to fix on her own. Thanks to the SIoT, the junction box in her car has embedded sensors which collect information and build a profile of her car and of this problem. This profile is then shared with SIoT networks that look for similar problems which have been addressed before by other similar cars in the network. Subsequently, the SIoT network either recommends the suggested corrective actions to Maria so that she can fix the problem by herself, or it fixes the problem itself by coordinating with sensors and actuators embedded in the network. Vehicle-to-vehicle (V2V) communications also enables coordination among vehicles, and so the SIoT network could search for a similar problem that has been fixed by other cars. If the SIoT network finds such a car in a close proxity of Maria, it could request such car's driver to visit Maria and help her to fix her problem. This option also promotes social relationships and community help.

5.3.4 Implementation Challenges

The framework we proposed is a conceptual framework and there are some implementation challenges for the realization of SIoT-based recommendation services across IoT applications. In this section, we discuss such challenges.

5.3.4.1 Interoperability

The lack of interoperability among IoT applications is one of the major challenge for the realization of a SIoT-based recommendation system, as it restricts data sharing among various IoT applications. IoT applications are generally developed in a vertical and standalone manner. Each IoT application has different data structures and semantics, making it difficult for one IoT application's data to be used and understood by another IoT application. There is some ongoing work on interoperability in the IoT; the two main reference models for IoT interoperability are oneM2M [56] and FIWARE [55]. There is a pressing need to consider interoperability in the IoT so that SIoT recommendation systems can utilize IoT applications' data to provide recommendation services. Furthermore, there are two EU projects, WISE-IoT [33] and FIESTA-IoT [115], that work on IoT interoperability and they can be good references for achieving this interoperability challenge.

5.3.4.2 Social Network Management

A SIoT recommendation system uses social information residing on smart devices, each of which has their own view of this information. It is important to consider where this social information should be stored in the social network, and to ensure that other smart devices and actors are able to access this social information in an efficient manner in order to fully exploit a SIoT recommendation system. Moreover, it would be advantageous to perform social network management based on current contextual IoT application information so that SIoT recommendation systems could have all the required information readily available, minimizing any delay and improving user experience.

5.3.4.3 Trust, Privacy and Security

Since SIoT-based recommendation system requires access to data from various IoT applications, trust, privacy and security are very important issues to consider. The access to and exploitation of various IoT applications' data can lead to misuse and fraudulent activities without a secure technology. Hence, the success of a SIoT recommendation system requires secure technology that can ensure safe communication, user privacy and trustworthy interactions. Existing approaches for achieving user privacy, trustworthiness and data confidentiality developed for other platforms can be taken into account as guidelines while developing approaches for SIoT-based recommendation system.

5.3.4.4 Self-Management, Self-Organization and Self-Healing

In the SIoT, the establishment and management of social relationships is performed without human intervention, and the SIoT is expected to be composed of billions of devices.

Therefore, it is imperative to have automatic operations, including self-management, self-organization and self-healing. Autonomic service discovery, composition and data analysis will also help to improve user experience. Moreover, a SIoT recommendation system should have self-learning capabilities and be self-adaptive based on the feedback from IoT applications in order to enhance its recommendation services.

5.3.4.5 Network Navigability

In the SIoT, objects look for their required services using their friendship circles in a distributed way. However, since SIoT is composed of large number of objects having social relationships with each other, each object has to maintain a large number of friends which will slow down the search operation for finding the desired services. Hence, the network navigability is an important issue in SIoT which should be taken into account. Some work has been performed in [116] for network navigability in SIoT by analyzing possible solutions to enable smart objects to select the appropriate links which can benefit the overall network navigability. This can serve as a base reference for investigating network navigability in SIoT.

5.3.4.6 Proof of Concept

The idea of SIoT-based recommendation services across IoT applications is a novel paradigm which has not been explored before. Some of the challenges are discussed above; however, a number of new challenges may be faced while implementing this concept. Therefore, developing a proof of concept to accommodate new challenges that will need to be addressed during the actual implementation of this system is a vital first step and can be considered as step 0.

In this study, we have proposed a concept of exploiting the SIoT for recommendation services across IoT applications. This framework helps to provide recommendation services across IoT application by taking the advantages of social IoT which offers things-to-things and things-to-human communications. In order to better understand this concept, we have also presented a sample application scenario for a more complete understanding this concept. Finally, we have discussed some implementation challenges that should be considered for the realization of this concept.

In the next study, we present a third type of SIoT communication at a global level, i.e., social cross-domain IoT enabling application-to-application communications.

5.4 SCDIoT: Social Cross-domain IoT enabling Application-to-Application Communication

5.4.1 Introduction

Traditionally, in the SIoT, there are two types of communications: things-to-things communication and things-to-human communication. In this study, we propose a third type of communication for the SIoT at a global level which enables application-to-application communication called social cross-domain IoT (SCDIoT). SCDIoT allows collaboration among IoT applications by enabling them to talk to each other, build social circles and relationships among each other, and to benefit from various useful services in order to completely exploit the advantage of interoperability. We have seen the significant benefit of things-to-things and things-to-human communication in traditional SIoT. With social application-to-application communication, enabled by the SCDIoT, the benefits can increase to many-fold by bridging the gap of IoT applications' isolation. We present the framework and some potential use case scenarios, together with some challenges and future research directions for SCDIoT.

5.4.2 Proposed Framework

In this section, we propose the framework of SCDIoT for social application-to-application communication, illustrated in Figure 5.3.

The three different applications at the bottom of the figure deploy IoT devices (e.g., sensors, actuators, RFIDs, cameras, microphones and smartphones); however, in contrast to the deployment of IoT devices in traditional IoT applications where IoT devices do not talk to each other, these applications (i.e., Applications 1, 2 and 3) support things-to-things communication in which the devices also talk to each other by having social relationships and circles among themselves enabled by the SIoT. This could be useful in various scenarios, such as in calculating room temperature where various temperature sensors are deployed in a room. Sensors can coordinate and collaborate with each other to correct their measurements. For example, if four temperature sensors are deployed in a room and three of them measures 20 degrees Celsius while one of them measured 4 degrees Celsius; it means that there is a problem with the fourth sensor. Thanks to the collaboration enabled by the SIoT, the fourth sensor will identify this problem and will either not forward its measurement to the gateway or will retake the measurement, rectifying the problem. Things-to-things communication supported by the SIoT brings the computing one level lower to edge computing (i.e., from cloud to edge to things-to-things); however, it mainly depends upon the application requirements because if the devices are resource-constrained with limited battery and/or processing power, things-to-things communication would not

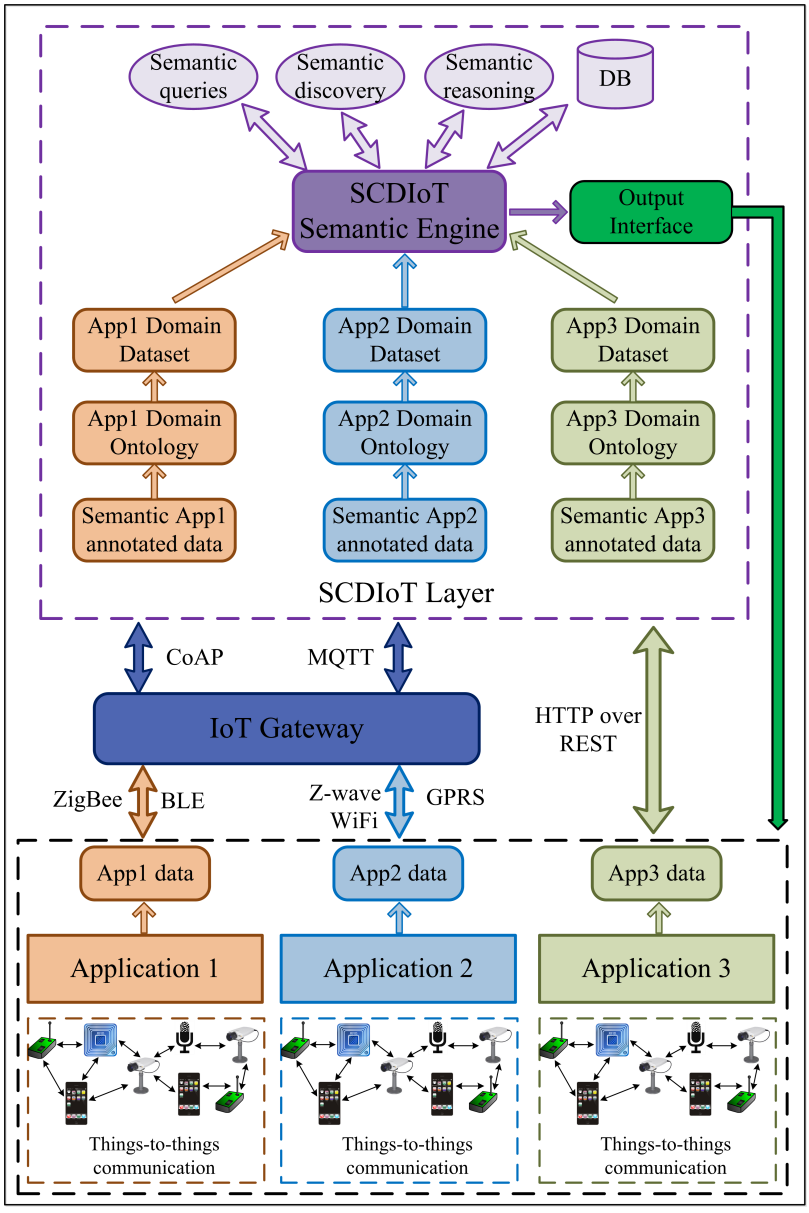


Figure 5.3 – Proposed Framework of SCDIoT.

be desirable. On the contrary, it could be desirable if devices have direct power and good processing capabilities, e.g., in smart home devices.

Each application collects data from its underlying deployment, and these collected data need to be forwarded to the SCDIoT layer to enable social application-to-application communication. Data forwarding could be achieved through various heterogeneous protocols. In Figure 5.3, each application uses a different communication protocol to forward its data.

Applications 1 and 2 first transfer their data to the IoT gateway using low power radio communication interfaces such as ZigBee, Bluetooth Low Energy (BLE), Z-wave, GPRS (General Packet Radio Service) or WiFi. The IoT gateway then forwards the data to the SCDIoT layer using any IoT communication protocol such as CoAP (Constrained Application Protocol) or MQTT (Message Queuing Telemetry Transport). Application 3 instead transfers its data directly to the SCDIoT layer through REST APIs, bypassing the IoT gateway. When the data reaches the SCDIoT layer, it needs to be enriched with semantics in order to achieve interoperability and enable social cross-domain IoT. The first step is to apply semantic annotations to the raw data of heterogeneous applications and domains. Semantic annotation is a very important step in the process of understanding and being able to apply logic (i.e., reasoning) to the applications' data because same data from various applications may have different meaning or different data from various applications may have the same meaning. For example, let's assume that all three applications data are related to temperature measurements. However, if they used their own notations to represent temperature, e.g., 't', 'temp' or 'temperature', it can be understandable by humans but not by machines [27,117]. Therefore, SCDIoT first applies semantic annotations to describe the data in order to make it understandable by the machines. The second step is to find the appropriate domain ontology of the applications' data. For example, temperature could be either environmental temperature or body temperature which corresponds to completely different domains, e.g., environmental temperature belongs to weather ontology while body temperature belongs to health ontology. After obtaining the relevant ontologies, the next step is to retrieve the most relevant datasets in order to acquire additional knowledge. Once the relevant ontologies and datasets have been identified, SCDIoT links the common concepts (e.g., 't', or 'temp' or 'temperature') to the identified ontologies and datasets using OWL (Web Ontology Language) with equivalent keywords, e.g.,`owl:equivalentClass` or `owl:sameAs` respectively. Finally, SCDIoT Semantic Engine performs reasoning over the semantic data using semantic queries and resource discoveries, and then it sends the required and relevant data back to the applications through its output interface.

5.4.3 Potential Use Case Scenarios

Social application-to-application communication can be advantageous in a number of use case scenarios. We present some potential use case scenarios below.

Traffic lights are generally operated either using fixed time intervals or based on the road load identified through sensors deployed on the roads. However, with the help of social application-to-application communication, traffic lights can be operated based to some extent on the users' profiles. For instance, a user's smartphone can provide the user's current route (e.g., assuming the user is using a navigation system) and this route

information can be passed to traffic lights management system which can take into account the user's route and operates accordingly. We propose that traffic lights management could be enhanced and become more efficient compared to current traffic lights management systems. Assuming an optimal case, let's consider a road intersection where there are four vehicles, one vehicle on each road, and the driver of each vehicle wants to make a turn to the right. In this situation, if all four traffic signals can turn green, there will be no collision and the drivers will not have to wait. If traffic light management system could communicate with users' navigation system and know the users' routes, it could allow all four turns by switching the traffic lights to green so that drivers can take right turn without any collision, which is otherwise not possible in currently operating traffic lights management systems. This is just a hypothetical case to highlight the usefulness of social application-to-application communication.

One step ahead, we could extend this scenario from traffic light management to traffic crowd management. A traffic crowd management system could consider the users route information and directly communicate with user's navigation systems to distribute users onto different routes to avoid traffic jams. Here, we would have two-way application-to-application communication, i.e., navigation systems provide users' routes information to a traffic crowd management system, the traffic crowd management system considers all the users' routes and communicates back to the navigation systems suggesting a different route with low traffic (in a way that the alternative route does not get congested, and so suggesting different routes to users) to avoid traffic jams.

Another use case scenario could be food recipe suggestions. This application can consider a user's profile (e.g., his/her food preferences), his/her health constraints (via a health monitoring application), the ingredients currently in the kitchen (through the smart home system) and weather conditions (via a weather monitoring application). Taking into account these three types of applications' data, a food recipe can recommend a recipe to a user which is in accordance with his/her health, is suitable for the current weather, for which the user has all the ingredients available in his/her home, and is according to his/her food preferences. When food ingredients run out of the stock, the smart kitchen can take appropriate actions (e.g., informing the user or ordering by itself) to refill those ingredients [117].

5.4.4 Challenges and Future Research Directions

In this section, we discuss the challenges and future research directions in SCDIoT, more specifically in social application-to-application communication.

5.4.4.1 Latency

The biggest challenge in an SCDIoT system is the latency. Since social application-to-application communication is possible through the SCDIoT framework, more latency can be incurred because the data must first be sent to the SCDIoT framework, which does the processing and then sends back the required information to the application. Hence, for delay-sensitive applications (e.g., health and emergency systems), this is a serious challenge which needs to be addressed.

5.4.4.2 Privacy and Trust

Privacy, trust and security are major issues in the SCDIoT. Since all the applications data pass through the SCDIoT framework, this presents privacy issues for applications which need to comply with strong privacy policies. Therefore privacy solutions for the SCDIoT must be developed or existing ones adapted to the SCDIoT. Trust is also an important parameter to be considered in order to ensure that each application involved in social cross-domain communication can be trustworthy to avoid malicious behaviors. Existing IoT solutions for privacy and trust could be considered as guidelines while developing solutions for the SCDIoT.

5.4.4.3 Autonomous Management

The SCDIoT framework receives applications' data, applies semantic technologies, performs reasoning and makes it shareable with other IoT applications. Therefore, all the operations and management need to be autonomous without human intervention [118]. It would also be desirable if the SCDIoT could learn from its environment and adapt itself according to new emerging requirements. Reinforcement learning could be a very beneficial tool for such self-learning from the environment.

5.4.4.4 Network and Storage Management

The SCDIoT processes applications' raw data, applies semantic technologies and obtains semantic data, as well as performs reasoning to infer new data. Such data must be securely stored and managed in order to be used in the future for enhanced services. Additionally, it is important to consider how the communication and access to the data and resources will be performed. Therefore, network and storage management is another important issue to be addressed.

5.4.4.5 Proof of Concept

The SCDIoT is a novel concept which needs more exploration and investigation. A proof of concept needs to be developed. We have discussed a few of the challenges that should be carefully considered while developing the proof of concept. New challenges may arise during the actual development of the proof of concept, and these will need to be addressed and incorporated. The relevant works [102], [103], and [111] should be considered as a valid starting point toward the proof of concept implementation.

The SIoT establishes social relationships among objects in the IoT, supporting two types of communications: things-to-things and things-to-human communication. In this study, we have proposed a third type of communication at a global level, i.e., a social cross-domain IoT (SCDIoT), which enables application-to-application communication thanks to social relationships and circles through which IoT applications can closely collaborate with each other. We have presented the basic concept of the SCDIoT, the specific framework that achieves interoperability and social relationships, and some potential use case scenarios together with challenges and future research directions.

5.5 Summary and Discussion

To summarize and conclude, this chapter presented the third contribution of this thesis of frameworks for SIoT-based recommendation services across smart cities/IoT applications and social cross-domain application-to-application communication. It is comprised of two parts, each proposing a framework. The first framework proposed SIoT-based recommendation services across IoT applications in smart cities. It presented the proposed concept and a sample application scenario where different IoT applications in smart cities collaborate with each other using SIoT to provide recommendation services to each other. It also discussed the implementation challenges to realize this system. The second framework proposed another type of communication for the SIoT at a global level that enables application-to-application communication, known as social cross-domain IoT (SCDIoT). It also presented the proposed concept and some potential use case scenarios, as well as the implementation challenges to realize this system. These two frameworks serve as the building blocks for social cross-domain recommendation services and application-to-application communication.

In the next chapter, we move towards the conclusion of this thesis and some future works for the extension of this thesis.

Chapter 6

Conclusion and Future Work

Contents

6.1 Conclusion

In this thesis, we proposed a novel privacy preserving IoT recommender for smart cities that provides recommendations by exploiting the IoT data of sensors and by considering various metrics. In addition, we proposed two frameworks for cross-domain recommendation services and application-to-application communications using social IoT in smart cities. The novelty and new ideas proposed in this thesis include i) a novel algorithm for the mapping of sensor and route coordinates; ii) the development of a novel IoT recommender for smart parking that considers IoT data of parking and traffic sensors, interacts with the Trust Monitoring component to obtain trust score and recommend trusted parking spots, provides GDPR-compliant implementation and occupancy statistics of parking areas; iii) the development of an IoT recommender for smart skiing that offers a novel routing engine for ski routes; iv) the application of two privacy preservation approaches: k-anonymity and differential privacy, for preserving the privacy of parking database that has not been explored in the past; and two novel frameworks on the use of social IoT for recommendation services.

6.1.1 Summary and Insights of Contributions

We provide the summary of each contribution, as well as the insights gained from each contribution in this section.

- **IoT Recommender for Smart Cities:** The first contribution is organized into three parts. The first part is the about mapping of sensors and route coordinates, the second part is about the IoT recommender for smart parking and the third part is about IoT recommender for smart skiing. We summarize each part separately and provide our insights.

  - In the first part, while working on traffic sensors in Santander to analyze the traffic on the roads, we obtained the coordinates of some alternative routes between two random points (considered as starting and ending points) and tried to map the traffic sensors on each route. However, we identified that there are some traffic sensors that did not map to the routes even though they existed within the route coordinates. This was because they were deviated from the coordinates of the roads. Therefore, we proposed an algorithm for the mapping of the sensors and route coordinates by introducing a deviation margin. We presented an algorithm and two illustrative examples that covered all the possible scenarios. We evaluated the performance of our mapping algorithm by measuring correct detection, missed detection and false detection by comparing with the

baseline scheme that does not consider the deviation margin. The experimental results showed signficant accuracy of our proposed algorithm.

- In the second part, we designed an EU GDPR-compliant IoT recommender system for smart parking that provides the recommendations of parking spots and routes by exploiting the data of parking and traffic sensors. The IoT recommender provide four-fold functions. Firstly, it helps users to find free parking spots based on different metrics (e.g., nearest or nearest trusted parking spot). Secondly, it recommends routes (the least crowded or the shortest route) leading to the recommended parking spots from the users' current location by using the mapping algorithm proposed in the first part. Thirdly, it provides the real-time provisioning of expected availability of parking areas (comprised of parking spots organized into groups) in a user-friendly manner. Lastly, it provides a GDPR-compliant implementation for operating in a privacy-aware environment. The IoT recommender was integrated into the smart parking use case of an H2020 EU-KR WISE-IoT project and was evaluated by the citizens of Santander Spain through a prototype. The evaluation results showed the high satisfaction of the citizens with the quality, functionalities, ease of use and reliability of the recommendations provided by the IoT recommender. The IoT recommender was also demonstrated at various occasions.
- In the third part, we designed an IoT recommender for smart skiing that provides skiing routes comprised of specific types of slopes. For skiing routes, there did not exist any stable routing engine, therefore, we developed a novel routing engine for skiing routes. This work was integrated into the smart skiing use case of WISE-IoT project.

- **Privacy Preservation of Smart Parking System:** The second contribution preserves the privacy of users in smart parking system, specifically against IoT parking recommender system while analyzing historic parking database for providing efficient and personalized recommendation services based on users past parking experience. Although the developed IoT recommender for smart parking was GDPR-compliant, however, it did not fully protect the privacy of the users because an indiscriminately sharing of users' data with an untrusted or semi-trusted third-party IoT parking recommender system violates the privacy, as user's behavior and mobility patterns could be inferred by analyzing the past travelling history of the users. Therefore, the privacy of users was preserved using k-anonymity (anonymization) and differential privacy (perturbation) techniques. The privacy and utility by both k-anonymity and differential privacy was thoroughly studied through extensive experiments. From the

experimental study, we learned that k-anonymity is stuiable for smaller values of k and for lower QIA sizes in order to have a good balance between privacy and utility. Otherwise, it achieves privacy but it makes the utility worst, hence unable to infer any useful information from the anonymized dataset. On the other hand, differential privacy is suitable for higher privacy budget ϵ and lower sensitivity values $\triangle f$ to have better utility while achieving the privacy.

- **Frameworks for Cross-Domain Recommendations in Smart Cities:** The third contribution proposed two frameworks for cross-domain recommendations in smart cities: one on how social IoT can be used for recommendation services across smart cities applications , and other on the new type of communication of social IoT at a global level, i.e., social cross-domain application-to-application communications. As smart cities applications are developed in a vertical manner and do not talk / communicate with each, i.e., each application is developed for a certain scenario which generally does not share data with other application, therefore, these frameworks could help to bridge this gap and provide cross-domain recommendation services by enabling application-to-application communications across IoT applications in smart cities and serve as the building blocks.

6.1.2 Practical Applicability

Some of our work (e.g., Contribution 1 in Chapter 3) has been practically applied in the real-world, specifically in an EU-KR H2020 WISE-IoT project, as well as been demonstrated at various occasions.

6.2 Future Work

This section summarizes some perspectives on the future work to extend the work in this thesis.

- **Consideration of Two-Way Roads in Sensor Mapping:** In our current work of mapping of sensor and route coordinates (see Section 3.2), we considered the roads to be one-way because our focus of study was Santander city where the traffic sensors are deployed on one-way streets. However, it is interesting to extend this work from one-way roads to two-way roads for the mapping the sensor and route coordinates. It could have its own challenges and implications. For instance, in the case of two-way roads, the deviation margin needs to be studied well in a way that it should not consider the sensors deployed on the other (reverse) side of the roads into consideration. It requires the modifications in our currently proposed mapping algorithm to consider this constraint.

- **Application of Machine Learning for Predicting the Least Congested Routes:** In our current work of mapping of sensor and route coordinates (see Section 3.2), we considered the bottleneck road load of real-time traffic to identify the level of congestion on the streets. However, it could be possible that although the road is less congested at the current time, but when the driver gets into the road, it becomes congested and our system could not identify it because of not considering the past history. Therefore, it would be interesting to apply Machine Learning that takes into account the past history and patterns of the traffic load on the roads and subsequently, recommends such routes that are expected to remain less congested in the near future when the driver gets there.

- **Application of Machine Learning for Predicting the Status of Parking Spots:** Currently, in the IoT recommender for smart parking (see Section 3.3), the parking spots are selected based on the real-time information. There are chances that a parking spot was available at the time of recommendation by the IoT recommender, however when the driver reaches there, it gets occupied. Therefore, the application of Machine Learning is very interesting that could predict the status of the parking spots in the near future that should be considered by the IoT recommender in the recommendations of parking spots.

- **Applying IoT Recommender to Other Cities:** The IoT recommenders for smart parking and smart skiing were currently developed and tested for Santander, Spain and Chamrousse, France, respectively. We claimed that they could be applied to other

places having similar infrastructure in a similar manner, however, we have not tested our claim. Therefore, applying these IoT recommenders to other cities is another extension to our work. An important factor while applying the IoT recommender to other cities is analysing the scalability in terms of the size of the city where the IoT sensors are deployed, and the number of concurrent requests made by the users. Additionally, the complexity and overhead also need to be analysed.

- **Studying the Impact of Privary Preservation on Personalized Recommendations:** We preserved the privacy of users against parking recommender by anonymizing and perturbing the parking database. When we go for privacy, we lose the data of individual users and correlation between the records, hence making it no longer possible to provide personalized recommendation with respect to the individual user's habits and preferences. This ultimately affects the quality of recommendations. Hence, there is a need to study the impact of privacy preservation on recommendations services. Also, there might be a need to either modify the existing recommender system or design a new one that can work on privacy preserved databases.

- **Exploitation of Privary Preserving Smart Parking System in Real-life Recommendations:** The privacy preserving smart parking system was designed after the completion of WISE-IoT project, and therefore, it was not possible to apply it in the real-world and check its feasibility. Although we evaluated the privacy and utility using our two considered privacy preservation techniques of k-anonymity and differential privacy, it would be interesting to use the anonymized dataset (in-case of k-anonymity) and perturbed query responses (in case of differential privacy) for providing the recommendations in real-life and check their feasibility, as well as the level of satisfaction with the citizens.

- **A Working Prototype of Frameworks for Cross-Domain Recommendations:** Finally, the two proposed frameworks on using social IoT for cross-domain recommendations services and application-to-application communications are mainly conceptual frameworks. There is a need to have a proof-of-concept prototype at the initial stage to validate the idea and subsquently, to develop a platform offering such cross-domain recommendation services in order to widely disseminate them.

References

[1] Y. Saleem, N. Crespi, M. H. Rehmani, R. Copeland, D. Hussein, and E. Bertin, "Exploitation of Social IoT for Recommendation Services," in *IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 359–364.

[2] S. Shepard, *RFID: Radio Frequency Identification.* McGraw Hill Professional, 2005.

[3] S. Farahani, *ZigBee Wireless Networks and Transceivers.* Newnes, 2011.

[4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[5] A. d. M. D. Esposte, E. F. Santana, L. Kanashiro, F. M. Costa, K. R. Braghetto, N. Lago, and F. Kon, "Design and Evaluation of a Scalable Smart City Software Platform with Large-Scale Simulations," *Future Generation Computer Systems*, vol. 93, pp. 427–441, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2018.10.026

[6] Smart Santander. Last accessed: October 2019. [Online]. Available: http://www.smartsantander.eu/index.php/testbeds/item/132-santander-summary

[7] Heidelberg Digital. Last accessed: October 2019. [Online]. Available: https://www.heidelberg.de/1192862.html

[8] Global Smart City. Last accessed: October 2019. [Online]. Available: http://www.k-smartcity.kr/english/index.php

[9] Amsterdam Smart City. Last accessed: October 2019. [Online]. Available: https://amsterdamsmartcity.com

[10] Connecting Copenhagen. Last accessed: October 2019. [Online]. Available: https://stateofgreen.com/en/partners/city-of-copenhagen/news/connecting-copenhagen-is-the-worlds-best-smart-city-project/

[11] MiNT Madrid Inteligente. Last accessed: October 2019. [Online]. Available: https://www.madridforyou.es/en/mint-madrid-intelligent

[12] Dubai Now. Last accessed: October 2019. [Online]. Available: https://dubainow.dubai.ae/en/Pages/default.aspx

[13] T. N. Pham, M.-F. Tsai, D. B. Nguyen, C.-R. Dow, and D.-J. Deng, "A Cloud-based Smart-parking System based on Internet-of-Things Technologies," *IEEE Access*, vol. 3, pp. 1581–1591, 2015.

[14] L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and R. Vergallo, "Integration of RFID and WSN technologies in a Smart Parking System," in *22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2014, pp. 104–110.

[15] C. W. Hsu, M. H. Shih, H. Y. Huang, Y. C. Shiue, and S. C. Huang, "Verification of Smart Guiding System to Search for Parking Space via DSRC Communication," in *12th International Conference on ITS Telecommunications*, 2012, pp. 77–81.

[16] P. Samarati, "Protecting Respondents' Identities in Microdata Release," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, 2001.

[17] T. Li, N. Li, J. Zhang, and I. Molloy, "Slicing: A New Approach for Privacy Preserving Data Publishing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 3, pp. 561–574, 2012.

[18] L. Sweeney, "k-anonymity: A Model for Protecting Privacy," *International Journal of Uncertainty, Puzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[19] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-Diversity: Privacy Beyond k-Anonymity," in *22nd International Conference on Data Engineering (ICDE)*, Atlanta, GA, USA, 2006, pp. 24–35.

[20] N. Li, T. Li, and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," in *IEEE 23rd International Conference on Data Engineering*, no. 3, 2007, pp. 106–115.

[21] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds., 2006, pp. 1–12.

[22] J. Salas, "Sanitizing and Measuring Privacy of Large Sparse Datasets for Recommender Systems," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2019. [Online]. Available: https://doi.org/10.1007/s12652-019-01391-2

[23] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The Social Internet of Things (SIoT)–When Social Networks Meet the Internet of Things: Concept, Architecture and Network Characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[24] M. Nitti, R. Girau, A. Floris, and L. Atzori, "On Adding the Social Dimension to the Internet of Vehicles: Friendship and Middleware," in *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2014, pp. 134–138.

[25] M. Nitti and L. Atzori, "What the SIoT needs: A New Caching System or New Friendship Selection Mechanism?" in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 424–429.

[26] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 28–37, 2001.

[27] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the Internet of Things: Early Progress and Back to the Future," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 8, no. 1, pp. 1–21, 2012.

[28] A. Gyrard, "Designing Cross-Domain Semantic Web of Things Applications," Ph.D. dissertation, 2015.

[29] D1.2 - Wise-IoT High-level Architecture, Reference Technologies and Standard. Last Accessed: January 2020. [Online]. Available: http://wise-iot.eu/wp-content/uploads/2017/06/D1.2-Wise-IoT-Architecture-PU-V1.0.pdf

[30] D2.1 - Morphing Mediation Gateway with Management and Configuration Functions R1. Last Accessed: January 2020. [Online]. Available: http://wise-iot.eu/wp-content/uploads/2017/03/D2.1-Morphing-Mediation-Gateway-with-Management-and-Configuration-Functions-R1-Final-version.pdf

[31] D3.2 - Integrated Platforms R2. Last Accessed: January 2020. [Online]. Available: http://wise-iot.eu/wp-content/uploads/2018/09/D3.2-Integrated-IoT-platforms-R2-1.0-Final.pdf

[32] D2.6 - Self-Adaptive Recommendation System. Last Accessed: October 2019. [Online]. Available: http://wise-iot.eu/wp-content/uploads/2017/08/D2.6-Self-Adaptive-Recommendation-System-V1.02.pdf

[33] Worldwide interoperability for semantics iot (wise-iot). [Online]. Available: http://wise-iot.eu/en/home/

[34] A. Fernandez-Ares, A. Mora, M. Arenas, P. Garcia-Sanchez, G. Romero, V. Rivas, P. Castillo, and J. Merelo, "Studying Real Traffic and Mobility Scenarios for a Smart City using a New Monitoring and Tracking System," *Future Generation Computer Systems*, vol. 76, pp. 163–179, 2017.

[35] A. Ziat, B. Leroy, N. Baskiotis, and L. Denoyer, "Joint Prediction of Road-Traffic and Parking Occupancy over a City with Representation Learning," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 725–730.

[36] R. Y. Lau, "Toward a Social Sensor Based Framework for Intelligent Transportation," in *IEEE 18th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM*, 2017, pp. 1–6.

[37] C. T. Barba, M. A. Mateos, P. R. Soto, A. M. Mezher, and M. A. Igartua, "Smart City for VANETs using Warning Messages, Traffic Statistics and Intelligent Traffic Lights," in *IEEE Intelligent Vehicles Symposium (IV)*, 2012, pp. 902–907.

[38] D. C. Shoup, "Cruising for Parking," *Transport Policy*, vol. 13, no. 6, pp. 479–486, 2006.

[39] Brouter Offline Routing Engine. Last Accessed: November 2018. [Online]. Available: http://brouter.de/brouter/offline.html

[40] Open street maps. Last Accessed: November 2018. [Online]. Available: https://www.openstreetmap.org

[41] Traffic sensors at santander, spain. Last Accessed: May 2018. [Online]. Available: https://mu.tlmat.unican.es:8443/v2/entities?limit=1000&type=TrafficFlowObserved

[42] S. Poslad, S. E. Middleton, F. Chaves, R. Tao, O. Necmioglu, and U. Bugel, "A Semantic IoT Early Warning System for Natural Environment Crisis Management," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 2, pp. 246–257, 2015.

[43] G. Vojkovic, "Will the gdpr slow down development of smart cities?"

[44] E. Mougiakou and M. Virvou, "Based on GDPR Privacy in UML: Case of e-Learning Program," in *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2017, pp. 1–8.

[45] P. Sotres, J. Lanza, L. Sanchez, J. R. Santana, C. Lopez, and L. Munoz, "Breaking Vendors and City Locks through a Semantic-enabled Global Interoperable Internet-of-Things System: A Smart Parking Case," *Sensors*, vol. 19, no. 2, p. 229, 2019.

[46] P. Sotres, J. R. Santana, L. Sanchez, J. Lanza, and L. Munoz, "Practical Lessons from the Deployment and Management of a Smart City Internet-of-Things Infrastructure: The SmartSantander Testbed Case," *IEEE Access*, vol. 5, pp. 14 309–14 322, 2017.

[47] L. Sanchez, L. Munoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, *et al.*, "SmartSantander: IoT experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[48] Smart Santander Data Sets. [Online]. Available: http://datos.santander.es/data/

[49] H. Baqa, N. B. Truong, N. Crespi, G. M. Lee, and F. Le Gall, "Quality of Information as an Indicator of Trust in the Internet of Things," in *IEEE 17th International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 204–211.

[50] D. Wuest, F. Fotrousi, and S. Fricker, "Combining Monitoring and Autonomous Feedback Requests to Elicit Actionable Knowledge of System Use," in *International Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ)*. Springer, 2019, pp. 209–225.

[51] P. Sotres, C. L. de la Torre, L. Sanchez, S. Jeong, and J. Kim, "Smart City Services Over a Global Interoperable Internet-of-Things System: The Smart Parking Case," in *Global Internet of Things Summit (GIoTS)*, 2018, pp. 1–6.

[52] R. E. Barone, T. Giuffre, S. M. Siniscalchi, M. A. Morgano, and G. Tesoriere, "Architecture for Parking Management in Smart Cities," *IET Intelligent Transport Systems*, vol. 8, no. 5, pp. 445–452, 2013.

[53] C. Shiyao, W. Ming, L. Chen, and R. Na, "The research and implement of the intelligent parking reservation management system based on zigbee technology," in *Sixth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*. IEEE, 2014, pp. 741–744.

[54] L. Lambrinos and A. Dosis, "DisAssist: An Internet of Things and Mobile Communications Platform for Disabled Parking Space Management," in *IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 2810–2815.

[55] FIWARE. Last accessed: March 2019. [Online]. Available: https://www.fiware.org

[56] "oneM2M - Standards for M2M and the Internet of Things," accessed: October 2019. [Online]. Available: http://onem2m.org/

[57] FIWARE Data Models. Last accessed: March 2019. [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/Parking/doc/introduction/index.html

[58] OMA, "Open Mobile Alliance, NGSI Context Management," 2012, last accessed: March 2019. [Online]. Available: http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf

[59] FIWARE Data Model for Parking. Last accessed: March 2019. [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/Parking/doc/introduction/index.html

[60] FIWARE Data Model for Traffic Flow. Last accessed: March 2019. [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/Transportation/TrafficFlowObserved/doc/spec/index.html

[61] D. Barth, "The Bright Side of Sitting in Traffic: Crowdsourcing Road Congestion Data," *Google Official Blog*, 2009.

[62] Orion Context Broker. Last accessed: April 2019. [Online]. Available: https://fiware-orion.readthedocs.io/en/master/

[63] Y. Saleem and N. Crespi, "Mapping of Sensor and Route Coordinates for Smart Cities," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2018, pp. 570–576.

[64] D. Protection, "Rules for the Protection of Personal Data Inside and Outside the EU," *Retrieved from European Comission: https://ec. europa. eu/info/law/law-topic/data-protection_en*, 2018.

[65] GDPR Explained In 5 Minutes: Everything You Need to Know. [Online]. Available: https://www.coredna.com/blogs/general-data-protection-regulation

[66] C. Bettstetter and S. Konig, "On the Message and Time complexity of a Distributed Mobility-Adaptive Clustering Algorithm in Wireless Ad Hoc Networks," in *Proceedings of the Fourth European Wireless Conference*. Citeseer, 2002, pp. 128–134.

[67] "Clustering Overhead and Convergence Time Analysis of the Mobility-based Multi-Hop Clustering Algorithm for Mobile Ad Hoc Networks, author=Er, Inn Inn and Seah, Winston KG, journal=Journal of Computer and System Sciences, volume=72, number=7, pages=1144–1155, year=2006."

[68] P. Sotres, C. L. D. L. Torre, L. Sanchez, S. Jeong, and J. Kim, "Smart city services over a global interoperable internet-of-things system: The smart parking case," *2018 Global Internet of Things Summit, GIoTS 2018*, no. March, 2018.

[69] D1.1 - Wise-IoT Pilot Use Case Technical Description, Business Requirements, and Draft High-Level Architecture. Last Accessed: October 2019. [Online]. Available: http://wise-iot.eu/wp-content/uploads/2016/12/D1.1-Use-Cases-PU-V1.0.pdf

[70] Open snow map routing engine. Last Accessed: November 2018. [Online]. Available: http://www.opensnowmap.org

[71] GraphHopper Routing Engine. Last Accessed: October 2019. [Online]. Available: https://www.graphhopper.com

[72] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential Privacy for Location-based Systems," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 901–914, 2013.

[73] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Privacy-Preserving Smart Parking Navigation Supporting Efficient Driving Guidance Retrieval," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6504–6517, 2018.

[74] J. Ni, K. Zhang, X. Lin, Y. Yu, and X. S. Shen, "Cloud-Based Privacy-Preserving Parking Navigation Through Vehicular Communications," in *International Conference on Security and Privacy in Communication Systems*, 2016, pp. 85—-103.

[75] I. Chatzigiannakis, A. Vitaletti, and A. Pyrgelis, "A Privacy-preserving Smart Parking System using an IoT Elliptic Curve based Security Platform," *Computer Communications*, vol. 89-90, pp. 165–177, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2016.03.014

[76] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure Automated Valet Parking: A Privacy-Preserving Reservation Scheme for Autonomous Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11 169–11 180, 2018.

[77] R. Lu, X. Lin, H. Zhu, and X. Shen, "SPARK: A New VANET-Based Smart Parking Scheme for Large Parking Lots," in *Proceedings - IEEE INFOCOM*, 2009, pp. 1413–1421.

[78] ——, "An Intelligent Secure and Privacy-Preserving Parking Scheme Through Vehicular Communications," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 6, pp. 2772–2785, 2010.

[79] G. Yan, W. Yang, D. B. Rawat, and S. Olariu, "SmartParking: A Secure and Intelligent Parking System," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 1, pp. 18–30, 2011.

[80] A. Alqazzaz, I. Alrashdi, E. Aloufi, M. Zohdy, and H. Ming, "SecSPS: A Secure and Privacy-Preserving Framework for Smart Parking Systems," *Journal of Information Security*, vol. 9, no. 4, pp. 299–314, 2018.

[81] R. Garra and S. Mart, "A Privacy-Preserving Pay-by-Phone Parking System," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 5697–5706, 2017.

[82] J. Hu, D. He, Q. Zhao, and K. K. R. Choo, "Parking Management: A Blockchain-Based Privacy-Preserving System," *IEEE Consumer Electronics Magazine*, vol. 8, no. 4, pp. 45–49, 2019.

[83] T. Zhu, G. Li, W. Zhou, and P. S. Yu, "Differentially Private Data Publishing and Analysis: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1619–1638, 2017.

[84] J. Soria-Comas, J. Domingo-Ferrer, D. Sanchez, and D. Megias, "Individual Differential Privacy: A Utility-Preserving Formulation of Differential Privacy Guarantees," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1418–1429, 2017.

[85] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain K-anonymity," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2005, pp. 49–60.

[86] R. J. Bayardo and R. Agrawal, "Data Privacy Through Optimal k-Anonymization," in *21st International Conference on Data Engineering (ICDE)*, 2005, pp. 217—-228.

[87] W. Fan, J. He, M. Guo, P. Li, Z. Han, and R. Wang, "Privacy Preserving Classification on Local Differential Privacy in Data Centers," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 70–82, 2020. [Online]. Available: https://doi.org/10.1016/j.jpdc.2019.09.009

[88] X. Su, G. Sperli, V. Moscato, A. Picariello, C. Esposito, and C. Choi, "An Edge Intelligence Empowered Recommender System Enabling Cultural Heritage Applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4266–4275, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8675979/

[89] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2013.03.012

[90] J. Yu, M. Gao, W. Rong, Y. Song, and Q. Xiong, "A Social Recommender Based on Factorization and Distance Metric Learning," *IEEE Access*, vol. 5, pp. 21 557–21 566, 2017.

[91] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen, "Smart-its Friends: A Technique for Users to Easily Establish Connections Between Smart Artefacts," in *international conference on Ubiquitous Computing*. Springer, 2001, pp. 116–122.

[92] J. Bleecker, "A Manifesto for Networked Objects—cohabiting with Pigeons, Arphids and Aibos in the Internet of Things," in *Proc. of the 13th International Conference on Human–Computer Interaction with Mobile Devices and Services, MobileHCI*, 2006, pp. 1–17.

[93] E. Nazzi and T. Sokoler, "Walky for Embodied Microblogging: Sharing Mundane Activities through Augmented Everyday Objects," in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 2011, pp. 563–568.

[94] P. Mendes, "Social-driven Internet of Connected Objects." IAB workshop on Interconnecting Smart Objects with the Internet, 2011.

[95] H. Ning and Z. Wang, "Future Internet of Things Architecture: Like Mankind Neural System or Social Organization Framework?" *IEEE Communications Letters*, vol. 15, no. 4, pp. 461–463, 2011.

[96] L. Ding, P. Shi, and B. Liu, "The Clustering of Internet, Internet of Things and Social Network," in *3rd International Symposium on Knowledge Acquisition and Modeling (KAM)*, 2010, pp. 417–420.

[97] D. Guinard, M. Fischer, and V. Trifa, "Sharing Using Social Networks in a Composable Web of Things," in *PerCom Workshops*, 2010, pp. 702–707.

[98] M. Kranz, L. Roalter, and F. Michahelles, "Things that Twitter: Social Networks and the Internet of Things," in *What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing (Pervasive 2010)*, 2010, pp. 1–10.

[99] J. An, X. Gui, W. Zhang, and J. Jiang, "Nodes Social Relations Cognition for Mobility-aware in the Internet of Things," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*. IEEE, 2011, pp. 687–691.

[100] K. Rasch, "Smart Assistants for Smart Homes," Ph.D. dissertation, KTH Royal Institute of Technology, 2013.

[101] D. Hussein, S. N. Han, G. M. Lee, and N. Crespi, "Social Cloud-Based Cognitive Reasoning for Task-Oriented Recommendation," *IEEE Cloud Computing*, vol. 2, no. 6, pp. 10–19, 2015.

[102] J. E. Kim, X. Fan, and D. Mosse, "Empowering End Users for Social Internet of Things," in *IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2017, pp. 71–82.

[103] R. Girau, S. Martis, and L. Atzori, "Lysis: A Platform for IoT Distributed Applications over Socially Connected Objects," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 40 – 51, 2017.

[104] J. F. Colom, H. Mora, D. Gil, and M. T. Signes-Pont, "Collaborative Building of Behavioural Models based on Internet of Things," *Computers & Electrical Engineering*, vol. 58, p. 385ñ396, 2017.

[105] S. Lee, S. Kim, K. Choi, and T. Shon, "Game theory-based Security Vulnerability Quantification for Social Internet of Things," *Future Generation Computer Systems*, 2017.

[106] A. Ahmad, M. Khan, A. Paul, S. Din, M. M. Rathore, G. Jeon, and G. S. Choi, "Toward Modeling and Optimization of Features Selection in Big Data based Social Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 715–726, 2018.

[107] G. Han, L. Zhou, H. Wang, W. Zhang, and S. Chan, "A Source Location Protection Protocol based on Dynamic Routing in WSNs for the Social Internet of Things," *Future Generation Computer Systems*, vol. 82, pp. 689–697, 2018.

[108] Z. Song, Y. Sun, J. Wan, L. Huang, Y. Xu, and C.-H. Hsu, "Exploring Robustness Management of Social Internet of Things for Customization Manufacturing," *Future Generation Computer Systems*, vol. 92, pp. 846–856, 2019.

[109] M. Z. Hasan and F. Al-Turjman, "SWARM-based data delivery in Social Internet of Things," *Future Generation Computer Systems*, vol. 92, pp. 821–836, 2019.

[110] F. Al-Turjman, "5G-enabled Devices and Smart-spaces in Social-IoT: An Overview," *Future Generation Computer Systems*, vol. 92, pp. 732–744, 2019.

[111] B. Afzal, M. Umair, G. A. Shah, and E. Ahmed, "Enabling IoT Platforms for Social IoT Applications: Vision, Feature Mapping, and Challenges," *Future Generation Computer Systems*, vol. 92, p. 718ñ731, 2019.

[112] X. Chen, L. Sun, H. Zhu, Y. Zhen, and H. Chen, "Application of Internet of Things in Power-Line Monitoring," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2012, pp. 423–426.

[113] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a social structure to the Internet of Things," *IEEE communications letters*, vol. 15, no. 11, pp. 1193–1195, 2011.

[114] V. Beltran, A. M. Ortiz, D. Hussein, and N. Crespi, "A Semantic Service Creation Platform for Social IoT," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 283–286.

[115] "FIESTA-IoT - Federated Interoperable Semantic IoT Testbeds and Applications," accessed: October 2019. [Online]. Available: http://fiesta-iot.eu/

[116] M. Nitti, L. Atzori, and I. P. Cvijikj, "Network Navigability in the Social Internet of Things," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 405–410.

[117] A. Gyrard, C. Bonnet, and K. Boudaoud, "Enrich Machine-to-Machine Data with Semantic Web Technologies for Cross-domain Applications," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 559–564.

[118] C. Savaglio and G. Fortino, "Autonomic and Cognitive Architectures for the Internet of Things," in *International Conference on Internet and Distributed Computing Systems*. Springer, 2015, pp. 39–47.

List of figures

List of tables