Facilitating mobile crowdsensing from both organizers' and participants' perspectives

Leye Wang

DOCTORAT EN CO-ACCREDITATION
TÉLÉCOM SUDPARIS - INSTITUT MINES-TÉLÉCOM
ET L'UNIVERSITÉ PIERRE ET MARIE CURIE - PARIS 6

Spécialité: Informatique

Ecole doctorale: Informatique, Télécommunications et Électronique de Paris

Présentée par

Leye WANG

Facilitating Mobile Crowdsensing from both Organizers' and Participants' Perspectives

Soutenue le 18 Mai 2016

Devant le jury composé de:

Hervé RIVANO	Rapporteur	Chargé de recherche, HDR, INRIA - France
Stéphane GALLAND	Rapporteur	Maître de conférences, HDR, UTBM - France
Steven MARTIN	Examinateur	Professeur, HDR, Université Paris-Sud - France
Djamal ZEGHLACHE	Examinateur	Professeur, HDR, Télécom SudParis - France
Farid NAIT-ABDESSELAM	Examinateur	Professeur, HDR, Université Paris Descartes - France
Abdallah MHAMED	Directeur de Thèse	Maître de conférences, HDR, Télécom SudParis - France
Daqing ZHANG	Co-Encadrant	Directeur d'études, Télécom SudParis - France

Thèse numéro : 2016TELE0008

Declaration

I, Leye Wang, hereby declare that this dissertation presents the results of my original research. I have not copied from any others' work or from any other sources except where due reference or acknowledgement is made explicitly in the text, nor has any part been written for me by another person.

Leye Wang

March 2016

Abstract

With the prevalence of sensor-rich equipped smartphones in recent years, *Mobile Crowd-Sensing (MCS)* becomes a promising paradigm to facilitate urban sensing applications, such as environment monitoring and traffic congestion detection. MCS achieves the urban sensing goal by leveraging the mobility of mobile users, the sensors embedded in mobile phones and the existing wireless infrastructure. Compared to the traditional urban sensing paradigms relying on the expensive specialized infrastructures, MCS can cheaply and efficiently sense large urban regions.

During the process of MCS, for both participants and organizers, there exist a variety of concerns affecting whether an MCS task can obtain enough satisfactory sensing results. For participants, these concerns include *smartphone energy consumption*, *mobile data cost*, *privacy*, and *incentive*, which significantly influence the participants' willingness to attend an MCS task. For organizers, *quality* and *budget* are two primary concerns, which, however, have some intrinsic conflicts, e.g., to achieve a better task quality, often more budget needs to be consumed; thus, balancing the trade-off between quality and budget is a vital issue for organizers to carry out satisfactory MCS tasks. How to appropriately address these participants' and organizers' concerns during the process of MCS has attracted a huge amount of research interest nowadays.

Following this research direction, in this dissertation, aiming to address both participants' and organizers' concerns, we propose two categories of mechanisms for MCS tasks.

The first category of mechanism is **collaborative data uploading** in crowdsensing, where participants help each other through opportunistic encounters in the data uploading process of crowdsensing, in order to save energy consumption, mobile data cost, etc. Specifically, two works in this dissertation belong to collaborative data uploading,

- **Save the participants' smartphone energy consumption and mobile data cost via collaborative data uploading.** Usually two classes of participants exist in MCS tasks: *data-plan users*, who are mostly concerned with energy consumption; *non-data-plan users*, who are more sensitive to data cost. Inspired by the observation that non-data-plan users can save mobile data cost by offloading data to data-plan users or uploading data via free Bluetooth/WiFi gateways, and data-plan users can save energy by piggybacking or using more energy-efficient networks than 3G, we propose a collaborative data uploading mechanism, called *effSense*, which can make intelligent decisions about the appropriate timing and network to upload data for each participant, in order to save both energy consumption and data cost.
- **Reduce the organizers' mobile data incentive budget in collaborative data uploading.** Paying participants' money to cover their mobile data cost is an effective incentive method for the organizers to eliminate the participants' concern about the mobile data cost. Under the collaborative data uploading mechanism (i.e., certain participants can offload data to others for saving mobile data cost), we further study how to partition the users into two groups corresponding to two price plans *Unlimited*

Data Plan and *Pay As You Go*, so as to minimize the overall mobile data cost for all participants, i.e., the mobile data incentive budget of the organizers. Based on predicting users' mobility patterns and sensed data size, we propose a genetic algorithm called *ecoSense* to do the participant partitioning.

The second category of mechanisms is called **sparse mobile crowdsensing**. To reduce the sensing costs, such as energy and incentive, while still achieving satisfactory data quality, we propose sparse mobile crowdsensing to intelligently select only a small part of the target area for sensing, while inferring the data of the remaining unsensed area with high accuracy. Specifically, we also conduct two works for sparse mobile crowdsensing.

- **Reduce the organizers' incentive budget while guaranteeing a required level of quality via sparse mobile crowdsensing.** Inspired by the spatial and temporal correlations among the data sensed in different sub-areas, we propose *sparse mobile crowdsensing*, which leverages such correlations to significantly reduce the required number of sensing tasks allocated, thus lowering the organizers' incentive budget, yet ensuring the data quality. We implement a sparse mobile crowdsensing framework, called *CCS-TA* (*Compressive CrowdSensing Task Allocation*), combining the state-of-the-art compressive sensing, Bayesian inference, and active learning techniques, to dynamically select a minimum number of sub-areas for sensing in each cycle, while inferring the missing data of unsensed sub-areas under the data quality guarantee.
- **Protect the participants' location privacy through quality-optimized differential location obfuscation in sparse mobile crowdsensing.** To protect participants' location privacy in sparse mobile crowdsensing, we adopt a differential location privacy notion called ϵ*-region-ambiguity* to provide guaranteed level of privacy regardless of an adversary's prior knowledge. As differential location privacy protection can cause data quality loss due to the discrepancy between the original and obfuscated locations, we propose a linear program, called *DUM-*ϵe (*Data Uncertainty Minimization* under the constraints of ϵ*-region-ambiguity* and *evenly-distributed obfuscation*), to select the optimal location obfuscation function that attempts to minimize the data quality loss incurred by the obfuscation.

Finally, we summarize the insights learned from both collaborative uploading and sparse mobile crowdsensing mechanisms, and discuss future research directions, such as how to enhance our mechanisms to cope with malicious behaviors of participants, how to adapt our mechanisms to more innovative MCS applications in smart city scenarios, and how to integrate all the techniques proposed in this dissertation into a unified MCS platform.

Keywords

Mobile Crowdsensing, Energy consumption, Mobile data cost, Data quality, Location privacy, Delay-tolerant data uploading, Data relay, Piggybacking, Task allocation, Compressive sensing, Bayesian inference, Active learning, Location obfuscation, Differential privacy, Linear program.

Acknowledgments

Four years, like a flush, finally lead me here. Not long, not short. But they must be the most important four years for me so far.

First of all, I would like to thank my advisors, *Prof. Daqing Zhang* and *Prof. Abdallah Mhamed*, for offering me the opportunity to get started on the way to the research. They encouraged and inspired my research, and shared me with their valuable experience in academia. Without their guidance, I would not be able to overcome the difficulties during my PhD study and to complete this dissertation. The talk and discussion with them always bring me new and deep understanding of what is a meaningful research work, how to do it, as well as how to express your ideas to others in an easy way. I also want to express my gratitude to *Prof. Bing Xie*, my master thesis advisor at Peking University, China. Without his encouragement, I would not choose to pursue the PhD degree; besides, I really appreciated his kindly offering that makes me work as a short-time visiting student at Peking University during my PhD studies, where I learned many research and life advices from him.

Second, I appreciate all my colleagues and friends during my studies at Institut Mines Télécom/Télécom SudParis, particularly *Dr. Xiao Han*, *Dr. Mingyue Qi*, *Dr. Chao Chen*, *Dr. Dingqi Yang*, *Dr. Haoyi Xiong*, *Dr. Jian Zhang*, *Longbiao Chen*, *Zaiyang Tang*, *Tianben Wang*. The life with them in France would be always a precious memory for me.

Third, I would like to thank my co-authors who assist me tremendously in conducting research works and writing paper manuscripts: *Dr. Zhixian Yan*, *Prof. J. Paul Gibson*, *Dr. Animesh Pathak*, *Dr. Brian Y. Lim*. Their brilliant suggestions and comments helped to enhance my research works substantially, and are indispensable factors giving rise to the birth of this dissertation.

Finally, I express my deepest gratitude to my parents *Weimin Sheng* and *Jian Wang* for their unconditional support and love since I came to this world. I would not be able to accomplish all these words without them.

Leye Wang

Feb. 2016

Evry, France

Publications

The following published, in press or submitted papers are partial outputs during my Ph.D. studies in Telecom SudParis and UPMC.

Journal Articles

- **Leye Wang**, Daqing Zhang, Yasha Wang, Chao Chen, Xiao Han and Abdallah Mhamed. *Sparse Mobile Crowdsensing: Challenges and Opportunities*. IEEE Communications Magazine, 2016, accepted.
- **Leye Wang**, Daqing Zhang, Haoyi Xiong, J. Paul Gibson, Chao Chen and Bing Xie. *ecoSense: Minimize Participants' Total 3G Data Cost in Mobile Crowdsensing Using Opportunistic Relays*. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016, accepted.
- **Leye Wang**, Daqing Zhang, Zhixian Yan, Haoyi Xiong and Bing Xie. *effSense: A Novel Mobile Crowdsensing Framework for Energy-Efficient and Cost-Effective Data Uploading*. IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 45, no. 12, 2015, pp. 1549-1563.
- Longbiao Chen, Daqing Zhang, Xiaojuan Ma, **Leye Wang**, Shijian Li, Zhaohui Wu and Gang Pan. *Container Port Performance Measurement and Comparison Leveraging Ship GPS Traces and Maritime Open Data*. IEEE Transactions on Intelligent Transportation Systems, 2016, accepted.
- Haoyi Xiong, Daqing Zhang, Guanling Chen, **Leye Wang**, Vincent Gauthier and Laura E. Barnes. *iCrowd: Near-Optimal Task Allocation for Piggyback Crowdsensing*. IEEE Transactions on Mobile Computing, 2016, accepted.
- Xiao Han, **Leye Wang**, Reza Farahbakhsh, Angel Cuevas, Ruben Cuevas and Noel Crespi. *CSD: A Multi-User Similarity Metric for Community Recommendation in Online Social Networks*. Expert Systems with Applications, 2016, accepted.
- Haoyi Xiong, Daqing Zhang, **Leye Wang**, J.Paul Gibson and Jie Zhu. *EEMC: Enabling Energy-Efficient Mobile Crowdsensing with Anonymous Participants*. ACM Transactions on Intelligent Systems and Technology, vol. 6, no. 3, 2015, pp. 39:1-39:27.
- Haoyi Xiong, Daqing Zhang, **Leye Wang** and Hakima Chaouchi. *EMC3: Energy-efficient Data Transfer in Mobile Crowdsensing under Full Coverage Constraint*. IEEE Transactions on Mobile Computing, vol. 14, no. 7, 2015, pp. 1355-1368.
- Xiao Han, **Leye Wang**, Noel Crespi, Soochang Park and Angel Cuevas. *Alike People, Alike Interests? Inferring Interest Similarity in Online Social Networks*. Decision Support Systems, vol. 69, 2015, pp. 92-106.

- Daqing Zhang, **Leye Wang**, Haoyi Xiong and Bin Guo. *4W1H in Mobile Crowd Sensing*. IEEE Communications Magazine, vol. 52, no. 8, 2014, pp. 42-48.

Conference Papers

- **Leye Wang**, Daqing Zhang, Animesh Pathak, Chao Chen, Haoyi Xiong, Dingqi Yang and Yasha Wang. *CCS-TA: Quality-Guaranteed Online Task Allocation in Compressive Crowdsensing*. Proc. UbiComp, Sept. 2015, Osaka, Japan, pp. 683-694.
- **Leye Wang**, Daqing Zhang and Haoyi Xiong, *effSense: Energy-Efficient and Cost-Effective Data Uploading in Mobile Crowdsensing*, Proc. UbiComp Adjunct, Sept. 2013, Zurich, Switzerland, pp. 1075-1086.
- Xiao Han, **Leye Wang**, Son N. Han, Chao Chen, Noel Crespi and Reza Farahbakhsh. *Link Prediction for New Users in Social Networks*. Proc. ICC, Jun. 2015, London, UK, pp. 1250-1255.
- Haoyi Xiong, Daqing Zhang, Guanling Chen, **Leye Wang** and Vincent Gauthier. *CrowdTasker: Maximizing Coverage Quality in Piggyback Crowdsensing under Budget Constraint*. Proc. PerCom, Mar. 2015, St. Louis, USA, pp. 55-62.
- Chao Chen, Daqing Zhang, **Leye Wang**, Xiaojuan Ma, Xiao Han and Edwin Sha. TaxiExp: A Novel Framework for City-wide Package Express Shipping via Taxi CrowdSourcing. Proc. UIC, Dec. 2014, Bali, Indonesia.
- Daqing Zhang, Haoyi Xiong, **Leye Wang** and Guanling Chen. *CrowdRecruiter: Selecting Participants for Piggyback Crowdsensing under Probabilistic Coverage Constraint*. Proc. UbiComp, Sept. 2014, Seattle, USA, pp. 703-714.
- Longbiao Chen, Daqing Zhang, Gang Pan, **Leye Wang**, Xiaojuan Ma, Chao Chen and Shijian Li. *Container Throughput Estimation Leveraging Ship GPS Traces and Open Data*. Proc. UbiComp, Sept. 2014, Seattle, USA, pp. 847-851.
- Xiao Han, **Leye Wang**, Soochang Park, Angel Cuevas and Noel Crespi. *Alike People, Alike Interests? A Large-Scale Study on Interest Similarity in Social Networks*. Proc. ASONAM, Aug. 2014, Beijing, China, pp. 491-496.

Under Review

- **Leye Wang**, Daqing Zhang, Dingqi Yang, Brian Y. Lim, Haoyi Xiong and Abdallah Mhamed. *Differential Location Privacy for Sparse Mobile Crowd-Sensing with Data Quality Loss Reduction*. Submitted to ACM/IEEE Transactions on Networking.

Contents

Part I

Research Background and Outline

Chapter 1

Introduction

Contents

1.1 What is Mobile Crowdsensing?

Mobile Crowd Sensing (MCS) — a term coined by *Ganti et al.* [1], has recently spurred lots of research interest. Similar to the notion of participatory sensing and human-centric computing [2], mobile crowd sensing refers to *the sensing paradigm* in which mobile users with sensing and computing devices are tasked to collect and contribute data in order to enable various applications. MCS applications leverage the sensing, computing and wireless communication capability offered by the millions of mobile devices, e.g., Android phones, iPhones or iPads, already "deployed in fields" and carried by people in their daily lives. MCS has successfully extended the sensing scope from single physical space to community and city scale, from recognizing hazard environmental situations to informing collective behavior of crowds.

Thus, nowadays, a huge number of tasks can be completed by MCS, even though these tasks could have heterogeneous properties from each other. From spatial and temporal perspectives, MCS tasks can be classified into four categories:

- *Short-range Short-term*: This category corresponds to the MCS task executed by participants in physical proximity for a short time, for example, sensing activities and face-to-face interactions among participants in a conference.
- *Long-range Short-term*: It corresponds to the MCS task executed by participants staying far apart for a short time, e.g., collecting users' captured images and audio

clips across the city for rescue when flooding occurs.

- *Short-range Long-term*: It corresponds to the MCS task executed by participants in physical proximity for a long time, e.g., sensing students' activities and interactions in a school for several weeks or semesters.
- *Long-range Long-term*: This category corresponds to the MCS task executed by participants staying far apart for a long time, e.g., engaging citizens to monitor environment (e.g. air quality, noise) of a city for several months.

To carry out such a broad spectrum of tasks, various kinds of MCS applications have been proposed and implemented in both academic and industry areas. Some representative MCS applications are listed as follows:

- *Environmental Applications*: Many MCS applications are designed to monitor environmental data, such as temperature, noise, and air quality. For example, PEIR [3] generates a personal environment influence report for each participant by collecting one's GPS data and other context data (e.g., weather and traffic). A crowdsourced urban noise map can be obtained via the Ear-phone system [4]. SakuraSensor [5] is a participatory sensing system to recommend cherry-lined roads for comfortable driving, through analyzing short videos recorded by participants' in-vehicle smartphones.
- *Infrastructure Applications*: This type of MCS applications attempts to measure the large-scale phenomenon related to public infrastructure, e.g., traffic condition monitoring and location characterization. Cartel [6] and Nericell [7] are two representative MCS applications of traffic congestion monitoring. CrowdSense@Place [8] links a place to place categories (e.g. store, restaurant) by aggregating the opportunistically captured images and audio clips from participants' smartphones.
- *Social Applications*: Another category of MCS applications aims to deal with social issues by collecting participants' daily life traces and social interactions. In SociableSense [9], for instance, users are provided with a quantitative measure of their sociability via their sensed office behavior from smartphones. DietSense [10] builds up a participatory sensing system for diabetics where they take photos of the food they eat and compare their eating habits with each other.

Despite the distinct design intention and objectives, in general, the life-cycle of an MCS application consists of four stages: *creating MCS applications* according to the requirements, *assigning sensing tasks* to participants, *executing the task* (sensing, computing and uploading) on the mobile device of individual participant, and *collecting and processing sensed results* from participants. Fig. 1.1 illustrates the four stages, and the key functionalities of each stage are described as follows:

- *Task Creation:* The MCS organizer creates an MCS task through providing the participants with the corresponding mobile sensing applications that would be deployed in the participants' smartphones later.
- *Task Assignment:* After the organizer creates an MCS task and the corresponding mobile task applications, the next stage is *task assignment* - recruiting participants and

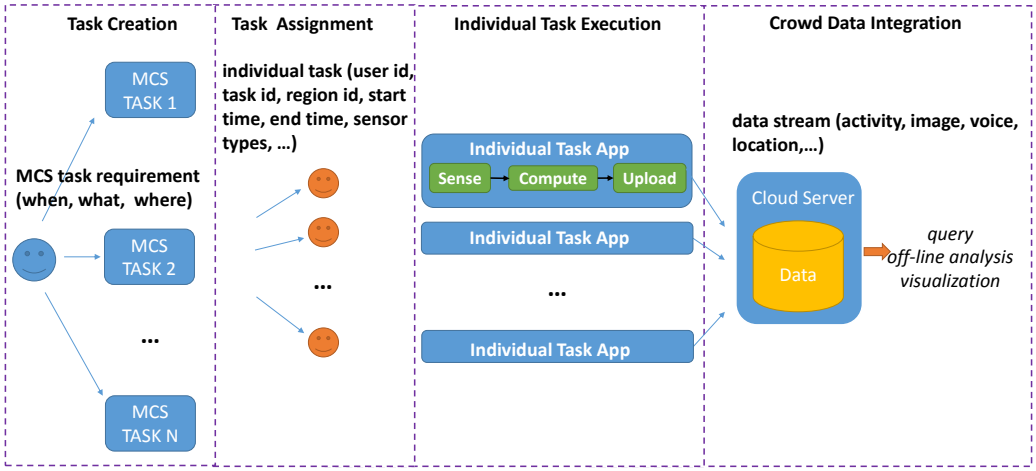


Figure 1.1: The Four-stage Life Cycle of Mobile Crowd Sensing Process

assigning them with individual sensing tasks that are supposed to run in each participant's mobile device. Finding enough and appropriate crowd sensing participants is the core issue in this stage.

- *Individual Task Execution:* Once receiving the assigned sensing task, a participant would try to finish it within a pre-defined MCS task duration in parallel with other tasks. This stage is called *individual task execution* stage, which can be further divided into 3 sub-stages - *sensing*, *computing*, and *data uploading*.
- *Crowd Data Integration:* This stage takes the data streams collected from all the participants as input, aggregates the data and provides end users with what they need in the appropriate format. For some MCS applications [11], the data processing in this stage is quite straightforward — central servers store the data and provide interfaces to end users for data query and sharing. While other MCS applications [8, 3, 9] employ complicated algorithms to integrate data and extract high-level collective intelligence from the raw data of large crowds.

1.2 Concerns for Mobile Crowdsensing Organizers and Participants

During the whole MCS task process, many concerns exist for both participants and organizers, which are necessary to be addressed well in order to successfully complete an MCS task. For participants, the concerns include energy consumption, mobile data cost, privacy, obtained incentive, etc.

- *Energy Consumption.* Energy consumption is directly related to the battery life of a

participant's smartphone. Usually the energy consumption is incurred during the "*individual task execution*" stage when the smartphone needs to do sensing, computing and uploading to finish a task. If the energy consumption of an MCS task is too high, it will severely discourage a potential user from becoming a crowd participant.

- *Mobile Data Cost*. Mobile data cost is associated with the fees that a participant needs to pay to her telecommunication operator, which is primarily caused by the *uploading* step in the "*individual task execution*" stage. According to the different data plans held, the participants may have different sensitiveness on the data cost concern: while the users with *unlimited data plan* may not consider it seriously, the others, who have to pay more for uploading more data (e.g., *pay-as-you-go* data plan), would see it as one of the most important factors relevant to the willingness of participation.
- *Privacy*. Privacy is an important factor that will impact the willingness of the participants to join in the sensing jobs. If a potential participant thinks that her privacy would be violated during the MCS task, apparently she would not like to participate in the task. One key privacy concern in MCS tasks is the *location privacy*, because usually to complete a sensing job such as environment monitoring, the participants will upload their sensed data accompanying with their actual locations, which may expose the participants under the attacks such as stalking, identity theft, and breach of sensitive information. In the *uploading* step of the "*individual task execution*" stage, some location protection techniques, e.g., anonymization and obfuscation, could be employed to relieve the participants' concern about the location privacy.
- *Incentive*. All the above-mentioned concerns may decrease a participant's willingness to take part in an MCS task, thus certain kinds of incentives should be offered to the participants to keep their motivation in conducting sensing tasks and contributing high-quality data. Generally, incentives can be classified into three categories, *money*, *love*, and *glory*. More specifically, *money* is a participant's external financial gain; *love* is her intrinsic enjoyment of joining an MCS task; *glory* is her desire to be recognized by peers for her contributions [12]. How to design an incentive mechanism considering all the three types of incentives is a key issue for MCS applications, as it significantly affects whether the "*task assignment*" stage can recruit enough qualified participants to complete an MCS task.

For organizers, the concerns include the completion quality of the MCS task and budget that needs to conduct the task.

- *Quality*. Making the quality of an MCS task achieve a predefined objective, or as good as possible, will be a primary objective for an organizer. How to measure the quality may depend on specific MCS tasks, while some common factors can affect the quality of a wide spectrum of MCS applications, such as the uncertain low-quality or even forgery data uploaded by the participants, and the spatial-temporal coverage of the target area and sensing duration offered by the crowd-contributed data. How to improve quality go through the whole life-cycle of the MCS task. For example, in the "*task assignment*" stage, how to recruit more participants to obtain larger

spatial-temporal coverage; in the "*individual task execution*" stage, how to design the mobile-client application to improve the quality of uploaded data and prevent malicious behaviors of participants; in the "*crowd data integration*" stage, how to aggregate individuals' data to obtain a convincing overall MCS result.

- *Budget*. Budget is another critical concern that must be considered for an MCS organizer. As budget and quality have certain intrinsic conflicts for an MCS task (e.g, the allowed budget may be too low to achieve the desired MCS task quality) , how to balance the trade-off between the two concerns become a vital issue for the organizer. Especially, the above-mentioned monetary incentives that need to be paid to participants is an important part of the budget. Thus, "*how to reduce the monetary incentive budget while ensuring the MCS task quality at a satisfactory level?*" and/or "*how to improve the MCS task quality most significantly under a limited monetary incentive budget?*" become critical problems for organizers.

1.3 Dissertation Contributions and Chapter Outline

In this dissertation, aiming to address the concerns for both participants and organizers, we have proposed two categories of crowdsensing mechanisms, in each of them we design two specific approaches to address some of the organizers' or participants' concerns. Figure 1.2 illustrates the overall organization of the rest dissertation. In **Chapter 2**, we first review the existing related work in mobile crowdsensing area. The next chapters describe our main contributions. **Chapter 3** and **4** illustrate the first category of mechanism, **collaborative data uploading** in crowdsensing, where participants help each other in the *individual data uploading* step of crowdsensing, in order to save energy consumption, mobile data cost, etc. Specifically,

- In **Chapter 3**, we propose *effSense* — an energy-efficient and cost-effective data uploading framework, which utilizes adaptive delay-tolerant uploading schemes within fixed data uploading cycles, in order to help participants save energy consumption and mobile data cost. We classify participants into two classes: *data-plan users*, who are mostly concerned with energy consumption; *non-data-plan users*, who are more sensitive to data cost. In each cycle, effSense empowers the participants with a distributed decision making scheme to choose the appropriate timing and network to upload data. effSense reduces data cost for non-data-plan users by maximally offloading data to free Bluetooth/WiFi gateways or data-plan users encountered; it reduces energy consumption for data-plan users by piggybacking data on a call or using more energy-efficient networks rather than initiating new 3G connections. By leveraging the predictability of users' calls and mobility, effSense selects proper uploading strategies for both types of users. Our evaluation with the MIT Reality Mining and Nodobo datasets shows that effSense can reduce 55-65% energy consumption for data-plan users, and 48-52% data cost for non-data-plan users, respectively, compared to traditional real-time uploading schemes.

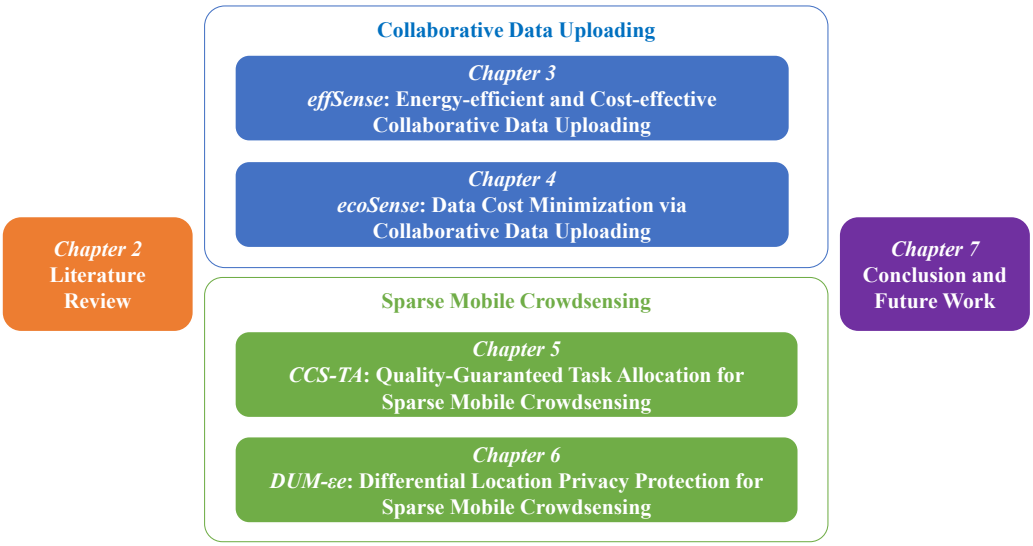


Figure 1.2: Organization of the rest dissertation

- In **Chapter 4**, to advance the work of effSense, we further study the problem of how to partition all the users into the two groups corresponding to the two price plans *Unlimited Data Plan* (*UnDP*, i.e., data-plan users in effSense) and *Pay As You Go* (*PAYG*, i.e., non-data-plan users in effSense), so as to minimize the total mobile data cost for all participants. The overall mechanism, including both participant partitioning and collaborative data uploading, is called *ecoSense*. The partitioning is based on the prediction of users' mobility patterns and sensed data size. The collaborative uploading, similar to effSense, is inspired by the observation that during the data uploading cycles, *UnDP* users could opportunistically relay *PAYG* users' data to the crowdsensing server without extra 3G cost, provided the two types of users are able to meet on a common local cost-free network (e.g. Bluetooth or WiFi Direct). We conduct our experiments using both the MIT Reality Mining and the SWIM simulation datasets. Evaluation results show that ecoSense could reduce total 3G data cost by up to ~50%, when compared to the direct-assignment method that assigns each participant to UnDP or PAYG directly according to the size of her sensed data.

Chapter 5 and **6** present the second category of mechanisms in this dissertation, called **sparse mobile crowdsensing**. To reduce the sensing costs, such as energy consumption and incentive budget, of crowdsensing tasks significantly while still achieving satisfactory data quality, sparse mobile crowdsensing intelligently selects only a small part of the target area for sensing in the *task assignment* step, while inferring the data of the remaining unsensed area with high accuracy in the *crowd data integration* step. Specifically,

- In **Chapter 5**, we leverage the spatial and temporal correlation among the data sensed in different sub-areas to significantly reduce the required number of sensing tasks allocated (corresponding to sensing costs), still ensuring the data quality. The proposed crowdsensing paradigm is named as *sparse mobile crowdsensing*. Then, we design a sparse mobile crowdsensing task allocation framework, called *CCS-TA* (*Compressive*

CrowdSensing Task Allocation), combining the state-of-the-art compressive sensing, Bayesian inference, and active learning techniques, to dynamically select a minimum number of sub-areas for sensing task allocation in each sensing cycle, while inferring the missing data of un-allocated sub-areas under a probabilistic data accuracy guarantee. Evaluations on real-life temperature and air quality monitoring datasets show that CCS-TA allocates 18.0-26.5% fewer tasks than baseline approaches. Specifically, for temperature monitoring, CCS-TA assigns tasks to only 15.5% of the sub-areas on average while the overall sensing error is kept below 0.25°C in 95% of the cycles.

- In **Chapter 6**, we aim to empower sparse mobile crowdsensing with a location obfuscation mechanism to protect participants' location privacy. Firstly we adopt a notion of differential location privacy for sparse mobile crowdsensing tasks, called ϵ*-region-ambiguity*, which provides a guaranteed level of privacy regardless of an adversary's prior knowledge about participants' location distribution; secondly we design a location privacy protection mechanism which requests each participant to obfuscate her location *in a distributed manner*, thus eliminating the need of a trustful third-party server. As differential location privacy protection can cause data quality loss due to the discrepancy between the original and obfuscated locations, we develop *DUM-*ϵe, a linear program which selects the optimal location obfuscation function and reduces data quality loss through *Data Uncertainty Minimization* under the constraints of ϵ*-region-ambiguity* and *evenly-distributed obfuscation*. We further speed up *DUM-*ϵe by reducing the number of constraints from $O(n^3)$ to $O(n^2)$ (n: number of sensing regions) with an approximation called *Fast DUM-*ϵe (*FDUM-*ϵe). Our evaluations with real-world environment and traffic monitoring datasets show that *DUM-*ϵe reduces the data quality loss by 15-45% compared to baseline mechanisms, with the same level of differential privacy guarantee; *FDUM-*ϵe incurs 2-6% more quality loss, but only needs 1% of the computation time compared to *DUM-*ϵe.

To conclude, **Chapter 7** summarizes the insights provided by this dissertation. Future research directions, such as how to enhance our mechanisms to cope with malicious behaviors of participants, how to adapt our mechanisms to more innovative MCS applications in smart city scenarios, and how to integrate all the techniques proposed in this dissertation into a unified MCS platform, are discussed in this final chapter.

Chapter 2

Literature Review

Contents

In this chapter, we review the existing research efforts in the MCS area. First, we go through the existing representative MCS applications and frameworks. Next, we extensively discuss the relevant works to various concerns for both participants and organizers. Specifically, for participants, we illustrate the existing works related to their concerns of *energy consumption*, *mobile data cost*, *privacy* and *incentive*; for organizers, we elaborate the studies focusing on the *quality* and *budget* issues. Finally, we briefly describe the relations of our works in this dissertation to the existing literature, and point out which participant and organizer concerns are addressed by each of our works.

2.1 Crowdsensing Applications and Frameworks

2.1.1 Crowdsensing Applications

Recent studies on mobile crowdsensing lead to a variety of applications in both academia and industry.

Applications in Academia

According to [1], MCS applications can be classified into three categories: *environment*, *infrastructure*, and *social* applications.

(i) Representative environment applications include *Ear-phone* [4], a urban-scale participatory noise sensing map, and *PEIR* [3], which provides the participants with personal environment impact reports. Smartphone based participatory air quality sensing mechanisms are also under active development in many research groups [13, 14]. Recently, a novel environment MCS application, *SakuraSensor* [5] is designed and implemented to leverage crowd-sourced video data to detect the beautiful cherry-lined roads.

(ii) For infrastructure applications, common crowdsensing scenarios include traffic congestion detection [6, 7], road speed monitoring [15], place characterization [8] and available parking spot detection [16]. For example, *CrowdSense@Place* [8] infers the category of a place (e.g. store, restaurant) based on the collective images and audio clips opportunistically captured and uploaded by the participants. *GreenGPS* [17] is a navigation service that uses participatory sensed data to map fuel consumption on city roads, thus guiding drivers to the most fuel-efficient routes between the original and target locations.

(iii) Social applications attempt to improve participants' experience in their daily social life via crowd data [9, 10, 18]. For instance, *SociableSense* [9] collects the data of participants' encounters and meetings with other people in the office, measures' each participant's sociability quantitatively, and gives the participants some suggestions for improving their social relationships. *Ubicon* [18] is a crowd social-computing platform, based on which *MyGroup* and *Conferator* applications are implemented to collect users' social interactions in a working group and a conference, respectively, so as to provide recommendations on users' future interactions.

Applications in Industry

In addition to academia, industry has also designed and implemented various off-the-shelf MCS applications. *Vaavud*[1] is a participatory wind speed sensing application, where the participants leverage the smartphone plug-in wind meters to sense the local wind speed and then share the information to other people. The crowdsourced traffic monitoring and navigation application, *WAZE*[2], has already appealed to more than 50 million users. *Placemeter*[3] recruits participants to suction-cup an old smartphone to their window and record the street view outside, so as to measure how many people come in and out of the place, with the objective of creating a crowdsourced placemeter world map. *Stereopublic*[4] requires participants to look for quiet places in their cities, geo-tag the places, and record the noise there, with the objective of crowdsourcing a quiet-place city map.

2.1.2 Crowdsensing Frameworks

A variety of general MCS frameworks are proposed in the literature, so as to support MCS application development, deployment, participant recruitment process, etc. To ease the

[1] http://vaavud.com
[2] https://www.waze.com
[3] https://www.placemeter.com
[4] http://www.stereopublic.net/

development of MCS applications, a programming framework, *Medusa* [19], is designed and implemented, which provides a high-level description language to specify an MCS application. Reddy et al. [20] propose a participant recruitment framework for MCS applications to identify appropriate participants considering their mobility patterns and participation habits. In addition to the mobility patterns, *McSense* [21] also attempts to consider social factors, such as collocations and social ties, when recruiting participants to enable various crowdsourced smart city applications. Regarding the data collection process, *CAROMM* [11] is designed to reduce the amount of data uploaded and the energy consumption on the smartphone side. Concerning participants' privacy, *AnonySense* [22] is proposed as a privacy-aware framework for realizing MCS applications over anonymous participants, where the sensed data from the participants is verified, yet anonymized. While ensuring user anonymity, Christin et al. [23] design a reputation framework for MCS applications, named *IncogniSense*, to measure the quality of user contributed data without needing to know users' true identities.

2.2 Participants' Concerns

For participants, a lot of concerns may affect their willingness and activeness in the participation of MCS tasks. In this section, we discuss the existing methods that can address some important participant concerns, including *energy consumption*, *mobile data cost*, *privacy* and *incentive*.

2.2.1 Energy Consumption

Recall that in the "*individual task execution*" stage of MCS life-cycle (Figure 1.1), three primary phases are included: *sensing*, *computing*, and *data uploading*. Accordingly, we summarize the exiting energy saving mechanisms for MCS participants into such three phases.

Energy Consumption in Sensing

Sensing is the process of acquiring data from sensors. Thus the energy consumption here is primarily incurred by the sensor itself. To reduce the energy consumption of sensing, one possible way is to implement a novel energy-efficient sensor to replace an old power-hungry sensor, while the sensing functionality is kept the same. Actually, this direction is especially pursued by industry sensor manufacturers as energy-efficiency is one of the most important criteria to measure the quality of a specific sensor. For example, the Bluetooth 4.0 sensors reduce the energy consumption significantly compared to the old-generation Bluetooth sensors by introducing the BLE low energy technology [24].

Another way of saving energy is to design new methodologies to use a kind of energy-efficient senors to finish a specific sensing task, which traditionally is done by another kind of more energy-hungry sensors. For example, for human body motion detection, Cohn

et al. [25] propose to use ultra-low-power passive static electric field sensing to replace traditional active accelerometer-based sensing.

In addition, how to schedule the sensors to do a sensing task also significantly impacts the energy consumption. Generally there are two scenarios for sensor scheduling. (i) When only a specific kind of sensor is needed to do the task, how to adjust the sampling frequency is often the key of sensor scheduling algorithm. (ii) When heterogeneous kinds of sensors can collaboratively conduct a sensing task, the scheduling is more challenging as the scheduling algorithms need to decide which sensors need to be activated under which scenarios. Take location tracking as a representative sensing task example. While GPS is the most common sensor to track a user's locations, other sensors such as accelerometers, compass, WiFi, and Bluetooth can also assist location tracking. In [26], focusing on the GPS sensor, authors propose an energy-efficient scheduling algorithm called *EnTracked* to adaptively adjust the GPS sampling rates to robustly track a user's location. By extending EnTracked, the same authors propose an energy-efficient trajectory tracking system named *EnTracked*$_T$ [27], which adds the adaptive sensor management for other sensors such as compass and accelerometers. *a-Loc* [28] and *RAPS* [29] consider the WiFi, celltower and Bluetooth into the location tracking process, and study how to schedule all these heterogeneous sensors to make an energy-accuracy trade-off for the smartphone positioning.

Energy Consumption in Computing

The energy consumption of *computing* primarily refers to the energy consumed by the smartphone processors during the individual task execution. In [30], authors propose a multi-processor architecture composed of main processors and supplementary low-power processors for smartphones; thus, a computing task can run on either main or low-power processors according to the task characterization, in order to optimize the overall energy consumption. Actually, a similar idea has been proposed by ARM, called *big.LITTLE* architecture[5], which is a heterogeneous computing architecture coupling low-power processor cores (*LITTLE*) and more powerful and power-hungry ones (*big*). Recently, many of the top-end smartphone processors have already followed this architecture, such as *Kirin 950* and *Exynos 8890*.

Another direction of reducing computing energy consumption lies on the technique called *code offloading* or *remote execution*, where the smartphone delegates the computation tasks to resource-rich infrastructure, e.g., a workstation in the same local area network. Actually, there exists a trade-off between the energy consumption of computation and data transmission. Only when the energy consumption of computation is larger than that of data transmission, the energy can be saved. Thus, in order to achieve the optimal energy-efficiency, deciding which part of the code should be offloaded and which should not is the key research challenge. A lot of research efforts have been done on this area, such as *MAUI* [31], *CloneCloud* [32], and *ThinkAir* [33]. As energy-efficient code offloading can be seen as an important use case in mobile cloud computing, interested readers can find much more relevant works from the surveys on mobile cloud computing [34, 35]

[5]http://www.arm.com/zh/products/processors/technologies/biglittleprocessing.php

Energy Consumption in Data Uploading

The final phase of individual sensing task execution is *uploading* the data to the MCS server. To reduce the energy consumption in data uploading, the first kind of solutions are trying to use relatively low-power wireless communication methods, such as WiFi, to upload data, rather than directly using 3G/4G. However, waiting for the smartphones to connect to the low-power wireless infrastructures often means some delay in data uploading. To deal with this energy-delay tradeoff, Ra et al. [36] design an online algorithm *SALSA*, which can automatically adapt to channel conditions and decide whether and when to defer a data transmission. Ma et al. [37] study the MCS paradigm whose data collection mainly relies on the opportunistic data transmission among mobile participants via short-range radio communications (e.g., Bluetooth, WiFi Direct) instead of 3G/4G, which can help participants to save energy consumption caused by data uploading.

Another possible way to upload data with low energy consumption is paralleling data uploading with phone calls, which can save up to 75–90% of energy [38]. Based on this observation, Lane et al. [39] propose an energy-efficient mobile crowdsensing system, called *Piggyback CrowdSensing (PCS)*, which attempts to upload data to the server when users place phone calls or use applications. Actually, such mobile smartphone app opportunities can also be exploited in the *sensing* and *computing* phase of an individual task, thus PCS is able to save smartphone energy during the whole process of individual task execution.

In addition, reducing the amount of data that needs to be uploaded may also lower the energy consumption. For example, compressing data before uploading can save energy if the energy required to compress data is less than the energy required to send it [40]. By compromising slightly on data quality, Musolesi et al. [41] propose the energy-efficient data uploading strategies that upload only part of continuous sensed data while inferring the rest on the server.

In summary, energy conservation is a widely studied research area in mobile sensing, which roughly consists of three phases, *sensing*, *computing* and *uploading*. It is worth noting that some proposed methods may save energy of one phase while increase the energy consumption for another phase. For example, *compression* [40] increases the computing energy consumption and decreases the uploading energy consumption; on the contrary, *code offloading* [31, 32, 33] does the opposite: consuming more energy in uploading and less energy in computing. Therefore, it is still quite challenging to integrate different kinds of energy-saving methods and trade off the energy consumption of different phases, in order to achieve the 'optimal' energy efficiency for the 'individual task execution' stage of a specific MCS application.

Note that in this section we primarily focus on the energy consumption for an individual participant and do not mention the overall energy consumption of all the participants. Concerning the overall energy consumption, besides the energy-saving methods used in the individual task execution, it also significantly depends on the design of task assignment and crowd data integration mechanisms. Briefly speaking, the fewer recruited participants and the less uploaded data, the less overall energy consumption. Actually, how to reduce the number of recruited participants and the amount of uploaded data, while still making

the MCS application achieve a satisfactory level of quality, is probably the most important issue for organizers, as they usually have a limited budget for collecting data and need to use it efficiently. We thus leave the discussion of the state-of-the-art works on this topic to the later Section 2.3, *Organizers' Concerns*.

2.2.2 Mobile Data Cost

Mobile data cost is a key concern of current MCS participants, as it is associated with the communication fees incurred, especially for the participants who only hold a limited data plan. A direct way to eliminate data cost concern is to allow participants to choose to upload data only via WiFi (can also save energy consumption as discussed above). This solution may incur some delay between sensing and uploading: according to a user study in Seoul [42], which consists of 97 participants and lasts for 18 days in 2010, WiFi can upload about 65% of total data immediately without any delay; for more than 90% of data to be uploaded, a delay more than 1 hour is usually required. Thus, WiFi-only uploading is suitable for delay-tolerant MCS applications, e.g., *CrowdSense@Place* [43] allows a 24-hour delay and waits WiFi connections to upload data. As the number of WiFi hotspots are increasing tremendously[6], it is expected that in near future, the delay between sensing and uploading by WiFi can be significantly shortened, so that more MCS applications may be satisfied via the WiFi-only uploading strategy.

While for the MCS applications that require real-time data uploading, e.g., traffic condition monitoring application where real-time user feedbacks are critical for improving the quality of service, the delay incurred by WiFi-only uploading may not be tolerant and thus other solutions are desired. Instead of eliminating mobile data cost, these applications usually loose the objective to reduce mobile data cost by decreasing the data uploaded as much as possible. To this end, the methods of lossless data compression [40] and partial data uploading [41], which have been discussed above to save energy, can also help to conserve mobile data cost. Inferring high-level data from raw sensed data and only uploading high-level data to the server can also save data cost for smartphones, but this may increase the energy consumption if the computation is intensive. A detailed analysis about the energy and data cost trade-off is conducted on *SoiableSense* by considering different configurations (whether running on smartphone or on server) of a speaker identity task [9].

2.2.3 Privacy

Participants share a variety of sensed data in MCS, including time, location, pictures, sound, biometric data, environment data, etc., which may incur privacy breaches if such data are exposed to adversaries. For example, time and location can leak participants' privacy-sensitive information including home and workplace locations, as well as their habits [44];

[6] According to the WiFi growth map (http://www.ipass.com/wifi-growth-map/), the growth of WiFi hotspots from 2013 to 2015 is more than 270%.

even the camera is oriented from the participant, the pictures may opportunistically capture some other people's faces and thus leak the participant's social relationship [45]. Therefore, providing countermeasures to these potential privacy attacks to participants is another important issue in MCS to encourage participants' activeness and protect their security. Inspired by a decent survey on preserving privacy in MCS [45], we categorize the state-of-the-art privacy protection approaches in MCS into three classes: *privacy preference setting*, *anonymous task allocation*, and *privacy-preserving data reporting*.

Privacy Preference Setting

A straightforward countermeasure to the privacy threats is to give the users the opportunities to express their privacy preferences, e.g., when and where does a participant allow her smartphone to do sensing tasks, and what degree of data granularity that she is willing to share with the MCS server (e.g., for location, sharing a precise GPS position or just a city name). This mechanism has been widely adopted in state-of-the-art MCS applications [46, 47]. However, often a participant cannot properly translate her conception of privacy into a privacy preference setting due to not fully understanding the implications of the setting [48]; especially, users usually underestimate the privacy risks they would face when setting privacy preferences [49].

Anonymous Task Allocation

During the task allocation stage, participants download tasks from the server and this would reveal the participants' locations (e.g., via IP address) at precise timestamps. Even with pseudonyms, participants' workplaces and home may be inferred over multiple task downloads [50]. To tackle this privacy leakage, *Kapadia et al.* [51] proposes to use *tasking beacons* to broadcast tasks to participants without the interaction between participants and central server. *Shin et al.* [52] suggest the participants to accept tasks where people density is high, so as to increase the difficulty for the server to identify the participants from their locations. Besides, the connections to the central server can also be anonymized via routing schemes such as *The Onion Router* (TOR) to hide participants' IP addresses, thus their locations [52].

Although hiding users' identities during the task allocation stage can protect participants' privacy, it also brings new challenges such as *how to pay the anonymous participants correct incentives* and *how to identify the malicious participants who report forgery data*. Some potential solutions have been proposed in the literature [53, 54, 23] to address these issues, but it still seems to be rarely applied in real MCS applications, perhaps due to the complexity of the proposed mechanisms.

Privacy-Preserving Data Reporting

Rather than directly sending raw sensed data to the server, how to manipulate the reported data to relieve the potential privacy breaches has attracted plenty of research interests.

Location privacy is perhaps the most widely studied among different kinds of sensed

data, as it is a common privacy issue in a variety of applications, such as location based services, online social networks, not restricted to MCS. Many location privacy-preserving mechanisms can be categorized into *obfuscation* methods, which seek to confuse the adversary with either *inaccurate* or *imprecise* locations [55, 56]. *Inaccuracy* means giving a different location from the actual one, and *imprecision* means giving a plurality of possible locations [56]. Among the obfuscation mechanisms, a popular mechanism is *cloaking* [57, 55, 58]. It employs *imprecision* to process location-based queries relative to a larger cloaked region, compared to the smaller regions or cells, where a user can be uniquely located. For instance, a cloaked region satisfying *l-indiscernible* [55] is generated by including the user's actual cell along with other nearby $l-1$ cells; and a cloaked region meeting *k-anonymity* means that at least k participants are located in this region [57]. Another popular type of obfuscation mechanism is using *dummy points*, which also employs *imprecision*, by adding the actual location to a set of dummy locations [59]. Recently, location based services have introduced *differential privacy* [60, 61] into location privacy protection, which is independent of an adversary's prior knowledge [62, 63, 64, 65]. Differential location privacy protection often employs *inaccuracy* to alter or transform the user's actual location to another obfuscated location by adding appropriate Laplace or Exponential noises [62, 63].

Besides location information, privacy-preserving technologies such as data perturbation, aggregation and pre-processing methods are also applied into other kinds of sensed data. *Epstein et al.* [66] employ value sensitive design to consider how to share fine grained step activity via data transformations that maximizes benefits while minimizing privacy harms. *Ganti et al.* [67] design a privacy-preserving architecture *PoolView*, which uses client-side data perturbation to ensure individuals' privacy, and community-wide reconstruction techniques to compute the aggregate information of interest such as traffic and weight. *Shi et al.* [68] propose a privacy-preserving solution, called *PriSense*, to support a wide range of statistical aggregation functions on the central server, such as Sum, Average, Variance, and Count, based on the concept of data slicing and mixing. Extracting features from raw data on the client-side to remove privacy-sensitive information, and just uploading the processed summaries is another kind of common countermeasures, e.g., used in social sensing [69] and noise monitoring [70]. In addition, secure data storages, such as *personal data vaults* [71] and *virtual individual servers* [72], are used to store raw sensed data from mobile device . These storages are fully controlled and accessed by the devices' owner. Then the owner can choose which information to report, or process the raw data to eliminate sensitive information before reporting.

2.2.4 Incentive

As above mentioned, participating in the MCS tasks may incur extra smartphone energy consumption, mobile data cost, and potential privacy leakage for users; thus, usually some incentives are offered to participants to mitigate such concerns, motivate them to accept the MCS tasks and contribute high-quality data. In this section, according to [12], we classify incentives into three classes including *money*, *love* and *glory*, and describe some representative works for each class. Specifically, *money* is a participant's external financial

gain; *love* is her intrinsic enjoyment of joining an MCS task; *glory* is her desire to be recognized by peers for her contributions. Actually, incentive mechanism design is one of the hottest research directions in MCS, so for a thorough overview, we recommend readers to refer to the survey on MCS incentive mechanism, such as [73].

Money

Paying participants some *money* may be the most commonly used incentive method in MCS. Designing an appropriate incentive mechanism to decide how much money should be paid to each participant, so as to keep the participants motivated and meantime the total incentive cost within the organizer's budget, is the key research challenge. To address this issue, generally two kinds of research works are conducted, one is to carry out experimental studies to compare different kinds of monetary incentive schemes in real MCS applications; the other is to design a theoretical incentive framework based on some mathematical abstraction and methodologies [73].

For experimental studies, Reddy et al. [74] compare *micro-payment* (money paid is proportional to the amount of data contributed) and *macro-payment* (each participant gets equal money), and verify that micro-payment works more efficiently; they also observe that introducing participant competition can increase user activeness at the beginning, but this high activeness would burn out soon, leading to a sharp decrease of the contributed data just after the first few days. Besides traditional *uniform* micro-payment, Musthag et al. [75] study *variable* micro-payment (the monetary incentive for each sensing task varies), and argue that variable micro-payment can help to find the appropriate incentive price settings for the tasks. Compared to micro-payment, Rula et al. [76] find that *weighted lottery* (a participant completing more tasks has higher opportunity to get a prize, which is often a relatively high monetary incentive) can recruit more participants, but the average number of completed tasks for each participant is lower. By running an MCS campaign among a pro-environmental group, called *Close the Door*, Massung et al. [77] show that although monetary incentive can significantly increase the participant activeness, it has a cost that intrinsic motivation for accepting an MCS task is reduced when the participation performance is explicitly inked to monetary rewards.

To design a theoretical monetary incentive framework, existing works usually rely on the techniques in mathematic tools such as auction or game theory. *RADP-VPC* [78] is a auction-based MCS incentive framework, where users sell their data to the organizer with their claimed bid prices. Specifically, a participant retaining mechanism, called *Virtual Participant Credit*, is implemented in *RADP-VPC* to motivate the users who lose the current auction round to still join the next auction rounds. Yang et al. [79] propose two kinds of MCS incentive frameworks based on game theory, *platform-centric* and *user-centric*. In the *platform-centric* model, the organizer distributes the total incentive budget to the participants according to the amount of their contributed data; the objective for the organizer is to find the optimal incentive budget to maximize her profit. In the *user-centric* model, authors propose a auction-based framework, which is *computationally efficient*, *individually rational*, *profitable*, and *truthful*. Peng et al. [80] further consider the data quality into the incentive framework design to motivate the participants to contribute high-quality sensed data.

Recently, some researchers begin to study the *online* incentive mechanism, where users come one by one, and the organizer needs to make an immediate decision about whether a task should be assigned to a user when she arrives [81, 82]. Besides, privacy-aware incentive framework is also studied to pay the anonymous participants correct incentive without knowing their true identities [53].

Love

Love is one of the intrinsic motivation of a participant to accept an MCS task. Although usually this varies for different participants, some incentive system design may affect it. For example, as mentioned above, monetary incentive might degrade a user's intrinsic motivation to participate in an MCS task [77]. Besides, by introducing game elements into an MCS application, i.e., *gamification* [83], a user's intrinsic enjoyment of participating in an MCS task can be increased. Representative gamified MCS applications include *CityExplorer* [84], a location-based game to collect geo-spatial urban data, and *PhotoCity* [85], a game to train its players to be experts at taking photos for creating 3D building models. A generalized middleware, called *Crowd Soft Control*, is proposed in [86], which can help to extend a traditional location-based MCS application into a location-based augmented reality game named *Ghost Hunter*. Kawajiri et al. [87] study in detail how to design the game point scheme to steer the participants to the locations with no or little data to contribute data, so as to collect the data more efficiently. Actually, there exist some location-based augmented reality games with millions of users, such as *Ingress*[7], which have a dramatic potential to carry out crowdsensing tasks on it. Although not officially announced by *Ingress*, it is probable that why Google[8] initially launched the *Ingress* project, to some extent, is to efficiently collect large-scale user mobility data so as to create crowd-sourced solution for some difficult problems, such as walking route planning.[9]

Glory

The last motivation factor is *glory*, which means a user's desire to be recognized for her contributions by other people. Explicitly showing the value of a participant's contribution may help her to feel glory, thus motivating her to contribute more data. Rashid et al. [88] verify this hypothesis in a movie recommendation system: by showing the users how much that their rating of a movie will contribute to the recommendation service for others, users do become more likely to rate the movie. Another way to notify a user of her contribution is building a contribution ranking list of all the participants, which has already been implemented in many MCS applications [77, 87]. Kawasaki et al. [89] argue that creating multiple ranking lists, instead of just one, can further improve the participant activeness as they have more probability to appear in certain top-ranking list. Moreover, creating a social group for the participants to let them feel a sense of involvement and belonging, may in-

[7] https://www.ingress.com/

[8] In August 2015, *Niantic Labs*, which developed *Ingress*, was split from Google and became an independent company.

[9] Some online discussions can be found on https://www.reddit.com/r/Android/comments/138res/google_launches_ingress_a_worldwide_mobile/c71v7yv?context=2

crease their engagement. Yu et al. [90] conduct a large-scale study in Amazon Mechanical Turk including thousands of participants, and verify that such a social group strategy can significantly increase not only the willingness of a user to participate in the task, but also the completion quality of the task.

To briefly wrap up, a real-life MCS application probably needs to aggregate a variety of incentive mechanisms to motivate participants from different aspects such as money, love and glory. However, how to integrate different kinds of incentive schemes may not be as trivial as it seems. It is found that merging multiple incentive schemes may not always help to improve the participant engagement (sometimes even reduce) [90]. Therefore, the implementation of the real-life MCS incentive mechanism is still a very critical and tricky problem nowadays, which needs the application designer to make careful and extensive investigation on users for the specific MCS application scenario.

2.3 Organizers' Concerns

Data quality and *Budget* are two primary concerns for MCS organizers, and they also have some intrinsic conflicts: generally, achieving higher data quality means that the organizers need to put more budget into the MCS application. In this section, we discuss

2.3.1 Data Quality

To successfully run an MCS application for an organizer, the quality of the collected data should achieve a satisfactory level. A straightforward method to quantify the quality of an MCS task is just counting the amount of the data collected [91], which can be applied for any MCS application. Specifically for location-based MCS applications, a more commonly used quality metric is *spatial-temporal coverage*. That is, in a specific time slot, how many sub-areas of the target sensing area can be covered by the collected data from participants. On one hand, many existing works attempt to guarantee an MCS application to achieve the *full-coverage* [92, 93] or *partial-coverage* [94, 95] of the target sensing area. On the other hand, a lot of works aim to *optimize the spatial-temporal coverage* of an MCS application with the limited resources in hand, e.g., a limited number of participants [20, 21] or incentive budget [96].

However, if an MCS application only achieves partial-coverage (especially the partial coverage is not very high), reporting the coverage ratio may not be enough for quantifying the data quality. Suppose that two cases both with the coverage ratio 50%. The first case iTruth Discovery in Crowd Sensing Systemss fully covering the left part of the sensing area, while the second case is scattering in the whole sensing area. Intuitively, despite the same coverage ratio, the data collected from the second case may achieve better quality considering the whole target sensing area for the MCS application. To differentiate such differences, some alternative measurements are proposed. One way to quantify the quality of a partial-covered sensing map is to infer the data of unsensed sub-areas and then compute

the inference error [15]: on one hand, *choosing what inference algorithm* is a critical issue here as it significantly impacts the inference error; on the other hand, as the ground truth data is not known in real life, *the computation of inference error is not trivial*, which has to be estimated via some methods such as cross validation [97], leading to a relatively low confidence level of the obtained inference error. Besides inference error, average or maximum predictive variance among all the sub-areas is also often used as a quality metric [98] for continuous sensing values (e.g., temperature). If the MCS task is an outbreak detection (e.g., detecting a source of the pollution or contaminants), then the expected detection time is a key quality metric [99]. In addition, for urban environment monitoring, Liu et al. [100] propose a metric, called *Urban Resolution*, to describe how sensitivity the urban MCS system could achieve.

While the above works give various useful methods to quantify the overall quality in the whole spatial-temporal space of an MCS task, they often make the assumption that the participants upload the *true* sensed data; but this assumption is not always true in real-life scenarios due to the low-precision and accidental error of the smartphone sensors, or even the malicious behaviors of the participants. Therefore, sometimes the organizer may receive multiple conflicting data from participants. By using the crowd data collected from all the participants to 'cross-check' the validity of each contributed data during the 'crowd data integration' stage, many works aim to find the 'true' data and drop the 'false' data among the conflicting data [101, 102, 103, 104]. This kinds of works are often called '*Truth Finding*' or '*Truth Discovery*'. Most of these works depend on the intuition that *the contributed data from reliable users is trustful* to find the fact; however, the challenge is that the reliability of a user is not known a priori. As one pioneering work in this direction, Yin et al. [101] build a graph model to represent *user reliability*, *user observation* (i.e., contributed data), and *objective fact* together, and then design an algorithm to iteratively compute the value of each user reliability and the confidence of each objective fact, based on all the collected observations. After [101], various other algorithms are proposed to improve the truth finding performance, such as *Expectation-Maximization* [102], *Bayesian Network* [103], and *Semi-supervised* [104] methods. Recently, a privacy-preserving truth discovery method is proposed in [105], where not only the truth of the fact can be found, but also the sensitive individual data (e.g., health and location data) and the reliability scores of the participants can be kept secret. To see more works on truth discovery, interested readers can refer to the survey on truth discovery [106].

In a nutshell, both the 'task assignment' and 'crowd data integration' stages significantly affect the quality of an MCS application: the 'task assignment' stage determines the number of participants and the amount of collected data to a large degree, thus dramatically impacting the spatial-temporal coverage (or other quality metrics) of the MCS application; while the 'crowd data integration' stage needs to handle different problems in the data integration process to achieve high task quality, such as *how to infer the missing data of unsensed sub-areas (i.e., missing data inference)* and *how to find truth from conflicting data (i.e., truth discovery)*. To obtain satisfactory quality, the algorithms involved in both stages need to be seriously designed to adapt to the specific MCS applications.

2.3.2 Budget

Budget is another serious issue that an MCS organizer cannot ignore for conducting the MCS task. As we have mentioned in the previous section, although the organizer wants to achieve the data quality as good as possible, it is always constrained by the budget the organizer holds. Therefore, how to efficiently leverage the budget so as to achieve the best possible task quality is usually the important and urgent pursuit for organizers.

Monetary incentive is often a primary part of the organizer's budget. Most existing works concerning the budget is focused on the incentive [20, 91, 93, 94, 96, 107, 108]. A commonly used incentive mechanism in these works is to pay some money for recruiting a participant into a task (*recruitment incentive*), and then to add more money that is proportional to the amount of data a participant contributes (*contribution incentive*) [96, 108]; some works suppose that only one of the two parts of incentive exists (only recruitment incentive [20, 94] or contribution incentive [93]); some works also consider varying incentive costs for different contributed data [107]. Then, based on the incentive mechanism supposed, these works attempt to optimize the data quality of the MCS task, e.g., the amount of collected data [91], spatial-temporal coverage [20, 93, 94, 96], and inferred data accuracy of unsensed sub-areas [107], under a certain amount of budget.

Besides incentive, another part of budget may be needed to spent on the computation resources that need to carry out the MCS task. As cloud computing is becoming prevalent nowadays, currently for MCS organizers, especially small business companies, it is a much more efficient and easier way to deploy the MCS server on the cloud instead of deploying their own IT structure. This kind of service provided by the cloud is often called *infrastructure as a service (IaaS)* [109]. While organizers need to pay for IaaS, how to minimize the cloud computing cost and meantime guarantee the quality of service provided by the MCS server becomes another important commercial issue in addition to incentive. As the price plans provided by the cloud computing providers are usually heterogeneous (e.g., , *reservation* and *on-demand*) and varying overtime, and the computation resources needed by the organizer are uncertain, how to achieve the optimal cost is very challenging. Some works have studied this problem to achieve the cost-effective resource allocation in the cloud platform [110, 111, 112]. Although these works are not specific for MCS servers, their insights and results are still valuable to guide MCS organizers to spend budget efficiently on the cloud computation resources.

2.4 Relations of Dissertation Contributions to Literature

In this dissertation, we attempt to advance the existing research works on MCS participants' and organizers' concerns from multiple perspectives. Before describing each work in detail, in the end of this section, we show what specific concerns each of our works focuses on, and briefly discuss how our works are different from existing literature. Figure 2.1 illustrates the concerns that each of our works is associated to.

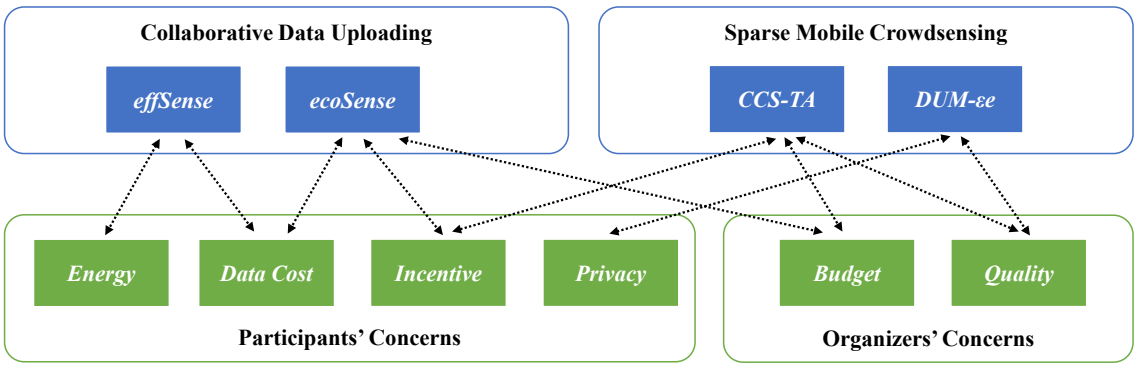


Figure 2.1: Relations of our contributions to participants' and organizers' concerns.

Our proposed *effSense* framework deals with the *data uploading* phase of the 'individual task execution' stage. Suppose participants usually belong to two categories with different main concerns, *data-plan* (main concern is energy consumption) and *non-data-plan* (main concern is data cost), *effSense* attempts to save energy for data-plan users and data cost for non-data-plan users simultaneously. Compared to the existing literature, to the best of our knowledge, this is the first work that aims to minimize *both energy consumption and mobile data cost* for MCS participants by leveraging *heterogeneous* networks (e.g., 3G, Bluetooth, and WiFi), *participant collaborations* (e.g., data relays), and *delay-tolerant* mechanisms.

In *ecoSense*, we suppose that the MCS organizer will give an enough *incentive* to participants to cover their *mobile data cost*. Still based on a similar collaborative data uploading framework like *effSense*, considering two mobile data cost plans — *unlimited data plan* and *pay as you go* — we study how to classify the participants into the two plans, in order to minimize the *incentive budget* of mobile data cost for an MCS organizer. As far as we know, no existing work has yet discussed this problem of minimizing mobile data cost incentive.

CCS-TA is an implementation of our proposed novel MCS paradigm, *Sparse Mobile Crowdsensing (Sparse MCS)*, where only a small part of the target area is selected for sensing, while the rest data of the unsensed area is inferred with high accuracy. The target of Sparse MCS is to reduce the amount of sensed data — suppose *incentive* is paid for each contributed data, then the organizer's *budget* is also reduced — and still guarantee the task *quality* on a satisfactory level. Although some previous works consider to infer missing data in MCS tasks [4, 15], we are the first to put forward the idea of Sparse MCS and propose the general framework for Sparse MCS applications, including three steps named *optimal task allocation*, *missing data inference*, and *task quality assessment*.

Finally, our effort of employing location *privacy* protection mechanism into Sparse MCS applications leads to the work of *DUM-εe*. Specifically, *DUM-εe* tries to introduce *differential privacy* [60, 62] into Sparse MCS to protect participants' location regardless of an adversary's prior knowledge; meantime, it aims to minimize the *data quality loss* incurred by the differential location obfuscation. Different from the previous location pri-

vacy preserving MCS systems in the literature, this is the first time that differential location obfuscation is adopted in MCS applications to provide location privacy protection.

Part II

Collaborative Data Uploading

Collaborative Data Uploading

To encourage users to participate in MCS tasks, it is paramount to minimize the inconvenience incurred for users. In this regard, energy consumption and mobile data cost are two critical concerns. While energy consumption is related to a mobile phone's battery life, mobile data cost is associated with the monetary fees, especially for the users who do not hold an unlimited data plan. Therefore, reducing energy consumption and data cost can encourage more people to actively participate in crowdsensing tasks.

In this part of the dissertation, we design the **collaborative data uploading** framework to address the two concerns of crowdsensing participants. In Chapter 3, we propose and implement the basic idea of collaborative data uploading, leading to a system called *effSense*. In effSense, via energy-efficient and cost-effective communication methods, participants help each other in the data uploading step so as to save energy consumption and data cost. In Chapter 4, we suppose that the organizer will pay participants monetary incentives to cover their mobile data cost during collaborative data uploading. Then, we design an incentive mechanism for participants, called *ecoSense*, which not only compensates participants' data cost concern, but also is economically efficient for the organizer.

Chapter 3

effSense: Energy-efficient and Cost-effective Collaborative Data Uploading

Contents

3.1 Introduction

Energy consumption and data cost are two important concerns for MCS participants. Researchers have developed several approaches to reduce energy and/or data cost for attracting engagements in mobile crowdsensing. The proposed solutions include adopting dynamic sensing duty cycle [9], making a trade-off between local and remote computation [11], reducing data uploading frequency by predicting missing data on the server side [41], and splitting the task intelligently among users [92], etc.

These existing works mostly assume that the sensed data should be sent to a central server as soon as the data is produced. In fact, some mobile crowdsensing tasks do not nec-

essarily require the sensed data to be uploaded in real-time. For example, in the MIT Reality Mining project [113], around 100 participants' mobile traces were collected to understand users' interests, activity patterns, etc. The project collected users' data in two ways. (1) 30 participants were provided with a mobile data plan[114]. These users uploaded their sensed data every night. (2) The other participants' data was stored on SD-cards and was collected after the mobile phones were returned at the end of the project. For both types of participants, a certain amount of time delay between sensing and uploading is allowed.

In this chapter, for the crowdsensing task that does not require real-time sensed data uploading (called a *delay-tolerant crowdsensing task*), we design a collaborative data uploading framework (named as *effSense*) leveraging *heterogeneous networks* (e.g., 3G, WiFi and Bluetooth) and *user collaborations* (data relays), in order to enable (1) users without a data plan or who are not willing to use their data plan for crowdsensing tasks (called *non-data-plan users*) to reduce data cost by relaying data to a Bluetooth gateway or other mobile phones encountered (rather than via 3G network), and (2) users with a data plan (called *data-plan users*) to consume less energy in data uploading.

With the mobile users classified into two groups with different optimization goals: non-data-plan users (reducing data cost) vs. data-plan users (reducing energy consumption), we propose to change the data uploading scheme in effSense from real-time to allowing a certain amount of delay with fixed uploading cycles. In such a way, sensed data uploading in mobile crowdsensing tasks becomes delay-tolerant and data only needs to be sent to the central server before the end of each data uploading cycle (rather than immediately after it is produced). As some delay is allowed, mobile participants might encounter each other or cheaper networks/devices to relay the sensed data before the end of each uploading cycle, so that their data cost or energy consumption can be preserved. Specifically, effSense empowers each mobile device with a distributed data relaying/uploading scheme to decide when and how to upload the sensed data, in order to reduce data cost for non-data-plan users and energy consumption for data-plan users.

effSense is designed based on the following observations:

1) Non-data-plan users can eliminate mobile data cost in data uploading by using zero-cost networks such as Bluetooth and WiFi. For example, they can upload data to the server directly via WiFi, or transfer data to another device via Bluetooth if the other device can relay data to the server without incurring extra cost.

2) Data-plan users can reduce energy consumption in data uploading via the energy-efficient methods other than establishing a new 3G connection. For example, piggybacking a data uploading task on a 3G voice call can save 75-90% energy consumption [38]. Alternatively, uploading data via WiFi or Bluetooth consumes less energy than via normal 3G.

A running example: Figure 1 shows a simple example to illustrate the basic idea of effSense. Here, u_1, u_2, and u_3 are three data-plan users, respectively, while u_1^*, u_2^*, and u_3^* are three non-data-plan users, respectively. In addition, there is a server (S) and a fixed-location Bluetooth gateway device (D). Instead of uploading the sensed data directly via a

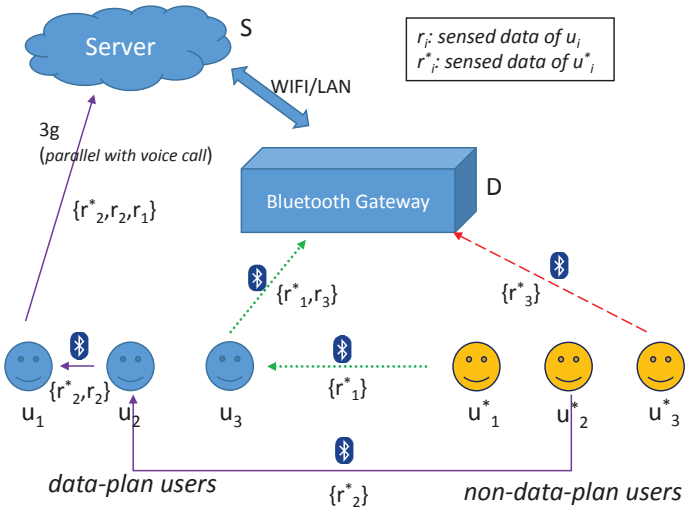


Figure 3.1: A running example of effSense

specific 3G link, effSense offers the following data uploading paths to save data cost and energy consumption:

Path 1 (red dashed line): $u_3^* \rightarrow D \rightarrow S$. A non-data-plan user ($u_3^*$) uploads data via a Bluetooth gateway.

Path 2 (green dot line): $u_1^* \rightarrow u_3 \rightarrow D \rightarrow S$. A non-data-plan user ($u_1^*$) relays data to a data-plan user (u_3) first and then u_3 uploads data via a Bluetooth gateway other than 3G directly to reduce both data cost and energy consumption.

Path 3 (purple solid line): $u_2^* \rightarrow u_2 \rightarrow u_1 \rightarrow S$. The primary difference between this path and the two above is that a data-plan user (u_1) piggybacks the data uploading task over a voice call in the end.

As shown in this example, non-data-plan users are guided to upload their sensed data without mobile data cost, while data-plan users are recommended to upload data via energy-efficient methods. Such preservation of data cost and energy consumption is exactly the design objective of effSense.

The key issues involved in designing effSense include:

1) **Identify and predict *critical events***. Typical critical events include *making a voice call*, *meeting another user*, *encountering a Bluetooth gateway*, *connecting to a WiFi AP*, etc. Predicting the critical events for each user is the basis for effSense to select the right data relaying strategy. By leveraging the state-of-the-art activity prediction methods [115, 116], effSense can predict critical events accurately.

2) **Estimate data uploading energy consumption associated to each critical event**. Specifically, for data uploading, the energy consumption is not always proportional to the data size. For instance, uploading a data packet smaller than 10KB via 3G always consumes about 12 joule, whatever the exact data size [117]. According to the existing literature about the energy consumption of mobile phones [117, 118], we estimate the energy consumption

of various critical events for different data sizes.

3) **Design real-time algorithms to decide if the data should be offloaded or kept at each individual event**. The algorithms should be lightweight and executed on each phone locally without incurring much energy consumption or data cost.

In summary, this work has the following contributions:

1) To the best of our knowledge, this is the first work that aims to minimize both energy consumption and mobile data cost in mobile crowdsensing tasks by leveraging heterogeneous networks and delay-tolerant mechanisms.

2) We consider two types of users (non-data-plan and data-plan) with different goals and propose a collaborative data uploading framework including two data uploading schemes for each kind of users, respectively: one is purely greedy, the other is based on the mobility/call predictions. While the former one is quite effective in handling the cold-start problem when the participants' historic call or mobility logs are not available, the latter one can achieve better performance by leveraging critical-event prediction according to the participants' historic logs.

3) We evaluate effSense with two real-world datasets - MIT Reality Mining [113] and Nodobo [119]. The results show that effSense could upload about 48-52% of non-data-plan users' data without extra data cost, and reduce 55-65% of data-plan users' data-uploading energy consumption compared to the traditional method, given the condition that data-plan users and non-data-plan users have the ratio of 3:4[1] and a data uploading cycle of 24 hours.

3.2 Preliminary: Delay-Tolerant Mobile Crowdsensing

Many crowdsensing tasks (e.g. MIT Reality Mining [113], environment monitoring [94, 93, 96, 91]) do not require immediate uploading of the data after it is sensed (called *delay-tolerant mobile crowdsensing task*). Such tasks allow some delay (max tolerable delay T_d) between collecting the data from sensors and uploading it to the server, i.e. the sensed data generated at t on a participant's phone can be uploaded during $[t, t+T_d]$.

Formally, we consider a crowdsensing task process that is composed of two kinds of cycles: *sensing cycles* and *delayed-uploading cycles* (see Figure 3.2).

- *Sensing Cycle*: A crowdsensing task process can be split into continuous sensing cycles. As shown in Figure 3.2, each sensing cycle lasts for T_s, i.e. the i^{th} sensing cycle starts at $t_{i-1} = t_0 + (i-1)T_s$ and ends at $t_i = t_0 + iT_s$. We assume that *each participant's sensed data is prepared for uploading right up until the end of each sensing cycle* (e.g. some aggregation algorithms need to be run on the raw sensed data before uploading).
- *Delayed-Uploading Cycle*: The i^{th} delayed-uploading cycle starts at the end of the

[1] 3:4 is the ratio when 30 users are selected as data-plan users in the MIT Reality Mining project, which is the actual project setting [114].

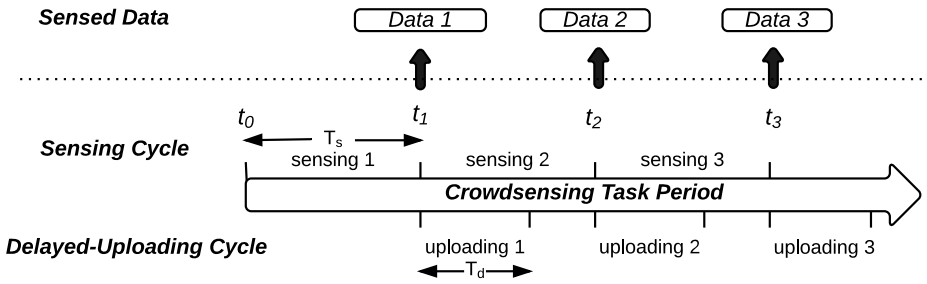


Figure 3.2: Sensing cycles and delayed-uploading cycles

i^{th} sensing cycle (i.e. t_i) and lasts for T_d (i.e. ends at $t_i + T_d$). In the i^{th} delayed-uploading cycle, each participant attempts to upload her i^{th} sensing cycle's data to the server. Various data uploading/relay strategies could be applied here. At the end of a delayed-uploading cycle, if some participants' data is still not uploaded to the server, these participants are forced to upload their data to the server via 3G, in order to ensure that all the sensed data could arrive at the server within the delay T_d. We consider only the condition that $t_i + T_d \leq t_{i+1}$ (i.e. $T_d \leq T_s$), which means that participants need to upload their i^{th} sensing cycle's data to the server before their $i + 1^{th}$ sensing cycle's data gets ready (so different delayed-uploading cycles will not overlap).

In summary, effSense can be applied to any mobile crowdsensing task process that meets two requirements: (1) all the participants' sensed data is ready for uploading at the end of each sensing cycle; and (2) the i^{th} sensing cycle's data needs to be uploaded to the server before the $i + 1^{th}$ sensing cycle's data gets ready. Therefore, in each delayed-uploading cycle, a participant only needs to upload one piece of sensed data.

3.3 Problem Statement

The research goal is to design the delay-tolerant data uploading schemes that can not only minimize mobile data cost for non-data-plan users (U_{ndp}) but also maximally decrease energy consumption for data-plan users (U_{dp}). We make the following assumptions in the crowdsensing process.

Assumption 1 - Offload and Dismiss: Once a user u offloads the sensed data to a recipient, no matter the recipient is the server, gateway, or another user, u will not be responsible for sending the data any more.

This assumption in effSense ensures having only one copy for all the sensed data and thus avoids redundant data uploading to meet the energy-saving objective.

Assumption 2 - Offload All Data: Once a user catches a chance to offload data, all the data will be offloaded to the recipient, no matter the data was sensed locally or received from other users.

This assumption seems quite strong at first glance, especially when we use very short Bluetooth encounters to transfer a large amount of data. However, there exist many crowdsensing applications that generate small amount of data; for such applications, this assumption holds most of the time. For example, after analyzing the user data collected in the MIT Reality Mining project, we find that even using the plain text to store users' sensed data without compression, the amount of data for one user per day is less than 100KB. Section 3.7.1 will discuss this issue in more details.

Before formulating the problem, we introduce some key concepts.

Definition 3.1 (Critical Events)**.** *A critical event ($e \in E$) refers to an encounter between a non-data-plan mobile terminal and another device that can help non-data-plan users offload data without cost, or/and a call/encounter that can help data-plan users save energy consumption in data uploading. The critical event set E contains two subsets:*

- *Server-related Critical Events (E_s): When a user encounters a server (including intermediate servers, e.g., Bluetooth gateway) or initiates/receives a call, the sensed data can be uploaded to the server directly.*
- *User-related Critical Events (E_u): When $user_a$ encounters $user_b$ who might be able to better accomplish data uploading, $user_a$ can offload data to $user_b$ to reduce data cost or energy consumption.*

In each crowdsensing cycle, users encounter a sequence of critical events ($EVENTS = \{e_1, e_2, \cdots, e_n\}$), where e_1 could be $user_a$ encountering a server, e_2 could be non-data-plan $user_b$ encountering data-plan $user_c$, and e_n could be data-plan $user_d$ receiving a phone call. Our effSense framework is designed to provide users with decisions at each event (e), either upload the data to the server (when $e \in E_s$) or offload the data to an encountered mobile device (when $e \in E_u$), or keep the data till next critical event occurs. Now, we formally define the "Decision Making" mechanism.

Definition 3.2 (Decision Making)**.** *In delay-tolerant data uploading with maximum delay d_{max}, when a user u_i with sensed data r_i encounters a critical event e at time t^* ($t^* \in [t_0, t_0 + d_{max}]$), effSense makes a decision about whether data r_i needs to be offloaded (i.e., true) or kept (i.e., false). We denote it as $DEC(u_i, r_i, t^*, e, t_0, d_{max}) \rightarrow \{true, false\}$.*

As the mobile data cost and energy consumption are two primary concerns when critical events occur, we define "Event Data Cost Function" and "Event Energy Consumption" as follows.

Definition 3.3 (Event Data Cost Function)**.** *The event data cost function represents whether an event e incurs mobile data cost or not.*

$$\delta(e) = \begin{cases} true, & \text{if } e \text{ incurs data cost} \\ false, & \text{otherwise} \end{cases}$$

Definition 3.4 (Event Energy Consumption)**.** *(1) For $e \in E_s$, the event energy consumption is the amount of energy that user u consumes to upload r to the server under e, marked*

as $W_s(u,r,e)$. (2) For $e \in E_u$, the total event energy consumption comprises two parts — user u_i sending data and user u_j receiving data, marked as $W_u(u_i,u_j,r,e) = W_{u\_sd}(u_i,r,e) + W_{u\_rv}(u_j,r,e)$.

As critical event prediction is helpful in event decision making, we also define "Event Probability" as follows:

Definition 3.5 (Event Probability)**.** *Given a user u, a critical event set E^*, a time t, the event probability is the probability that u will encounter any event e ($e \in E^*$) from t to the end of data uploading cycle t_d (i.e., $t_0 + d_{max}$), marked as $P_e(u,E^*,t,t_d)$.*

Based on these definitions, we formulate our problem as follows:

Problem Statement: In a crowdsensing task with some delayed uploading cycle (max delay d_{max}), data-plan users (U_{dp}) and non-data-plan users (U_{ndp}) would encounter a critical event sequence (*EVENTS*). Each mobile device aims to obtain a decision making sequence *DECISIONS* corresponding to *EVENTS* (i.e., each $d \in DECISIONS$ corresponds to the decision (*true* or $false$) at an event $e \in EVENTS$), in order to achieve the following two goals dedicated to U_{ndp} and U_{dp}, respectively.

First goal: Maximize the number of non-data-plan users whose data is uploaded to the server with zero data cost.

$$\max |\{u | u \in U_{ndp}, R_u(t_0) \in R_s(t_d)\}|$$

where

- $R_u(t_0)$ is the sensed data of u produced at t_0.
- $R_s(t_d)$ is the sensed data on the server at t_d.

Second goal: Minimize the energy consumption for data-plan users during the data uploading process.

$$\min \sum_{u \in U_{dp}} EnergyCons_u(t_0,t_d)$$

where $EnergyCons_u(t_0,t_d)$ is the amount of energy that u consumes in data uploading during $[t_0,t_d]$.

It is worth noting that we do not know when the critical events (*EVENTS*) would occur in advance. In practice, the event appears one after another. When an event occurs for a mobile device, a decision should be made instantly in a distributed manner. Although future events are unknown, they could be predicted to help decision making for data offloading.

3.4 The effSense Framework

In order to solve the two-goal optimization problem formulated in the previous section, we design effSense to accomplish effective data uploading for mobile crowdsensing applications. Our effSense framework is shown in Figure 3.3.

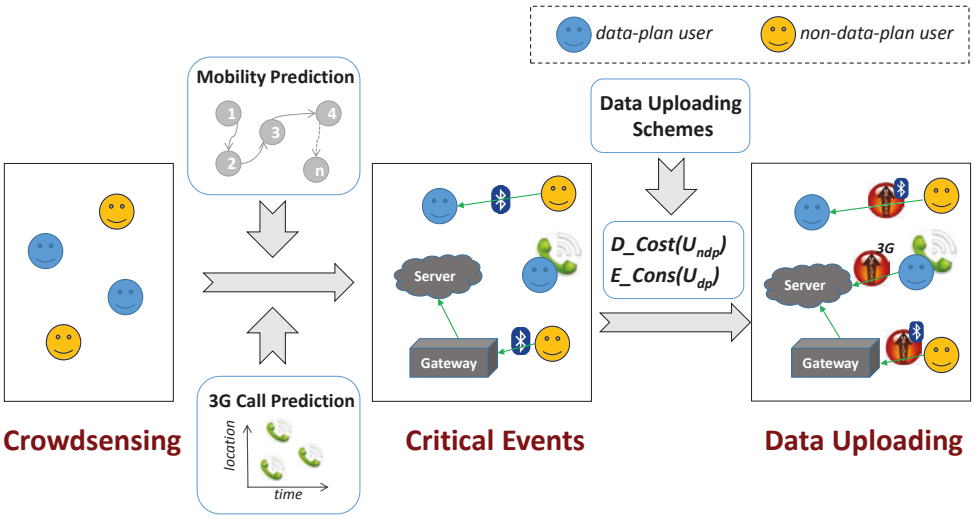


Figure 3.3: The effSense framework

As shown in the left part of Figure 3.3, we have two types of crowdsensing users, i.e., data-plan (U_{dp}) and non-data-plan users (U_{ndp}). Each mobile device identifies future critical events in a distributed manner using state-of-the-art prediction techniques — for both *mobility prediction* [115] and *call prediction* [116]. Accordingly, we obtain a sequence of critical events (*EVENTS*) for each mobile device in the middle of Figure 3.3. For U_{dp}, critical events are used to reduce energy consumption by offloading data via Bluetooth gateways encountered or piggybacking sensed data on a 3G phone call as predicted. For U_{ndp}, critical events are used to eliminate mobile data cost by offloading data to Bluetooth gateways or U_{dp} devices encountered as predicted.

In each crowdsensing uploading cycle, effSense selects the data uploading schemes by analyzing the critical events. When a user encounters a critical event, effSense makes the decision to offload or keep the data. As shown in the right part of Figure 3.3, the generated data uploading schemes might include two non-data-plan users sending data to a data-plan user and a gateway via Bluetooth, respectively, and a data-plan user sending data via 3G when he makes a voice call.

There are two kinds of schemes proposed in effSense for both types of users.

- *Cold-start scheme*: It does not require users' historic event traces, and applies a straightforward greedy algorithm to offload data as soon as it encounters a "promising" event that can eliminate data cost for non-data-plan users $D\_Cost(U_{ndp})$, or reduce energy consumption for data-plan users $E\_Cons(U_{dp})$.
- *Prediction-based scheme*: It compares the current uploading cost (i.e., offload data at current event) with future predicted uploading cost (i.e., keep data at current event), to decide whether the sensed data should be offloaded or kept.

Before the end of each data uploading cycle, effSense checks with all mobile devices to see whether they have un-uploaded data: for U_{ndp} with data, effSense forces them to

symbol	*definition*
u	a user
r	sensed data size
e	a critical event
t_0	sensed data generated time (i.e., uploading cycle start time)
d_{max}	max tolerable delay
t_d	uploading cycle deadline (i.e., $t_0 + d_{max}$)
U_{ndp}	non-data-plan users
U_{dp}	data-plan users
E_s	server-related critical events
E_u	user-related critical events
$\delta(e)$	whether e incurs data cost or not
$W_s(u,r,e)$	energy consumption for u to upload r when $e \in E_s$
$W_{u\_sd}(u,r,e)$	energy consumption for u to send r when $e \in E_u$
$W_{u\_rv}(u,r,e)$	energy consumption for u to receive r when $e \in E_u$
$P_e(u,E^*,t,t_d)$	event probability of u encountering any $e \in E^*$ from t to t_d

Table 3.1: Notations

offload data to nearby U_{dp} if possible; for U_{dp} with data, effSense forces them to create a 3G connection to upload data.

3.5 Uploading Schemes

We propose two uploading schemes (*cold-start* and *prediction-based*) for both non-data-plan and data-plan users. Note that the event probability (P_e) prediction is not the focus of this work, so the prediction method will be later described in the experiment (Section 3.6). The important notations are listed in Table 3.1.

3.5.1 Uploading Schemes for Non-Data-Plan Users

Suppose a non-data-plan user u_i encounters a critical event e at time t (the encountered user is u_j if e is user-related), we propose two schemes for non-data-plan users to offload data: $SimpleGreedy_{ndp}$ (cold-start) and $AdvancedGreedy_{ndp}$ (prediction-based).

Cold-start Scheme: $SimpleGreedy_{ndp}$

$SimpleGreedy_{ndp}$ follows the logic below:

- When a non-data-plan user u_i encounters a server-related event,
  - If the event is an encounter with a Bluetooth gateway or a WiFi AP, u_i uploads the sensed data as it will not incur any data cost.

  - If the event is a 3G call, u_i will not upload data, as piggybacking data on a call only reduces energy but still incurs 3G data cost.

- When a non-data-plan user u_i encounters a user-related event,
  - If the encountered user u_j is a data-plan user, u_i offloads data to u_j, because data-plan users can ensure uploading data before the uploading cycle deadline.
  - If the encountered user u_j is a non-data-plan user, u_i does not offload data.

For generality, we use E_{ndp_like} to represent the above events that make the decision making *true*:

$$E_{ndp\_like} = \{e|e \in E_s, \delta(e) = false\} \cup \{e|e \in E_u, \delta(e) = false, u_j \in U_{dp}\}$$

Prediction-based Scheme: $AdvancedGreedy_{ndp}$

On top of $SimpleGreedy_{ndp}$, $AdvancedGreedy_{ndp}$ adds a new data offloading condition when a non-data-plan user u_i meets another non-data-plan user u_j:

- When a non-data-plan user u_i meets another non-data-plan user u_j, if u_j has higher probability to meet data-plan users or to upload data via Bluetooth or WiFi gateways (i.e., encountering $e \in E_{ndp\_like}$) than u_i, u_i will offload data to u_j.

So the events which would trigger data offloading are generalized as:

1. $e \in E_{ndp\_like}$

   (same as $SimpleGreedy_{ndp}$)

2. $e \in \{e'|e' \in E_u, u_j \in U_{ndp}\}$ *and* $P_e(u_i, E_{ndp\_like}, t, t_d) < P_e(u_j, E_{ndp\_like}, t, t_d)$

   (new data offloading condition)

3.5.2 Uploading Schemes for Data-Plan Users

Suppose that a data-plan user u_i encounters a critical event e at time t (the encountered user is u_j if e is user-related), we design two energy-saving schemes for data-plan users to upload data: $Greedy_{dp}$ (cold-start) and $ExpectationBased_{dp}$ (prediction-based).

Cold-start Scheme: $Greedy_{dp}$

$Greedy_{dp}$ follows the intuitions below to upload data:

- If a data-plan user u_i encounters a server-related event, whether the event is making a call, encountering a Bluetooth gateway, or connecting to WiFi, u_i will upload data, because all these events cost less energy than creating a new 3G connection for data uploading.
- If a data-plan user u_i encounters a user-related event, u_i will not offload data.

We use E_{dp_like} to represent the events described above which make the decision making *true*:

$$E_{dp\_like} = \{e | e \in E_s, W_s(u,r,e) < W_{3G}(u,r)\}$$

where $W_{3G}(u,r)$ is the energy consumption for user u to upload data of size r by creating a new 3G connection.

Prediction-based Scheme: $ExpectationBased_{dp}$

The intuition behind $ExpectationBased_{dp}$ is to compare the expected energy consumptions (i.e., *expEnergy*) needed for different data offloading schemes corresponding to possible events predicted before the end of each data uploading cycle (e.g., *uploading data to the server* or *keeping data* under a server-related event) and select the one with least expected energy consumption. In this process, $ExpectationBased_{dp}$ leverages users' mobility and call prediction to predict future events.

$ExpectationBased_{dp}$ might discard the current energy-efficient event to wait for another *more* energy-efficient event later, while $Greedy_{dp}$ always triggers the first-coming energy-efficient event, specifically:

1. When encountering the server-related events that consume less energy than 3G, $Greedy_{dp}$ will always make u_i upload data, while $ExpectationBased_{dp}$ will sometimes make u_i keep data. For example, if u_i currently makes a 3G call, and u_i is predicted to have a very high probability of meeting a Bluetooth gateway soon, then keeping data till the next event of encountering the Bluetooth gateway could be a better strategy in terms of energy saving (because uploading data via Bluetooth consumes less energy than piggybacking data on a call).

2. When encountering the user-related events, $Greedy_{dp}$ will always make u_i keep data, while $ExpectationBased_{dp}$ provides the possibility of offloading data between encountered data-plan users u_i and u_j. For example, if u_i has a much lower probability to upload data via energy-efficient methods (i.e., encounter $e \in E_{dp\_like}$) in the future than u_j, then offloading data from u_i to u_j can hopefully reduce the users' total energy consumption, because u_j later could probably upload data via an energy-efficient method.

Figure 3.4 illustrates the basic decision making process of $ExpectationBased_{dp}$, where *expEnergy* refers to the user's total energy consumption predicted from the current event time (i.e., when u encounters e) to the end of uploading cycle (t_d). For example, suppose t_d is 12:00 and current time is 8:25. If user *Bob* makes a phone call, then we calculate Bob's *expEnergy* from 8:25 to 12:00 under two distinct conditions: *uploading data over this call* vs. *keeping data locally*.

When encountering the server-related events, we calculate *expEnergy* for two possible conditions:

- u_i uploads data to the server ($expEnergy_{u_i \to S}$)
- u_i keeps data ($expEnergy_{u_i \not\to S}$)

And we select the scheme with smaller *expEnergy*.

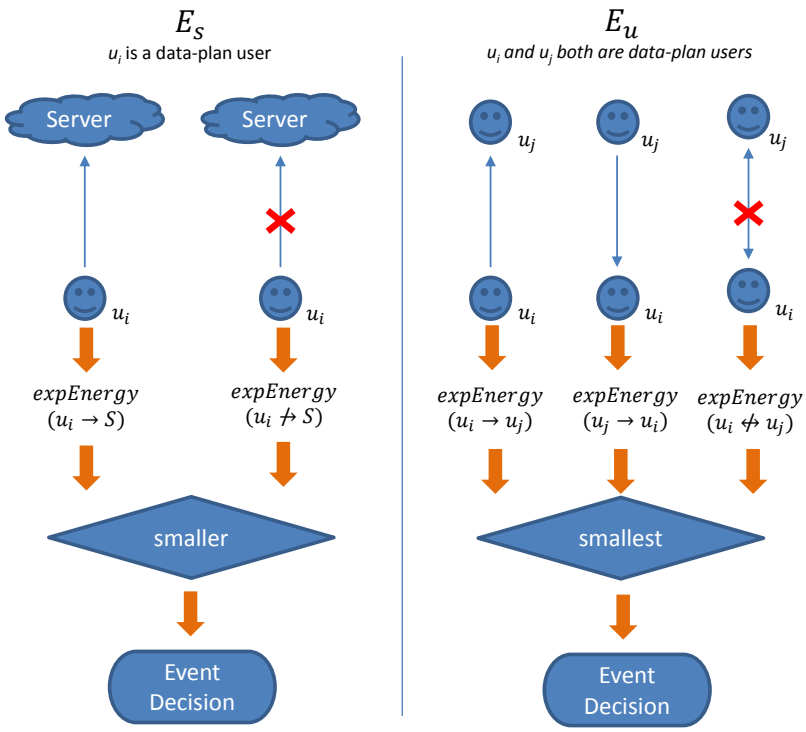


Figure 3.4: The decision making process of $ExpectationBased_{dp}$

When encountering the user-related events, if a data-plan user u_i meets a non-data-plan user u_j, u_i will not offload data to u_j. (In fact, according to the non-data-plan user data uploading schemes, the non-data-plan user u_j will offload data to u_i if u_j has data.)

If a data-plan user u_i meets another data-plan user u_j, the decision making process is a little more complicated. We need to take both u_i and u_j's expected energy consumption into account. If we only consider one user's own energy consumption, u_i and u_j might both decide to send data to the counterpart, which leads to more energy consumption. We thus compute *expEnergy* for three possible conditions:

- u_i offloads data to u_j ($expEnergy_{u_i \rightarrow u_j}$)
- u_j offloads data to u_i ($expEnergy_{u_j \rightarrow u_i}$)
- u_i and u_j both keep data ($expEnergy_{u_i \leftrightarrow u_j}$)

We choose the scheme with smallest *expEnergy*. To calculate the aforementioned *expEnergy* in Figure 3.4, two basic components are involved.

The first component is the *Event Energy Consumption* (defined in Section 3.3, e.g., data transmission energy associated with 3G, WiFi, and Bluetooth). To estimate this part, we refer to existing work [117, 118]. Further details are shown in the section of *Evaluation*.

The second component is $expEnergy_{keep}(u, r, t, t_d)$, which represents a user u's expected energy consumption (from t to t_d) if u keeps the data of size r at time t. To compute $expEnergy_{keep}$, we first predict u's each event probability during $[t, t_d]$. Based on these

Problem:
Given a data-plan user u, local data (*size=r)* and time t, calculate the expected energy consumption during $[t, t_d]$ if u keeps data r at t.

Notation:
e_i: all critical events belonging to E_{dp_like}
p_i: event probability, abbr. for $P_e(u,\{e_i\},t,t_d)$
w_i: event energy consumption, abbr. for $W_s(u, r, e_i)$
w_{3G}: energy consumption under new 3G connection, abbr. for $W_{3G}(u, r)$
$P_{rec}(r)$: the probability for u to receive data (*size=r*) from t to t_d, $\sum_{r\geq 0} P_{rec}(r) = 1$ (specifically, *r=0* means the probability not receiving any data later)

Solution:
Step1. Sort critical events, make $w_1<=w_2<=...<=w_n$
Step 2. Calculate the expected energy consumption for uploading data (*size=r)*:

$$energy_{up}(u,r,t,t_d) = \sum_{j=1}^{n}\left(p_j * w_j * \prod_{i=1}^{j-1}(1-p_i)\right) + w_{3G} * \prod_{i=1}^{n}(1-p_i)$$

Step 3. As u may receive data from the other users later, so the expected energy consumption from t to t_d is:

$$expEnergy_{keep}(u,r,t,t_d) = \sum_{r'\geq 0}\left(P_{rec}(r') * energy_{up}(u,r'+r,t,t_d)\right)$$

Figure 3.5: Calculation for $expEnergy_{keep}$

event probabilities, we sum up all the event energy consumptions for offloading/uploading r. (Figure 3.5 shows the detailed algorithm.) It is possible that $expEnergy_{keep}(u,0,t,t_d) > 0$ (even though u holds no data at t), because u may receive data from other users during $[t,t_d]$ (see *Step 3* in Figure 3.5) .

With these two components, we can calculate all the aforementioned *expEnergy* in Figure 3.4. The detailed formulas are given as follows:

1. For e ∈ E_s

1) When u_i uploads data to the server ($expEnergy_{u_i\to S}$)

$expEnergy_{u_i\to S}$ includes two parts: (1) the energy to upload data, (2) the expected energy consumption after this uploading (so u_i keeps no local data).

$$expEnergy_{u_i\to S} = W_s(u_i,r_i,e) + expEnergy_{keep}(u_i,0,t,t_d)$$

2) When u_i keeps data ($expEnergy_{u_i\not\to S}$)

$expEnergy_{u_i \not\mapsto S}$ includes only one part: the expected energy consumption when u_i still keeps data of size r_i.

$$expEnergy_{u_i \not\mapsto S} = expEnergy_{keep}(u_i, r_i, t, t_d)$$

2. For e ∈ E_u

For $e \in E_u$, we consider both u_i and u_j's energy consumptions together.

1) When u_i offloads data to u_j ($expEnergy_{u_i \to u_j}$)

$expEnergy_{u_i \to u_j}$ includes four parts: (1) u_i's energy consumption for sending data, (2) u_j's energy consumption for receiving data, (3) u_i's expected energy consumption after sending data (so u_i keeps no local data), (4) u_j's expected energy consumption after receiving data (so u_j keeps both u_i's and u_j's data).

$$\begin{aligned} expEnergy_{u_i \to u_j} = {} & W_{u\_sd}(u_i, r_i, e) + W_{u\_rv}(u_j, r_i, e) \\ & + expEnergy_{keep}(u_i, 0, t, t_d) \\ & + expEnergy_{keep}(u_j, r_i + r_j, t, t_d) \end{aligned}$$

2) When u_j offloads data to u_i ($expEnergy_{u_j \to u_i}$)

Similar to $expEneryg_{u_i \to u_j}$, we only need to exchange i and j in the above formula.

3) When u_i and u_j keep data ($expEnergy_{u_i \leftrightarrow u_j}$)

$expEnergy_{u_i \leftrightarrow u_j}$ includes two parts: Both u_i's and u_j's expected energy consumption when they keep their own data (r_i and r_j).

$$expEnergy_{u_i \leftrightarrow u_j} = expEnergy_{keep}(u_i, r_i, t, t_d) + expEnergy_{keep}(u_j, r_j, t, t_d)$$

3.5.3 Scheme Selection

When a new user first participates in the crowdsensing task, since there is no activity history about this user, the cold-start scheme $\{SimpleGreedy_{ndp}, Greedy_{dp}\}$ is recommended. After a period of time (T_{change}), when the new user accumulates certain activity logs, he can change to the prediction-based schemes $\{AdvancedGreedy_{ndp}, ExpectationBased_{dp}\}$ to get better performance.

Deciding T_{change} is an important issue. The optimal T_{change} may vary from one specific mobile crowdsensing task to another. In this study we do not discuss how to choose T_{change} but focus on designing and evaluating the overall framework.

3.5.4 Additional Features

Here, we briefly introduce some other features in the implementation of effSense.

Information Exchange between Users

When two users meet, they need to exchange some information with each other for running effSense. For example, when two data-plan users meet, one needs to know the value of $expEnergy_{keep}$ of the counterpart (for $ExpectationBased_{dp}$). We propose to exchange information between encountered users as follows: encoding all the event probabilities in the device name according to a predefined protocol. Then, when a user meets another one, each user will know the event probabilities that can be used to calculate the other user's $expEnergy_{keep}$. One defect of this solution is that the user's device name needs to change after a time period because the event probabilities change over time. Fortunately, this is a simple operation without much energy consumption.

Flexibility to User-Type Interchange

An interesting and valuable feature of effSense is that one user can change his user type at any time, so that he can decide whether he cares more about energy consumption or data cost. We can still use the previously proposed solution: encoding the user type in the device name. Then, one user could easily know the other user's type when they meet each other.

Mechanisms for Exit-Users

A user might exit the task during a uploading cycle, which means that the un-uploaded data in this exit-user's phone might fail to be uploaded to the server. Note that as we have the assumption of "*Offload and Dismiss*", the un-uploaded data might include multiple users' data. To minimize the data loss incurred by exit-users, we propose the following mechanisms:

- *Forced-uploading* (known-exit): If effSense knows when a user exits from a crowdsensing task (e.g., a user instructs to exit from the mobile task app menu), then the app can upload/offload the un-uploaded data immediately (e.g., a data-plan user would initialize a new 3G connection).
- *Notify-reuploading* (sudden-exit): A user might exit suddenly (e.g., due to operating system errors). In this case, the mobile task app fails to operate normally for forced uploading. To relieve this problem, at the end of each uploading cycle, the server can use a method like *Dial-to-Deliver* [120] to notify the users whose data has not been uploaded, to upload/offload their data again. Thus, if a user u's data was relayed to a sudden-exit user but u does not exit, u can still have a chance to upload his data again in the end.

3.6 Evaluation

In this section, we evaluate the effSense framework using two real-world crowdsensing datasets: MIT Reality Mining [113] and Nodobo [119]. While the MIT Reality Mining dataset recorded more mobile users' call and mobility traces, the Nodobo dataset reported a more up-to-date mobile users' traces with different patterns.

	Small-size(<10KB)	*Big-size(xKB)*
3G	12 J	12+0.025x J
Bluetooth	1 J	1+0.003x J

Table 3.2: Energy consumption estimation of 3G/Bluetooth

3.6.1 Experimental Setup for the MIT Dataset

For the MIT dataset, we choose 7 weeks of sensing data (2004.10.4-2004.11.21) from 71 active users. The 30 users who consumed the largest volume of mobile data are considered as data-plan users, and the remaining 41 users are non-data-plan users — as the MIT data campaign provided 30 users with data plans subsidy [114]. We used the first five weeks of user data to build the model, and evaluated the performance of effSense using the data of the last two weeks. The data uploading cycle was set to one day, i.e., each uploading cycle starts at 00:00 and ends at 24:00. Thus 14 rounds of data uploading occurred during the last two weeks.

This experiment involved 3 types of critical events:

- e_{3g_call}: making a 3G voice call.
- e_{bt_device}: encountering a Bluetooth gateway. Two Bluetooth gateways are in the experiment: *localhost.media.mit.edu* and *studies.media.mit.edu*.
- e_{bt_user}: encountering another user via Bluetooth.

Based on the literature about the mobile phone energy consumption [117, 118], we estimate the energy consumption for transmitting data through 3G and Bluetooth (Table 3.2). Table 3.3 shows the estimation results of energy consumption for each critical event type.

Bluetooth Encounters

We clarify two practical issues related to the Bluetooth encounters in our experiment.

Bluetooth Contact Duration: Sometimes, the Bluetooth encounter between two users is too short to transfer all data successfully. As the MIT Reality Mining campaign only activates Bluetooth scanning every 5 minutes [114], we cannot accurately know how long two users really meet when they are in contact via Bluetooth. In the evaluation, we eliminated the short Bluetooth encounters that do not have enough time to transfer data between two devices as follows: we only use the encounters that can be discovered in two continuous 5-minute scannings. For example, if user u_i meets user u_j at 12:00 via Bluetooth, we will use this encounter in our evaluation only if u_i can still meet u_j at 12:05. Due to the dataset limitation, u_i and u_j might meet each other just at the two time points of 12:00 and 12:05, while keeping away from each other between 12:00 and 12:05. However, with this data preprocessing about the device encounters, we believe that most of the device encounters could allow successful data transfer between devices via Bluetooth.

	Small-size(<10KB)	*Big-size(xKB)*
e_{3g_call}[2]	$12 \times (1 - 75\%) = 3$ J	3+0.006x J
e_{bt_device}	1 J	1+0.003x J
e_{bt_user}[3]	2 J	2+0.006x J

Table 3.3: Energy consumption estimation for critical events

Action	*Energy*
Idle (1 minute)	0.9 J
3G Call (1 minute)	75.9 J
SMS (1 message)	3.5 J
Bluetooth Scanning (20 seconds)	4.5 J

Table 3.4: Energy consumption for different phone usages

Bluetooth Scanning Energy: Note that the energy consumption of Bluetooth scanning is not considered in the evaluation due to two reasons: (1) Intermittent Bluetooth scanning is required by some crowdsensing tasks such as MIT Reality Mining and SociableSense [9]; so the energy consumption of Bluetooth scanning is not caused by data uploading, but is required by the crowdsensing task. (2) Bluetooth 4.0 low energy (BLE) technology is adopted by more and more up-to-date smartphones (*iPhone 5s, Nexus 5,* etc.). With BLE, the battery drain of Bluetooth scanning is dramatically decreased [24]. Due to the energy efficiency of BLE, a lot of novel real-time smartphone sensing applications are emerging recently, which *require the smartphone users to turn on Bluetooth and do intermittent scanning all the time* (e.g., fitness sensing with FitBit wristbands[4]). Thus, we believe that in the near future, more smartphone users would like to have Bluetooth (with BLE) always on to support such novel applications; again the energy consumption of Bluetooth scanning is not caused by the data uploading of effSense, but is required by these applications.

Energy Calculation and Battery Constraints

If a user's phone battery has already reached a low energy level, he is typically not willing to relay data for other users. Thus, we prevent a user from relaying data when his phone battery level is lower than a predefined limit, e.g., 50% battery level.

Due to the lack of explicit battery information in the MIT dataset, we simulate a user phone's battery level based on real-time phone usage records, including calls, messages, mobile data usage, etc. Our simulation makes the following basic assumptions:

1. Each user's phone battery is fully charged as 100% level at 00:00 a.m. every day and the phone would not be charged during the day.
2. We adopted the basic energy consumption for each phone usage type using exiting statistics of Nokia N95 [118] (see Table 3.4). According to the specification, N95's

[2] 3G data transmission during call saves ~75% energy [38].

[3] The energy consumption of e_{bt_user} is twice of e_{bt_device} because of one user sending and one user receiving.

[4] https://www.fitbit.com/

full battery is 950mAh/3.7v, which means the total battery energy is $950 \times 0.001 \times 3.7 \times 3600 = 12654J$.

We set the battery level limit to 50%. This limit is purposely set high for two reasons:

1. The phone usage records are not a complete set. Many application usage logs, such as games, are not included. The actual energy consumption should be higher than our simulation setting.
2. We set a high battery limit to ensure that relaying others' data will not bring significant inconvenience to phone users' own experiences.

3.6.2 Prediction of Critical Events

To estimate the critical event occurring probability $P_e(u, E^*, t_1, t_2)$, we use a Poisson distribution model [116, 115]. The computation process contains the following steps (Note that E^* might include different kinds of events, e.g., $E^* = \{e_{3g\_call}, e_{bt\_device}\}$ represents the energy-efficient events for DP users.):

Step 1. Split one week into 24×7 non-overlapping timeslots — each timeslot lasts for one hour; then, map t_1, t_2 to the corresponding timeslots in a week, i.e., ts_1 and ts_2, respectively.

Step 2. Assume the event $e_i \in E^*$ follows a Poisson process, then the probability of event e_i happening k times for user u during $[ts_1, ts_2]$ is:

$$p(e_i, u, k, ts_1, ts_2) = \mu^k_{e_i,u,ts_1,ts_2} \cdot exp(-\mu_{e_i,u,ts_1,ts_2})/k!$$

where μ_{e_i,u,ts_1,ts_2} is estimated as the average number of the occurrences that user u encounters event e_i during $[ts_1, ts_2]$ from the history data.

Step 3. As $P_e(u, E^*, t_1, t_2)$ is the probability of at least one event $e_i \in E^*$ occurring at least once during $[t_1, t_2]$, we thus calculate it as follows:

$$\begin{aligned} P_e(u, E^*, t_1, t_2) &= 1 - \prod_{e_i \in E^*} p(e_i, u, k = 0, ts_1, ts_2) \\ &= 1 - \prod_{e_i \in E^*} exp(-\mu_{e_i,u,ts_1,ts_2}) \end{aligned}$$

To measure the performance of the Poisson method, we compare it with a simple method that directly counts the frequency of event occurrences.

Frequency: Suppose that the history data includes m weeks, and there are n weeks u encounters any event $e_i \in E^*$ during $[ts_1, ts_2]$, then $P_e(u, E^*, t_1, t_2) = n/m$.

To compare the performance between the two methods, we draw ROC curves [121] and calculate their AUC values. Figure 3.6 and 3.7 show the ROC curves associated to the prediction of 3G calls and Bluetooth encounters between users. We can see that the Poisson method outperforms the Frequency method in both cases.

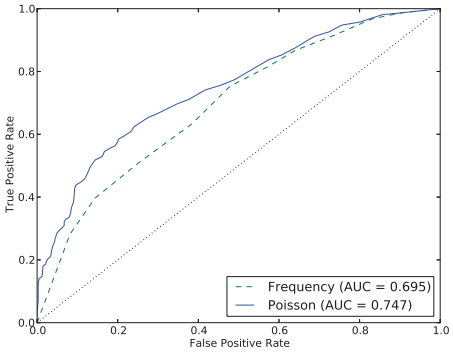


Figure 3.6: ROC curve of call prediction (e_{3g_call}) on the MIT dataset

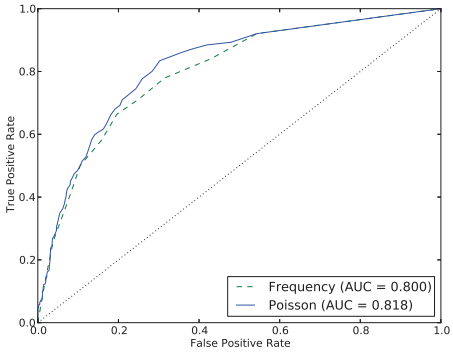


Figure 3.7: ROC curve of Bluetooth encounter prediction (e_{bt_user}) on the MIT dataset

	Weekday	*Weekend*	*Overall*
BT Activity	28.8	10.5	23.6
SimpleGreedy	24.5 (85.7%)	7.8 (74.3%)	19.7 (83.5%)
AdvancedGreedy	26.2 (91.0%)	8.5 (80.6%)	21.2 (89.8%)

(values in the brackets are the proportions to the value of BT Activity)

Table 3.5: Average N_{nd_upload} in one data uploading cycle on the MIT dataset

3.6.3 Experimental Results on the MIT Dataset

In order to evaluate the performance of proposed data uploading schemes in effSense, we design the experiments to address the following key questions:

1. *Data Cost*: How many non-data-plan (NDP) users can upload their data before the uploading cycle deadline without incurring data cost?
2. *Energy Consumption*: How much phone energy is consumed for data-plan (DP) users during the data uploading process?
3. *Event Triggering*: How many critical events have triggered data uploading/offloading during the data uploading process?

After answering the three questions, we also evaluate the influence of exit-users and different parameter settings (ratio of NDP/DP users and max tolerable delay) on the performance of effSense.

Data Cost Conservation

First, we investigate the performance of two effSense schemes for NDP users — the number of NDP users who upload data successfully in each data uploading cycle (noted as N_{nd_upload}). Figure 3.8 plots the detailed N_{nd_upload} in two weeks for the two schemes, together with the upper-bound of N_{nd_upload} — the number of NDP users having Bluetooth activities (*BT Activity*). This is because only Bluetooth activities can trigger successful data uploading for NDP users without data cost. In Figure 3.8, effSense does not perform very

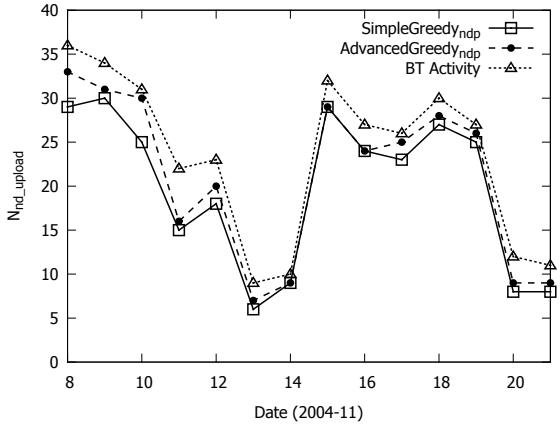
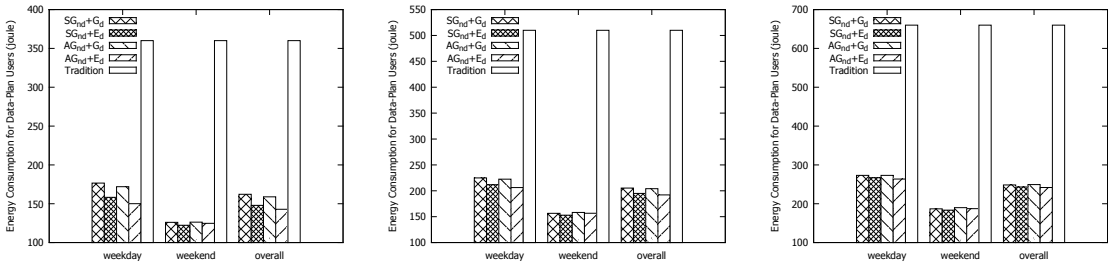


Figure 3.8: N_{nd_upload} for 2 weeks on the MIT dataset



(a) small (1KB) (b) 100KB (c) 200KB

Figure 3.9: Energy consumption of data-plan users, $totalEnergy_{dp}$, for uploading small(1KB)/100KB/200KB data per cycle on the MIT dataset (SG_{nd}: $SimpleGreedy_{ndp}$; AG_{nd}: $AdvancedGreedy_{ndp}$; G_d: $Greedy_{dp}$; E_d: $ExpectationBased_{dp}$)

well in weekends (11/13, 11/14, 11/20, 11/21) and Veterans day (11/11), as few users came to school so that the opportunities for Bluetooth relay dropped.

In Table 3.5, we further list the statistics, observing that effSense helps on average 19.7 NDP users using $SimpleGreedy_{ndp}$, and 21.2 NDP users using $AdvancedGreedy_{ndp}$, corresponding to the success rate of 48% and 52% for 41 NDP users, respectively. $AdvancedGreedy_{ndp}$ outperforms $SimpleGreedy_{ndp}$ by 4%. This improvement is quite significant for $AdvancedGreedy_{ndp}$, as $SimpleGreedy_{ndp}$ has already achieved more than 85% of the upper bound *BT Activity* on weekdays.

Energy Conservation

Figure 3.9 shows the energy consumption for DP users in one data uploading cycle (noted as $totalEnergy_{dp}$) when three different types of sensed data sizes are considered — small (1KB), 100KB, and 200KB. effSense reduces 55-65% of the energy consumption for DP users compared with the traditional method[6]. When using the same NDP scheme (ei-

[6]The traditional method means that DP users upload their data via a new 3G connection every day, while NDP users store their data in the SD-cards instead of uploading to the server. So NDP users will not consume

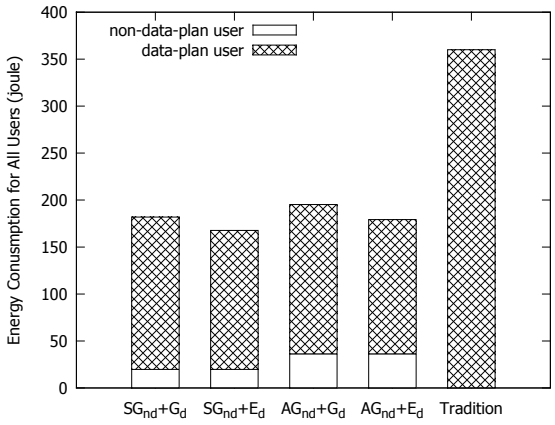


Figure 3.10: $totalEnergy_{all}$ for small-size data on the MIT dataset

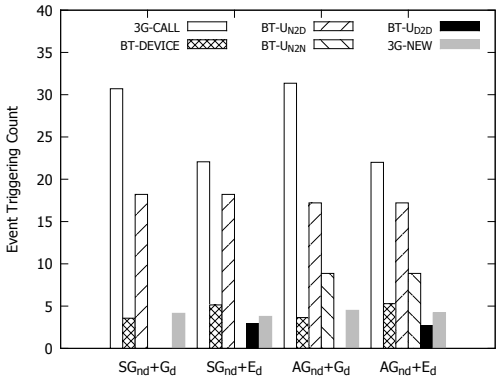


Figure 3.11: Event triggering per uploading cycle on the MIT dataset (small data)

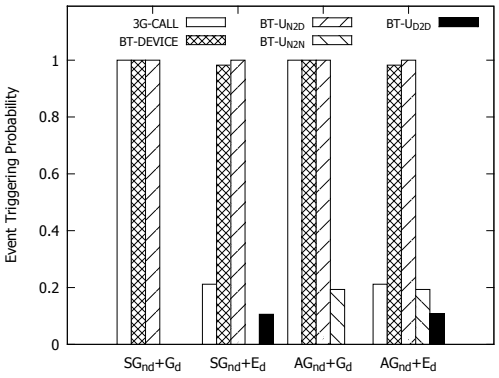


Figure 3.12: Event triggering probability per uploading cycle on the MIT dataset (small data)

ther $SimpleGreedy_{ndp}$ or $AdvancedGreedy_{ndp}$), $ExpectationBased_{dp}$ can save up to about 13% extra energy for DP users compared with $Greedy_{dp}$ on weekdays. On weekends, the energy consumption difference between $ExpectationBased_{dp}$ and $Greedy_{dp}$ is not significant, because few students go to school on weekends and thus data relays between DP users will rarely happen. In addition, as the data size increases, the performance gap between $ExpectationBased_{dp}$ and $Greedy_{dp}$ decreases. The reason is that as the data size increases, the overhead for one user to help others relay data becomes larger.

Figure 3.10 shows the total energy consumption for all DP and NDP users ($totalEnergy_{all}$). Though effSense causes NDP users to consume energy, which does not exist in the traditional method, $totalEnergy_{all}$ is still reduced by 46-54%.

Event Triggering

Figure 3.11 shows the total number of events that trigger data uploading/offloading for each event type, and Figure 3.12 shows the corresponding triggering probability. Here, the events are *3G-call*, *BT-device*, and *BT-user*. In particular, *BT-user* can be further divided

energy in data uploading.

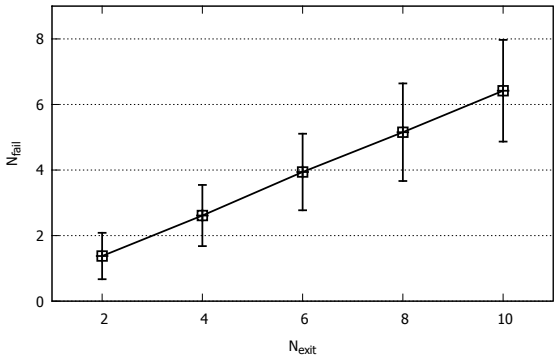


Figure 3.13: N_{fail} for different N_{exit} on the MIT dataset

into 3 categories according to user type: (1) $NDP \rightarrow DP$ (BT_U_{N2D}), (2) $NDP \rightarrow NDP$ (BT_U_{N2N}), and (3) $DP \rightarrow DP$ (BT_U_{D2D}).[7]

First, the frequency of requiring a new 3G connection decreases significantly in effSense compared with the traditional method that would incur 30 new 3G connections in each data uploading cycle. As establishing a new 3G connection is energy-demanding, this is the primary reason why effSense can reduce energy consumption significantly. In addition, for the cold-start scheme pair $\{SimpleGreedy_{ndp}, Greedy_{dp}\}$, the data exchanges between users are only carried out when $NDP \rightarrow DP$. However, $AdvancedGreedy_{ndp}$ and $ExpectationBased_{dp}$ introduce the data relays at $NDP \rightarrow NDP$ and $DP \rightarrow DP$, respectively. Furthermore, given the same NDP scheme, $ExpectationBased_{dp}$ triggers much fewer e_{3g_call} and more e_{bt_device} than $Greedy_{dp}$. This is why $ExpectationBased_{dp}$ can further conserve energy compared with $Greedy_{dp}$, as the energy consumption of e_{bt_device} is less than e_{3g_call}.

Impact of Exit-Users

Exit-users can incur data loss and thus effSense needs some mechanisms to deal with exit-users (Section 3.5.4). Here we evaluate how many users' sensed data would fail to be uploaded (N_{fail}) due to exit-users in one uploading cycle by simulations.

Exit-Model: We randomly choose N_{exit} users to be exit-users (either known-exit or sudden-exit). For each selected user, we randomly assign a time line within the delayed uploading cycle as his exit timing, via uniform distribution.

Figure 3.13 shows N_{fail} and its standard deviation, varying with N_{exit} (2-10) based on 100 simulations. Generally, N_{fail} is less than N_{exit}, which means that in most simulations, some exit-users have already uploaded/offloaded their sensed data before they exit. The ratio N_{fail}/N_{exit} is around 65%, no distinction between different N_{exit}. Therefore, if N_{exit} users exit in one cycle, the expected N_{fail} is about $0.65N_{exit}$, less than N_{exit}. We also run the simulations when effSense does not deploy the exit-user mechanisms described in Section 3.5.4, and the ratio N_{fail}/N_{exit} is larger (75-80%). This verifies the effectiveness of

[7] $DP \rightarrow NDP$ does not exist in effSense because this direction conflicts the goal of saving data cost for NDP users.

$\|U_{dp}\|$	$\|U_{ndp}\|$	$N_{nd\_upload}/\|U_{ndp}\|$	$totalEnergy_{dp}/\|U_{dp}\|$
10	61	38%	6.06 J
20	51	47%	5.23 J
30	41	51%	4.76 J
40	31	53%	4.40 J
50	21	57%	4.26 J
60	11	68%	4.05 J

Table 3.6: Results of different DP/NDP ratios (24-hour delay)

our proposed exit-user mechanisms.

Parametric Analysis

We then analyze the two key parameters in effSense and study how they affect the performance of the framework. The parameters are *DP/NDP user ratio* and *max tolerable delay (d_{max})*. Our experiments here focus on the scheme pair $\{AdvancedGreedy_{ndp}, ExpectationBased_{dp}\}$ and small sensed data size.

DP/NDP User Ratio

In the previous experiments, we selected 30 out of 71 participants as DP users, as this is the actual setting in the MIT Reality Mining project[114]; thus, the DP/NDP ratio is 30/41. Real-life crowdsensing tasks face various DP/NDP ratios. Furthermore, in a long period of crowdsensing, existing participants could leave and new participants could join. The objective of evaluations here is to verify effSense's performance with different DP/NDP ratios. Instead of investigating the absolute N_{nd_upload} and $totalEnergy_{dp}$ metrics, we investigate their relative values to better present effSense's robustness. The two relative metrics are: $N_{nd\_upload}/|U_{ndp}|$ (the percentage of NDP users who upload data successfully) and $totalEnergy_{dp}/|U_{dp}|$ (the average energy consumption of each DP user).

Table 3.6 shows the evaluation results under six different DP/NDP user ratio settings, and from the table we have the following observations:

- With more DP users, greater percentage of NDP users can upload data successfully without incurring data cost (i.e., $N_{nd\_upload}/|U_{ndp}|$ increases), because NDP users could have more chances to encounter DP users and relay data to them.
- With more DP users, average energy consumption for each DP user decreases (i.e., $totalEnergy_{dp}/|U_{dp}|$ decreases), because each DP user needs to help fewer NDP users to relay data.

Though effSense performs better with more DP users, it also works well when only few DP users exist. As shown in Table 3.6, even under low $|U_{dp}|/|U_{ndp}|$ such as 10/61, nearly 40% of the NDP users could upload data successfully without data cost, and a DP user usually consumes less than 50% energy on average, compared with uploading data by creating a new 3G connection which consumes *12J* energy.

d_{max}	N_{nd_upload}	$totalEnergy_{dp}$
3-hour	13.3	206.1 J
6-hour	17.0	184.4 J
12-hour	19.0	152.5 J
24-hour	21.2	142.9 J

Table 3.7: Results for different max delays d_{max} (30 DP users)

Start	N_{nd_upload}	$totalEnergy_{dp}$
8:00	7.2	189.4 J
11:00	14.3	209.1 J
14:00	13.2	195.4 J
17:00	10.8	181.4 J

Table 3.8: Results for different start time (30 DP users, 3-hour delay)

Max Tolerable Delay

All the evaluation results reported so far were carried out assuming the max tolerable delay (d_{max}) for data uploading is 24-hour, as the MIT Reality Mining project asked DP users to upload data once a day[114]. Nowadays, 24-hour is a long uploading delay as users most likely be able to upload data using free WiFi at home. Thus, we conduct extra experiments to test the performance at smaller d_{max} (see Table 3.7). As d_{max} decreases, NDP users have less chance to upload data without incurring data cost (i.e., N_{nd_upload} decreases) and DP users consume more energy (i.e., $totalEnergy_{dp}$ increases) in each data uploading cycle. This is because with a shorter delay, there are fewer critical events occurring for effSense to trigger data offloading in order to reduce data cost and/or energy consumption.

In addition, we observe that the start time of data uploading cycle affects N_{nd_upload} and $totalEnergy_{dp}$ when applying smaller d_{max}, as the number of each type of critical events differ greatly between different time periods. Table 3.8 shows the evaluation results with different data uploading cycle start time and a max tolerable delay of three hours. N_{nd_upload} is higher in the afternoon than in the morning/evening, as NDP users are more likely to encounter DP users to relay in the afternoon; $totalEnergy_{dp}$ is lower in the evening than in the morning/afternoon, as users are likely to make calls in the evening.

3.6.4 Evaluation on the Nodobo Dataset

We also evaluated effSense on the Nodobo dataset with users' WiFi traces. It contains a type of critical event, e_{wifi}, different from the MIT dataset, when a user connects to a WiFi Access Point (AP). Free WiFi APs can be used for NDP users to upload data without incurring data cost. To simplify the experiment, we assume all WiFi APs in the Nodobo project are free. In the Nodobo project, there is no Bluetooth gateway, so we excluded e_{bt_device} events, and focus on the e_{wifi} events. According to [117], uploading small-size data via WiFi consumes 2J energy. In the experiment of the Nodobo dataset, we set the same d_{max} (i.e., 24-hour), and DP/NDP ratio as 11/16 (similar to 30/41 on the MIT dataset).

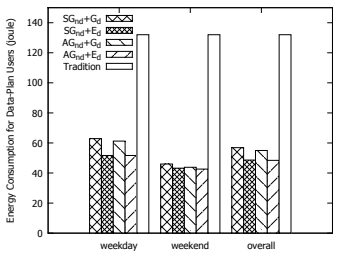


Figure 3.14: $totalEnergy_{dp}$ for small-size data on the Nodobo dataset

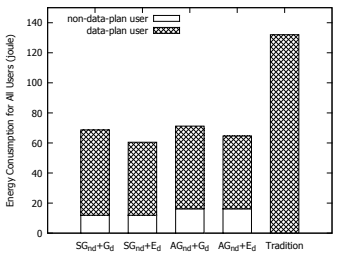


Figure 3.15: $totalEnergy_{all}$ for small-size data on the Nodobo dataset

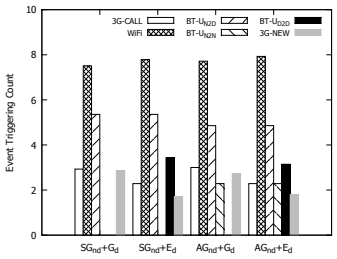


Figure 3.16: Event triggering per uploading cycle on the Nodobo dataset

	Weekday	*Weekend*	*Overall*
BT/WiFi Activity	10.3	7.2	9.2
SimpleGreedy	10.0 (96.8%)	6.2 (86.1%)	8.6 (93.8%)
AdvancedGreedy	10.3 (100%)	6.4 (88.9%)	8.9 (96.9%)

(values in the brackets are the proportions to the value of BT/WiFi Activity)

Table 3.9: N_{nd_upload} on the Nodobo dataset

The results are shown in Figure 3.14, Figure 3.15, Figure 3.16, and Table 3.9 (we set the upper bound of N_{nd_upload} as the number of NDP users who have either Bluetooth or WiFi activity, called *BT/WiFi Activity*). Like the previous experiments on the MIT dataset, these results show similar performance in conserving both data cost and energy consumption.

Figure 3.16 shows that e_{wifi} plays an active role in improving effSense's performance as it frequently triggers data uploading to conserve data cost or energy consumption. Note that N_{nd_upload} can reach the upper bound when using $AdvancedGreedy_{ndp}$ scheme for NDP users in weekdays (see Table 3.9). These results show the flexibility and extensibility of the effSense framework, and reveal the potential that effSense can become more powerful when introducing more critical events.

3.7 Discussion

3.7.1 Data Size

In the experiment, we set three data sizes — small (1KB), 100KB, and 200KB, which are not large, and thus our assumption of "Offload All Data" can probably be satisfied. However, some mobile crowdsensing tasks can generate sensed data in a large volume up to several MB (e.g., taking photos). When the data size is large, the assumption of "Offload All Data" will become a bit unrealistic, especially for the events such as Bluetooth encounters (often temporary and unstable). In the future, effSense can be improved to handle sensing a large volume of data.

3.7.2 User Incentives for Data Relay

Data relay between users is important in effSense to help reduce data cost and energy consumption. In the experiment, we assumed that users are always willing to help others relay data when their phone battery is not low (>50%). However, this is not always true in real life; effSense in the future could additionally design incentive mechanisms for encouraging users with the sufficient battery life to actively relay data for others.

3.7.3 Diversity of Critical Events

Although we identified several useful critical events in this study, there are still other critical events that we could consider in our future work. One example is the event of charging the mobile phone battery. When a user's phone is charging, energy consumption of data uploading could be assumed to be zero. Because of the dataset limitation, we cannot include these critical events in our current evaluation. However, effSense is flexible to include new critical events in the framework. Like these existing critical events, adding a new event in effSense should address two key practical issues, i.e., estimating energy consumption of uploading/offloading data under such an event, and providing the model for event prediction.

3.7.4 Delay-Tolerant and Real-Time Tasks

The main purpose of effSense is to support delay-tolerant data uploading, but it can be integrated with real-time data uploading to better serve application requirements. For example, users use effSense to deal with delay-tolerant crowdsensing tasks, and initialize new 3G connections (or WiFi if possible) to upload data for real-time crowdsensing tasks. Although it seems like a simple integration, this solution can further improve the performance of effSense. For example, if a data-plan user participates in both delay-tolerant and real-time crowdsensing tasks, then he regularly uploads real-time sensed data at certain time every day. effSense can leverage this routine uploading event to reduce his energy consumption by uploading the delay-tolerant data simultaneously, because uploading multiple data together to the server can reduce the transmission energy overhead [39].

3.7.5 Prediction of Event Probability

We use a Poisson distribution model to predict the critical event probability. As this work does not focus on the mobility/activity prediction, we apply this state-of-the-art method [116, 115] and focus on the data uploading schemes to show the feasibility and effectiveness of effSense. A more advanced prediction method may further enhance the framework. However, when trying a complex prediction method, it is inevitable to consider that the prediction itself could consume a certain amount of energy.

3.8 Concluding Remarks

In this chapter, we investigate the problem of how to minimize both energy consumption and data cost caused by data uploading in mobile crowdsensing. We address the key concerns for both data-plan and non-data-plan users (energy consumption vs. data cost) simultaneously, and design a collaborative data uploading framework called *effSense* to improve both types of users' experience in mobile crowdsensing tasks.

For non-data-plan users, we propose to reduce or eliminate data cost in data uploading by using the least expensive network (e.g., Bluetooth) other than 3G. For data-plan users, we propose to choose the appropriate critical events (e.g., making a voice call) to trigger data uploading that requires less energy rather than create a new 3G connection. We design two dedicated schemes for both user types to help users make proper decisions about whether to offload or keep data when encountering a critical event. Our effSense framework was evaluated with the MIT and Nodobo datasets. The experiment results verified the efficiency and effectiveness of effSense for data uploading in mobile crowdsensing tasks.

Chapter 4

ecoSense: Data Cost Minimization via Collaborative Data Uploading

Contents

4.1 Introduction

In MCS, for organizers, refunding participants money to cover the 3G data cost[1] of sensed data uploading is an efficient marketing strategy to compensate participants' data cost concern [77]. As this 3G data cost refund would increase organizers' incentive budget, especially for tasks needing a large number of participants, *how to reduce the 3G data refund budget* becomes a critical problem for MCS organizers.

To address this problem, first, we study the common price plans of 3G data cost. Currently two price plans are widely used by most telecom operators: *Unlimited Data Plan* and *Pay As You Go*.

[1] In this chapter, for brevity, we use 3G data cost to represent all kinds of communication fees incurred by data uploading via cellular networks (also including 2G, 4G, etc.).

- *Unlimited Data Plan (UnDP)*: with Unlimited Data Plan, a user can transfer an unlimited amount of data during a period of time (usually for a month). The cost for an unlimited data plan is fixed, e.g. \$7/month (denoted as $Price_u$).
- *Pay As You Go (PAYG)*: with Pay As You Go, a user pays 3G data cost according to the amount of data transferred via 3G, e.g. \$0.1/MB (denoted as $Price_p$).

With the above two 3G price plans, a simple solution to refunding participants' 3G data cost is to choose the right refund scheme for each mobile user according to the amount of her uploaded data. Specifically, this *direct-assignment* method works as follows:

1. For each participant u_i, estimate her amount of sensed data to be uploaded each month (d MB).
2. Two possible ***refund schemes*** exist:
   - *UnDP*: refund is $Price_u$.
   - *PAYG*: refund is $d * Price_p$.

   Choose the cheaper one as the refund for u_i.
3. If assigned to *UnDP*, u_i needs to set her 3G price plan to *UnDP* before the next month starts (this is why we need to estimate d). At the end of the next month, the organizer pays $Price_u$ to u_i.
4. If assigned to *PAYG*, u_i can keep her original personal 3G price plan for next month (independent of whether u_i's original price plan is *PAYG* or *UnDP*, the organizer does not need to know[2]). In next month, the organizer counts the actual amount of sensed data that u_i uploads (d' MB). At the end of the next month, the organizer pays $d' * Price_p$ to u_i.

When the sensing task is determined, one participant's sensed data size in a month can usually be estimated within reasonable error bounds (i.e. $d \approx d'$), which makes direct-assignment applicable.

Although direct-assignment can support real-time data uploading reasonably well, for many delay-tolerant MCS tasks (Section 3.2), the refund budget of direct-assignment may be still high. Recall in the collaborative data uploading mechanism, the following events can be leveraged to reduce participants' 3G data cost during the delay period:

1) *Cost-free Network.* A PAYG participant can use a cost-free network, such as Bluetooth or WiFi (e.g. at home or in the office), to upload sensed data to the server within the delay period, which reduces her 3G data cost.

2) *User Collaboration.* UnDP participants can help relay PAYG participants' sensed data to the server. This kind of relay reduces 3G data cost for PAYG participants, without increasing 3G data cost for UnDP participants, thus decreasing the organizer's refund budget.

[2]A participant's *refund scheme* may be different from her *3G price plan*. Refer to Section 4.7.2 and 4.7.3 for more details. In this chapter, unless specified otherwise, *UnDP* and *PAYG* represent refund schemes.

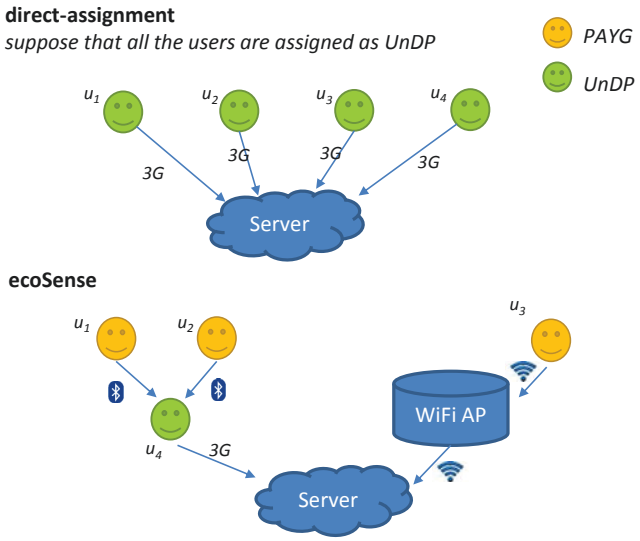


Figure 4.1: Comparison between ecoSense and direct-assignment

Based on these events, we incorporate a new *participant partition* module into the collaborative data uploading framework, whose goal is to optimally partition the participants into UnDP or PAYG sub-groups, in order to minimize the organizer's 3G data refund budget, via maximumly taking advantage of cost-free networks and user collaborations. The resultant framework is called *ecoSense*. Note that the difference between ecoSense and effSense is: in effSense, whether a participant holds an UnDP is foreknown (e.g. according to each user's own preference) and we focus on designing data uploading strategies; while in ecoSense, in addition to uploading strategies, another key technical issue is deciding whether to assign a participant to the UnDP or PAYG group.

To better illustrate the basic idea of ecoSense, we use an example to compare ecoSense and direct-assignment (see Figure 4.1). Suppose that $Price_u$ is \$7/month, $Price_p$ is \$0.1/MB, and each participant's sensed data will be larger than 70MB/month, so direct-assignment refunds each of the four participants as UnDP. The refund budget of direct-assignment is $\$7 * 4 = \$28$ for a month.

By allowing some delay between data sensing and uploading, ecoSense enables the following new data uploading paths:

- u_1 and u_2 have high probability to encounter u_4 via Bluetooth within the delay period, thus u_4 can relay u_1 and u_2's data.
- u_3 is likely to meet a free WiFi access point (AP) within the delay period, so u_3 could upload data via that AP without 3G data cost.

As a result, by using delay-tolerant data uploading mechanisms, u_1, u_2 and u_3 could significantly reduce the amount of uploaded data via 3G. By adopting the above mechanisms, if we assume that u_1, u_2 and u_3's 3G-uploaded sensed data size can decrease to less than 70MB/month, e.g. 50MB, 60MB, and 40MB respectively, then ecoSense would choose the PAYG scheme for the three participants instead of UnDP, reducing the organizer's refund budget to $\$0.1 * (50 + 60 + 40) + \$7 * 1 = \$22$ for a month, compared to the \$28 of

direct-assignment.

Two important issues are involved in designing ecoSense:

1) *How to transfer data when two PAYG participants meet?*

Unlike the clear relay strategy between a PAYG participant and an UnDP participant (i.e. PAYG $\xrightarrow{data}$ UnDP), relay strategy between two PAYG participants is more complicated and will affect the organizer's refund budget. Among all the possible strategies, flooding (i.e. always exchanging data between two PAYG participants) is expected to produce the smallest refund budget. Because after flooding, if any one of two PAYG participants could meet an UnDP participant or a cost-free network, both of their data could be uploaded without 3G cost. However, flooding might incur too many redundant relays that rapidly drain the batteries of participants' mobile phones. Though our work focuses on minimizing the organizer's 3G data refund budget, the participants' energy concerns should also be taken into account to some extent. Otherwise, even if ecoSense "successfully" minimizes the refund budget, participant phones' energy consumption might be too high, making ecoSense impractical. Thus, to study the trade-off between the organizer's 3G data refund budget and participant phones' energy consumption, we try different data uploading/relay strategies.

2) *How to decide each participant's refund scheme — PAYG or UnDP?*

To minimize the organizer's 3G data refund budget, another key issue is to determine which participants should be assigned to each of the schemes (and not just the percentage of participants allocated to each scheme). Thus, the participant partition algorithm needs to consider each participant's mobility pattern and sensed data size:

- *Mobility Pattern*. To maximize data relay opportunities between PAYG and UnDP participants, accurately profiling each participant's mobility pattern is necessary for deciding whether she should be assigned to the UnDP or PAYG scheme. Intuitively, "active" participants who can help more other participants relay data should be assigned to UnDP.
- *Sensed Data Size*. A participant's sensed data size would also impact whether she is assigned to PAYG or UnDP. Generally a participant who uploads a larger amount of sensed data should be assigned to UnDP.

Our proposed partition algorithm first predicts each participant's mobility pattern and estimates her sensed data size, and finally uses a genetic algorithm to obtain a participant partition for PAYG and UnDP groups. This algorithm is run before a new month starts (only once a month) on the MCS organizer's server.[3]

In summary, this work makes the following contributions:

1) To the best of our knowledge, this is the first work that aims to minimize the organizer's 3G data refund budget by leveraging heterogeneous networks (e.g. 3G, Bluetooth, WiFi) and user collaborations in MCS.

[3]Most telecom operators need a participant to decide her 3G data price plan for the upcoming month before the end of the current month.

2) We propose a collaborative data uploading framework, called *ecoSense*, attempting to minimize the organizer's 3G data refund budget. ecoSense considers two 3G price plans to refund the participants — *Pay As You Go* and *Unlimited Data Plan* — and proposes data uploading strategies for both UnDP and PAYG participants in the delayed uploading period. Furthermore, a participant partition algorithm is designed to split all the participants between PAYG/UnDP participant groups, in order to minimize the organizer's 3G data refund budget, via maximumly taking advantage of opportunistically encountered cost-free networks and participant collaborations (data relays).

3) We use a real-life dataset, the MIT Reality Mining [113], and a larger SWIM [122] simulation dataset to evaluate our approach. The evaluation results show that ecoSense can reduce the organizer's 3G data refund budget by up to ~50% compared to the direct-assignment solution.

4.2 Problem Statement

Considering the delaty-tolerant mobile crowdsensing task process (Section 3.2), ecoSense aims to reduce the crowdsensing organizer's refund budget for participants' 3G data cost. Before formulating the problem, we introduce some key concepts.

Definition 4.1 (Cost-free Events)**.** *A cost-free event refers to an encounter between a PAYG participant and another participant or device that can probably help PAYG participants upload data to the server without 3G data cost*[4].

Definition 4.2 (Uploading Decision Making)**.** *In a delayed-uploading cycle with maximum tolerable delay T_d, when a PAYG participant u_i with sensed data r_i (generated at t) encounters a cost-free event e at time t^* ($t^* \in [t, t+T_d]$), ecoSense makes a decision about whether data r_i needs to be uploaded/relayed (i.e. true) or not (i.e. false). We express this decision function as: $\mathcal{D}(u_i, r_i, t^*, e, t, T_d) \rightarrow \{true, false\}$.*

Similar to effSense, if the decision $\mathcal{D}$ is *true*, we assume that a PAYG participant can always relay/upload data successfully.

Definition 4.3 (Participant Partition)**.** *Given all the crowdsensing participants U, a participant partition assigns each participant to UnDP group (U_u) or PAYG group (U_p). We express this partition function as: $\mathcal{P}(U) \rightarrow [U_u, U_p]$, where $U_u \cup U_p = U$ and $U_u \cap U_p = \phi$.*

Based on these definitions, we formulate our problem as follows.

Problem Formulation: In a crowdsensing task allowing certain data uploading delay (T_d), given all the participants (U), unit prices for both 3G price plans UnDP ($Price_u$, e.g. \$7/user for a month) and PAYG ($Price_p$, e.g. \$0.1/MB), we require an uploading decision making strategy for PAYG participants ($\mathcal{D}$) and a PAYG/UnDP participant partition

[4] *Cost-free events* are considered only for PAYG participants, as UnDP participants can upload an unlimited amount of data via 3G.

function ($\mathcal{P}$), in order to minimize the crowdsensing organizer's refund budget for all the participants' additional 3G data cost in one month[5]:

$$\begin{aligned}\underset{\mathcal{D},\mathcal{P}}{\operatorname{argmin}}\ Refund &= \underset{\mathcal{D},\mathcal{P}}{\operatorname{argmin}}\ (Refund_u + Refund_p) \\ &= \underset{\mathcal{D},\mathcal{P}}{\operatorname{argmin}}\ (|U_u| * Price_u + \sum_{i \in U_p} d_i * Price_p)\end{aligned}$$

where

- $Refund_u$: refund budget for UnDP participants.
- $Refund_p$: refund budget for PAYG participants.
- d_i: amount of data uploaded via 3G by participant i in a month.

The solution to this problem is non-trivial, because:

1) We can neither foresee the participants' mobility traces in the next month, nor how much sensed data that needs to be uploaded. Thus, obtaining d_i is not straight-forward: both participant mobility and sensed data size prediction methods should be combined in order to estimate d_i.

2) Different $\mathcal{D}$ and $\mathcal{P}$ would affect $Refund_u$ and $Refund_p$ jointly. For example, if $\mathcal{P}$ assigns more participants to UnDP, then $Refund_u$ increases and $Refund_p$ decreases, so that whether overall $Refund$ increases or decreases remains uncertain. Even if $\mathcal{P}$ is determined, $\mathcal{D}$ can still impact $Refund_p$, because PAYG participants hold different uploading strategies under different $\mathcal{D}$.

4.3 The ecoSense Framework

To solve the problem formulated in the previous section, we design a novel mobile crowdsensing data uploading framework named ecoSense. The ecoSense framework is shown in Figure 4.2, which contains two key components:

1) **Uploading Decision Making** (*client component*). This component runs on every crowdsensing participant's smartphone. It is triggered to decide whether to upload/relay or keep data when a participant encounters a cost-free event, such as meeting another participant or discovering a Bluetooth/WiFi gateway. The *Uploading Decision Making* component will be further elaborated in Section 4.4.

2) **Participant Partition** (*server component*). This component runs on the crowdsensing organizer's server to assign the participants to either the UnDP or PAYG group. It relies on two modules - *mobility prediction* and *sensed data size estimation*.

[5]The problem is parameterized by the frequency with which we can update the participants' type of price plan. Currently, this is usually done on a monthly basis, but our approach is generalizable to different update frequencies which may be more likely in the future provision of crowdsensing services.

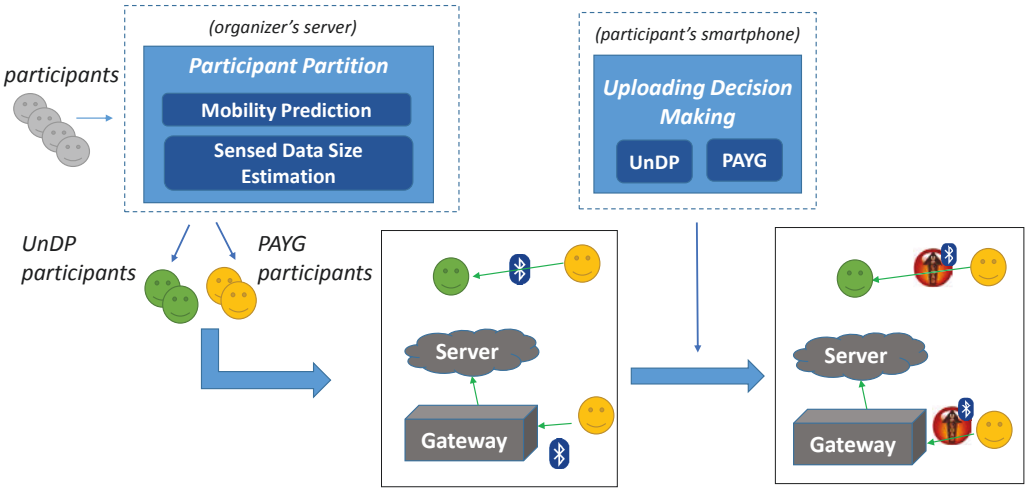


Figure 4.2: Overview of the ecoSense Framework

- *Mobility Prediction* module predicts participants' mobility patterns in the next month. With mobility prediction results, we can forecast how often a participant might meet another participant, a Bluetooth gateway, or a WiFi access point, etc.
- *Sensed Data Size Estimation* module estimates the amount of sensed data that a specific participant would contribute in the following month. For different participants, sensed data size might vary according to their activeness, privacy concerns, visited locations, etc.

Currently, most telecom operators' 3G data plans can change once a month, so this component needs to run once at the end of a month, to obtain the group partition for the following month. The *Participant Partition* component will be further elaborated in Section 4.5.

Now we briefly explain ecoSense's workflow during a crowdsensing task period:

1) As shown in the left part of Figure 4.2, before a new month begins, the *Participant Partition* component partitions all the participants into two groups with two different 3G refund schemes: UnDP and PAYG.

2) After the new month starts, in each delayed-uploading cycle, when a participant encounters a cost-free event (e.g. encountering a Bluetooth gateway or another participant), the *Uploading Decision Making* component decides whether to upload/relay or keep data. For example, in the right part of Figure 4.2, after making the decision, a PAYG participant relays data to an UnDP participant via Bluetooth, while another PAYG participant relays data to a Bluetooth gateway.

3) At the end of each delayed-uploading cycle, ecoSense checks all the participants to see whether they have *non-uploaded* data, which can include the sensed data collected by a participant himself and relayed data received from other participants. Then, ecoSense forces those participants with outstanding non-uploaded data to create 3G connections in order to upload it at the end of the cycle. In fact, only under this condition will PAYG participants upload sensed data with 3G data cost in a particular cycle.

4.4 Uploading Decision Making

In this section, we focus on introducing the strategies used in the uploading decision making component (called *uploading strategy*), especially for PAYG participants, because their uploading strategy will affect the organizer's 3G data refund budget. To make the work complete, we also introduce the uploading strategy for UnDP participants.

4.4.1 PAYG Uploading Strategy

We provide PAYG participants with three candidate uploading strategies: *OneRelay*, *OneHopFlooding* and *Epidemic*.

1) *OneRelay*: a PAYG participant u_p would relay her data when she encounters an UnDP participant u_u (or a Bluetooth/WiFi gateway) at the first time in the delayed-uploading cycle. After this relay, u_p's data uploading process ends successfully as u_u (or the gateway) would help upload u_p's data to the server. The name "*OneRelay*" just means that a PAYG participant will relay her data at most once. Note that using *OneRelay*, a PAYG participant will not relay her data to another PAYG participant.[6]

2) *OneHopFlooding*: a PAYG participant u_p would relay her data unconditionally to another PAYG participant encountered until u_p meets either one of the two following stopping criteria: (1) u_p directly encounters an UnDP participant (or a gateway) u_u and relays data to u_u, or (2) the server notifies u_p that she could stop flooding (which we will further discuss in the next subsection – *UnDP uploading strategy*). The name "*OneHopFlooding*" just means that the flooding is only one-hop, i.e. a PAYG participant u_{p1} will only flood her own data (i.e. u_{p1}'s data) to the PAYG participants encountered. That is, even if u_{p1} previously received a PAYG participant u_{p2}'s data, u_{p1} will not further flood u_{p2}'s data to other PAYG participants. However, if u_{p1} encounters an UnDP participant (or a gateway) u_u, u_{p1} will relay both u_{p1} and u_{p2}'s data to u_u.

3) *Epidemic* [123]: Removing the one-hop restriction of *OneHopFlooding* can lead to a complete flooding strategy, i.e., *Epidemic* routing. Using the *Epidemic* strategy, when two PAYG users meet, they will exchange all the data that they do not have in common, independent of whether the data is generated by themselves or received from someone else. To avoid redundant connections, a *reconnection time threshold* is set to ensure that two specific users exchange data at most once within a predefined time period [123]. For example, supposing the threshold is 30 minutes, then if two users u_1 and u_2 exchange data at 12:00, they will not reconnect with each other and exchange data until 12:30, even though they re-encounter between 12:00 and 12:30. Although a smaller threshold could incur a lower refund budget, in practice, we cannot greatly reduce the reconnection time threshold due to the energy consumption issue.[7]

[6] Actually, *OneRelay* is the same as $SimpleGreedy_{ndp}$ in effSense (Section 3.5.1). While to make this section self-contained, we still describe it here.

[7] We set the reconnection time threshold of *Epidemic* to one hour in the experiment.

In general, among the three strategies, *Epidemic* incurs the most data relays, while it can help the organizer to pay the smallest refund budget. *OneRelay* makes every relay valuable, but the refund budget is likely to be higher than for the other two strategies. All the strategies take the energy consumption (i.e. relay count) into account, as we cannot make the delayed uploading process too energy-draining in real-life scenarios. A detailed comparison of the three uploading strategies with respect to refund budget and energy consumption will be examined in our experiments.

Currently, we restrict our comparison of PAYG uploading strategies to *OneRelay*, *OneHopFlooding* and *Epidemic*, as they are easily implemented in participants' mobile phones. In our future work, we will evaluate other strategies such as *Binary Spray and Wait* [124].

4.4.2 UnDP Uploading Strategy

How UnDP participants upload data during delayed-uploading cycles does not affect the crowdsensing organizer's 3G data refund budget. For brevity, we assume that the UnDP participants use the *UpEnd* strategy, defined as follows:

UpEnd: UnDP participants keep all the sensed data collected by themselves, and received from other PAYG participants, until the end of the delayed-uploading cycle; at which point they upload all the data together.

Specifically, if PAYG participants adopt *OneHopFlooding* or *Epidemic*, UnDP participants will perform an additional action when they receive PAYG participants' data: If an UnDP participant u_u receives a PAYG participant u_p's data, u_u will notify the server that u_p's data will definitely be uploaded to the server by u_u before the end of the delayed-uploading cycle. Then, the server will notify u_p to stop flooding her data,[8] which corresponds to the second stopping criterion that we described previously in *OneHopFlooding* (the same stopping criteria also apply to *Epidemic*). While this additional action would not affect the organizer's 3G data refund budget, it could decrease the relay count of PAYG participants significantly and make *OneHopFlooding* and *Epidemic* more energy-efficient in real-life scenarios.

4.5 Participant Partition

After choosing the uploading strategy for the participants, the crowdsensing organizer also needs to partition the participants into two groups — PAYG and UnDP — in order to minimize the 3G data cost that needs to be refunded. Figure 4.3 shows the overview of the participant partition framework. To achieve a reasonable participant partition, two factors need to be considered:

[8]This notification will consume some data cost for PAYG participants. However, the amount of data cost incurred by the notification is usually small compared to the sensed data to upload. So we currently ignore it.

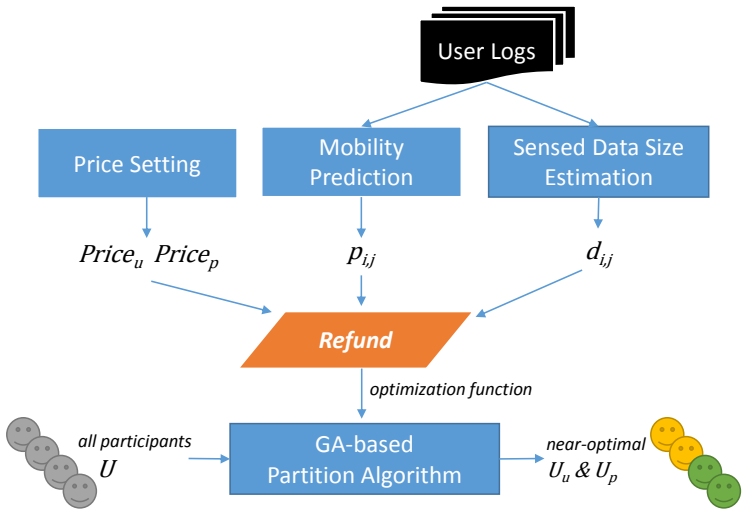


Figure 4.3: Overview of participant partition framework

- *Mobility Pattern.* A participant's mobility pattern affects how often she could meet another participant or a Bluetooth/WiFi gateway.
- *Sensed Data Size.* Different participants will most likely contribute different sizes of sensed data due to variant behaviors such as their degree of activity and their privacy concerns.

In this section, we first describe our methods to predict participants' *mobility pattern* and to estimate participants' *sensed data size*. Then we propose a genetic algorithm to partition the participants into UnDP and PAYG groups.

4.5.1 Mobility Pattern Prediction

Though mobility prediction has been studied comprehensively, most of the existing work focuses on short-term "Next Place" prediction [125, 126] and could not be applied directly to ecoSense. In ecoSense, we want to solve a long-term mobility pattern prediction problem:

In the next upcoming month, for each delayed-uploading cycle, what is the probability that a participant encounters another participant or a Bluetooth/WiFi gateway?

Specifically, through previous mobility trace analysis, we predict whether a PAYG participant can upload data without 3G data cost in a delayed-uploading cycle, i.e. the probability that she could meet at least one UnDP participant or one Bluetooth/WiFi gateway. We note this probability as $p_{i,j}$ for participant i and delayed-uploading cycle j.

Similar to effSense, we also use a Poisson-distribution-based method to predict $p_{i,j}$:

1. Represent a delayed-uploading cycle j as a triple: $(t_{start}, t_{end}, day\_type)$, e.g. $(8{:}00, 12{:}00, weekday)$, where t_{start} and t_{end} are the start and end time of the cycle, respectively, and day_type refers to *weekday* or *weekend*.

2. Find all the historic time spans with the same triple as the delayed-uploading cycle j (denoting the set of these historic time spans as HS_j), then count the total number of the cost-free events that could help u_i to upload data without additional 3G cost occurred in HS_j, denoted as $\#event_{free}(u_i, HS_j)$. Then $p_{i,j}$ can be predicted as follows, assuming the Poisson distribution stands:

$$p_{i,j} = 1 - e^{\#event_{free}(u_i, HS_j)/|HS_j|}$$

4.5.2 Sensed Data Size Estimation

Estimating how much sensed data each participant needs to upload is also important to obtain a participant partition to reduce total 3G cost. In this subsection, we attempt to model a participant i's sensed data size during the sensing cycle j, denoted as $d_{i,j}$.

1) *Fixed-Size Sensed Data.* Some sensing tasks will generate similar sensed data size in a sensing cycle. For example, the size of each user's accelerometer record is approximately proportional to the sensing duration time. So if all the users participate in a common sensing task for the same duration (e.g. activity recognition during daytime) in a sensing cycle, their contributed sensed data size will be similar.

For *fixed-size* sensing tasks, we model $d_{i,j}$ as:

$$d_{i,j} = c$$

where c is a constant, which means that for different participants and different sensing cycles, this sensing task would generate the same sensed data size.

2) *Varied-Size Sensed Data.* Some sensing tasks will generate different sensed data sizes for different participants in a sensing cycle for various reasons, e.g.:

- *Location-centric sensing.* This kind of sensing task usually triggers sensing when a participant enters a new location within a target sensing area, e.g. air quality and noise monitoring. Thus, the more places a participant visits, the more sensed data she would gather.
- *Activeness in participatory sensing.* Participatory sensing needs participants to be actively involved in the sensing tasks, e.g. taking photos at specific locations. Different participants can have different levels of activity, leading to different contributed sensed data sizes.
- *Private concern.* To protect privacy, some participants might choose not to upload part of the sensed data.

For *varied-size* sensing tasks, here we consider only the influence of locations visited on the size of the sensed data and model $d_{i,j}$ as:

$$d_{i,j} = c + k * l_{i,j}$$

where

Algorithm 1 GA-based Partition Algorithm

Input: $Refund$: optimization function; U: all participants
Output: U_p and U_u: participant partition

```
B ← bivector(U)   ▷ Change U into a bi-vector, where each element marks a participant as PAYG (0) or UnDP (1).
N(population) ← initialize with randomly assigning 0 or 1 in B for each candidate participant partition in the population.
i ← 0
while i < iter_max do
    K ← keepbest(N)   ▷ The best partitions are the ones that can get the smallest values on Refund function.
    C ← crossover(N)
    M ← mutation(N)
    N ← {K, C, M}
    i ← i + 1
end while
U_p, U_u ← the best participant partition that can achieve the smallest Refund during all the iterations.
```

- c is the size of the constant part of the sensed data. Though the data sizes of *varied-size* sensing tasks usually vary for different participants, still some part of the sensed data size is constant (e.g. some aggregation/summary information of the sensing task). In fact, we can also see *fixed-size* sensing tasks as a special form of *varied-size* sensing tasks.
- $k * l_{i,j}$ is the sensed data size for location-centric sensing, where k is the unit sensed data size for one location, while $l_{i,j}$ is the number of the locations that participant i would visit during the sensing cycle j. We predict $l_{i,j}$ via the same mobility prediction method previously discussed.

In real life, the sensed data size distribution is likely to be more complicated than suggested by our estimation formula, due to our abstracting away from subtle variations between different instances of sensing tasks. In our future work, we will model the sensed data size in a more precise way, taking into account more task-dependent and participant-dependent factors.

4.5.3 GA-based Partition Algorithm

Based on the mobility prediction and sensed data size estimation results, we can approximate the organizer's 3G data refund budget for the upcoming month. This 3G data refund budget approximation function takes a specific UnDP/PAYG participant partition as input: $Refund(U_u, U_p)$.

In other words, given a participant partition result (i.e. knowing U_u and U_p), we can approximate $Refund$ for the next upcoming month (supposing all the sensing cycles for the following month is C):

$$Refund = |U_u| * Price_u + \sum_{j \in C} \sum_{i \in U_p} d_{i,j} * (1 - p_{i,j}) * Price_p$$

Considering $Refund$ as the optimization function, participant partition can be directly expressed as a set split problem for minimizing $Refund$, which is NP-hard. Thus we use a genetic algorithm [127] to obtain a near-optimal solution. Algorithm 1 shows the pseudocode of the genetic algorithm. We use a bi-vector to mark each participant as an PAYG (0) or UnDP (1) participant (Line 1). In each generation, we keep the *best* participant partitions from the previous generation by evaluating $Refund$ function on every partition (Line 5). In addition to the best partitions, applying *crossover* (Line 6) and *mutation* (Line 7) functions [127] on the previous generation's population, we obtain the other two sets of candidate partitions, in order to compose the whole population that will be examined in the new generation (Line 8). Finally, the genetic algorithm could generate a near-optimal participant partition result. Note that we adopt a fixed number of iterations as the stopping criterion for the genetic algorithm (Line 4).

4.6 Evaluation

In this section, we first evaluate ecoSense using the MIT Reality Mining dataset [113] including 48 active users. Then, to further evaluate both ecoSense's budget saving efficacy and algorithm computation efficiency with a larger number of users, we simulate a 500-user mobility trace leveraging SWIM [122] and show the corresponding results.

4.6.1 Experimental Setup on MIT Reality Mining

For the MIT Reality Mining dataset, we choose two months' data (2004.10 and 2004.11) from 48 active users who have more than 20-day records per month. The mobility traces in the MIT Reality dataset include each user's Bluetooth encounters with other users and gateways deployed by the organizer. Note that to ensure that the sensed data can be successfully offloaded between two users via Bluetooth, we only use the Bluetooth encounters lasting for more than 5 minutes in the experiment. We use the first month's user data (2004.10) as the historic record to obtain a participant partition of PAYG and UnDP groups, and evaluate the performance on the second month (2004.11). The sensing cycle lasts for one day and the delayed-uploading cycle lasts for 3 hours[9] (recall Figure 3.2 for the illustration of sensing and delayed-uploading cycles). In this experiment, we set two Bluetooth gateways named *localhost.media.mit.edu* and *studies.media.mit.edu*. The costs of PAYG and UnDP price plans are set to \$0.1/MB and \$7/month respectively.

Adding up-to-date WiFi usage logs. To mitigate the shortage of WiFi usage logs in the MIT dataset[10], we use an up-to-date mobile phone usage dataset, Cambridge Device

[9]The delayed-uploading cycle starts at noon every day.

[10]Back to 2004 when the MIT Reality Mining campaign was conducted, WiFi was not a popular data

Analyzer [128], to complement the MIT dataset. Specifically, for each MIT user u_{mit}, we randomly select a Cambridge user u_{cam} and uses u_{cam}'s latest two-month WiFi usage logs to represent u_{mit}'s WiFi usage.

Setting the battery threshold for relaying data. Similar like the previous work effSense, in this experiment, a user will relay others' data only when the battery level of her phone is above a predefined threshold (50%). More details about how to simulate the battery consumption please refer back to Section 3.6.1.

4.6.2 Baseline Methods

To compare with ecoSense, we introduce the following two methods:

1. direct-assignment is the baseline method that assigns each participant to the UnDP or PAYG group directly according to her estimated sensed data size in the upcoming month.
2. *ecoSense-ideal* does not do mobility prediction and directly leverages the second month's mobility trace (2004.11) in the genetic algorithm in order to get an optimal participant partition. In real life, this future data is not available; however, this result serves as a bound of the organizer's refund budget and shows the potential improvement that we could make by designing a better mobility prediction method in the future.

4.6.3 Evaluation Results on MIT Reality Mining

The most important issue is how much ecoSense could save for the organizer's 3G data refund budget when compared to direct-assignment. We will first show the results when all the participants upload the same size of sensed data in each sensing cycle (i.e. *fixed-size data* setting). Then, we will change the *fixed-size data* setting to the *varied-size data* setting, where the size of the sensed data that a participant would contribute is proportional to the number of visited locations. For the sake of simplicity, the above experiments are conducted using the *OneRelay* uploading strategy (the "uploading strategy" in the evaluation section refers to "PAYG uploading strategy"). Afterwards, we will illustrate the difference between the *OneRelay*, *OneHopFlooding* and *Epidemic* uploading strategies, from two perspectives: refund budget and energy efficiency. Finally, we evaluate how two parameters — *max tolerable delay* and *3G price plan cost* — affect ecoSense's performance.

Fixed-Size Sensed Data Uploading

Here, we show how much refund cost ecoSense could save compared to direct-assignment for sensing tasks generating fixed-size sensed data. We adopt *OneRelay* as the uploading strategy. To see whether ecoSense's Poisson-distribution-based mobility prediction method

transmission method for mobile phones.

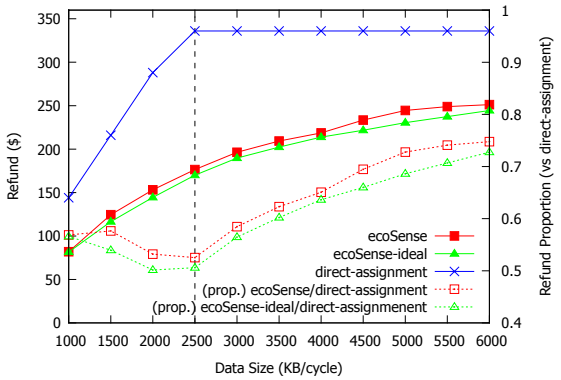


Figure 4.4: Refund budget for fixed-size data uploading

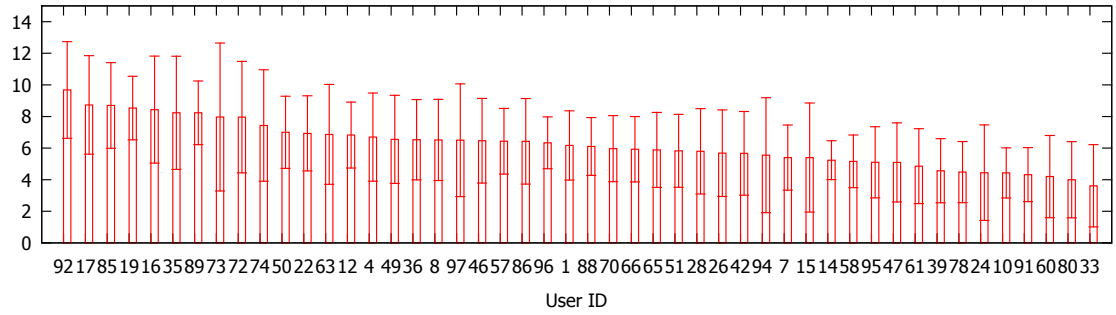


Figure 4.5: Average number of visited cell towers per day for each participant in 2004.11

has already achieved good enough performance or not, we also compare ecoSense with *ecoSense-ideal*.

Figure 4.4 shows the mobile data cost refund under different fixed data size settings: 1000-6000 KB/cycle. ecoSense could save 25-48% monetary refund compared to direct-assignment. We find that the data size setting where ecoSense can get the most significant effect ($\sim 48\%$ saving) is around 2500 KB/cycle, where direct-assignment changes from assigning the participants with PAYG to UnDP (denoted as *turning point*, the vertical dashed line in Figure 4.4).

Compared to *ecoSense-ideal*, ecoSense's refund budget is larger by only 1-4%. This demonstrates that with the Poisson-distribution-based mobility prediction method, ecoSense has already achieved good performance. It may be possible to design a better mobility prediction method, but even with higher accuracy, the improvement on refund budget saving is not likely to be significant.

Varied-Size Sensed Data Uploading

We now consider the 3G data refund budget for varied-size sensing tasks. As mentioned in Section 4.5.2, the estimated varied data size for participant i in sensing cycle j can be modeled as $d_{i,j} = c + k * l_{i,j}$.

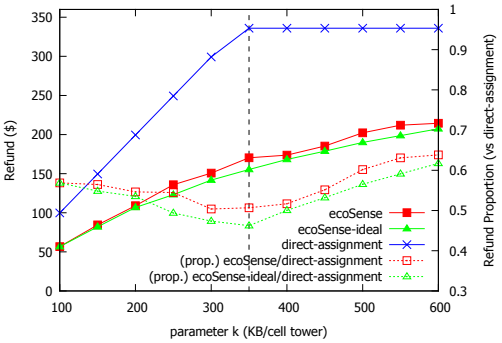


Figure 4.6: Refund budget for varied-size data uploading ($c = 0$)

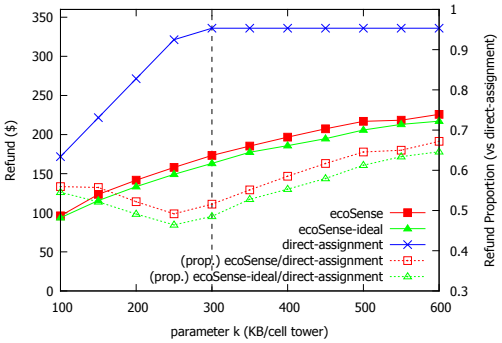


Figure 4.7: Refund budget for varied-size data uploading ($c = 500KB$)

We write $l_{i,j}$ for the number of cell towers the participant i stayed in during sensing cycle j.[11] Figure 4.5 illustrates the average number ($avg \pm std$) of the visited cell towers per sensing cycle for each participant in 2004.11. The average number of visited cell towers for different participants ranges from 3.6 to 9.7 per sensing cycle. For constant c in the $d_{i,j}$ model, when $c = 0$, we assume that the participants sense only in each visited place. When $c \neq 0$, we assume that besides the sensed data for each visited place, participants also upload some other sensed data, e.g. aggregated coarse activity log, which accounts for approximately the same size of data for different participants (e.g. 500 KB).

Figures 4.6 and 4.7 show the organizer's 3G data refund budget when the sensed data size for each cell tower is set as k ranges from 100 to 600 KB/cell tower, where c is set to 0 and 500 KB respectively. Similar to the evaluation results for the fixed-size data setting, compared to direct-assignment, ecoSense could save 33-51% of the refund budget. Besides, the most significant budget saving also appears around the *turning point* (the vertical lines in Figure 4.6 and 4.7), where direct-assignment begins to assign all the participants to UnDP.

Compared to *ecoSense-ideal*, ecoSense's refund budget is larger by 1-5%, which gives the range for improvement through the introduction of a more precise mobility prediction method.

Different Uploading Strategies

Previous experiments all run under the *OneRelay* uploading strategy. Here we study how different uploading strategies (*OneRelay*, *OneHopFlooding*, or *Epidemic*) would affect ecoSense's performance. In addition to the 3G data refund budget, we also consider the energy consumption in participants' smartphones by counting participants' relays and battery drain per delayed-uploading cycle. For simplicity, we consider only the fixed-size data setting.

Figure 4.8 shows the 3G data refund budget when PAYG participants adopt different uploading strategies for fixed-size data. As expected, *Epidemic* achieves the smallest refund budget, while *OneRelay* incurs the largest. Specifically, *Epidemic* could save 2-15% more budget than *OneRelay*. For smaller sensed data sizes, the improvement is more pronounced

[11] To avoid those cell towers that a participant just passed by, we record a cell tower only if a participant stayed in the vicinity of the cell tower for more than 5 minutes.

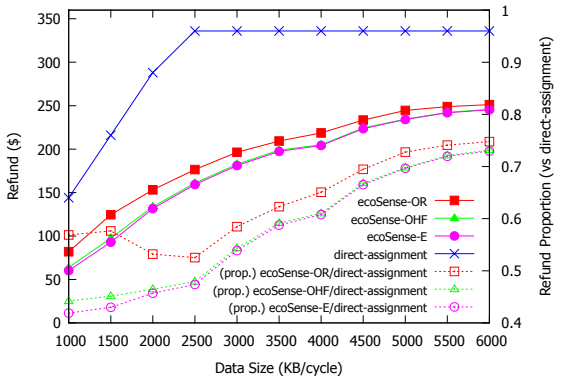


Figure 4.8: Refund budget for different uploading strategies (*OR: OneRelay*, *OHF: OneHopFlooding*, *E: Epidemic*)

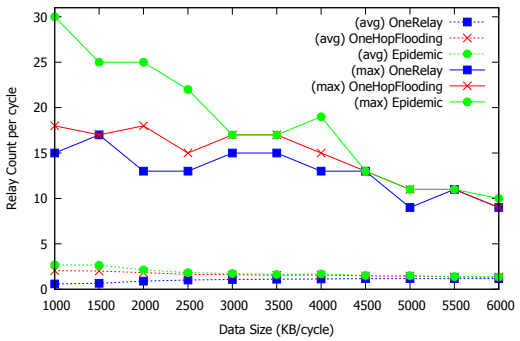


Figure 4.9: Relay count for different uploading strategies

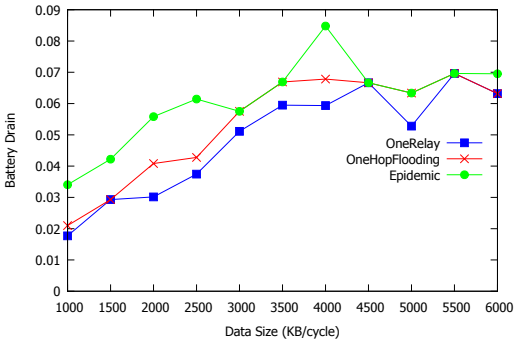


Figure 4.10: 2000mAh battery drain of the worst participant for different uploading strategies

(e.g. 15% for 1000-1500 KB/cycle, while 2% for 6000 KB/cycle). This occurs because, for smaller data packets, fewer participants will be assigned to UnDP, leading to more PAYG participants. The large number of the PAYG participants consequently improve the performance of *Epidemic*. For *OneHopFlooding*, it achieves almost the same refund budget as *Epidemic* with the increase in data size. This indicates that with more UnDP participants for larger data packets, flooding a PAYG user's data within only one hop has already achieved high probability of making the data received by an UnDP participant or gateway (thus reducing the budget); using *Epidemic*, i.e., removing the restriction of one-hop of *OneHopFlooding*, does not increase this probability significantly.

Figure 4.9 shows the average relay count per delayed-uploading cycle for a participant, as well as the maximum relay count among all the participants (including both PAYG and UnDP groups). *Epidemic* and *OneHopFlooding* incur the similar average relay count, which is 1.2–4.6 times larger than *OneRelay*. Considering the maximum relay count for a participant in a delayed-uploading cycle (i.e. the worst energy consumption case), *Epidemic* is significantly larger than *OneHopFlooding* and *OneRelay*, especially when the data size is less than 3000 KB/cycle. For example, when data size is 1000 KB/cycle, the maximum relay count for *Epidemic* is 30, while only 18 and 15 for *OneHopFlooding* and *OneRelay*, respectively. The maximum relay count for *OneHopFlooding* is also larger than *OneRelay*,

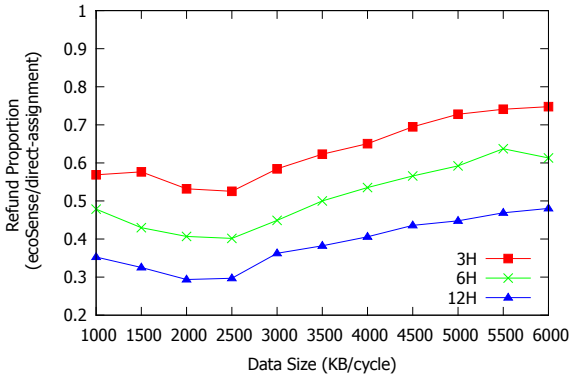


Figure 4.11: Refund budget proportions for varying maximum tolerable delays

however the difference is usually smaller than the average relay count (less than 1.5 times). Specifically, using *OneHopFlooding* would trigger at most 5 more relays than *OneRelay* for the "worst participant" who consumes the most energy for relays. Based on previous research on mobile phone energy consumption [117], we calculate the mobile phones' battery drain for the "worst participant" with different uploading strategies. Figure 4.10 shows that — for a 2000mAh battery — the battery drain with *OneHopFlooding* and *OneRelay* is always less than 7%, while the battery drain difference between the two strategies is at most 2%; in comparison, the battery drain with *Epidemic* is a bit higher, leading to 8.5% in the worst case. For real-life scenarios, the strategy selection among *Epidemic*, *OneHopFlooding* and *OneRelay* needs to be further studied in order to better balance the 3G refund saving and the participant mobile phones' battery drain.

Other Experimental Parameter Analysis

To better understand how *max tolerable delay* and *3G price plan cost* affect ecoSense's performance, in this section, we use some different parameter settings from the ones used in the previous experiments.

Varying Maximum Tolerable Delays

Here, we conduct the experiments to test ecoSense's performance for various delays other than 3-hour. Suppose the uploading strategy is *OneRelay*, Figure 4.11 shows the 3G data refund budget proportion (ecoSense vs. direct-assignment) for 3/6/12-hour maximum delays. As the delay increases, the 3G refund budget becomes lower. This is because adopting a longer delay, participants have more opportunities to relay data via another participant or a Bluetooth/WiFi gateway to reduce 3G data cost.

Varying 3G Price Plan Costs

The costs of different telecom operators' 3G price plans can vary. Thus, we also examine the evaluation results for various price plan settings.

Suppose the uploading strategy is *OneRelay*, Figure 4.12 shows the refund budget proportion (ecoSense vs. direct-assignment) for three different price plan settings — where

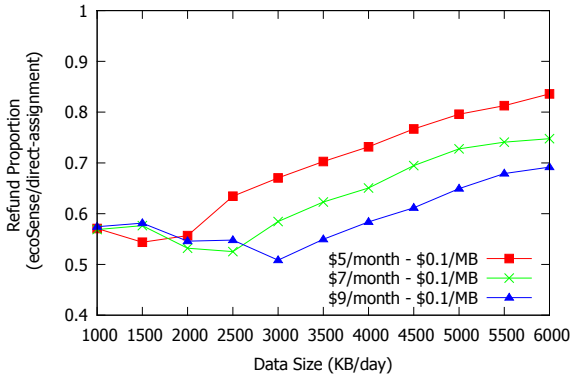


Figure 4.12: Refund budget proportions for varying price settings

data size	ecoSense	direct-assignment	*saving ratio*
500 KB	$368	$750	51%
1500 KB	$573	$2250	75%
2500 KB	$728	$3500	79%
3500 KB	$849	$3500	76%
4500 KB	$956	$3500	73%

Table 4.1: Refund budget for 500-user SWIM simulation

PAYG prices are all set to \$0.1/MB and UnDP prices are set to \$5, \$7, \$9/month respectively. The most significant difference among different settings is that the *turning point* (i.e. the data size where the refund budget proportion is the lowest) changes, because the *turning point* is the sensed data size where direct-assignment begins to assign all the participants to UnDP, which is sensitive to the price plan setting. Except for the *turning point*, most observations are similar for different settings. For example, whatever the price plan settings, ecoSense could save at most about 50% of the refund budget compared to direct-assignment.

4.6.4 Experiment on SWIM Simulation

To evaluate ecoSense's budget saving efficacy and algorithm computation efficiency (especially the participant partition algorithm) on a larger number of users, we simulate a user-encounter dataset containing 500 users' two-month traces by SWIM [122]. The simulation parameters are selected according to the "Dartmouth" setting in [122], in order to simulate user encounters in a campus. The experimental settings are similar to those used with the MIT Reality Mining dataset: the sensing cycle lasts for one day and delayed-uploading cycle lasts for 3 hours. For simplicity, we show only the evaluation results when the uploading strategy is *OneRelay* and the sensing task is fixed-size.

Budget Saving Efficacy

Table 4.1 shows the refund budget with the 500-user simulation dataset. As the user number increases to 500 and the users' activity area remains within the campus-like scale of the MIT dataset, the user spatial density in the SWIM simulation is much larger than that for the MIT dataset, which leads to more refund budget saving. For example, when the fixed data size is 2500 KB, ecoSense can save 79% in the 500-user simulation dataset, with respect to 48% on the 48-user MIT dataset (see Figure 4.4). In other words, if more users are active in a smaller area, they will have more chances to meet each other so that ecoSense could save more 3G data cost.

Algorithm Computation Efficiency

With respect to computation efficiency, we focus on the run-time performance of the server-side participant partition component, as the smartphone-side uploading decision making component's strategy (*OneRelay*, *OneHopFlooding* or *Epidemic*) is simple. For the participant partition component, the genetic algorithm dominates the running time, because our proposed mobility prediction and sensed data size estimation methods are simple and run much faster than the genetic algorithm.

For the genetic algorithm, the main difference between 48 users (MIT) and 500 users (SWIM simulation) is that for more users we need more iterations to obtain a near-optimal result. In our experiment environment[12], we can get a good solution in 50 rounds for the 48-user MIT dataset (approximately 5 minutes of execution time), while we need 500 rounds for the 500-user simulation dataset (approximately 90 minutes of execution time). As the participant partition is an offline algorithm, which only needs to run once a month, we believe that this execution efficiency is already adequate for most real-life conditions. Furthermore, the performance of the genetic algorithm can be easily (and dramatically) improved by leveraging its inherent parallelism.

4.7 Discussion

As this is the first research work investigating how to refund crowdsensing participants' 3G data usage incurred by sensing tasks and it is still at an early age, we will discuss some issues which are not addressed due to space limit, and point out some future potential research directions.

4.7.1 Other Kinds of Monetary Incentives

Currently, we consider only the refund to cover crowdsensing participants' additional 3G data cost. In real life, other kinds of monetary incentives can also be provided to the participants, such as a fixed payment for the participation and higher reward (prize) for a small number of most active participants.

[12]Software: DEAP (https://github.com/DEAP/deap) with python 2.7, Windows 7; Hardware: Intel core i7-3612QM@2.1GHz, 8G RAM.

Though the refund of 3G data cost is only a part of the total monetary incentives, it could be a reasonable baseline because refunding each participant with her 3G data cost can mitigate the participant's worry about whether the 3G data usage would exceed her data plan and incur extra fees [129, 77]. In other words, refunding 3G data cost could at least prevent the participants from paying extra fees because of participation in a crowdsensing task. Other kinds of monetary incentives could be added to the 3G data cost refund to further encourage users to participate in the sensing task.

4.7.2 Participants' Personal 3G Data Usage

In addition to the 3G data usage for the crowdsensing task, participants also consume 3G data for their personal application usage. Here, we prove that ecoSense can always work effectively (i.e. refund can cover additional 3G cost incurred by the crowdsensing task) even if we do not know participants' actual personal data usage.

For an UnDP participant, personal data usage does not matter because the unlimited 3G data plan covers it automatically. However, for a PAYG participant, each of the two price plan cases requires further clarification and analysis:

1. If a PAYG participant u_p's original price plan is also PAYG (i.e. small personal data usage), then ecoSense's refund for u_p just equals the additional 3G data cost incurred by the sensing task.
2. If a PAYG participant u_p's original price plan is UnDP (i.e. large personal data usage), obviously u_p's reasonable choice is *still using UnDP as price plan after participating in the sensing task*, which means that u_p's additional 3G data cost incurred by the sensing task is 0. Currently, ecoSense would still refund this kind of participant through the PAYG scheme (i.e. refunding them the money proportional to their 3G-uploaded data size), because we cannot easily distinguish them from the previous *small-personal-data-usage* kind of participant.[13]

In summary, without knowing the actual personal data usage for each participant, ecoSense's refund mechanism can always cover each participant's additional 3G data cost incurred by the crowdsensing task.

4.7.3 Other 3G Price Plans

Usually telecom operators offer users 3G price plans other than PAYG and UnDP. For example, *D100MB* price plan: \$1 for the first 100MB data and then \$0.1/MB (like PAYG). Here, we discuss the problem: whether ecoSense's refund mechanism can still cover participants' additional 3G cost when other 3G price plans exist?

On one hand, by still using only two refund schemes of PAYG and UnDP, even if other 3G price plans exist, ecoSense can effectively refund participants to cover their additional

[13] We can distinguish these two kinds of participants provided we assume that all the participants trustfully report their original price plans. However, due to the privacy concern and user selfishness, we do not make this assumption in our current work.

3G cost. For example, assume that a participant's original price plan is D100MB. If refunded as UnDP, she can change next month's price plan to UnDP; and if refunded as PAYG, she can keep the D100MB price plan. As long as a participant follows the above rule, ecoSense's refund can always cover her additional 3G cost.

On the other hand, it is promising to introduce new 3G price plans, such as D100MB, into refund schemes of ecoSense. For example, after importing D100MB as a new refund scheme (i.e. refund schemes increase to three types: *UnDP*, *D100MB*, and *PAYG*), we could model the participants with D100MB refund scheme as "UnDP" participants when data usage is less than 100MB, and as "PAYG" participants when data usage exceeds 100MB. This type of polymorphic participant partition mechanism requires further research.

4.8 Concluding Remarks

Refunding mobile crowdsensing participants for additional 3G data cost incurred during the crowdsensing process is an effective marketing strategy for the organizer. In this chapter, we investigate the problem of how to minimize total 3G data refund budget for the crowdsensing organizer who follows such a marketing strategy, especially via heterogeneous network communications and user collaborations. Based on two widely-used 3G price plans, *Pay As You Go* and *Unlimited Data Plan*, we propose a delay-tolerant collaborative data uploading framework called *ecoSense*, whose goal is to minimize the organizer's 3G refund budget. Specifically, ecoSense includes the data uploading strategies for both PAYG and UnDP participants, as well as a participant partition algorithm to determine whether a participant should be assigned to PAYG or UnDP. Our ecoSense framework was evaluated using the MIT Reality Mining dataset and a larger SWIM simulation dataset. The evaluation results showed that ecoSense could save up to ~50% of the refund budget compared to direct-assignment that assigns each participant to UnDP or PAYG directly according to the size of her sensed data.

Part III

Sparse Mobile Crowdsensing

Sparse Mobile Crowdsensing

In a traditional MCS paradigm, to obtain a high-quality sensed result, organizers usually need to recruit enough participants so that their uploaded sensed data can cover almost the whole target sensing area (full or high coverage). Nevertheless, this strategy may still incur high sensing cost, including overall smartphone energy and network bandwidth consumption, as well as incentives paid to the participants. Then, one question arises: *is it possible to reduce the sensing cost by only sensing a small number of the sub-areas, meanwhile still guarantee satisfactory data quality for the whole target area?*

To address this question, in this part of the dissertation, we consider the spatial and temporal correlation among the data sensed in different sub-areas to significantly reduce the required number of sensing tasks allocated, and thus propose a new MCS paradigm called **Sparse Mobile Crowdsensing (Sparse MCS)**. Sparse MCS applications intelligently select only a small part of the target area for sensing while inferring the data of the remaining unsensed area with high accuracy. In Chapter 5, we describe the basic idea and research issues in Sparse MCS, as well as implement a prototype Sparse MCS system, called *CCS-TA*, for urban environment monitoring.

As location privacy is an important concern which impacts MCS participants' activeness, in Chapter 6, we also study this issue specifically for Sparse MCS applications. To this end, we introduce a differential location privacy mechanism, called *DUM-*ϵe, to obfuscate participants' actual locations before data uploading, and meanwhile minimize the data quality loss incurred by the privacy protection.

Chapter 5

CCS-TA: Quality-Guaranteed Task Allocation for Sparse Mobile Crowdsensing

Contents

5.1 Introduction

To obtain high-quality sensed results in MCS applications, existing work usually aims at recruiting enough participants so as to ensure that their sensed data can cover almost the whole target area, i.e., *full coverage* [92, 93] or *high probabilistic coverage* [95, 8, 130, 94]. Specifically, by ensuring coverage of each sensing sub-area or cell (full coverage) or most of the cells (probabilistic coverage) by mobile participants, the MCS applications attempt to obtain accurate and reliable sensing values throughout the target area, in order to compile a full sensing picture which meets the needs of the MCS organizers. The underlying connection between full coverage (or high probabilistic coverage) and data quality is that the full coverage (or high probabilistic coverage) can ensure the MCS organizers get representative sensing values for the target area. In essence, for the MCS organizers, the data quality requirement is to *obtain a reasonably accurate sensing value for each sub-area (cell)*.

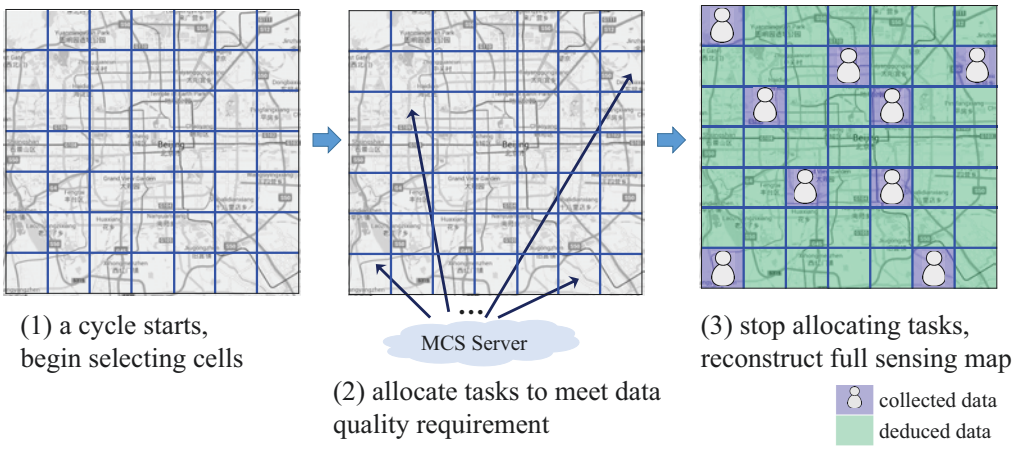


Figure 5.1: Use case: temperature monitoring in an urban area.

While ensuring full or high probabilistic coverage is a straightforward approach to obtain representative data for each cell, the down side is that whatever the efficient task allocation mechanism is used, the MCS organizer has to allocate tasks to almost every cell at least once [93]. This may still lead to high incentive budget, especially when the organizer plans to carry out a variety of MCS campaigns at a large scale.

In order to further reduce the number of recruited participants, an alternative approach is to select only a few cells for physical data sensing, while inferring the data of the rest of cells of the target area and ensuring the data accuracy. This alternative method is actually feasible as many kinds of environmental data, such as temperature [131] and noise [4, 132], have been shown to be able to be inferred with good quality due to the strong temporal and spatial correlations among the data.

With this insight in mind, we propose to use the overall sensing data accuracy, rather than the sensing area coverage, as the data quality metric. We further exploit the temporal and spatial correlations among the sensing data to infer the missing data of unsensed cells from the sensing data in selected cells. By actively selecting a minimal number of cells for task allocation, we try to minimize the number of recruited participants, while ensuring the overall data accuracy meets a predefined bound. We name this novel MCS paradigm, which applies inference algorithms to construct the complete sensed data map from partial sensed data, as *Sparse Mobile Crowdsensing* (*Sparse MCS*).

The basic idea can be illustrated by the following use case (Figure 5.1): an MCS organizer launches an environment temperature monitoring task in a target urban area, which has already been divided into cells according to the organizer's requirement. The organizer needs to update the full temperature sensing map once every hour (sensing cycle), and in each sensing cycle, the data quality requirement is that the mean absolute error for the whole area should be less than $0.25^{\circ}C$. In each sensing cycle, to meet the data quality requirement while minimizing the number of the allocated tasks, the organizer actively selects a subset of the cells to sense physically, i.e., allocating tasks to the participants in those selected cells. Based on the sensed temperature values of those selected cells, the temperature values of the remaining cells are inferred.

To accomplish the Sparse MCS task described in the use case above, we need to address the following two research challenges:

1) How many and which cells should be chosen for task allocation? In each cycle, to minimize the number of allocated tasks while ensuring the data quality, the organizer needs to select a minimum subset of the cells to allocate tasks. In order to find this minimum cell combination, we need to identify the *salient* cells whose sensing values, if collected, can help infer the values of other cells to the maximum extent. However, how to identify the salient cells in an online manner is not trivial, because without foreknowing the true sensing value of a cell, it is hard to predict how much that value can help increase the data accuracy if collected.

2) How to quantitatively measure and estimate the data quality online throughout an MCS task without knowing the true sensing values of the unsensed cells? Since the true sensing values of the unsensed cells are unknown, we cannot measure the data accuracy directly by comparing the inferred data with the unknown ground truth. We thus need a method to efficiently estimate such sensing data accuracy online in each sensing cycle. Furthermore, as the estimated data accuracy intrinsically has certain discrepancy from the actual accuracy, it is practically hard to strictly guarantee *all* the sensing cycles to meet a predefined error bound. Therefore, we need to find a practical way to define the data quality requirement for an MCS task instead of merely setting the error bound for each sensing cycle.

With the above-mentioned research objective and challenges, the main contributions of the work are:

1) We propose a *novel sensing data quality metric* to assess the quality of an MCS task, called (ϵ, p)-*quality*, which considers not only the required error bound ϵ but also what fraction p of sensing cycles should meet the bound ϵ. We further propose to exploit the temporal and spatial correlations among the sensing values of different cells to significantly reduce the number of allocated tasks. To the best of our knowledge, this is the first work in MCS that attempts to reduce the number of cells being sensed by intelligently selecting which cells are best for sensing while inferring the missing values of the remaining cells, to ensure that the overall data accuracy meets a predefined quality requirement. Specifically, we coin a new term, *Sparse Mobile Crowdsensing*, to refer to such a novel MCS paradigm.

2) In order to reduce the number of sensing cells required for task allocation in each sensing cycle while achieving the predefined (ϵ, p)-quality in Sparse MCS, we propose a two-phase online task allocation framework, called *CCS-TA (Compressive CrowdSensing Task Allocation)*. With CCS-TA, in phase one we decide which cell is the best to add for sensing in each cycle according to *active learning* techniques. After collecting the sensing data from the selected cells, in phase two we propose a *Bayesian inference* based method to estimate the overall data accuracy online after inferring the data of the remaining cells using *compressive sensing*. Based on the estimated overall data accuracy, CCS-TA decides whether more cells should be selected for task allocation to ensure the predefined (ϵ, p)-quality.

3) We conduct extensive evaluations on real-life temperature [133] and air quality [134] monitoring datasets to show the effectiveness of CCS-TA. In the case of temperature monitoring, CCS-TA needs to assign tasks to only 15.5% of the cells on average which can ensure the overall sensing error below 0.25°C in 95% of the cycles, i.e., satisfying $(0.25°C, 0.95)$-quality. As a comparison, the baseline approaches need to allocate 18.0-26.5% more tasks to ensure the same data quality.

5.2 Problem Statement

In this section, we first define the key concepts used throughout this chapter. Then, we clarify our assumptions, followed by the problem formulation.

5.2.1 Definitions

To formally define the sensing data quality metric, we first define the concepts about the *sensing and selection matrices* (see Figure 5.3 for an example).

Definition 5.1. *Full Sensing Matrix. For a location-centric MCS task involving m cells and n sensing cycles, its full sensing matrix is denoted as F_{m*n}, where each entry $F[i, j]$ denotes the true sensing data of cell i in cycle j.*

Definition 5.2. *Cell-Selection Matrix. In a cell-selection matrix S_{m*n}, each entry $S[i, j]$ denotes whether or not the corresponding entry in the full sensing matrix $F[i, j]$ is selected for sensing: if cell i is selected for sensing in cycle j, $S[i, j] = 1$; otherwise, $S[i, j] = 0$.*

Definition 5.3. *Collected Sensing Matrix. A collected sensing matrix C_{m*n} records the actual collected sensing data:*

$$C = F \circ S$$

where $\circ$ denotes the element-wise product of two matrices.

Definition 5.4. *Sensing Matrix Reconstruction Algorithm. A sensing matrix reconstruction algorithm $\mathcal{R}$ attempts to reconstruct a full sensing matrix $\hat{F}_{m*n}$ from the collected sensing matrix C_{m*n}:*

$$\mathcal{R}(C_{m*n}) = \hat{F}_{m*n} \approx F_{m*n}$$

Now, we define the *overall sensing error*, which represents the data quality.

Definition 5.5. *Overall Sensing Error. It quantifies the difference between the reconstructed full sensing matrix $\hat{F}$ and the true full sensing matrix F. In this work, we focus on the*

overall sensing error of each sensing cycle separately. For sensing cycle k, the overall sensing error is defined as:

$$\mathcal{E}_k = error(\hat{F}[:,k], F[:,k])$$

where $F[:,k]$ is the kth column of F, i.e., the true sensing values of all the m cells in cycle k, and $\hat{F}[:,k]$ contains the corresponding inferred sensing values by using the reconstruction algorithm $\mathcal{R}$.

Note that the specific technique to calculate the overall sensing error (i.e., the *error*() function) depends on the type of sensing data. In this work, we focus on two widely-used metrics: *mean absolute error* (for continuous values, e.g., temperature [135]), and *classification error* (for classification labels, e.g., PM2.5 air quality index (AQI) descriptors [134]).

- *Mean Absolute Error*

$$error(\hat{F}[:,k], F[:,k]) = \frac{\sum_{i=1}^{m} |\hat{F}[i,k] - F[i,k]|}{m} \tag{5.1}$$

- *Classification Error*

$$error(\hat{F}[:,k], F[:,k]) = 1 - \frac{\sum_{i=1}^{m} I(\psi(\hat{F}[i,k]), \psi(F[i,k]))}{m} \tag{5.2}$$

  where $\psi()$ is the function to map a value to its classification label. $I(x,y) = 1$ if $x = y$; otherwise, 0.

With the overall sensing error, we define the data quality for an MCS task including n sensing cycles:

Definition 5.6. *(ϵ, p)-quality. For an MCS task lasting for n sensing cycles, it satisfies (ϵ, p)-quality, iff*

$$|\{k|\mathcal{E}_k \leq \epsilon, 1 \leq k \leq n\}| \geq n \cdot p$$

where ϵ is a predefined error bound for the overall sensing error of each cycle, and p is a predefined probability threshold to quantify the minimum fraction of the cycles whose errors should satisfy the error bound ϵ.

Ideally, for a predefined error bound ϵ, we expect that an MCS task keeps the overall sensing error lower than ϵ in *all* ($p = 1$) the cycles. However, it is intractable for a real-life MCS task to satisfy $(\epsilon, 1)$-quality because we cannot know the accurate overall sensing error $\mathcal{E}_k$ but have to estimate it (as the ground truth F cannot be foreknown). Thus we focus on the cases where p is large (e.g., 0.9 or 0.95), to guarantee the overall sensing error bounded by ϵ in *most* (e.g., 90% or 95%) cycles. Later we show how this allows us to use some techniques from probability theory and Bayesian statistics to handle the problem.

5.2.2 Assumptions

We make the following assumptions.

Assumption 1. *Massive Candidate Participants. There are sufficient number of candidate participants across the target sensing area, so for any cell in any sensing cycle, the organizer can always find a participant to allocate a sensing task.*

We believe this assumption is realistic in many current and future MCS applications. For example, the crowdsourcing traffic monitoring application *WAZE* (https://www.waze.com) has already appealed to more than 50 million users. With this massive user base, this assumption can be easily satisfied, especially in the user-intensive urban areas.

Assumption 2. *High Quality Sensing. Every participant returns the accurate sensing value if a task is allocated to her.*

The *assumption 2* also appears in other existing research work [15]. We note that in real life, it is not always true due to possible issues such as sensor error or varying conditions. But with an attractive incentive scheme in place, this assumption is also reasonable.

Assumption 3. *Not Moving Out During Sensing. After a participant receives a sensing task in a cell, she will not move out of the cell before she finishes sensing.*

Assumption 3 ensures that if we allocate a sensing task to a participant in cell i, her returned sensing value will actually represent cell i. This assumption can usually be satisfied if the sensing task does not consume much time. For example, with an embedded ambient temperature sensor, a smartphone can obtain the temperature reading in a few seconds;[1] for air quality monitoring, usually the sensor needs 30-60 seconds to be prepared to start sensing and then the sampling cycle is 2-10 seconds [137, 13].

Assumption 4. *Cycle-Length-Satisfying Sequential Sensing. Each sensing cycle is sufficiently long to collect enough sensing values by sequentially allocating tasks — the next task is allocated after the sensing value of the previous task is returned.*

Like *assumption 3*, if the sensing task is quick (e.g., a few seconds for temperature sensing), then *assumption 4* can also be satisfied for most MCS tasks with reasonable-length cycles (e.g., 30-minute cycle).

In summary, the above assumptions are made for the following reasons:

- *Assumption 1* allows us to reduce the task allocation problem to a *cell selection* problem.
- Combining *assumptions 2 and 3*, we only need to allocate one task to one participant in cell i during cycle j in order to get the true sensing value from cell i in cycle j.
- *Assumption 4* allows us to use an iterative method, i.e., progressively selecting the cells for sensing, in each cycle, for solving the *cell selection* problem.

[1] The response time of the temperature/humidity sensor SHTC1 of Galaxy S4 is about 8 seconds [136].

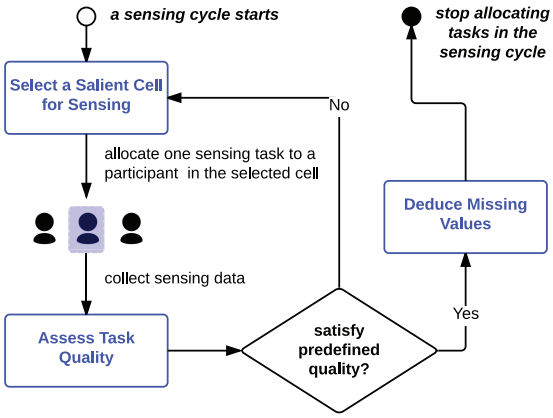


Figure 5.2: Workflow of CCS-TA.

5.2.3 Problem Formulation

Based on the previous definitions and assumptions, we formulate the research problem as follows: *Given an MCS task with m cells and n cycles, and a sensing matrix reconstruction algorithm $\mathcal{R}$, select a minimal subset of sensing cells during the whole MCS task process (i.e., minimize the number of non-zero entries in the cell-selection matrix S), while ensuring that the overall sensing errors of at least $n \cdot p$ cycles are below the predefined error bound ϵ (i.e., satisfying (ϵ, p)-quality):*

$$\begin{aligned} \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} S[i,j] \\ s.t., \quad & |\{k|\mathcal{E}_k \le \epsilon, 1 \le k \le n\}| \ge n \cdot p \\ where \quad & \mathcal{E}_k = error(\hat{F}[:,k], F[:,k]) \\ & \hat{F} = \mathcal{R}(C),\ C = F \circ S \end{aligned}$$

If we know F in advance, it is possible for us to get the optimal S by enumerating all the possibilities (given sufficient computation time). However, in real life, we cannot foreknow F, which makes the problem challenging: (1) $\mathcal{E}_k$ cannot be directly obtained, and (2) the cell selection process is monotonic (i.e., only after we set $S[i,j] = 1$ can we get $F[i,j]$, and this selection cannot be retracted to save incentive costs). To overcome these difficulties, we design a Sparse MCS task allocation mechanism, CCS-TA, which leverages an iterative process to select cells for sensing in each cycle, with details elaborated in the following sections.

5.3 Overview of CCS-TA

In this section, we introduce the design of CCS-TA. Figure 5.2 shows the workflow of CCS-TA. In each cycle, CCS-TA iteratively *selects the next salient cell for sensing* and

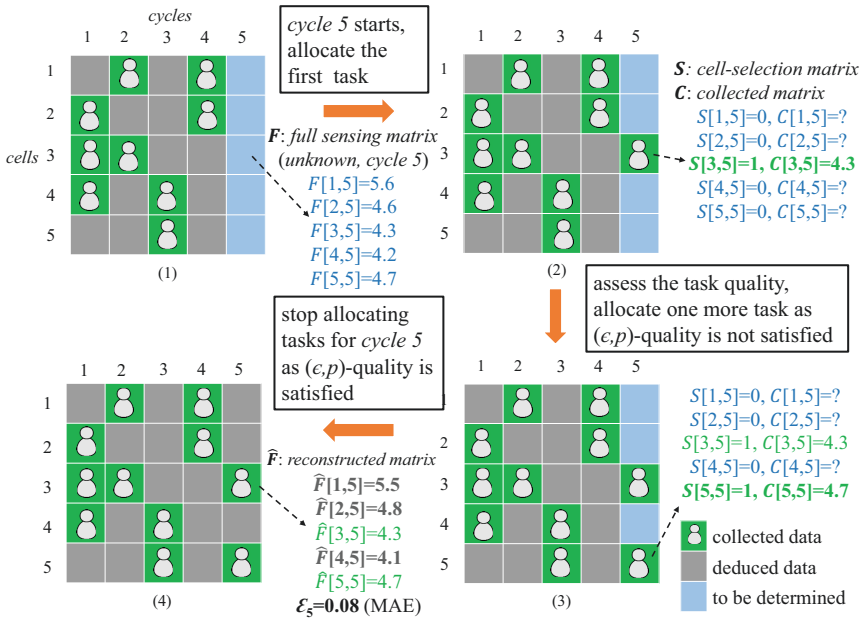


Figure 5.3: An example of CCS-TA process (5 cells and 5 cycles, temperature).

waits for allocating a sensing task to a participant present in that cell, until the *estimated data quality* satisfies the predefined (ϵ, p)-*quality* requirement. Then, the task allocation stops and *missing data values of the unsensed cells are inferred*.

Figure 5.3 shows an example to illustrate the task allocation process of CCS-TA in one sensing cycle. Suppose the target sensing area contains five cells and the fifth sensing cycle starts currently; in the beginning, no sensing data is collected in *cycle 5* (Figure 5.3-1, ground truth F is unknown). CCS-TA works as follows:

1) CCS-TA selects the first salient cell (*cell 3*, $S[3,5]$=1) and allocates a sensing task to one participant appearing in *cell 3*. This participant performs the sensing task and returns the sensing data to CCS-TA (Figure 5.3-2, $C[3,5]$=$F[3,5]$=4.3°C).

2) After CCS-TA gets the sensing data of *cell 3*, it assesses whether the data quality satisfies the predefined (ϵ, p)-quality. Assume the assessment result is *no*, CCS-TA continues selecting the next salient cell (*cell 5*, $S[5,5]$=1) to allocate another sensing task (Figure 5.3-3, $C[5,5]$=$F[5,5]$=4.7°C).

3) After collecting the sensing data from *cells 3 and 5* in *cycle 5*, CCS-TA assesses whether the data quality satisfies the predefined (ϵ, p)-quality again. If the obtained result is *yes*, CCS-TA stops further task allocations for *cycle 5* and infers the missing data of the unsensed cells (Figure 5.3-4, $\hat{F}[1,5]$, $\hat{F}[2,5]$, and $\hat{F}[4,5]$ are inferred).

5.4 Detailed Design of CCS-TA

In this section, we describe the algorithms used in CCS-TA: *inferring missing values*, *determining task allocation stopping criterion*, and *selecting salient cell for sensing*.

5.4.1 Inferring Missing Values

To reconstruct the full sensing matrix from the partically collected sensing values, *Compressive Sensing* (*CS*) is commonly used in the literature [131, 15]. In this section, we first introduce the basic idea of CS, and then illustrate an enhanced version of CS, called *Spatio-Temporal Compressive Sensing* (*STCS*), which considers the spatial and temporal correlations among the environmental data explicitly to further improve the reconstruction performance [131, 138]. We use STCS to infer missing values in CCS-TA, given its improved reconstruction accuracy over normal CS and the other methods [131].

CS: Compressive Sensing

Given a partially collected sensing matrix C, *compressive sensing* reconstructs the full sensing matrix $\hat{F}$ based on the low-rank property:

$$\begin{aligned} &\min\ rank(\hat{F}) \\ &s.t.,\ \hat{F} \circ S = C \end{aligned} \tag{5.3}$$

Directly solving this problem is hard because it is nonconvex. Based on the *singular value decomposition*, i.e., $\hat{F} = LR^T$, and compressive sensing theory [139, 140, 141], a more practical optimization problem is formulated [131, 138, 15], which changes rank minimization to minimizing the sum of L and R's Frobenius norms:

$$\min\ \lambda(\|L\|_F^2 + \|R\|_F^2) + \|LR^T \circ S - C\|_F^2 \tag{5.4}$$

where λ is used to make the trade-off between rank minimization and accuracy fitness. To get the optimal $\hat{F}$, we use an alternating least squares [131, 138, 15] procedure to estimate L and R iteratively ($\hat{F} = LR^T$).

STCS: Spatio-Temporal Compressive Sensing

As environment data such as temperature usually exhibits strong spatial and temporal correlations, explicit spatio-temporal correlations are introduced into compressive sensing in recent work [131, 138], called *spatio-temporal compressive sensing*, which focuses on the optimization function below:

$$\begin{aligned} \min\ &\lambda_r(\|L\|_F^2 + \|R\|_F^2) + \|LR^T \circ S - C\|_F^2 \\ &+ \lambda_s\|\mathbb{S}(LR^T)\|_F^2 + \lambda_t\|(LR^T)\mathbb{T}^T\|_F^2 \end{aligned} \tag{5.5}$$

where $\mathbb{S}$ and $\mathbb{T}$ are spatial and temporal constraint matrices respectively; λ_r, λ_s, and λ_t are chosen to balance the weights of different elements in the optimization problem.

Similar to the *CS* optimization problem (5.4), the above *STCS* optimization problem (5.5) could be solved by using alternating least squares [131, 138]. We elaborate below our strategies of choosing the temporal and spatial constraint matrices.

Temporal constraint matrix ($\mathbb{T}$): Like [131, 138], we choose the temporal constraint matrix $\mathbb{T}$ as $Toeplitz(0, 1, -1)_{n*n}$ (total n sensing cycles), which considers the temporal cor-

relation in the following manner — for a specific cell, its sensing values in two continuous sensing cycles should be similar.

Spatial constraint matrix ($\mathbb{S}$): The spatial constraint matrix $\mathbb{S}$ is used to express the correlations between the sensing data from different cells. Generally, if two cells are close to each other, their sensing data might be similar. In [131], due to the lack of the GPS information, the authors have to use sensor network topology to construct $\mathbb{S}$. Here, since we can get the GPS positions of the cells, we directly use the distance to model the correlation between cell i and j, $c_{i,j}$, as $1/distance(cell_i, cell_j)$. Then, we get $\mathbb{S}$ as follows:

$$\mathbb{S}_{i,i} = -1;\ \mathbb{S}_{i,j} = c_{i,j} / \sum_{k \neq i} c_{i,k},\ \text{if } i \neq j$$

5.4.2 Data Quality Assessment

In CCS-TA, for each sensing cycle, the data quailty assessment step, which decides when to stop task allocation, is a key issue: if we stop too early, the server might not collect enough data to achieve the predefined (ϵ, p)-quality for the MCS task; if we stop too late, then the server might collect redundant data, which would lead to waste in the organizer's budget. A well-designed stopping criterion should guarantee the data quality to satisfy the predefined (ϵ, p)-quality requirement, while allocating sensing tasks to as few cells as possible in each sensing cycle.

To this end, we propose a *Leave-One-Out Bayesian-Inference* (*LOO-BI*) based method to decide the stopping criterion for each sensing cycle, as shown in Algorithm 2. First, LOO-BI uses *leave-one-out* re-sampling method to obtain a set of re-inferred sensing data with the corresponding true collected data. Then, comparing the re-inferred data to the true collected data, *Bayesian inference* is leveraged to assess whether the current data quality can satsify the predefined (ϵ, p)-quality requirement or not.

Leave-One-Out Re-Sampling

In statistics, *leave-one-out* is a popular re-sampling method to measure the performance for many prediction and classification algorithms [97]. Suppose we have m true observations, the basic idea of leave-one-out is for each time, we leave one observation out and using the other $m-1$ observations (as training data set) to make a prediction for the excluded observation. By running this process on all m observations, we get m predictions accompanying with the m true observations, which can be used to estimate the prediction error.

We adopt the basic idea of leave-one-out in the LOO-BI (Algorithm 2), where we actually run leave-one-out on the collected data of the current cycle k (line 3-11). In each iteration, LOO-BI attempts to temporarily remove one piece of collected data of the current cycle k and then run the reconstruction algorithm $\mathcal{R}$ to re-infer the removed data (line 6-9). After enumerating all the collected data in cycle k, we finally get two vectors $\mathbf{x}$ and $\mathbf{y}$: $\mathbf{x}$ stores the true collected data for the current cycle k, while $\mathbf{y}$ stores the corresponding re-

Algorithm 2 LOO-BI task allocation stopping criterion

Input:

C_{m*k}: collected sensing matrix with m locations and k cycles, non-collected entry is *null* (current cycle is k).
$\mathcal{R}()$: a sensing matrix reconstruction algorithm (e.g., *STCS*).
error: an error metric (e.g., *mean absolute error*).
ϵ, p: predefined (ϵ, p)-quality requirement.

Output:

stop: *true*/*false* – stop/continue task allocation.

```
1:  x ← [ ]                                          ▷ vector storing collected data
2:  y ← [ ]                                          ▷ vector storing re-inferred data
3:  for i ← 1 to m do
4:      if C[i,k] ≠ null then                        ▷ if data is collected in C[i,k]
5:          x.append(C[i,k])
6:          C' ← C
7:          C'[i,k] ← null                           ▷ remove one collected data
8:          F̂' ← R(C')                               ▷ re-infer the removed data
9:          y.append(F̂'[i,k])
10:     end if
11: end for
12: P(E_k ≤ ε) ← BayesianInference(error, x, y, ε)
13: if P(E_k ≤ ε) ≥ p then
14:     stop ← true
15: else
16:     stop ← false
17: end if
18: return stop
```

inferred data by using leave-one-out. Suppose we have already collected data from m' cells for the current cycle, then both $\mathbf{x}$ and $\mathbf{y}$ have m' elements:

$$\mathbf{x} = \langle x_1, x_2, \cdots, x_{m'} \rangle, \quad \mathbf{y} = \langle y_1, y_2, \cdots, y_{m'} \rangle$$

where x_i is the i *th* ground truth data collected in cycle k, and y_i is the corresponding re-inferred data by leaving x_i out of the collected data.

Based on the ground truth set $\mathbf{x}$ and the leave-one-out re-inferred set $\mathbf{y}$, in the next section, we will describe how to assess whether the task could satisfy (ϵ, p)-quality or not.

Assessing Task Quality by Bayesian Inference

According to the *law of large numbers* in probability theory [142], ensuring a task satisfies (ϵ, p)-quality can be achieved by making sure that the probability of the error of each sensing cycle being at most ϵ be at least p, formally:

$$\forall k, 1 \leq k \leq n : P(\mathcal{E}_k \leq \epsilon) \geq p \tag{5.6}$$

Thus, the problem of assessing whether a task can satisfy (ϵ, p)-quality is converted to calculate $P(\mathcal{E}_k \leq \epsilon)$. To this end, we need to estimate the probability distribution for the cycle k's overall sensing error $\mathcal{E}_k$, which can be done with Bayesian inference [143].

In Bayesian inference, we see $\mathcal{E}_k$ as an unknown parameter with a *prior* probability distribution $g(\mathcal{E}_k)$;[2] based on our observation $\boldsymbol{\theta}$ (obtained from the leave-one-out re-inferred data, will be explained later), we update the probability distribution of $\mathcal{E}_k$, getting the *posterior* probability distribution $g(\mathcal{E}_k|\boldsymbol{\theta})$ according to the *Bayes' Theorem*:

$$g(\mathcal{E}_k|\boldsymbol{\theta}) = \frac{f(\boldsymbol{\theta}|\mathcal{E}_k)g(\mathcal{E}_k)}{\int_{-\infty}^{\infty} f(\boldsymbol{\theta}|\mathcal{E}_k)g(\mathcal{E}_k)d\mathcal{E}_k} \tag{5.7}$$

where $f(\boldsymbol{\theta}|\mathcal{E}_k)$ is the *likelihood* function that represents the conditional probability of observing $\boldsymbol{\theta}$ given $\mathcal{E}_k$.

The posterior $g(\mathcal{E}_k|\boldsymbol{\theta})$ is thus the estimated probability distribution of $\mathcal{E}_k$, based on which we can approximate $P(\mathcal{E}_k \leq \epsilon)$:

$$P(\mathcal{E}_k \leq \epsilon) \approx \int_{-\infty}^{\epsilon} g(\mathcal{E}_k|\boldsymbol{\theta})d\mathcal{E}_k \tag{5.8}$$

If $P(\mathcal{E}_k \leq \epsilon) \geq p$, then CCS-TA stops the task allocation for current cycle k and waits for the start of the next cycle; otherwise, CCS-TA continues selecting a new cell to collect sensing data (see Algorithm 2, line 12-18).

Note that for different error metrics, the calculation processes for the posterior $g(\mathcal{E}_k|\boldsymbol{\theta})$ are different (as the likelihood functions $f(\boldsymbol{\theta}|\mathcal{E}_k)$ are usually different). In the next two sub-sections, we introduce how to compute the posterior $g(\mathcal{E}_k|\boldsymbol{\theta})$ for two-widely used error metrics, *mean absolute error* (for continuous value, e.g., temperature [135]) and *classification error* (for classification label, e.g., AQI descriptor [134]).

Bayesian Inference for Mean Absolute Error

When $\mathcal{E}_k$ is defined as mean absolute error (MAE, Eq. (5.1)), we use the absolute difference of $\mathbf{y}$ (leave-one-out re-inferred data set) and $\mathbf{x}$ (true collected data set) as the observation $\boldsymbol{\theta}$ (suppose m' sensed data has already been collected in the current cycle):

$$\begin{aligned}\boldsymbol{\theta} &= abs(\mathbf{y} - \mathbf{x}) \\ &= \langle |y_1 - x_1|, |y_2 - x_2|, \ldots, |y_{m'} - x_{m'}| \rangle\end{aligned}$$

After inspecting our evaluation temperature dataset (which will be described in detail later), we find that the MAE in each sensing cycle follows the normal distribution. Figure 5.4 shows the histogram of the standardized MAE (i.e., MAE divided by the standard deviation of MAE in each cycle) when we keep the data of 10% of the cells and infer that of the remaining 90%. Thus, by supposing that the sampled absolute errors satisfy the normal distribution around mean $\mathcal{E}_k$ and variance σ^2, we get the likelihood function (the probability of observing $\boldsymbol{\theta}$ given a specific $\mathcal{E}_k$):

$$f(\boldsymbol{\theta}|\mathcal{E}_k) : \ \theta_i = |y_i - x_i| \sim \mathcal{N}(\mathcal{E}_k, \sigma^2)$$

[2]The prior distribution is often selected as a non-informative probability distribution (such as uniform distribution) if we do not have specific prior knowledge about $\mathcal{E}_k$.

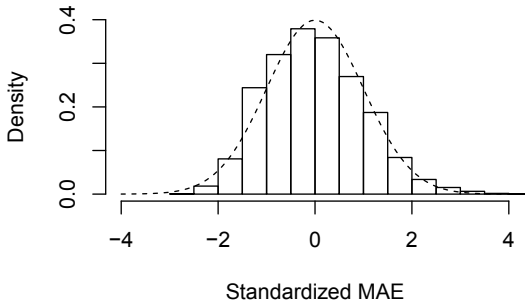


Figure 5.4: Histogram of mean absolute error with fitted normal curve.

Given the above, calculating the posterior $g(\mathcal{E}_k|\boldsymbol{\theta})$ from the above likelihood function and observation is a classic Bayesian statistics problem: *inferring normal mean with unknown variance*, which can be solved by fixing the variance σ^2 to the sample variance s^2 and then directly calculating the posterior $g(\mathcal{E}_k|\boldsymbol{\theta})$ by t-distribution [144]. For the prior $g(\mathcal{E}_k)$, we select the *Jeffreys' flat prior* [145]: $g(\mathcal{E}_k) = 1, \ \forall \mathcal{E}_k$. Then, the posterior $g(\mathcal{E}_k|\boldsymbol{\theta})$ satisfies the following (m'-1) degree-of-freedom t-distribution:

$$g(\mathcal{E}_k|\boldsymbol{\theta}) \sim t_{m'-1}(\bar{\theta}, s^2) \tag{5.9}$$

where $\bar{\theta}$ is the sample mean of the values in $\boldsymbol{\theta}$. Based on the posterior $g(\mathcal{E}_k|\boldsymbol{\theta})$ obtained from Eq. (5.9), we can then use Eq. (5.8) to calculate the $P(\mathcal{E}_k \leq \epsilon)$ and decide whether more cells should be selected in the current cycle k.

Bayesian Inference for Classification Error

For classification problems, the classification error (Eq. (5.2)) measures the percentage of the test data that is classified into a wrong label. We now show how we use Bayesian inference to estimate the posterior distribution for the classification error $\mathcal{E}_k$. First, as our reconstruction algorithm $\mathcal{R}$ deals with continuous values, we map $\mathbf{x}$ and $\mathbf{y}$ to their corresponding classification labels using the mapping function $\psi()$ in Eq. (5.2), e.g., for the PM2.5 AQI value between 0 and 50, we map it into the AQI descriptor label "Good". Afterward, we use the $I()$ function on $\psi(\mathbf{x})$ and $\psi(\mathbf{y})$ to get our observation $\boldsymbol{\theta}$:

$$\begin{aligned}\boldsymbol{\theta} &= I(\psi(\mathbf{x}), \psi(\mathbf{y})) \\ &= \langle I(\psi(x_1), \psi(y_1)), I(\psi(x_2), \psi(y_2)), \ldots, I(\psi(x_{m'}), \psi(y_{m'}))\rangle\end{aligned}$$

Each θ_i is either 1 (success, $\psi(x_i) = \psi(y_i)$) or 0 (failure, $\psi(x_i) \neq \psi(y_i)$), and the classification error $\mathcal{E}_k$ is exactly the failure ratio. Suppose each θ_i is independent, then it satisfies the *Bernoulli* distribution with the probability of $1 - \mathcal{E}_k$:

$$f(\boldsymbol{\theta}|\mathcal{E}_k): \ \theta_i = I(\psi(x_i), \psi(y_i)) \sim Bernoulli(1 - \mathcal{E}_k)$$

Based on this likelihood function $f(\boldsymbol{\theta}|\mathcal{E}_k)$, the problem to infer the posterior $g(\mathcal{E}_k|\boldsymbol{\theta})$ is converted to a classic Bayesian statistics problem, *Coin Flipping* [144, 143]. We choose the

uniform prior for $\mathcal{E}_k$: $g(\mathcal{E}_k) = 1 \quad \text{for} \quad 0 \leq \mathcal{E}_k \leq 1$. Then the posterior for $\mathcal{E}_k$ follows *Beta* distribution [144, 143]:

$$g(\mathcal{E}_k|\boldsymbol{\theta}) \sim Beta(m' - z + 1, z + 1) \tag{5.10}$$

where $z = \sum_{i=1}^{m'} \theta_i$, i.e., the number of successes. Then, for the classification error $\mathcal{E}_k$, combing Eq. (5.10) and Eq. (5.8), we can calculate the $P(\mathcal{E}_k \leq \epsilon)$ to decide whether we need to continue the task allocation.

Computation Complexity of LOO-BI

As there are two phases for the computation of LOO-BI, we discuss the computation complexity of both phases respectively. First, to use leave-one-out to estimate the sensing error, LOO-BI needs to run the reconstruction algorithm $\mathcal{R}$ for m' times, where m' is the number of the already collected sensing values in the current cycle. This time consumption dominates the running time so that the computation complexity is $O(m' \cdot T_{\mathcal{R}})$ for the leave-one-out part, where $T_{\mathcal{R}}$ is the complexity of the reconstruction algorithm $\mathcal{R}$. For the second part, Bayesian inference, recalling Eq. (5.9) and Eq. (5.10), we can simply use two distributions, *t*-distribution and *Beta* distribution respectively, to calculate the posterior for mean absolution error and classification error, which makes the computation process of Bayesian inference run much faster than the leave-one-out part. In summary, the computation complexity of LOO-BI is dominated by the leave-one-out part, which is $O(m' \cdot T_{\mathcal{R}})$. If m' is large, sequentially executing LOO-BI might consume much time. Fortunately, though we need to run $\mathcal{R}$ for m' times, each run is independent, so LOO-BI can be easily parallelized to accelerate as needed.

5.4.3 Selecting Salient Cell for Sensing

When the output of the stopping criterion LOO-BI is *false*, CCS-TA will continue selecting more cells for sensing. During this process, selecting some salient cells for sensing may reduce the overall sensing error more significantly, e.g., the missing values of some cells might incur more inference errors and are thus more uncertain. If CCS-TA can identify these salient cells, the number of the allocated tasks can possibly be reduced to make the data quality satisfy the predefined (ϵ, p)-quality requirement earlier, compared to other simple cell selection methods such as random selection.

Based on the recent research advances in *active learning* on matrix completion, we use a method proposed in [146], called *Query by Committee (QBC)*, to select the salient cell to allocate the next task (*committee* here refers to a set of various matrix reconstruction algorithms). QBC attempts to use each algorithm in the committee to reconstruct the full sensing matrix. Then it allocates the next task to the cell with the largest variance among the inferred values of different algorithms [146].

In CCS-TA, the committee includes *CS*, *STCS*, *KNN-S*, and *KNN-T*. *CS* and *STCS* are described previously, and *KNN-S* and *KNN-T* use the classic *K-Nearest Neighbors*

Algorithm 3 QBC: Selecting salient cells using Query By Committee

Input:

C_{m*k}: collected sensing matrix with m cells and k sensing cycles, non-collected entry is *null*. Current sensing cycle is *cycle k*.

$C_{\mathcal{R}}$: a set of sensing matrix recovery algorithms (called *committee*, e.g. *CS, STCS, KNN-S, and KNN-T*).

Output:

s: salient cell for next task allocation in current cycle.

1: **for** each $\mathcal{R}_i \in C_{\mathcal{R}}$ **do** ▷ for each algorithm $\mathcal{R}_i$ in the committee
2: $\hat{F}_i \leftarrow \mathcal{R}_i(C)$ ▷ $\hat{F}_i$ is the reconstructed matrix via algorithm $\mathcal{R}_i$
3: **end for**
4: $s \leftarrow 0$
5: $v_{max} \leftarrow 0$ ▷ initialize the max variance
6: **for** $j \leftarrow 1 \sim m$ **do**
7: **if** $C[j,k] = null$ **then** ▷ for each unsensed cell j in cycle k
8: $v_j \leftarrow variance(\hat{F}_1[j,k], \hat{F}_2[j,k], ..., \hat{F}_k[j,k])$
9: ▷ $\hat{F}_i[j,k]$ is the inferred value of cell j in cycle k via algorithm $\mathcal{R}_i$
10: **if** $v_j > v_{max}$ **then** ▷ select the cell whose inferred value variance is the largest
11: $s \leftarrow j$
12: $v_{max} \leftarrow v_j$
13: **end if**
14: **end if**
15: **end for**
16: **return** s

(KNN) [147] method to interpolate missing values. For a missing value, KNN uses a weighted average of the values of the k nearest neighbors. In sensing matrix reconstruction, we can perform KNN on columns or rows, i.e., using spatial (KNN-S) or temporal (KNN-T) K nearest neighbors. Specifically, for a missing value $F[i, j]$ (cell i in cycle j), KNN-S attempts to find K nearest spatial neighbors $F[i', j]$ (weight $\propto 1/distance(cell_i, cell_{i'})$), while KNN-T attempts to find K nearest temporal neighbors $F[i, j']$ (weight $\propto 1/|j - j'|$).

Computation Complexity of QBC

The running time of the QBC method is primarily spent on using all the algorithms in the committee to reconstruct the sensing matrix. Suppose for each reconstruction algorithm $\mathcal{R}_i$ in the committee, the computation complexity is $T_{\mathcal{R}_i}$, then the complexity of QBC is $O(\sum_i T_{\mathcal{R}_i})$. If the committee contains more algorithms, then running QBC sequentially will cost more time. However, like LOO-BI, since the executions of different reconstruction algorithms are independent, QBC can also be parallelized to improve runtime performance.

5.5 Evaluation

5.5.1 Experiment Setup

To evaluate the real-world applicability of our work, we use two real-life sensing datasets, *temperature* (*TEMP*) and *PM 2.5 air quality* (*PM25*), to create two experimental scenarios.

Note that though the two datasets are collected by sensor networks or static stations, we assume the MCS participants can obtain them using smartphone applications [137, 13].

The *TEMP* dataset has been collected by the *SensorScope* project [133]. The project maintains nearly one hundred sensors across the EPFL campus (500m×300m) to collect various environment readings, e.g., temperature, solar radiation, and humidity. For our evaluation, we divide the target area into 100 cells (each cell is 50m×30m), and find that 57 cells are deployed with the temperature sensors. If a cell has more than one sensor, we select the sensor collecting the most number of the temperature readings. In summary, *TEMP* dataset includes the temperature readings in 57 cells from 2007-07-01 to 2007-07-07; each sensing cycle lasts for 30 minutes. To measure the data quality for temperature, referring to [135], we use the *mean absolute error* (Eq. (5.1)) as the metric.

PM25 dataset has been collected by the *U-Air* project [134], which includes PM2.5, PM10, and NO2 AQI (air quality index) values reported by 36 air quality monitoring stations in Beijing. For our evaluation, like [134], we also split the Beijing urban area to 1km×1km cells and only use the cells where the stations are situated. In summary, *PM25* dataset includes the PM2.5 AQI values on 36 station-situated cells from 2013-11-10 to 2013-11-20; each sensing cycle lasts for one hour. To measure the data quality for PM2.5 AQI, we follow the methods used in [134] — each AQI value is classified to a range called *descriptor*, which is used as the basis for computing the *classification error* discussed in Eq. (5.2). Six levels of descriptors are defined: *Good* (0∼50), *Moderate* (51∼100), *Unhealthy for Sensitive Groups* (101∼150), *Unhealthy* (150∼200), *Very Unhealthy* (201∼300), and *Hazardous* (301∼500).

5.5.2 Performance Analysis: Inferring Missing Values

First, we aim to verify the effectiveness of *STCS* in inferring missing values for temperature and PM 2.5 compared to the other state-of-the-art matrix reconstruction algorithms described before, including CS, KNN-S, and KNN-T. Note that for STCS and CS, optimization parameters are selected as in [131]; for KNN-S and KNN-T, we set K to 3 because by comparing different values of K, we find that K = 3 achieves the best performance for KNN-S and KNN-T.

Figure 5.5 and 5.6 show the overall sensing error (with standard deviation) of different reconstruction algorithms on TEMP and PM25 datasets, respectively. In this experiment, we iteratively consider each sensing cycle k as the latest cycle, reconstruct the full sensing matrix based on the collected sensing matrix from cycle *1* to k, and calculate the overall sensing error for the cycle k. The x axis, i.e., *sparsity ratio*, denotes the fraction of unsensed entries in the collected sensing matrix. Consistent with the literature [131], our evaluation results also show the improved accuracy of STCS over the other methods, verifying that compressive sensing is effective in inferring the missing environmental data such as temperature and air quality, especially when the explicit spatio-temporal correlations are incorporated. Therefore, for both TEMP and PM25 datasets, we use STCS to infer the missing values.

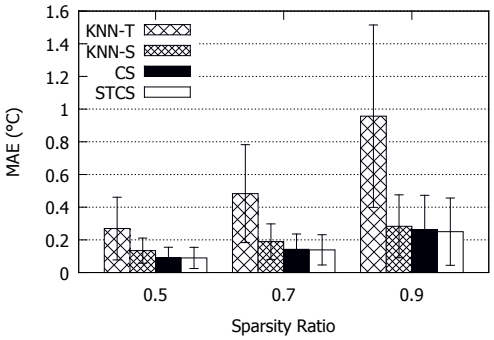


Figure 5.5: Error of different reconstruction algorithms (TEMP).

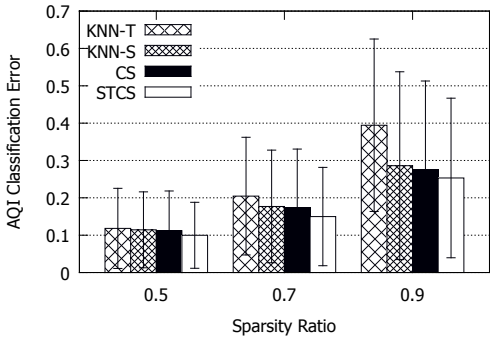


Figure 5.6: Error of different reconstruction algorithms (PM25).

	TEMP		*PM25*	
p \ ϵ	0.25°C	0.30°C	6/36	9/36
0.90	0.915	0.919	0.904	0.912
0.95	0.943	0.949	0.944	0.965

Table 5.1: Fraction of the cycles whose errors are lower than the error bound ϵ.

5.5.3 Performance Analysis: Stopping Criterion

Then, we evaluate the effectiveness of the stopping criterion *LOO-BI*. We use various settings of (ϵ, p)-quality to see what fraction of sensing cycles can actually keep the overall sensing error less than ϵ. Table 5.1 shows the results for both TEMP and PM25 datasets. For p, we purposely set it to a large value as 0.90 and 0.95, i.e., ensuring *most* (90% or 95%) sensing cycles' error to be less than the predefined error bound ϵ, which we think is a more reasonable and realistic scenario than small p for MCS organizers. For ϵ, we vary it from 0.25°C to 0.30°C for TEMP and 6/36 to 9/36 for PM25. Note that for PM25, the error bound $X/36$ represents that to satisfy this error bound, more than $36-X$ cells have the correct AQI level.

From Table 5.1, we see that for any predefined error bound ϵ, the actual fraction of the cycles whose errors are less than ϵ is quite similar to the p in the predefined (ϵ, p)-quality. Specifically, for $p = 0.90$, all the actual fractions are larger than 0.90; for $p = 0.95$, even though the the actual fractions sometimes are slightly less than 0.95, the values are still quite near 0.95 (in our settings, the smallest actual fraction is 0.943, only 0.007 smaller than 0.95). Based on these results, we verify that, by using LOO-BI as the stopping criterion, CCS-TA can well satisfy the predefined (ϵ, p)-quality.

5.5.4 Performance Analysis: Number of Allocated Tasks

After selecting the best sensing data reconstruction algorithm and verifying the effectiveness of the stopping criterion, now we focus on analyzing the research objective — *how many allocated tasks could CCS-TA reduce while ensuring a certain data quality?*

TEMP	*CCS-TA*	*RAND-TA*	*ICDM-FIX-k*
$p = 0.9$	0.915	0.900	0.911 ($k = 9$)
$p = 0.95$	0.943	0.943	0.949 ($k = 12$)

Table 5.2: Fraction of the cycles whose errors are lower than 0.25°C (TEMP).

We first compare CCS-TA with baseline approaches to see the advantage of CCS-TA. Afterward, we compare CCS-TA with the *OPTIMAL* solution, i.e., if the sensing data of all the cells for each cycle is foreknown, what is the minimal number of allocated tasks that we can achieve under a predefined error bound ϵ. Although the OPTIMAL solution is unrealistic from a performance perspective, we want to show the performance gap between CCS-TA and the ideal condition to identify the potential improvement space for CCS-TA.

CCS-TA vs. Baselines

To compare with CCS-TA, we use the following baselines:

- **ICDM-FIX-*k***: An alternative way to extend the QBC active learning method [146] from ICDM'13 to the MCS task allocation mechanism is fixing the task number k in each sensing cycle, while still using QBC to actively select cells to allocate tasks; we call this modified algorithm ICDM-FIX-k. Compared to ICDM-FIX-k, CCS-TA shows the benefit brought by LOO-BI, which helps the organizer decide when to stop the task allocation, thus adaptively adjusting the number of the sensed cells for different cycles.
- **RAND-TA**: In this baseline, we *randomly* select the next cell for sensing, but still leverage LOO-BI as the task allocation stopping criterion. Compared to RAND-TA, CCS-TA shows the advantage of applying QBC to select the salient cells for sensing.

On the TEMP dataset, for the predefined (ϵ, p)-quality, we set the error bound ϵ as 0.25°C and p as 0.9 or 0.95. Before comparing the number of allocated tasks, we need to ensure that all the methods can achieve the similar task quality. While CCS-TA has already been verified to be able to satisfy (ϵ, p)-quality in the previous section, now we need to check the two baselines. Table 5.2 shows the results. We can see that RAND-TA can also satisfy (ϵ, p)-quality well, as it adopts LOO-BI as the stopping criterion like CCS-TA. For ICDM-FIX-k, we tune k to achieve the similar task quality, which leads to $k = 9$ for $p = 0.9$ and $k = 12$ for $p = 0.95$.

As all the methods can satisfy similar task quality, we compare their numbers of the allocated tasks (i.e., number of selected cells) in Figure 5.7. When $p = 0.9$, CCS-TA can allocate 11.1% fewer tasks than RAND-TA, and 18.0% fewer tasks than ICDM-FIX-9; when $p = 0.95$, CCS-TA outperforms RAND-TA and ICDM-FIX-12 by assigning 18.0% and 26.5% fewer tasks, respectively. Specifically, CCS-TA allocates tasks to only 12.9% (15.5%) cells on average, while ensuring the overall sensing error below 0.25°C in 90% (95%) of the cycles.

On the PM25 dataset, we get similar observations. See Table 5.3 and Figure 5.8 for the results when we set ϵ to 9/36 (i.e., 0.25) and p to 0.90 or 0.95. In general, while achieving

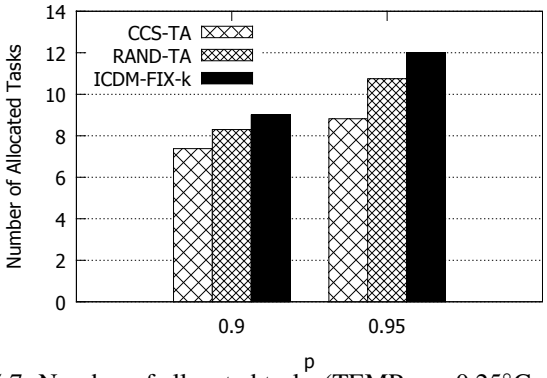


Figure 5.7: Number of allocated tasks (TEMP, $\epsilon = 0.25$°C, varying p).

PM25	*CCS-TA*	*RAND-TA*	*ICDM-FIX-k*
$p = 0.9$	0.912	0.916	0.884 ($k = 16$)
$p = 0.95$	0.965	0.969	0.961 ($k = 18$)

Table 5.3: Fraction of the cycles whose errors are lower than 9/36 (PM25).

similar data quality, CCS-TA allocates 7.5-9.9% and 25.0-31.9% fewer tasks than RAND-TA and ICDM-FIX-k, respectively.

CCS-TA vs. OPTIMAL

Now we compare CCS-TA with the *OPTIMAL* solution. The OPTIMAL solution assumes the sensing data of all the cells for each cycle is foreknown, so it can choose the optimal (i.e., minimal-size) cell combination to satisfy the error bound ϵ. OPTIMAL is not practical in real life, but it can serve as the bound for CCS-TA. To get the OPTIMAL solution, we enumerate all the possible cell combinations from size 1 to m until we find the smallest-size cell combination that can meet the error bound. The enumeration process is highly time-consuming, so we trim the TEMP dataset to a small number of cells, i.e., 15 cells, to conduct this experiment.

Suppose the error bound ϵ is 0.25°C, for CCS-TA, we set $p = 0.95$ and it actually guarantees the overall sensing error below 0.25°C in 95.4% of the cycles, and the average task number in each cycle is 4.74. In contrast, OPTIMAL allocates only 2.23 tasks per cycle (53.0% fewer tasks than CCS-TA) while ensuring the error below 0.25°C in 100% of the cycles (4.6% more cycles than CCS-TA). Therefore, *a noticeable gap still exists between CCS-TA and OPTIMAL*, which inspires us to to improve CCS-TA to narrow this gap in the future work.

5.5.5 Running Time Analysis

Finally, we study the running time of CCS-TA to see whether it can satisfy the real-life MCS scenario, as well as the speedup it provides vis-a-vis the OPTIMAL solution. We run the experiments on a laptop (Intel core i7-3612QM, 8GB RAM, Windows 7) with Python 2.7. Table 5.4 shows the running time for different parts of CCS-TA. The most

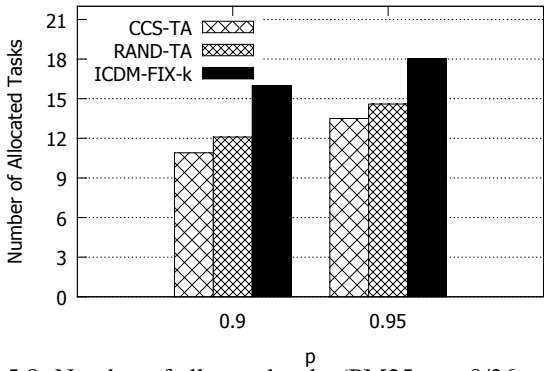


Figure 5.8: Number of allocated tasks (PM25, $\epsilon = 9/36$, varying p).

	Reconstruction (STCS)	*Stopping Criterion (LOO-BI)*	*Cell Selection (QBC)*
TEMP	0.95s	<9s	1.04s
PM25	0.75s	<7s	0.91s

Table 5.4: Running time for each sub-step of CCS-TA.

time-consuming part is the LOO-BI stopping criterion, which needs 9 seconds at most. As described previously, LOO-BI is suitable for being parallelized, which can help improve its performance. In summary, on our experimental setup, CCS-TA spends ∼10 seconds to allocate one task, i.e., estimating the task quality once and, if it cannot meet the predefined (ϵ, p)-quality, finds the next sensing cell. Thus, for the sensing tasks requiring a few seconds, such as temperature sensing, if we can find a participant and receive her sensing data in 10 seconds, CCS-TA can allocate tasks to ∼180 cells in an hour; even for the air quality sensing that needs 60 seconds to get a valid reading [137, 13], CCS-TA can allocate tasks to ∼50 cells in an hour. We believe this efficiency can meet most real-life MCS scenarios, especially with more powerful servers in CCS-TA deployment environment and more efficient smartphone-equipped sensors in the future. As a comparison, the OPTIMAL solution, even in the trimmed TEMP dataset with 15 cells, needs >30 minutes to find an optimal combination containing only 6 cells.

In summary, we have shown that each constituent of the our approach performs better than the baseline. We have shown (in Figures 5.5 and 5.6) how STCS is the best choice for reconstruction, and comparison with ICDM-FIX-k shows how LOO-BI as the stopping criterion is better. Since CCS-TA and RAND-TA share the reconstruction and stopping criterion techniques, differing only in the cell-selection process, they are close in performance, although our approach is still marginally better.

5.6 Concluding Remarks

In this chapter, we attempt to reduce the number of the required sensing cells and thus the number of the allocated tasks to participants in MCS applications by considering the tem-

poral and spatial correlations among the sensing data from different cells. To that end, we propose a novel crowdsensing paradigm, *Sparse Mobile Crowdsensing*, and further design a Sparse MCS task allocation framework called *CCS-TA*, combining the state-of-the-art compressive sensing, Bayesian inference, and active learning techniques to actively select a minimum number of sensing cells in each cycle while inferring the missing values of the remaining cells, and ensuring that the overall data accuracy meets a predefined bound. Evaluation results on real-world temperature and air quality monitoring datasets show the effectiveness and applicability of CCS-TA.

Chapter 6

DUM-ϵe: Differential Location Privacy Protection for Sparse Mobile Crowdsensing

Contents

6.1 Introduction

In Chapter 5, we have shown that Sparse MCS can be used to facilitate various MCS applications such as environment monitoring (e.g., noise [4]). In these applications, the participants report not only the sensed data, but also their corresponding location and time. This has serious privacy implications for MCS participants. Knowing each participant's location, an adversary, who wants to exploit information about the participant, can stage a broad spectrum of attacks, such as physical surveillance and stalking, identity theft, and breach of sensitive information [148]. Thus, ensuring location privacy is an essential aspect of MCS, because mobile users will not accept to engage in an MCS task if their privacy may be violated.

Location privacy has been widely addressed in the context of location-based systems [56, 149]. There are two general mechanisms to protect a user's location privacy: (i) protect the

user's identity through *anonymity*, so that the user's location traces cannot be linked to the individual user, and (ii) using location *obfuscation* to alter the user's actual locations in order to reduce the location information exposed to the service provider. In this work, we focus on the latter mechanism of privacy protection.

Much research has been conducted in location privacy protection with various obfuscation techniques [55, 56, 149]. These protection mechanisms are based on either hiding or perturbing the user's actual location to increase the uncertainty of the adversary's knowledge about the locations. The most popular mechanism is cloaking [150, 151, 152], where the user's location is represented as a cloaked region (containing multiple fine-grained cells) instead of a specific place or cell. Usually, the number of fine-grained cells in the cloaked region is leveraged to measure the privacy protection level. However, one common shortcoming of the cloaking mechanisms is that they are sensitive to the adversary's prior knowledge about the user's location distribution [62]. For example, if a user appears in a cloaked region including one school and one government office, and the adversary knows in advance that the user is likely to be at schools (e.g., the user is a student), then the adversary will have high confidence that the user would be at the school in the cloaked area, instead of the government office. This violates the intended protection effect of cloaking.

Differential privacy [60, 61] has been proposed to remedy this shortcoming of the obfuscation mechanisms regarding the adversary's prior knowledge. From the domain of statistical databases, its original goal is to protect an individual's data while publishing aggregate metrics (e.g., count and sum) from the database. Differential privacy specifies that modifying a single user's data will have an insignificant effect on the query outcome. This is typically done by adding a controlled amount of random noise to the query output. Suppose an adversary attempts to find out a user's certain attribute value (e.g., age) in a database, even if the adversary already has the query results using the aggregate of all other attribute values, he cannot gain significantly more knowledge about the single attribute value from the perturbed query output, regardless of what prior knowledge the adversary holds.

In applying differential privacy to location-based services (LBS), Andres et al. [62] proposed *geo-indistinguishability*, which gives a user l-privacy within a circular area R with certain radius. The probability to report the same obfuscated location r' from any two actual locations within R is similar (the level of similarity depends on l). Thus, after observing a user's obfuscated location r', the adversary gains little additional knowledge about which location within R produces r', regardless of the adversary's prior knowledge about the user's location distribution. To protect location privacy while maintaining the quality of service in LBS (such as Point-of-Interest queries), both the cloaking mechanisms and differential privacy mechanisms assume that the distance between the user's actual location and the obfuscated location is small. This assumption works well for LBS, as the quality for the output of a privacy-preserving location-based query is usually only degraded with the increase of the distance between the obfuscated and actual locations.

However, in Sparse MCS, the data quality loss is affected by the difference of sensed data between the actual location and obfuscated location, other than just the distance between the two locations. To limit the data quality loss for Sparse MCS, as long as the

sensing values of the two locations are close, the participant's location may be mapped to a location far from the actual one. For example, when sensing air quality in a city, two parks may have similar air quality values. If a participant is at a park, obfuscating her location to another park would incur little loss in data quality, even if the two parks are far from each other. Due to this distinction between the qualities of LBS and Sparse MCS, instead of directly using the mechanisms for LBS (e.g., [62]), we need to redesign the obfuscation mechanisms used in Sparse MCS to protect location privacy, while ensuring data quality.

With this insight in mind, in this chapter, we explore how to balance the location privacy for participants and the overall data quality obtained for Sparse MCS applications. We consider three key elements in the location-privacy preserving mechanism design: the *participant's privacy requirements*, the *adversary's prior knowledge about the participant's actual location distribution*, and the *data quality degradation* stemming from the obfuscation of actual locations. The main contributions are:

1. Bringing differential location privacy to Sparse MCS by introducing the privacy notion of ϵ*-region-ambiguity*, to restrict what an adversary may learn of participants regardless of his prior knowledge about participants' location distribution.
2. Addressing the two issues of location obfuscation and sensed data mapping simultaneously for data quality in Sparse MCS by modeling the data-quality-aware location privacy-preserving requirement as an optimization problem. Specifically, we propose a novel linear program called *DUM-*ϵe, which selects the optimal location obfuscation function and reduces data quality loss through *Data Uncertainty Minimization* under the constraints of ϵ*-region-ambiguity* and an *even distribution of obfuscated locations*. Hence the resulting optimal obfuscation function ensures differential location privacy with significantly reduced loss of data quality.
3. Reducing the number of constraints of *DUM-*ϵe from $O(n^3)$ to $O(n^2)$ by proposing an approximation linear program *Fast DUM-*ϵe (*FDUM-*ϵe). As the number of constraints affects both the time and space complexity of the linear programming solver techniques [153], *FDUM-*ϵe requires much less computation time and memory usage than *DUM-*ϵe. Therefore, it can be applied in large-scale MCS tasks that *DUM-*ϵe cannot handle.

To the best of our knowledge, this is the first work to apply differential location privacy to MCS while reducing the loss of data quality due to obfuscation. We evaluate our optimization mechanisms, *DUM-*ϵe and *FDUM-*ϵe, with experiments using real-world environment (temperature and humidity) and traffic monitoring datasets. Our results show that compared to three baseline approaches, *DUM-*ϵe can reduce the data quality loss for Sparse MCS tasks by 15-45%, with the same level of differential privacy guarantee. Compared to *DUM-*ϵe, *FDUM-*ϵe can achieve similar data quality (2-6% more quality loss), while only needing less than 1% computation time for generating the obfuscation function.

6.2 Preliminaries

6.2.1 Target Privacy Protection Scenarios

In this work, we focus on the MCS scenarios when the participants upload their sensed data and locations in a *sporadic* manner [154]. That is, in each sensing cycle, the participants will be re-selected; then with a large number of candidate participants, no single participant will report her locations in a *continuous* manner (e.g., continuous cycles). In a sporadic manner, an adversary may do a *snapshot* localization attack [154, 152], i.e., inferring a participant's current location via her currently reported (obfuscated) location. We aim to protect the users' location privacy against this snapshot localization attack. Protecting the users' location privacy in the *continuous* data uploading manner (against the adversary's *trajectory* attack) will be our future work (brief discussion in Section 6.7.2).

6.2.2 Inferring Missing Data in Sparse MCS

Recall that the data inference in Sparse MCS is modeled as a matrix completion problem: let *Collected Sensing Matrix* (C) be a matrix to record all the sensed data values collected from the participants, such that $C[r,t]$ represents the sensed data value of region r in sensing cycle t. If no participant uploads data from region r in cycle t, then $C[r,t]$ is unknown. The key to a successful Sparse MCS task is to determine a high quality, low uncertainty inference algorithm to infer such missing data. In this work, we still use the *Spatio-Temporal Compressive Sensing (STCS)* method (Section 5.4.1), which has been verified to outperform other methods in CCS-TA (Section 5.5.2).

As a corollary from compressive sensing theory, *Candes et al.* [155] have proven that *one can recover an unknown matrix of low rank, given a small number (compared to the size of the matrix) of noisy entries* ***uniformly sampled****, with* ***an error which is proportional to the noise level****.* Restated, applying compressive sensing to the problem of matrix completion makes two inherent assumptions:

1. *Even Data Distribution*. To ensure the data inference algorithm performs effectively, uniform distribution of the observed data is required. In Sparse MCS, this means that the sensed regions in the target sensing area should be evenly distributed. If the distribution is biased, e.g., one row of a matrix contains no observation (i.e., one region is not covered by participants in all sensing cycles), then it is impossible to infer the missing data for this row [155, 139].
2. *Small Data Uncertainty*. When there is no noise or uncertainty in the sampled entries, the missing values in the matrix can be accurately inferred as long as the previous assumption also holds. When the sampled entries have noise or uncertainty involved, the total inference error is proportional to the uncertainty level of the sampled entries [155]. The smaller the uncertainty of the sampled entries, the better the overall inference performance.

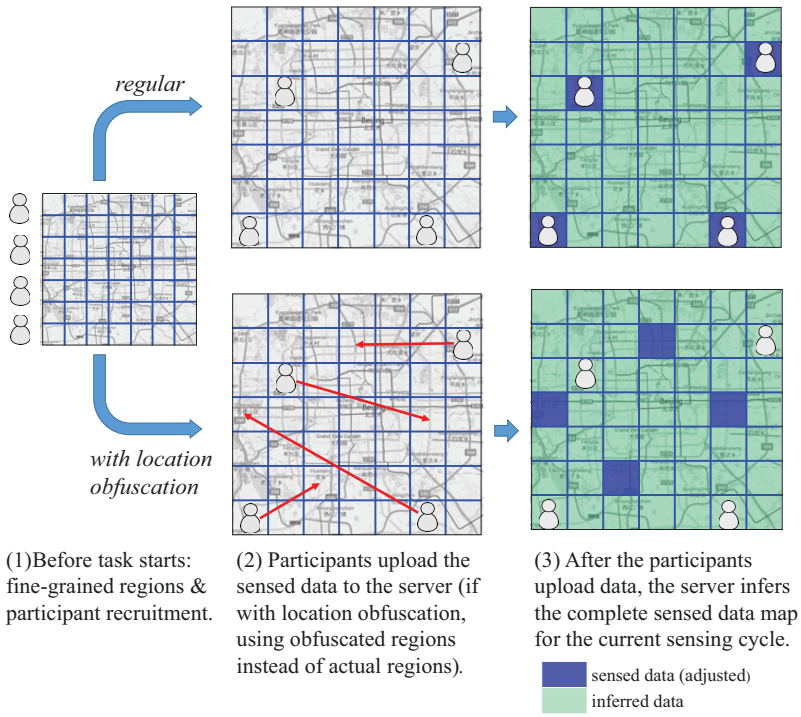


Figure 6.1: Regular data reporting for Sparse MCS (Top) and with location privacy protection using obfuscation (Bottom).

Therefore, these two assumptions in matrix completion require that the sensed regions in Sparse MCS should be evenly distributed and the uncertainty in the reported sensed data should be as small as possible.

6.3 Location Privacy-Preserving Framework

Regular Sparse MCS does not consider the participants' location privacy such that the participants report their actual regions. Using obfuscation to add location privacy protection can allay participants' privacy concerns, but will lead to data quality loss as the sensed data of the original actual region may not be representative of the obfuscated region. Therefore, we design a location privacy-preserving framework, which incorporates two unique components: *location obfuscation* and *data adjustment*.

Figure 6.1 (Bottom) illustrates the process of privacy-preserving Sparse MCS for the temperature monitoring use case compared to the regular case (Top). In each sensing cycle, a participant reports her obfuscated region to the server, so that the server never knows her actual region. However, as the sensed data from the actual region might not well represent the obfuscated region, the data to report also needs to be adjusted on the mobile phone side before uploading. Thus, ⟨*obfuscated region, adjusted data*⟩ is uploaded to the server by each participant. After receiving all the (adjusted) data from the participants, the server

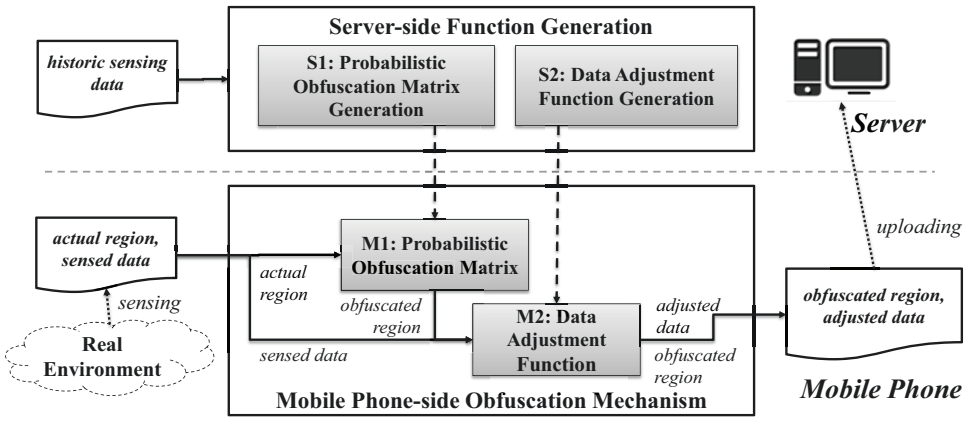


Figure 6.2: Location privacy-preserving framework for Sparse MCS.

infers the complete sensing map, following the same process of regular Sparse MCS.

Figure 6.2 shows the overview of our proposed location privacy-preserving framework for Sparse MCS. It consists of two tiers — server side and mobile client side. On the server side, before a Sparse MCS task starts, based on the historical sensed data, it generates the *probabilistic obfuscation matrix* (Step S1) and *data adjustment function* (Step S2) in an offline manner. The probabilistic obfuscation matrix encodes the probabilities of obfuscating any one region to another one. By carefully selecting the probabilities in the matrix, we can guarantee the participant's location privacy so that the participant's actual region cannot be accurately inferred from the obfuscated region, *even if the adversary knows the obfuscation matrix*.[1] The data adjustment function is used to reduce the error in the sensed data due to the region obfuscation. It is learned by studying the correlation between any two regions' sensed data in the historical log. For example, the sensed temperature value can be adjusted to be higher if the temperature of the obfuscated region is historically higher than the participant's actual region.

Before task execution, participants pre-download both obfuscation matrix and data adjustment function to their mobile phones. When executing a task, the workflow on the mobile phone client is as follows. First, the mobile phone obtains the sensed data in the actual region. Then, based on the probabilistic obfuscation matrix, it obfuscates the actual region to another region with the corresponding probability (Step M1). Afterward, knowing both the actual region and obfuscated region, the data adjustment function adjusts the original sensed data to estimate the real sensed data of obfuscated region (Step M2). Finally, the mobile phone accomplishes a sensing task by uploading the obfuscated region and adjusted data to the server.

With this framework overview in mind, in the following sections, we introduce our notion of differential location privacy for Sparse MCS and describe how we construct the probabilistic obfuscation matrix and data adjustment function.

[1] Same as an adversary, the server cannot know the participant's actual region even though the obfuscation matrix is generated by the server.

6.4 Differential Location Privacy for Sparse MCS

In the context of Sparse MCS where the sensing area is divided into regions, we introduce *ϵ-region-ambiguity*, a notion of differential location privacy for Sparse MCS, which can guarantee the differential location privacy regardless of the adversary's prior knowledge about the participant's location distribution.

6.4.1 Defining ϵ-Region-Ambiguity

Let $\mathcal{R}$ represent the set of regions in the target sensing area and r^* represent the obfuscated region. We define the aforementioned probabilistic obfuscation matrix as P. For a target sensing area of m regions (i.e., $|\mathcal{R}| = m$), we need an $m \times m$ obfuscation matrix P, where $P[r, r^*]$ is the probability of assigning region r to obfuscated region r^*. If an adversary knows the obfuscation matrix P and some prior information about a participant's location, e.g., a set of likely visited regions $\widetilde{\mathcal{R}} \subset \mathcal{R}$ (where $\widetilde{\mathcal{R}}$ includes the user's actual location), then the adversary can compute the probability of generating r^* for each region in $\widetilde{\mathcal{R}}$. If a certain region in $\widetilde{\mathcal{R}}$ has a significantly high probability, the adversary can confidently infer the user's actual region. To prevent the adversary from tracing back to the user's actual location, the basic idea of differential location privacy is to make the obfuscation probabilities from any two regions in $\widetilde{\mathcal{R}}$ to the obfuscated region r^* similar. In such a way, there is no single region in $\widetilde{\mathcal{R}}$ that has significantly higher obfuscation probability than the others, thus the adversary cannot accurately infer which region in $\widetilde{\mathcal{R}}$ is the actual location of the user.

However, an adversary's prior knowledge may be unknown in advance, we thus take the whole set of regions $\mathcal{R}$ as $\widetilde{\mathcal{R}}$. That is to say, given the obfuscated region r^*, we attempt to make any two regions in $\mathcal{R}$ have similar obfuscation probabilities to be mapped to r^*. So regardless of which set of regions the adversary has prior knowledge about, he gains little additional knowledge about which region in $\mathcal{R}$ is the user's actual location after observing the obfuscated region r^*.

With this insight, we formally define the differential location privacy notion used in Sparse MCS, *ϵ-region-ambiguity*, as follows:

Definition 6.1. *ϵ-Region-Ambiguity. Suppose the target sensing area consists of a set of regions $\mathcal{R}$, then a probabilistic obfuscation matrix P satisfies ϵ-region-ambiguity iff:*

$$P[r, r^*] \leq e^{\epsilon} \cdot P[r', r^*], \quad \forall r, r', r^* \in \mathcal{R} \tag{6.1}$$

where r and r' are any region in $\mathcal{R}$, $P[r, r^*]$ is the probability of obfuscating region r to region r^*, and ϵ is the differential privacy parameter indicating the level of privacy.

Eq. (6.1) can also be represented as:

$$\frac{1}{e^{\epsilon}} \leq \frac{P[r, r^*]}{P[r', r^*]} \leq e^{\epsilon}$$

region	*1*	*2*	*3*
1	0.50	0.25	0.25
2	0.25	0.50	0.25
3	0.25	0.25	0.50

region	*1*	*2*	*3*
1	0.25	0.25	0.50
2	0.25	0.50	0.25
3	0.50	0.25	0.25

Table 6.1: Two obfuscation matrices satisfying *ln(2)-region-ambiguity* for three regions. The entry $[i, j]$ refers to the probability of obfuscating region i to region j.

For the same obfuscated region r^*, it bounds the maximum ratio difference between any two entries of $P[r, r^*]$, $\forall r$. Note that the smaller ϵ an obfuscation matrix P can satisfy, the higher-level location privacy P can provide, because the probability of obfuscation from any actual region is more uniform such that it is less distinguishable to know which is the original actual region. However, setting ϵ too small would restrict P too much and compromise data quality, which will be explained later.

To illustrate the implications of Eq. (6.1), Table 6.1 shows two examples of obfuscation matrices designed for a Sparse MCS task with three regions, both satisfying ϵ-region-ambiguity (where $\epsilon = \ln(2)$). Given an obfuscated region, e.g., region 1, ϵ-region-ambiguity specifies that any two probabilities in the column 1 of the obfuscation matrix are bounded within a constant factor of e^{ϵ}. In other words, the ratio of the largest probability to the smallest one in each column cannot be larger than e^{ϵ}. For example, we can see that for each column in the left obfuscation matrix in Table 6.1, the largest probability is 0.5 and the smallest is 0.25, so the ratio is bounded within 2 (i.e., e^{ϵ} where $\epsilon = \ln(2)$). Thus, the left obfuscation matrix satisfies ln(2)-region-ambiguity. Similarly, the right obfuscation matrix in Table 6.1 satisfies ln(2)-region-ambiguity as well. By showing two obfuscation matrices satisfying ln(2)-region-ambiguity, we want to point out that there can exist *multiple* probability matrices satisfying ϵ-region-ambiguity, given a predefined privacy level ϵ.

6.4.2 Privacy Guarantee of ϵ-Region-Ambiguity

Next, we show how the differential privacy characteristic of ϵ-region-ambiguity can limit the knowledge gained by an adversary, regardless of his prior knowledge about the user's location distribution.

Adversary Model: Suppose an adversary has some *prior* knowledge about the probabilistic distribution of a user's actual region r, denoted as $\pi(r)$. Now if the adversary knows the user's obfuscated region r^* and the obfuscation matrix P, then based on Bayes' rule, the adversary can get a *posterior* distribution of the user's location, noted as $\sigma(r)$, as follows:

$$\sigma(r) = \frac{P[r, r^*] \cdot \pi(r)}{\sum_{r' \in \mathcal{R}} P[r', r^*] \cdot \pi(r')} \tag{6.2}$$

With this adversary model, if the obfuscation matrix P satisfies ϵ-region-ambiguity, the adversary's knowledge about the user's actual location distribution is not significantly changed after observing the obfuscated region r^*. In other words, the improvement of the adversary's knowledge caused by the observation, i.e., $\sigma(r)/\pi(r)$, is bounded within a certain range. Formally, we have the following theorem:

Theorem 6.1. *If an obfuscation matrix satisfies ϵ-region-ambiguity, then for an adversary with any prior knowledge π, his posterior knowledge σ satisfies:*

$$\frac{1}{e^{\epsilon}} \leq \frac{\sigma(r)}{\pi(r)} \leq e^{\epsilon}, \quad \forall r \in \mathcal{R} \tag{6.3}$$

Proof.

$$\sigma(r) = \frac{P[r, r^*] \cdot \pi(r)}{\sum_{r' \in \mathcal{R}} P[r', r^*] \cdot \pi(r')} \tag{6.4}$$

$$\implies \sigma(r) = \frac{\pi(r)}{\sum_{r' \in \mathcal{R}} \frac{P[r', r^*]}{P[r, r^*]} \cdot \pi(r')} \tag{6.5}$$

$$\implies \sigma(r) \leq \frac{\pi(r)}{\frac{1}{e^{\epsilon}} \cdot \sum_{r' \in \mathcal{R}} \pi(r')} \tag{6.6}$$

$$\implies \frac{\sigma(r)}{\pi(r)} \leq e^{\epsilon} \qquad (\sum_{r' \in \mathcal{R}} \pi(r') = 1) \tag{6.7}$$

The proof for $\frac{\sigma(r)}{\pi(r)} \geq \frac{1}{e^{\epsilon}}$ is similar, change Eq. (6.6) to:

$$\sigma(r) \geq \frac{\pi(r)}{e^{\epsilon} \cdot \sum_{r' \in \mathcal{R}} \pi(r')} \tag{6.8}$$

□

Hence, by modeling the location privacy requirement with differential privacy (Eq. (6.1)), we can restrict the *knowledge leakage* caused by the observed obfuscated region r^*. Recall that this leaves many choices for the obfuscation matrix P. In the next section, we describe how we find the optimal P with the consideration of data quality.

6.4.3 Relationship with Other Differential Location Privacy Notions

The definition of *ϵ-region-ambiguity* is inspired by a few existing works about the differential location privacy. A *generalized* differential privacy notion is proposed in [156] and used in the location context recently [65]:

$$P[r, r^*] \leq e^{\epsilon \cdot d(r, r')} \cdot P[r', r^*], \quad \forall r, r', r^* \in \mathcal{R} \tag{6.9}$$

where $d(r, r')$ can be arbitrary distance metric. Thus, ϵ-region-ambiguity can be seen as an instance of Eq. (6.9) by setting $d(r, r')$ as *discrete distance* [65]. By comparison, the existing differential location privacy notion used in LBS, *Geo-indistinguishability* [62, 64], is another instance of Eq. (6.9) by setting $d(r, r')$ as *Euclidean distance*. Other distance metrics, such as *Manhattan distance* and *location semantic distance*, can also be used according to specific applications [65].

In this work, we use *discrete metric* due to its simplicity for explaining the effect of differential privacy protection. Note that our proposed mechanism to obtain the obfuscation matrix (Section 6.5) can be easily applied to the differential location privacy notion with other distance metrics, or even *multiple* differential location privacy notions *simultaneously*.

6.5 Differential Location Privacy with Data Quality Loss Reduction

With our definition of ϵ-region-ambiguity to ensure the differential location privacy in Sparse MCS, we can select a probabilistic obfuscation matrix P to obfuscate the participant's location region, but there can be many suitable matrices. The choice of P affects two things: (i) the spatial distribution of obfuscated regions and (ii) the sensed data uncertainty of the obfuscated region. Hence, among the set of P, which satisfies ϵ-region-ambiguity, we aim to select the optimal P which produces an even spatial distribution of obfuscated regions and minimum data uncertainty in order to reduce the loss in data quality. In this section, we formulate the differential location privacy-preserving problem in Sparse MCS as a data quality optimization problem, with *minimizing the data uncertainty in the obfuscated regions* as the optimization objective and with *ϵ-region-ambiguity* and *even obfuscated region distribution* as constraints.

6.5.1 Data Quality Requirements for Obfuscation

Recall that in regular Sparse MCS tasks, to infer the complete sensing matrix, compressive sensing theory assumes that (1) the participants report from uniformly or *evenly* distributed regions, and (2) their reported sensed data are *accurate* [155, 139]. However, introducing differential location privacy into Sparse MCS may compromise these two requirements:

1. *Even Obfuscated Region Distribution*. Even if the selected participants' actual location distribution is even, the distribution of the obfuscated regions may be uneven. For instance, consider an extreme case of the obfuscation matrix where no region can be obfuscated to region i. Then in the collected sensing matrix C, all the values of the ith row will be unknown, thus it is impossible to recover the values in this row accurately [139].
2. *Small Data Uncertainty in Obfuscated Regions*. The participant's actual sensed data corresponds to the original actual region, not the obfuscated region. Although the data adjustment step can reduce the discrepancy or uncertainty in the reported data, there will still be a baseline uncertainty due to the choice of obfuscation matrix. We describe later how an optimal choice of probabilistic obfuscation matrix can further minimize this uncertainty.

With these problem statements, we next formulate our optimization problem.

6.5.2 Optimal Obfuscation Matrix Generation

We seek to reduce the data uncertainty and control the distribution evenness of the obfuscated regions which arise due to location obfuscation. To reduce data uncertainty, we optimally select a probabilistic obfuscation matrix that can minimize the *expectation of*

region	*1*	*2*	*3*
1	0	0.1	0.2
2	0.1	0	0.1
3	0.2	0.1	0

Table 6.2: An example of uncertainty matrix for three regions. The entry $[i, j]$ refers to the data uncertainty incurred by obfuscating region i to region j.

data uncertainty between the reported and true data in the obfuscated regions. To keep the obfuscated region evenly distributed, we introduce an *evenness constraint* to the obfuscation matrix so that different regions have similar probabilities of being the obfuscated region. Finally, we propose a mechanism, *DUM-ϵe*, to combine these two aspects and ϵ-region-ambiguity together, to obtain an optimal obfuscation matrix which achieves better data quality while ensuring the differential privacy protection for Sparse MCS.

Objective: Data Uncertainty Minimization

The first step to reduce the data uncertainty in the obfuscated regions is to estimate the corresponding sensed data of the region by incorporating a data adjustment function in our mechanism (Figure 6.2). As environmental data (e.g., temperature, humidity) usually has high spatial correlations [131, 157], for simplicity, we use *linear regression* as the data adjustment model. In our privacy-preserving framework (Figure 6.2), the linear regression function is learned or trained in the server (Step S2), while the linear fit estimation is performed on the mobile phone (Step M2). We will discuss how more sophisticated adjustment functions may be used in a later section.

Since all data adjustment models have intrinsic uncertainty, the selection of the probabilistic obfuscation matrix will impact the inferred data quality. We define an *uncertainty matrix*, U, to represent the data uncertainty for every location obfuscation, where $U[r, r^*]$ is the data uncertainty incurred by the obfuscating region r to region r^*. Note that the uncertainty matrix U is intrinsic to the mapping $\langle r \rightarrow r^* \rangle, \forall r, r^*$ and is independent of obfuscation matrix P. For the case of using linear regression as the data adjustment model, the uncertainty $U[r, r^*]$ can be computed by the *residual standard error* [97]. Intuitively, obfuscating a certain region to different regions would result in different data uncertainties. For example, when sensing environmental data such as air quality, if the original region, r, is a park, then obfuscating it to another park, r_1^*, will likely lead to lower data uncertainty than that of obfuscating it to a transportation hub, r_2^*, i.e., $U[r, r_1^*] < U[r, r_2^*]$. We therefore seek to find an obfuscation matrix P which is likely to obfuscate r to r_1^* instead of r_2^*, i.e., P contains the relation $P[r, r_1^*] > P[r, r_2^*]$.

In general, given a data adjustment function, our objective is to find an obfuscation matrix P that can minimize the overall *expectation* of data uncertainty in the uncertainty matrix U. Table 6.2 shows an example uncertainty matrix for three regions. Note that $U[r, r] = 0$ because there is no uncertainty if the obfuscated and actual regions are same.

The *overall expectation of data uncertainty* across the whole target sensing area is just

region	*1*	*2*	*3*
1	0	0.75	0.25
2	0	0.75	0.25
3	0	0.50	0.50

Table 6.3: An obfuscation matrix satisfying *ln(2)-region-ambiguity* for three regions, where no region can be obfuscated to region 1.

the normalized sum product of data uncertainty over all region obfuscations $\mathcal{R}$, i.e.,

$$\bar{U} = \sum_{r \in \mathcal{R}} p(r) \cdot \sum_{r^* \in \mathcal{R}} U[r, r^*] \cdot P[r, r^*] \tag{6.10}$$

where $U[r, r^*]$ is the data uncertainty of obfuscating r to r^*, and $P[r, r^*]$ is the probability of the obfuscation; $p(r)$ is the overall probability that any one participant will appear at the region r ($\sum_{r \in \mathcal{R}} p(r) = 1$). Usually we can assume $p(r)$ as uniform distribution or estimate it via the overall human mobility pattern (e.g., via anonymous mobile phone call records [93, 94]). Actually, for a well-designed Sparse MCS task using compressive sensing, the distribution of $p(r)$ should also be roughly even to get the good data quality (recall the assumption of '*Even Data Distribution*' in Section 6.2.2). For simplicity, $p(r)$ is set to follow uniform distribution here, i.e., $p(r) = 1/|\mathcal{R}|$.

With Eq. (6.10), we can calculate the expectation of data uncertainty for the obfuscation matrices in Table 6.1: 0.067 (Left) and 0.100 (Right). This implies that under the same privacy level of ϵ-region-ambiguity (in this case, $\epsilon = \ln(2)$), the left obfuscation matrix in Table 6.1 produces better data quality than the right one.

Therefore, to improve data quality for Sparse MCS, the objective for the optimization problem is minimizing the *expectation of data uncertainty* (see Eq. (6.13)). This optimization is performed under several constraints which we discuss next.

Constraint 1: Differential Privacy with ϵ-Region-Ambiguity

The first constraint is that the optimal probabilistic obfuscation matrix P must satisfy *ϵ-region-ambiguity* to provide a guaranteed level of privacy, which is tunable by setting the value of ϵ. We have described this previously in Eq. (6.1).

Constraint 2: Even Obfuscated Region Distribution

The second constraint concerns the inference of missing data for Sparse MCS. In addition to minimizing the data uncertainties in the obfuscated regions, the obfuscated regions need to be evenly distributed to ensure the inferred data quality. Formally, we represent this evenness with the probability of a certain region r^* to be the obfuscated region:

$$\psi(r^*) = \sum_{r \in \mathcal{R}} p(r) \cdot P[r, r^*] \tag{6.11}$$

Both example obfuscation matrices in Table 6.1 have even obfuscated region distribution: $\psi(r^*) = 1/3$, $\forall r^* \in \{1,2,3\}$. However, this is not always the case. While the obfuscation

matrix in Table 6.3 still satisfies ln(2)-region-ambiguity, no actual region is ever obfuscated to region 1, i.e., $\psi(1) = 0$. With this obfuscation matrix for location obfuscation, the server would never receive any data from region 1, such that all the values in the corresponding row of the collected sensing matrix C will be unknown. Thus, it is impossible to recover the missing data for region 1 accurately via compressive sensing [155, 139]. To avoid uneven obfuscation, we introduce an *evenness* constraint:

$$\psi(r^*) = \frac{1}{|\mathcal{R}|}, \quad \forall r^* \in \mathcal{R} \tag{6.12}$$

which means that every region has exactly the equal probability to be the obfuscated region. In the experiment, we will evaluate that the evenly-distributed obfuscation outperforms different levels of unevenly-distributed obfuscation in terms of the data quality for a Sparse MCS task.

Optimization: Reduced Data Quality Loss

With the objective of reducing data quality loss, by considering the data uncertainty, differential location privacy and the distribution of the obfuscated regions, we formulate the following linear optimization program, ***D**ata **U**ncertainty-**M**inimization under constraints of **ϵ**-region-ambiguity and **e**venly-distributed obfuscation* (*DUM-ϵe*), to obtain the optimal obfuscation matrix:

Choose $P[r, r^*], \forall r, r^* \in \mathcal{R}$, in order to

$$\min_{P} \sum_{r \in \mathcal{R}} p(r) \cdot \sum_{r^* \in \mathcal{R}} U[r, r^*] \cdot P[r, r^*] \tag{6.13}$$

$$\text{s.t. } P[r, r^*] \leq e^{\epsilon} \cdot P[r', r^*] \qquad \forall r, r', r^* \in \mathcal{R} \tag{6.14}$$

$$\sum_{r \in \mathcal{R}} p(r) \cdot P[r, r^*] = \frac{1}{|\mathcal{R}|} \qquad \forall r^* \in \mathcal{R} \tag{6.15}$$

$$P[r, r^*] \geq 0 \qquad \forall r, r^* \in \mathcal{R} \tag{6.16}$$

$$\sum_{r^* \in \mathcal{R}} P[r, r^*] = 1 \qquad \forall r \in \mathcal{R} \tag{6.17}$$

where Eq. (6.13) is the objective function to minimize data uncertainty, $\mathcal{R}$ is the set of all regions in the target sensing area, $|\mathcal{R}|$ is the number of regions, and the following are the series of linear constraints: Eq. (6.14) is the requirement for *ϵ-region-ambiguity* differential location privacy; Eq. (6.15) is the *evenness* constraint; Eq. (6.16) and Eq. (6.17) are general constraints for probabilities. ϵ represents the privacy level, which is a constant specified for the application. δ is also predefined and we discuss the method to select its value next.

In summary, we propose a location privacy-preserving framework to generate an optimal obfuscation matrix P. The obfuscation matrix satisfies differential privacy as defined as *ϵ-region-ambiguity* (Eq. (6.14)), and is optimized via *DUM-ϵe* to produce *an even obfuscation distribution* (Eq. (6.15)) with *minimized data uncertainty* (Eq. (6.13)). This supports a guaranteed privacy level while still satisfying the two requirements for compressive sensing to achieve good inference performance in Sparse MCS.

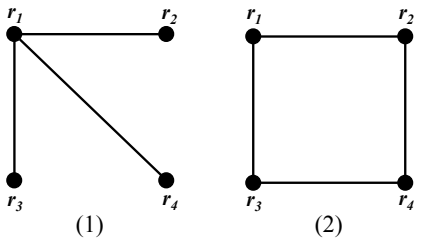


Figure 6.3: Two examples of diameter-2-critical graphs for 4 regions.

6.5.3 Approximation of Optimal Obfuscation Matrix

Although *DUM-ϵe* is expected to obtain good performance in Sparse MCS, it has $O(|\mathcal{R}|^3)$ constraints, which makes it hard to be scaled up to a large number of regions. In this subsection, we approximate the optimal obfuscation matrix by reducing the number of the constraints in the linear programming to $O(|\mathcal{R}|^2)$. The basic idea is: instead of comparing the probabilities of obfuscating any two regions r, r' to a given region r^* (Eq. (6.14)), we restrict the comparison only between some specific region pairs but still ensure the privacy protection of ϵ-region-ambiguity.

To mark which two regions need to be compared, we define a *region-comparison* graph $\mathcal{G}(\mathcal{R}, \mathcal{E})$ where each vertex $r \in \mathcal{R}$ represents a region. Two regions r, r' are required to be compared if there exists an edge $\langle r, r' \rangle \in \mathcal{E}$. Intuitively, for *DUM-ϵe*, $\mathcal{G}$ is a complete graph as every two regions should be compared.

Now, to describe the approximation mechanism, we introduce the definition of *diameter-2-critical* graph [158]. A diameter-2-critical graph is a graph whose diameter (i.e., the maximum distance between any pair of vertexes) is 2 and the deletion of any edge increases its diameter (examples in Figure 6.3). Then, the following theorem holds (see the appendix for proof):

Theorem 6.2. *If $\mathcal{G}(\mathcal{R}, \mathcal{E})$ is a diameter-2-critical graph, an obfuscation matrix P satisfies ϵ-region-ambiguity if:*

$$P[r, r^*] \leq e^{\frac{\epsilon}{2}} \cdot P[r', r^*], \quad \forall \langle r, r' \rangle \in \mathcal{E}, r^* \in \mathcal{R} \tag{6.18}$$

Proof. As $\mathcal{G}(\mathcal{R}, \mathcal{E})$ is a diameter-2-critical graph, for any two regions $r_1, r_2 \in \mathcal{R}$, we can find a region r' and $\langle r_1, r' \rangle, \langle r', r_2 \rangle \in \mathcal{E}$, then for any region $r^* \in \mathcal{R}$:

$$P[r_1, r^*] \leq e^{\frac{\epsilon}{2}} \cdot P[r', r^*] \tag{6.19}$$

$$\implies P[r_1, r^*] \leq e^{\frac{\epsilon}{2}} \cdot (e^{\frac{\epsilon}{2}} \cdot P[r_2, r^*]) \tag{6.20}$$

$$\implies P[r_1, r^*] \leq e^{\epsilon} \cdot P[r_2, r^*] \tag{6.21}$$

□

Note that the number of comparisons in Eq. (6.18) is $|\mathcal{E}||\mathcal{R}|$. To minimize $|\mathcal{E}||\mathcal{R}|$, we expect to find the diameter-2-critical graph with the minimal number of edges. Fortunately,

the minimal diameter-2-critical graph can be constructed as follows: one vertex is joined by an edge with all others [159] (Figure 6.3 (1) is the minimal diameter-2-critical graph for 4 regions). For the minimal diameter-2-critical graph, $|\mathcal{E}| = |\mathcal{R}| - 1$, so the number of comparisons to ensure differential location privacy can be reduced from $|\mathcal{R}|^3$ (Eq. (6.14)) to $|\mathcal{R}|^2 - |\mathcal{R}|$ (Eq. (6.18)). To wrap up, we propose the following linear program to approximate *DUM-ϵe* by replacing Eq. (6.14) with Eq. (6.18), called *Fast DUM-ϵe* (*FDUM-ϵe*):

$$\min_{P} \sum_{r \in \mathcal{R}} p(r) \cdot \sum_{r^* \in \mathcal{R}} U[r, r^*] \cdot P[r, r^*]$$

$$\text{s.t. } P[r, r^*] \leq e^{\frac{\epsilon}{2}} \cdot P[r', r^*] \qquad \forall \langle r, r' \rangle \in \mathcal{E}, r^* \in \mathcal{R}$$
$$constraints\ (6.15) - (6.17)$$

Selecting the "central" vertex: When we construct the minimal diameter-2-critical graph, any region can be chosen as the "central" vertex that connects to all others, which leads to different formulations of *FDUM-ϵe* (e.g., in Figure 6.3 (1), r_1 is the "central" vertex). Our empirical experiments show that this selection does not affect the data quality of *FDUM-ϵe* obviously, thus we choose the region whose *ID* is 1 as the "central" vertex in the evaluation.

6.6 Evaluation

In the experiment, we use two real-life datasets, *environment* and *traffic* monitoring, to evaluate the performance of our proposed mechanisms.

6.6.1 Baseline Methods and Evaluation Scenarios

We compare the data quality of our proposed location privacy-preserving mechanisms, *DUM-ϵe* and *FDUM-ϵe*, to three privacy-preserving baseline mechanisms. The difference between the three privacy-preserving mechanisms and our mechanisms is the resultant selected obfuscation matrix; we use the same data adjustment model (linear regression) for all the mechanisms.

We further measure the data quality when no location privacy (***No-Privacy***) is applied in the Sparse MCS and use it as the data quality upper bound for comparison purpose. Below are the three privacy-preserving baseline mechanisms.

Self [160]: If a region is obfuscated to itself, the data uncertainty of the obfuscation is zero. Thus, if an obfuscation matrix assigns higher probability to self-obfuscation pairs, i.e., $P[r, r]$, it is likely that the overall data uncertainty will be lower. Based on the intuition, we use a computationally simple obfuscation matrix putting more probabilities on the self-

obfuscation while satisfying ϵ-region-ambiguity:

$$P_{self}[r, r^*] \propto \begin{cases} e^{\epsilon}, & r^* = r \\ 1, & r^* \neq r \end{cases} \tag{6.22}$$

This diagonal obfuscation matrix was also previously used in [160] to improve the privacy-preserving mining of association rules. Note that the "$\propto$" in eq. (6.22) means "proportional to", and the exact value of $P_{self}[r, r^*]$ should be normalized to ensure that $\sum_{r^* \in \mathcal{R}} P_{self}[r, r^*] = 1$. For example, Table 6.1 is the *Self* obfuscation matrix for 3 regions for $\epsilon = \ln(2)$.

Planar-Laplace [62]: The state-of-the-art approach in differential location privacy is to add Laplace noise to the actual location data to probabilistically obfuscate the location [62, 63, 60]. To compare with this kind of mechanism, we adapt the *Planar-Laplace* mechanism [62] into the MCS scenario. *Planar-Laplace* first adds Laplace noise to the actual location coordinate to generate an obfuscated location point (might be outside of the target sensing area), and then remaps the obfuscated location point to the nearest region as the obfuscated region. Intuitively, *Planar-Laplace* tends to obfuscate a region to its nearby regions with high probability.

Exponential [61]: Another widely-used mechanism to achieve differential privacy is the exponential mechanism [61]. By considering the uncertainty matrix as the scoring function, we construct the *Exponential* obfuscation matrix that attempts to reduce the uncertainty incurred by the obfuscation, while satisfying ϵ-region-ambiguity. Intuitively, the smaller the uncertainty of obfuscating r to r^* is, the larger the probability of obfuscating r to r^* will be for the *Exponential* mechanism.

In the following sections, we use two Sparse MCS task scenarios to evaluate our proposed mechanisms:

Environment Monitoring: this task collects environment temperature and humidity sensing data in a modest number of regions (57 regions); thus we can run *DUM-ϵe* for this scenario. The objective of this experiment is to compare the data quality of our optimal mechanism *DUM-ϵe* to three baseline mechanisms with the same differential location privacy guarantee.

Traffic Monitoring: this task collects travel speed from taxis in a number of road segments, i.e., regions. For some large settings ($\geq$200 road segments in our experiment), *DUM-ϵe* fails with the error of 'out of memory' on our test computer, while *FDUM-ϵe* can work efficiently. Besides comparing our mechanisms to the baselines, another objective of this experiment is to compare the performance between our proposed two mechanisms, *DUM-ϵe* and *FDUM-ϵe*, with respect to computation time, memory usage and data quality.

6.6.2 Evaluation on Environment Monitoring

We first evaluate our privacy-preserving mechanism using the temperature and humidity sensing datasets from the *SensorScope* project [133], which deployed 97 sensors across the EPFL campus (300m×500m). We divided the target area into 100 equal-sized regions

(each region is 30m×50m), and found that 57 regions are deployed with the sensors. Since Sparse MCS requires all regions to be sensed at least once, we limit our target area to these 57 regions. If a region had more than one sensor, we only used the sensor with the most number of the temperature or humidity readings to ensure that all the data from the same region is generated by the same sensor. The sensed data spanned one week, from 2007-07-01 to 2007-07-07, with a sensing cycle of 30-minute intervals. We used the sensed data of the first day as historical data to train the data adjustment function, and the remaining six days as the evaluation dataset. Thus, we have totally 288 sensing cycles. As the number of regions is modest, we run *DUM-ϵe* to generate the optimal obfuscation matrix on this dataset.

Evaluation Experiment Method

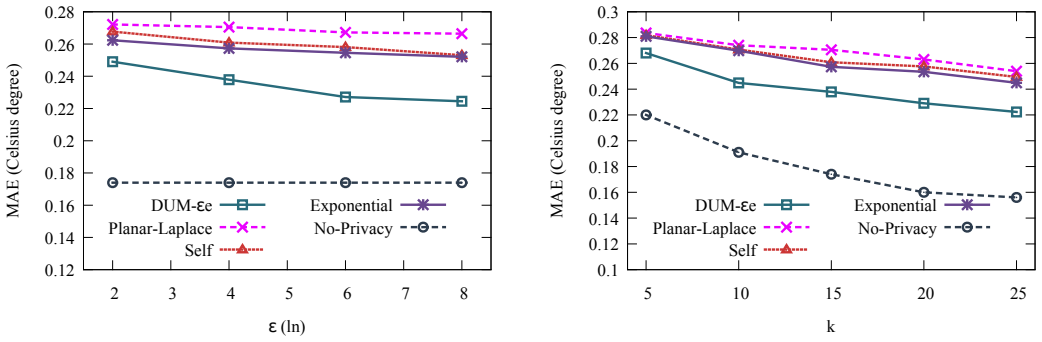
We set the experiment up with the number of participants selected in each cycle, k, and privacy level, ϵ, as independent variables. ϵ is usually chosen by participants and could be personalized.[2] For simplicity, we assume all the participants require the same privacy level ϵ. The MCS organizer typically decides on the number of participants k, based on the incentive budget, the level of privacy guarantee, and the expected data quality. We assume a large number of candidates, N, from which the organizer can select the actual participants. As k ranges from 5 to 25 in our experiment, we set N to a larger value, 1000, and simulate the 1000 candidates' mobility traces around the target sensing area using a state-of-the-art mobility model, SWIM [122]. The SWIM parameters are chosen as the 'Dartmouth' setting [122] to simulate the user mobility traces on campus.

In each sensing cycle, we randomly selected k participants from N candidates. Each selected participant was placed in one of the 57 regions (w.r.t. the simulated trace). We take the sensing values of the placed regions as ground truth. If the server received multiple data reports for one region (it is possible due to the probabilistic nature of the obfuscation process), we used the *average* value as the reported reading for that region. For the *No-Privacy* mechanism, the data in placed regions was reported with the ground truth data, while the data in uncovered regions without reported data was inferred using the STCS algorithm. For the privacy-preserving mechanisms, a participant would have her placed region obfuscated to another region (different obfuscation matrices for different mechanisms). Then her sensed data from the placed region would be adjusted, using the learned linear regression functions, and reported with the obfuscated region. Finally, the data of the uncovered regions was inferred by the STCS algorithm like for the no-privacy mechanism.

As measures of data quality, we calculated the *Mean Absolute Error (MAE)* between the captured data (whether participant reported or inferred via STCS) and ground truth data on all the 57 regions for each sensing cycle, and report the mean values over 288 sensing cycles.[3] Due to the probabilistic nature of the obfuscation matrices, for each experiment setting of k and ϵ values, *MAE* are calculated over 10 repeated trials. Furthermore, since we

[2]The server can pre-generate obfuscation matrices for different privacy levels ϵ. Then, a participant can decide to download the one that satisfies her privacy requirement.

[3]We also calculated the *Root Mean Squared Error (RMSE)* and the evaluation results are similar.



(a) k=15, varying ϵ (b) ϵ = ln(4), varying k

Figure 6.4: *MAE* on the temperature monitoring.

focus on the *data quality* ***loss*** due to privacy protection, we are also interested in the *relative* errors with respect to the *No-Privacy* mechanism. For example, the *MAE* data quality loss due to *DUM-ϵe* is:

$$Loss_{MAE}(DUM\text{-}\epsilon e) = MAE(DUM\text{-}\epsilon e) - MAE(No\text{-}Privacy)$$

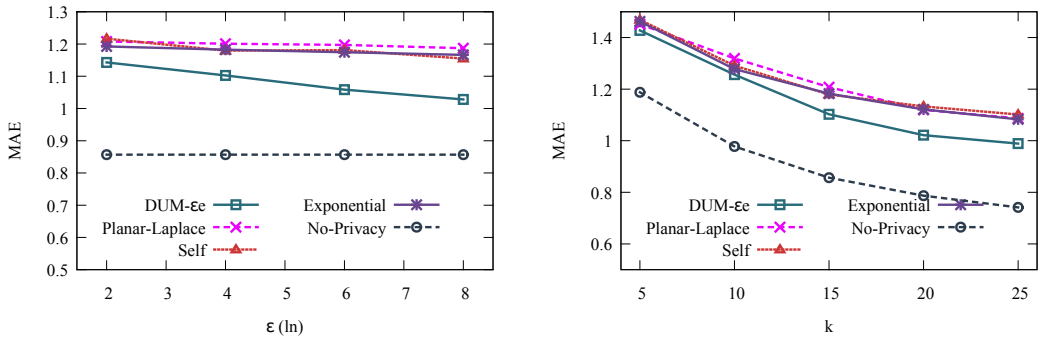
For our evaluation, we compare $Loss_{MAE}$ for *DUM-ϵe*, *Self*, *Planar-Laplace* and *Exponential*.

Performance Results

In general, our results show that *DUM-ϵe* can reduce data quality loss by 15-45% compared to three baseline privacy-preserving mechanisms, with the same level of differential privacy guarantee ϵ. The detailed results are illustrated as follows.

Figure 6.4 (a) shows the *MAE* of temperature under varying privacy levels ϵ for different mechanisms, with a fixed number of participants (k = 15). As expected, *No-Privacy* achieves the best data quality (the smallest *MAE*, i.e., 0.175). Among the privacy-preserving mechanisms, *DUM-ϵe* incurs the smallest $Loss_{MAE}$ at each privacy level. For example, when the privacy level ϵ is ln(6), $Loss_{MAE}(DUM\text{-}\epsilon e) = 0.056$, smaller than $Loss_{MAE}(Planar\text{-}Laplace) = 0.093$, $Loss_{MAE}(Self) = 0.084$, and $Loss_{MAE}(Exponential) = 0.081$. Generally, when varying ϵ in [ln(2), ln(8)], *DUM-ϵe* can reduce the data quality loss, $Loss_{MAE}$, by 23.6-45.4%, 19.9.7-36.1% and 15.3-35.3% compared to *Planar-Laplace*, *Self* and *Exponential*, respectively. We can also see that the *MAE* of *DUM-ϵe* decreases more sharply with increasing ϵ compared to the three baseline privacy-preserving mechanisms. This indicates a *less demanding compromise* of data quality for the privacy level. Loosening the privacy level leads to more significant improvements in data quality for *DUM-ϵe* compared to the other mechanisms.

Figure 6.4 (b) shows that the *MAE* of temperature decreases for all the mechanisms with more participants (higher k). When the privacy level ϵ is fixed to ln(4), by varying k from 5 to 25, $Loss_{MAE}$ of *DUM-ϵe* is always the smallest among all the privacy-preserving mechanisms. Specifically, $Loss_{MAE}$ of *DUM-ϵe* is smaller than the three baseline mechanisms by 21.3-35.3%. Furthermore, we can see that if the task organizer requires a certain



(a) k=15, varying ϵ (b) $\epsilon = \ln(4)$, varying k

Figure 6.5: *MAE* on the humidity monitoring.

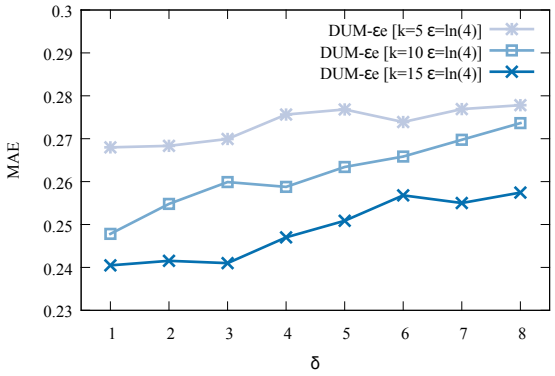


Figure 6.6: *MAE* on the temperature dataset, varying δ in the δ-evenness constraint of *DUM-ϵe*.

data quality level, he will have to trade-off between smaller, more manageable recruitment populations (k) and the participants' privacy level (ϵ). For example, suppose the organizer requires $MAE \leq 0.23$ for temperature sensing, this can be achieved with *No-Privacy* by recruiting $k = 5$ participants. To achieve the same *MAE* using *DUM-ϵe*, the organizer will need to recruit 20 more participants ($k = 25$) to ensure ϵ-region-ambiguity for $\epsilon = \ln(4)$. On the other hand, decreasing the privacy level to $\epsilon = \ln(6)$ can allow a smaller recruitment size $k = 15$.

The results for the humidity dataset are similar to the temperature, shown in Figure 6.5.

Evenness vs. Unevenness

Here, we verify that by ensuring the exactly even obfuscated region distribution (Eq. (6.15)), *DUM-ϵe* can achieve better quality for a Sparse MCS task than uneven obfuscated region distribution. To compare even distribution to different levels of uneven distribution, we replace the exact *evenness* constaint (Eq. (6.15)) in *DUM-ϵe* by introducing another constraint called δ-*evenness*:

$$\forall r_1^*, r_2^* \in \mathcal{R}, \quad \psi(r_1^*) \leq \delta \cdot \psi(r_2^*) \tag{6.23}$$

where δ is a predefined constant ($\delta \geq 1$). δ-*evenness* ensures that for any two regions, the difference between their probabilities of being the obfuscated region is bounded within δ.

Server-Side Function Generation	
S1: obfuscation matrix	$17.2s$
S2: linear regression function	$8.3s$
Mobile-Side Real-time Running	
M1: region obfuscation	$3.6\times10^{-4}s$
M2: data adjustment	$5.7\times10^{-6}s$

Table 6.4: Average durations for server and mobile computations.

Note that when $\delta = 1$, δ-*evenness* is equivalent to the exact *evenness* constraint (Eq. (6.15)).

We evaluated how δ impacts the quality for a Sparse MCS task. Figure 6.6 shows the *MAE* of *DUM-ϵe* when setting different δ-evenness constraints on the temperature dataset. Generally, with the increase of δ, the *MAE* becomes larger, illustrating that the data quality degrades when the distribution of the obfuscated regions becomes *uneven*. Therefore, we use the exact *evenness* constraint ($\delta = 1$) in our implementation of *DUM-ϵe*.

Computation Time

We used the *CPLEX optimizer* to solve *DUM-ϵe* to obtain the optimal obfuscation matrix.[4] Recall that our proposed privacy-preserving framework includes two stages: (i) the offline server-side process to learn both probabilistic obfuscation matrix and data adjustment function, and (ii) the online mobile client-side process to real-time obfuscate a participant's actual location and adjust her raw sensed data. Table 6.4 shows the computation time of each stage on the temperature dataset. The first stage is more computation-intensive, especially for solving the linear program *DUM-ϵe* to get the optimal obfuscation matrix. This lasts for about 17 seconds using *CPLEX* on our test computer (CPU: Intel Core i7-3612QM@2.10GHz, RAM: 8 GB, OS: Windows 7). As the first stage is an offline process, this computation duration is acceptable for the 57-region environment monitoring task.[5] For the second stage, our mechanism runs in less than 1 millisecond on the Nexus 5, which is also reasonable on a mobile phone.

6.6.3 Evaluation on Traffic Monitoring

We also evaluate our mechanism in another Sparse MCS scenario — traffic monitoring [15]. In this scenario, the MCS organizer is interested in the real-time travel speed of some road segments in the urban area, while many participants (e.g., taxis) are willing to upload their travel speed to the organizer. Due to the budget constraint, the organizer cannot collect data from all the participants. [15] has already verified that by collecting the travel speed from only a small portion of road segments, the travel speed of the other road segments can be inferred with high accuracy via compressive sensing. By considering road segments as

[4]CPLEX: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[5]In the next experiment of traffic monitoring, we will show that when the number of regions increases, *DUM-ϵe* might not be solvable.

#Segments	*DUM-ϵe*	*FDUM-ϵe*
100	337*s* / 1000MB	2*s* / 28MB
150	1119*s* / 1433MB*	4*s* / 60MB
200	N/A	8*s* / 108MB
300	N/A	22*s* / 236MB
400	N/A	48*s* / 410MB
500	N/A	121*s* / 633MB

*turn on the 'memory reduction switch' in CPLEX; otherwise N/A

Table 6.5: Computation time (second) and memory usage (MB) of the obfuscation matrix generation for 100-500 road segments.

'regions', our privacy-preserving framework is applicable to this MCS scenario as well.

Experiment Setting

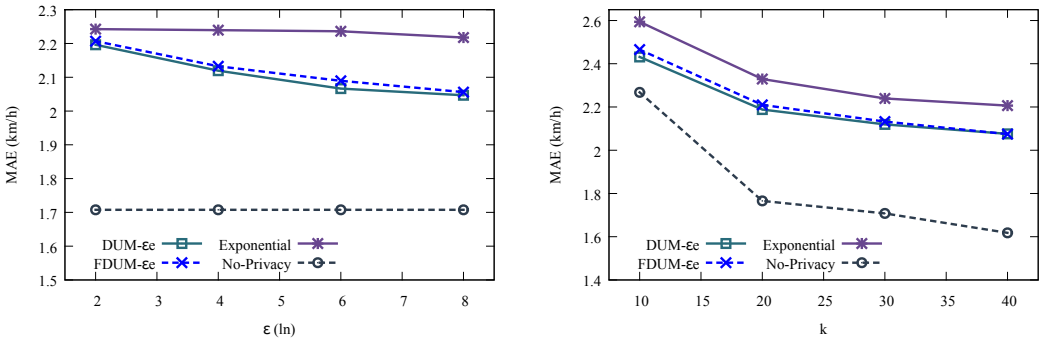
We use a trajectory dataset generated by more than 30,000 taxis in Beijing [161] to conduct this experiment. The sensing cycle is set to 1 hour. Like [15], as a road segment might not be covered by any one taxi in some sensing cycles (i.e., without ground truth travel speed), we only keep a subset of the road segments in our experiment to make the ground truth sensing matrix have as fewer vacancies as possible. We thus choose the top 100-500 road segments which are covered by the taxis most frequently to construct the target sensing area. In each sensing cycle, among all the taxis in the target sensing area, we randomly choose k (10-50% of the number of the road segments) taxis to upload their travel speed with their (obfuscated) road segments. One selected taxi only uploads the travel speed of one road segment even though it can cover multiple segments in one cycle. We adopt the same assumption as [107]: every selected taxi can obtain the ground truth travel speed, which is the average travel speed of the road segment in the cycle. This taxi dataset contains data for four days: the first day's data is used to train the data adjustment functions, while the remaining three days' data are for test purposes.

Computation Time and Memory Usage

Due to the large number of constraints ($O(|\mathcal{R}|^3)$), we can run *DUM-ϵe* only on the top 100/150 road segments in our experiment environment; for more than 200 road segments, *DUM-ϵe* fails with the error of 'out of memory' on our test computer. In comparison, *FDUM-ϵe* can deal with the scenario up to 500 road segments. The detailed computation costs, including both time and memory usage, are shown in Table 6.5. We can see that the computation time of *FDUM-ϵe* is less than 1% of the computation time of *DUM-ϵe*, if *DUM-ϵe* is available. As *FDUM-ϵe* reduces the number of constraints from $O(|\mathcal{R}|^3)$ to $O(|\mathcal{R}|^2)$, the memory usage of *FDUM-ϵe* is much less than *DUM-ϵe*. Therefore, *FDUM-ϵe* is capable of dealing with the large-scale problems that *DUM-ϵe* cannot handle.

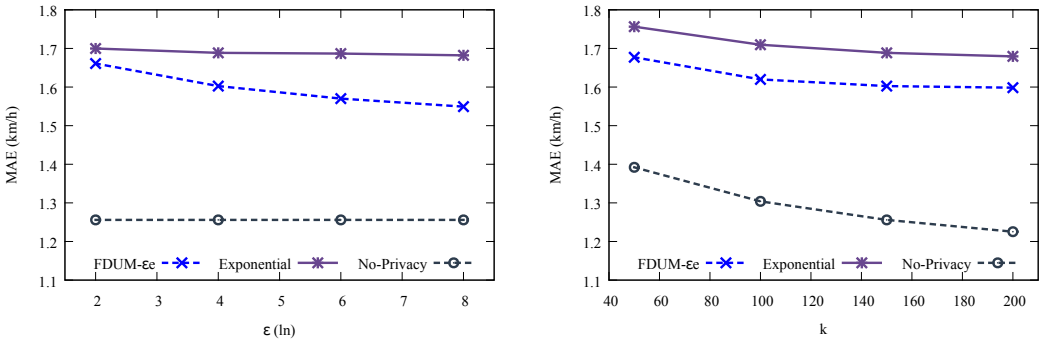
Data Quality

We compare the data quality obtained by our proposed mechanisms to the baseline



(a) k=30, varying ϵ (b) ϵ = ln(4), varying k

Figure 6.7: *MAE* on traffic monitoring (100 road segments).



(a) k=150, varying ϵ (b) ϵ = ln(4), varying k

Figure 6.8: *MAE* on traffic monitoring (500 road segments).

mechanisms. For brevity, we only show the results with 100 (see Figure 6.7) and 500 road segments (see Figure 6.8, *DUM-ϵe* is not available for this problem scale); the data quality metric is *MAE*. For clarity, among all the baseline mechanisms, we only show the results of the best one, *Exponential*. Generally, *DUM-ϵe* and *FDUM-ϵe* achieve much better data quality than *Exponential* with the same level of differential privacy guarantee. Furthermore, *FDUM-ϵe* achieves quite similar data quality as *DUM-ϵe*. For example, *FDUM-ϵe* increases the quality loss only by 2.1-6.0% compared to *DUM-ϵe* when 30 taxis are randomly selected for the traffic monitoring on 100 road segments with different levels of differential privacy guarantee (Figure 6.7 (a)).

6.7 Discussion

As the first work to balance differential location privacy and data quality in MCS, this work focuses on the major algorithms, process and evaluation results. We discuss future work to address some limitations.

6.7.1 Need for Historical Sensed Data

Our mechanism requires some initial ground truth historical sensed data to generate the data adjustment functions and the optimal location obfuscation matrix. To get this initial data, before the privacy-preserving Sparse MCS task begins, the MCS organizer needs to employ a group of workers to cover the target sensing area and collect accurate sensed data. After the data adjustment functions and obfuscation matrix are learned, we can stop the initialization stage and recruit participants for the privacy-preserving Sparse MCS task. Note that the learned data adjustment functions and obfuscation matrix could be less effective after a period of time (e.g., the temperature correlations between different regions will probably change for different seasons) and thus need to be updated periodically. Deciding when to update them will be one direction of our future work.

6.7.2 Repeated-Observations Trajectory Attack

In this work, we have focused on privacy protection from a *snapshot* attack [152, 154], i.e., the adversary attempts to find a user's actual region r only using the obfuscated region r^* of one sensing cycle. This kind of attack is reasonable as we make the assumption that there exist a large number of candidate participants so that no participant needs to continuously report her locations. However, if this assumption does not hold (i.e., a participant is required to contribute her sensed data and locations in continuous cycles), then an alternative attack is a *trajectory* attack, where an adversary observes a user's obfuscated regions over multiple or repeated sensing cycles, $\langle r_1^*, r_2^*, ..., r_n^* \rangle$. Specifically, if P satisfies ϵ-region-ambiguity under the snapshot case, then it only satisfies $n\epsilon$-region-ambiguity under the trajectory case. This is a common limitation of differential privacy [62, 60] and fortunately, new methods have been proposed to achieve better privacy protection under trajectory attacks (e.g., [162]), which may be applied in our future work.

6.7.3 Location Correlations between Participants

To achieve the desired protection effect, differential privacy implicitly assumes that the different data are generated independently [163]. That is, to protect participants' location privacy in Sparse MCS, differential privacy requires that the locations of different participants should not have strong correlations. Supposing an extreme case that two participants are always together, if they both upload sensed data with obfuscated regions to the server at the same cycle, the adversary can use two obfuscated regions to infer their actual region. Then the differential location privacy protection is degraded to 2ϵ-region-ambiguity. Fortunately, this concern can be mitigated largely via a well-designed participant selection scheme: As we assume a large number of candidate participants exist, it is probable for an organizer to select a small subset of participants whose locations are almost independent.

6.7.4 Alternative Data Adjustment Models

Although we use linear regression as the data adjustment model, our proposed privacy-preserving mechanism is not limited to it. Given an alternative data adjustment model, M, the key adaptation is to generate the corresponding data uncertainty matrix U for our privacy-preserving framework. For any model M, we can use *resampling* methods [97], such as *cross-validation* and *bootstrapping*, to estimate the data uncertainty incurred by location obfuscation, and then construct the uncertainty matrix U.

6.7.5 Auxiliary Knowledge beyond Location Distribution

If an adversary has auxiliary knowledge beyond the location distribution (which cannot be written in the form of $\pi(r)$), then the differential privacy protection effect (Section 6.4.2) may fail. For example, if the adversary foreknows that the participant is "in the hottest region" and he gets the temperature sensing map from the MCS task, then he can infer the participant's actual location (suppose the temperature sensing map is accurate enough). In our future work, we will try to deal with this kind of auxiliary information — in addition to location obfuscation, some sensed data perturbation (e.g., temperature) may also be needed.

6.8 Concluding Remarks

In this chapter we present a framework for achieving differential location privacy in sparse mobile crowd-sensing. This takes into account the desired level of privacy protection, the prior knowledge that an adversary might have, and the data quality loss due to location obfuscation. Our contributions are (i) the introduction of the differential location privacy notion, called *ϵ-region-ambiguity*, into Sparse MCS, (ii) a mechanism to obtain the optimal location obfuscation matrix, called *DUM-ϵe*, which provides a guaranteed level of location privacy with reduced loss in data quality, and (iii) a less computation-intensive mechanism to approximate the optimal obfuscation matrix, called *FDUM-ϵe*, which can achieve similar data quality compared to *DUM-ϵe* while significantly reducing the computation time and memory usage. We evaluated our mechanisms *DUM-ϵe* and *FDUM-ϵe* with real-world environment and traffic monitoring datasets, and showed that they achieved better data quality than the state-of-the-art baseline mechanisms, with the same level of differential privacy guarantee.

Part IV

Reflections

Chapter 7

Conclusion and Future Work

Contents

As an alternative and complementary sensing paradigm to the traditional infrastructure-based sensing methods, Mobile CrowdSensing (MCS) provides an efficient way to conduct large-scale sensing campaigns for tasks such as environment monitoring, traffic congestion detection, and social relationship sensing, with the help of large crowds owning up-to-date sensor-rich equipped smartphones. Primarily, there are two human roles in MCS, *organizers* and *participants*; accordingly, there exist a variety of research challenges and opportunities, in order to help both organizers and participants play their corresponding roles better. This dissertation, following this research direction, attempts to address some of the research issues from two roles' perspectives, such as reducing smartphone energy consumption for participants and achieving efficient quality-budget trade-off for organizers.

In this final chapter, the main contributions of the dissertation are summarized. Then, future research opportunities are discussed.

7.1 Dissertation Summary

The main contributions of this dissertation include two MCS mechanisms: *collaborative data uploading* and *sparse mobile crowdsensing*.

Collaborative Data Uploading. *Energy consumption* and *mobile data cost* are two key factors affecting users' willingness to participate in MCS tasks. By allowing some delay between data uploading and sensing for the participants, various methods, such piggyback data uploading and data relays between participants, are studied to relieve the burden of the participants in both energy consumption and mobile data cost, which finally leads to the

collaborative data uploading mechanism. Specifically,

- **effSense**: In Chapter 3, we consider two kinds of participants: *data-plan* and *non-data-plan*. Data-plan users are mostly concerned with energy consumption as they hold enough mobile data plan (e.g., unlimited data plan); non-data-plan users are more sensitive to data cost. The proposed framework, called *effSense*, attempts to save energy consumption for data-plan users and data cost for non-data-plan users, respectively, by appropriately selecting the timing and network to transfer data. The possible networks include WiFi, Bluetooth, and piggybacking on 3G calls. Especially, the short-range user encounters via Bluetooth are leveraged to trigger user data relays, i.e., *collaborations*, to further improve the system efficiency. Based on real-life dataset, effSense is verified that it can reduce 55-65% of energy consumption for data-plan users, and 48-52% of data cost for non-data-plan users, respectively, compared to direct 3G data uploading.
- **ecoSense**: In Chapter 4, suppose that the MCS organizer pays participants incentives to cover their 3G data cost, a participant partition problem is formulated in *collaborative data uploading*: how to partition all the participants into two 3G data price plans — *Unlimited Data Plan* or *Pay As You Go* — in order to minimize the total incentives for the organizer? Based on user mobility prediction and sensed data size estimation, a genetic algorithm based partition method, called *ecoSense*, is proposed to achieve an efficient user partition, which can save around 50% of total 3G cost incentives compared to the basic partition method that directly assigns each participant's data plan according to her sensed data size.

Sparse Mobile Crowdsensing. *Budget* and *quality* are two primary concerns for MCS organizers. To get a high-quality urban sensing map, the traditional way is recruiting enough participants to cover almost all the sub-areas of a city; but the incentive budget can be high. Considering the sptio-temporal correlations of urban data, we propose a novel MCS paradigm that significantly reduces the number of tasks allocated (thus budget) by only sensing partial sub-areas and inferring missing values for the rest sub-areas; meanwhile, the overall data quality of the sensing map is guaranteed. Due to the sparsity of the sensed sub-areas, we name this novel crowdsensing paradigm as *sparse mobile crowdsensing*. Specifically,

- **CCS-TA**: In Chapter 5, a pioneering sparse MCS implementation is proposed for urban environment monitoring applications. A three-step sparse MCS framework, called *CCS-TA*, is introduced, including *optimal task allocation*, *missing data inference*, and *data quality assessment*. A combination of state-of-the-art data mining and machine learning techniques, such as *active learning*, *compressive sensing*, and *Bayesian inference*, are implemented in CCS-TA to ensure task quality via only sparsely sensed data. Evaluation results on a temperature dataset show that by sensing only 13% of cells, the mean absolute error of temperature is ensured to be less than 0.25°C in 90% of sensing cycles.
- **DUM-ϵe**: In Chapter 6, a location privacy preserving mechanism based on *differential privacy* is designed for the participants who take part in sparse MCS tasks.

Differential location privacy uses the location obfuscation method to change a user's actual location to another one before sensed data is uploaded to the server. Then, to reduce the data quality loss (i.e., inference error in sparse MCS), an optimal obfuscation mechanism is designed by prioritizing the location obfuscation between two areas that have similar or correlated sensing values. This mechanism is called *DUM-ϵe*, meaning *Data Uncertainty Minimization* under the constraints of *ϵ-region-ambiguity* and *evenly-distributed obfuscation*. Experiment results show that DUM-ϵe can reduce the data quality loss by 15-45% compared to two state-of-the-art differential privacy mechanisms, *Laplace* and *Exponential* obfuscation.

7.2 Future Research Opportunities

This dissertation proposes two novel MCS frameworks, *collaborative data uploading* and *sparse mobile crowdsensing*. While several key issues and challenges have been studied, still a variety of research opportunities exist for improving the two MCS frameworks in the future.

Malicious Participants. Sometimes, malicious participants may exist, which can bring unexpected threats and losses to other participants and the organizer. In collaborative data uploading, a malicious participant might abuse another participant's personal sensed data, which is received via Bluetooth encounters, for the malicious user's own purpose; this can bring privacy violations to other participants. In sparse mobile crowdsensing, a malicious participant can ruin the quality of the inferred sensing map by uploading faked sensed data. Against such attacks from malicious participants, it is urgently required to introduce security mechanisms, such as encryption, and participant reputation evaluation systems into our proposed MCS frameworks.

Multiple Task Scenario. In reality, an organizer may carry out multiple MCS tasks simultaneously; a participant also can take part in multiple tasks at the same time. By considering certain relationships between different tasks, it is probable that our proposed frameworks can be enhanced. For collaborative data uploading, if there exist sensed data redundancy between different tasks of different participants, the data to be uploaded may be reduced by merging or aggregating different participants' data when they make data relays, which finally results in lower energy consumption and data cost. For sparse mobile crowdsensing, diverse data of multiple tasks may show certain correlations, which can boost the inference accuracy of all the tasks together (e.g., via *transfer learning* [164]).

Event Detection. While the studied MCS applications in this dissertation are primarily monitoring, such as user activity and environment monitoring, another important usage of MCS is for event or anomaly detection. To better suit the event detection purpose, our designed mechanisms in this dissertation are also necessary for certain adaptation. For example, in collaborative data uploading, while some delay (even relatively long) is tolerant for normal human activity monitoring scenario, it may not suit the requirement of abnormal event detection; in sparse mobile crowdsensing, the data are collected only in partial target

area, thus the anomalies happening in unsensed areas may fail to be identified. Therefore, to adapt our proposed MCS frameworks to event detection still needs much research effort in the future.

Generalized MCS Platform. To provide organizers with an efficient way to create and conduct various kinds of MCS tasks, as well as participants with an easy portal to discover and join in interested MCS campaigns, designing a generalized MCS platform or middleware is another critical research and practical topic. To implement such a platform that can incorporate the techniques proposed in this dissertation, as well as other state-of-the-art research advances in MCS, requires extensive future endeavors from both academia and industry.

Bibliography

[1] R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Communications Magazine*, vol. 49, pp. 32–39, 2011.

[2] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn, "The rise of people-centric sensing," *Internet Computing, IEEE*, vol. 12, no. 4, pp. 12–21, 2008.

[3] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, "Peir, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 55–68.

[4] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: an end-to-end participatory urban noise mapping system," in *IPSN*, 2010, pp. 105–116.

[5] S. Morishita, S. Maenaka, D. Nagata, M. Tamai, K. Yasumoto, T. Fukukura, and K. Sato, "Sakurasensor: quasi-realtime cherry-lined roads detection through participatory video sensing by cars," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 695–705.

[6] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "Cartel: A distributed mobile sensor computing system," in *SenSys*, 2006, pp. 125–138.

[7] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 323–336.

[8] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, "Understanding the coverage and scalability of place-centric crowdsensing," in *UbiComp*, 2013, pp. 3–12.

[9] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "SociableSense: exploring the trade-offs of adaptive sampling and computation offloading for social sensing," in *MobiCom*, 2011, pp. 73–84.

[10] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen, "Image browsing, processing, and clustering for participatory sensing: lessons from a dietsense prototype," in *Proceedings of the 4th workshop on Embedded networked sensors*. ACM, 2007, pp. 13–17.

[11] W. Sherchan, P. P. Jayaraman, S. Krishnaswamy, A. Zaslavsky, S. Loke, and A. Sinha, "Using on-the-move mining for mobile crowdsensing," in *MDM*, 2012, pp. 115–124.

[12] T. W. Malone, R. Laubacher, and C. Dellarocas, "The collective intelligence genome," *IEEE Engineering Management Review*, vol. 38, no. 3, p. 38, 2010.

[13] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele, "Participatory air pollution monitoring using smartphones," in *International Workshop on Mobile Sensing, IPSN*, 2012.

[14] B. Predic, Z. Yan, J. Eberle, D. Stojanovic, and K. Aberer, "Exposuresense: Integrating daily activities with air quality using mobile participatory sensing," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 2013, pp. 303–305.

[15] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A compressive sensing approach to urban traffic estimation with probe vehicles," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2289–2302, 2013.

[16] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "Parknet: drive-by sensing of road-side parking statistics," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 123–136.

[17] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "Greengps: a participatory sensing fuel-efficient maps application," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 151–164.

[18] M. Atzmueller, M. Becker, M. Kibanov, C. Scholz, S. Doerfel, A. Hotho, B.-E. Macek, F. Mitzlaff, J. Mueller, and G. Stumme, "Ubicon and its applications for ubiquitous social computing," *New Review of Hypermedia and Multimedia*, vol. 20, no. 1, pp. 53–77, 2014.

[19] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: A programming framework for crowd-sensing applications," in *MobiSys*, 2012, pp. 337–350.

[20] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing*, 2010, pp. 138–155.

[21] G. Cardone, L. Foschini, C. Borcea, P. Bellavista, A. Corradi, M. Talasila, and R. Curtmola, "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, 2013.

[22] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "Anonysense: privacy-aware people-centric sensing," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 211–224.

[23] D. Christin, C. Roßkopf, M. Hollick, L. A. Martucci, and S. S. Kanhere, "Incognisense: An anonymity-preserving reputation framework for participatory sensing applications," *Pervasive and Mobile Computing*, vol. 9, no. 3, pp. 353–371, 2013.

[24] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[25] G. Cohn, S. Gupta, T.-J. Lee, D. Morris, J. R. Smith, M. S. Reynolds, D. S. Tan, and S. N. Patel, "An ultra-low-power human body motion sensor using static electric field sensing," in *UbiComp*, 2012, pp. 99–102.

[26] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, "Entracked: energy-efficient robust position tracking for mobile devices," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 221–234.

[27] M. B. Kjærgaard, S. Bhattacharya, H. Blunck, and P. Nurmi, "Energy-efficient trajectory tracking for mobile devices," in *MobiSys*, 2011, pp. 307–320.

[28] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, "Energy-accuracy trade-off for continuous mobile device location," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 285–298.

[29] J. Paek, J. Kim, and R. Govindan, "Energy-efficient rate-adaptive gps-based positioning for smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 299–314.

[30] M.-R. Ra, B. Priyantha, A. Kansal, and J. Liu, "Improving energy efficiency of personal sensing applications with heterogeneous multi-processors," in *UbiComp*, 2012, pp. 1–10.

[31] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in *MobiSys*, 2010, pp. 49–62.

[32] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[33] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *INFOCOM*. IEEE, 2012, pp. 945–953.

[34] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[35] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[36] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *MobiSys*, 2010, pp. 255–270.

[37] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 29–35, 2014.

[38] J. Nurminen, "Parallel connections and their effect on the battery consumption of a mobile phone," in *CCNC*, 2010, pp. 1–5.

[39] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha, "Piggyback crowdsensing (pcs): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *SenSys*, 2013, pp. 7:1–7:14.

[40] K. C. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 3, pp. 250–291, 2006.

[41] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A. T. Campbell, "Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones," in *Pervasive Computing*, 2010, pp. 355–372.

[42] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can wifi deliver?" *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 2, pp. 536–550, 2013.

[43] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *UbiComp*, 2012, pp. 481–490.

[44] K. Shilton, "Four billion little brothers?: Privacy, mobile phones, and ubiquitous data collection," *Communications of the ACM*, vol. 52, no. 11, pp. 48–53, 2009.

[45] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick, "A survey on privacy in mobile participatory sensing applications," *Journal of Systems and Software*, vol. 84, no. 11, pp. 1928–1946, 2011.

[46] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "Prism: platform for remote sensing using smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 63–76.

[47] B. N. Schilit, A. LaMarca, G. Borriello, W. G. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson, "Challenge: Ubiquitous location-aware computing and the place lab initiative," in *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*. ACM, 2003, pp. 29–35.

[48] B. Debatin, J. P. Lovejoy, A.-K. Horn, and B. N. Hughes, "Facebook and online privacy: Attitudes, behaviors, and unintended consequences," *Journal of Computer-Mediated Communication*, vol. 15, no. 1, pp. 83–108, 2009.

[49] A. Acquisti and J. Grossklags, "Privacy and rationality in individual decision making," *IEEE Security & Privacy*, no. 1, pp. 26–33, 2005.

[50] J. Krumm, "Inference attacks on location tracks," in *Pervasive Computing*. Springer, 2007, pp. 127–143.

[51] A. Kapadia, D. Kotz, and N. Triandopoulos, "Opportunistic sensing: Security challenges for the new paradigm," in *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*. IEEE, 2009, pp. 1–10.

[52] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos, "AnonySense: a system for anonymous opportunistic sensing," *PMC*, vol. 7, no. 1, p. 16–30, 2011.

[53] Q. Li and G. Cao, "Providing privacy-aware incentives for mobile sensing," in *PerCom*, 2013, pp. 76–84.

[54] ——, "Providing efficient privacy-aware incentives for mobile sensing," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*. IEEE, 2014, pp. 208–217.

[55] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Pervasive Computing*. Springer, 2005, pp. 152–170.

[56] J. Krumm, "A survey of computational location privacy," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 391–399, 2009.

[57] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting anonymous location queries in mobile environments with privacygrid," in *WWW*, 2008, pp. 237–246.

[58] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: query processing for location services without compromising privacy," in *VLDB*, 2006, pp. 763–774.

[59] H. Kido, Y. Yanagisawa, and T. Satoh, "Protection of location privacy using dummies for location-based services," in *ICDE Workshops*, 2005, pp. 1248–1248.

[60] C. Dwork, "Differential privacy," in *ICALP*, 2006, pp. 1–12.

[61] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *FOCS*, 2007, pp. 94–103.

[62] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *CCS*, 2013, pp. 901–914.

[63] R. Dewri, "Local differential perturbations: Location privacy under approximate knowledge attackers," *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2360–2372, 2013.

[64] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Optimal geo-indistinguishable mechanisms for location privacy," in *CCS*, 2014, pp. 251–262.

[65] R. Shokri, "Privacy games: Optimal user-centric data obfuscation," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 1–17, 2015.

[66] D. A. Epstein, A. Borning, and J. Fogarty, "Fine-grained sharing of sensed physical activity: a value sensitive approach," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 489–498.

[67] R. K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher, "Poolview: stream privacy for grassroots participatory sensing," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 281–294.

[68] J. Shi, R. Zhang, Y. Liu, and Y. Zhang, "Prisense: privacy-preserving data aggregation in people-centric urban sensing systems," in *Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[69] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application," in *SenSys*, 2008.

[70] N. Maisonneuve, M. Stevens, M. E. Niessen, and L. Steels, "Noisetube: Measuring and mapping noise pollution with mobile phones," in *Information Technologies in Environmental Engineering*. Springer, 2009, pp. 215–228.

[71] M. Mun, S. Hao, N. Mishra, K. Shilton, J. Burke, D. Estrin, M. Hansen, and R. Govindan, "Personal data vaults: a locus of control for personal data streams," in *Proceedings of the 6th International COnference*. ACM, 2010, p. 17.

[72] R. Cáceres, L. Cox, H. Lim, A. Shakimov, and A. Varshavsky, "Virtual individual servers as privacy-preserving proxies for mobile devices," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*. ACM, 2009, pp. 37–42.

[73] H. Gao, C. Liu, W. Wang, J. Zhao, Z. Song, X. Su, J. Crowcroft, and K. Leung, "A survey of incentive mechanisms for participatory sensing," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 918–943, 2015.

[74] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava, "Examining micro-payments for participatory sensing data collections," in *Proceedings of the 12th ACM international conference on Ubiquitous computing*. ACM, 2010, pp. 33–36.

[75] M. Musthag, A. Raij, D. Ganesan, S. Kumar, and S. Shiffman, "Exploring micro-incentive strategies for participant compensation in high-burden studies," in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 435–444.

[76] J. P. Rula, V. Navda, F. E. Bustamante, R. Bhagwan, and S. Guha, "No one-size fits all: Towards a principled approach for incentives in mobile crowdsourcing," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014, p. 3.

[77] E. Massung, D. Coyle, K. F. Cater, M. Jay, and C. Preist, "Using crowdsourcing to support pro-environmental community activism," in *CHI*, 2013, pp. 371–380.

[78] J.-S. Lee and B. Hoh, "Sell your experiences: a market mechanism based incentive for participatory sensing," in *PerCom*, 2010, pp. 60–68.

[79] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *MobiCom*, 2012, pp. 173–184.

[80] D. Peng, F. Wu, and G. Chen, "Pay as how well you do: A quality based incentive mechanism for crowdsensing," in *MobiHoc*, 2015.

[81] D. Zhao, X.-Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 1213–1221.

[82] X. Zhang, Z. Yang, Z. Zhou, H. Cai, L. Chen, and X. Li, "Free market of crowdsourcing: Incentive mechanism design for mobile sensing," *IEEE Transactions on Parallel and Distributed Systems*, no. 12, pp. 3190–3200, 2014.

[83] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. ACM, 2011, pp. 9–15.

[84] S. Matyas, C. Matyas, C. Schlieder, P. Kiefer, H. Mitarai, and M. Kamata, "Designing location-based mobile games with a purpose: collecting geospatial data with cityexplorer," in *Proceedings of the 2008 international conference on advances in computer entertainment technology*. ACM, 2008, pp. 244–247.

[85] K. Tuite, N. Snavely, D.-y. Hsiao, N. Tabing, and Z. Popovic, "Photocity: training experts at large-scale image acquisition through a competitive game," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011, pp. 1383–1392.

[86] J. Rula and F. E. Bustamante, "Crowd (soft) control: moving beyond the opportunistic," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012, p. 3.

[87] R. Kawajiri, M. Shimosaka, and H. Kahima, "Steered crowdsensing: incentive design towards quality-oriented place-centric crowdsensing," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2014, pp. 691–701.

[88] A. M. Rashid, K. Ling, R. D. Tassone, P. Resnick, R. Kraut, and J. Riedl, "Motivating participation by displaying the value of contribution," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 955–958.

[89] H. Kawasaki, A. Yamamoto, H. Kurasawa, H. Sato, M. Nakamura, and H. Matsumura, "Top of worlds: method for improving motivation to participate in sensing services," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 594–595.

[90] L. Yu, P. André, A. Kittur, and R. Kraut, "A comparison of social, learning, and financial strategies on crowd engagement and output quality," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 2014, pp. 967–978.

[91] H. Xiong, D. Zhang, L. Wang, J. P. Gibson, and J. Zhu, "Eemc: Enabling energy-efficient mobile crowdsensing with anonymous participants," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 39, 2015.

[92] X. Sheng, J. Tang, and W. Zhang, "Energy-efficient collaborative sensing with mobile phones," in *INFOCOM*, 2012, pp. 1916–1924.

[93] H. Xiong, D. Zhang, L. Wang, and H. Chaouchi, "Emc3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint," *IEEE Transactions on Mobile Computing, in press*, 2014.

[94] D. Zhang, H. Xiong, L. Wang, and G. Chen, "Crowdrecruiter: Selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *UbiComp*, 2014, pp. 703–714.

[95] A. Ahmed, K. Yasumoto, Y. Yamauchi, and M. Ito, "Distance and time based node selection for probabilistic coverage in people-centric sensing," in *Proceedings of SECON*. IEEE, 2011, pp. 134–142.

[96] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, "Crowdtasker: Maximizing coverage quality in piggyback crowdsensing under budget constraint," in *PerCom*, 2015, pp. 55–62.

[97] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013.

[98] A. Krause, H. B. McMahan, C. Guestrin, and A. Gupta, "Robust submodular observation selection," *Journal of Machine Learning Research*, vol. 9, pp. 2761–2801, 2008.

[99] A. Ostfeld, J. G. Uber, E. Salomons, J. W. Berry, W. E. Hart, C. A. Phillips, J.-P. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan *et al.*, "The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 556–568, 2008.

[100] L. Liu, W. Wei, D. Zhao, and H. Ma, "Urban resolution: New metric for measuring the quality of urban sensing," *IEEE Transactions on Mobile Computing, online*, no. c, pp. 1–1, 2015.

[101] X. Yin, J. Han, and P. S. Yu, "Truth discovery with multiple conflicting information providers on the web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 796–808, 2008.

[102] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*. ACM, 2012, pp. 233–244.

[103] B. Zhao, B. I. Rubinstein, J. Gemmell, and J. Han, "A bayesian approach to discovering truth from conflicting sources for data integration," *Proceedings of the VLDB Endowment*, vol. 5, no. 6, pp. 550–561, 2012.

[104] X. Yin and W. Tan, "Semi-supervised truth discovery," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 217–226.

[105] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, J. Gao, and K. Ren, "Cloud-enabled privacy-preserving truth discovery in crowd sensing systems," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 183–196.

[106] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, "A survey on truth discovery," *arXiv preprint arXiv:1505.02463*, 2015.

[107] L. Xu, X. Hao, N. D. Lane, X. Liu, and T. Moscibroda, "Cost-aware compressive sensing for networked sensing systems," in *IPSN*, 2015, pp. 130–141.

[108] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. Barnes, "icrowd: Near-optimal task allocation for piggyback crowdsensing," *IEEE Transactions on Mobile Computing, to appear*, 2015.

[109] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," in *Grid Computing, 2009 10th IEEE/ACM International Conference on*. IEEE, 2009, pp. 17–25.

[110] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, 2012.

[111] L. Wu, S. K. Garg, and R. Buyya, "Sla-based resource allocation for software as a service provider (saas) in cloud computing environments," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 2011, pp. 195–204.

[112] D. Villegas, A. Antoniou, S. M. Sadjadi, and A. Iosup, "An analysis of provisioning and allocation policies for infrastructure-as-a-service clouds," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 2012, pp. 612–619.

[113] N. Eagle and A. (Sandy) Pentland, "Reality mining: sensing complex social systems," *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 255–268, 2006.

[114] N. Eagle, "The reality mining data readme."

[115] R. I. Ciobanu and C. Dobre, "Predicting encounters in opportunistic networks," in *HP-MOSys*, 2012, pp. 9–14.

[116] J. Weinberg, L. D. Brown, and J. R. Stroud, "Bayesian forecasting of an inhomogeneous poisson process with applications to call center data," *Journal of the American Statistical Association*, vol. 102, no. 480, pp. 1185–1198, 2007.

[117] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *IMC*, 2009, pp. 280–293.

[118] G. Perrucci, F. Fitzek, and J. Widmer, "Survey on energy consumption entities on the smartphone platform," in *VTC Spring*, 2011, pp. 1–6.

[119] S. Bell, A. McDiarmid, and J. Irvine, "Nodobo: Mobile phone as a software sensor for social network research," in *VTC Spring*, 2011, pp. 1–5.

[120] L. Ravindranath, A. Thiagarajan, H. Balakrishnan, and S. Madden, "Code in the air: simplifying sensing and coordination tasks on smartphones," in *HotMobile*, 2012, pp. 4:1–4:6.

[121] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[122] S. Kosta, A. Mei, and J. Stefa, "Large-scale synthetic social mobile networks with swim," *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 116–129, 2014.

[123] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[124] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *WDTN*, 2005, pp. 252–259.

[125] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, "Nextplace: a spatio-temporal prediction framework for pervasive systems," *Pervasive Computing*, pp. 152–169, 2011.

[126] Y. Chon, H. Shin, E. Talipov, and H. Cha, "Evaluating mobility models for temporal prediction with high-granularity mobility data," in *PerCom*, 2012, pp. 206–212.

[127] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[128] D. T. Wagner, A. Rice, and A. R. Beresford, "Device analyzer: Understanding smartphone usage," in *MobiQuitous*, 2014, pp. 195–208.

[129] T. Luo and C.-K. Tham, "Fairness and social welfare in incentivizing participatory sensing," in *SECON*, 2012, pp. 425–433.

[130] S. Hachem, A. Pathak, and V. Issarny, "Probabilistic registration for large-scale mobile participatory sensing," in *PerCom*, 2013, pp. 132–140.

[131] L. Kong, M. Xia, X.-Y. Liu, M.-Y. Wu, and X. Liu, "Data loss and reconstruction in sensor networks," in *INFOCOM*, 2013, pp. 1654–1662.

[132] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang, "Diagnosing new york city's noises with ubiquitous data," in *UbiComp*, 2014, pp. 715–725.

[133] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Application-specific sensor network for environmental monitoring," *ACM Transactions on Sensor Networks*, vol. 6, no. 2, pp. 17:1–17:32, 2010.

[134] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-air: when urban air quality inference meets big data," in *KDD*, 2013, pp. 1436–1444.

[135] P. V. Bolstad, L. Swift, F. Collins, and J. Régnière, "Measured and predicted air temperatures at basin to regional scales in the southern appalachian mountains," *Agricultural and Forest Meteorology*, vol. 91, no. 3, pp. 161–176, 1998.

[136] "Shtc1 - digital temperature and humidity sensor," http://www.sensirion.com/en/products/humidity-temperature/humidity-temperature-sensor-shtc1/, accessed: 2015-06-24.

[137] S. Devarakonda, P. Sevusu, H. Liu, R. Liu, L. Iftode, and B. Nath, "Real-time air quality monitoring through mobile sensing in metropolitan areas," in *UrbComp*, 2013, pp. 15:1–15:8.

[138] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices," in *SIGCOMM*, 2009, pp. 267–278.

[139] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[140] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[141] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review*, vol. 52, no. 3, pp. 471–501, 2010.

[142] W. Feller, *An introduction to probability theory and its applications*. John Wiley & Sons, 2008, vol. 2.

[143] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. CRC press, 2013.

[144] W. M. Bolstad, *Introduction to Bayesian statistics*. John Wiley & Sons, 2007.

[145] H. Jeffreys, "An invariant form for the prior probability in estimation problems," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 186, no. 1007, pp. 453–461, 1946.

[146] S. Chakraborty, J. Zhou, V. Balasubramanian, S. Panchanathan, I. Davidson, and J. Ye, "Active matrix completion," in *ICDM*, 2013, pp. 81–90.

[147] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[148] J. E. Dobson and P. F. Fisher, "Geoslavery," *IEEE Technology and Society Magazine*, vol. 22, no. 1, pp. 47–52, 2003.

[149] K. G. Shin, X. Ju, Z. Chen, and X. Hu, "Privacy protection for users of location-based services," *IEEE Wireless Communications*, vol. 19, no. 1, pp. 30–39, 2012.

[150] I. Krontiris and T. Dimitriou, "Privacy-respecting discovery of data providers in crowd-sensing applications," in *DCOSS*, 2013, pp. 249–257.

[151] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *MDM*, 2014, pp. 73–82.

[152] I. J. Vergara-Laurens, D. Mendez, and M. A. Labrador, "Privacy, quality of information, and energy consumption in participatory sensing systems," in *PerCom*, 2014, pp. 199–207.

[153] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.

[154] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting location privacy: optimal strategy against localization attacks," in *CCS*, 2012, pp. 617–627.

[155] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[156] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi, "Broadening the scope of differential privacy using metrics." in *Privacy Enhancing Technologies*. Springer, 2013, pp. 82–102.

[157] G. Quer, R. Masiero, G. Pillonetto, M. Rossi, and M. Zorzi, "Sensing, compression, and recovery for WSNs: Sparse signal modeling and monitoring framework," *IEEE Transactions on Wireless Communications*, vol. 11, no. 10, pp. 3447–3461, 2012.

[158] L. Caccetta and R. Häggkvist, "On diameter critical graphs," *Discrete Mathematics*, vol. 28, no. 3, pp. 223–229, 1979.

[159] P. Erdős, A. Rényi, and V. Sós, "On a problem of graph theory," *Studia Sci. Math. Hungar*, vol. 1, pp. 215–235, 1966.

[160] S. Agrawal and J. R. Haritsa, "A framework for high-accuracy privacy-preserving mining," in *ICDE*, 2005, pp. 193–204.

[161] J. Shang, Y. Zheng, W. Tong, E. Chang, and Y. Yu, "Inferring gas consumption and pollution emission of vehicles throughout a city," in *KDD*, 2014, pp. 1027–1036.

[162] A. Roth and T. Roughgarden, "Interactive privacy via the median mechanism," in *STOC*, 2010, pp. 765–774.

[163] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *SIGMOD*, 2011, pp. 193–204.

[164] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

Résumé de la thèse en français

La collecte participative des données mobiles est un nouveau paradigme dédié aux applications de détection urbaines utilisant une foule de participants munis de téléphones intelligents. Pour mener à bien les tâches de collecte participative des données mobiles, diverses préoccupations relatives aux participants et aux organisateurs doivent être soigneusement prises en considération. Pour les participants, la principale préoccupation porte sur la consommation d'énergie, le coût des données mobiles, etc. Pour les organisateurs, la qualité des données et le budget sont les deux préoccupations essentielles. Dans cette thèse, deux mécanismes de collecte participative des données mobiles sont proposés : le téléchargement montant collaboratif des données et la collecte clairsemée des données mobiles. Pour le téléchargement montant collaboratif des données, deux procédés sont proposés 1) « effSense », qui fournit la meilleure solution permettant d'économiser la consommation d'énergie aux participants ayant un débit suffisant, et de réduire le coût des communications mobiles aux participants ayant un débit limité; 2) « ecoSense », qui permet de réduire le remboursement incitatif par les organisateurs des frais associés au coût des données mobiles des participants. Dans la collecte clairsemée des données mobiles, les corrélations spatiales et temporelles entre les données détectées sont exploitées pour réduire de manière significative le nombre de tâches allouées et, par conséquent, le budget associé aux organisateurs, tout en assurant la qualité des données. De plus, l'intimité différentielle est afin de répondre au besoin de préservation de la localisation des participants.

Facilitation de la collecte participative des données mobiles (MCS : Mobile Crowd Sensing) du point de vue des organisateurs et des participants

Avec la prévalence des téléphones intelligents (smartphones) équipés de capteurs, au cours des dernières années, la collecte participative des données mobiles (MCS) devient un paradigme prometteur pour favoriser les applications de détection urbaine, telles que la surveillance de l'environnement et la détection de la congestion du trafic. Grâce à cette collecte participative, MCS atteint l'objectif de détection urbaine en tirant parti de la mobilité des utilisateurs mobiles, des capteurs intégrés dans les téléphones mobiles et de l'infrastructure sans fil existante. Par rapport aux paradigmes de détection urbains traditionnels reposant sur les infrastructures spécialisées coûteuses, MCS peut couvrir la détection dans de grands espaces urbains, de manière efficace et à un coût réduit.

Pour les participants et les organisateurs, dans le processus MCS, il existe une variété de préoccupations relatives à l'obtention, par une tâche MCS, d'une quantité suffisante de résultats de détection satisfaisants. Pour les participants, ces préoccupations comprennent la consommation d'énergie des téléphones intelligents (Smartphones), le coût des données mobiles, la vie privée, et les incitations qui influencent de manière significative la volonté des participants à contribuer à une tâche MCS. Pour les organisateurs, la qualité et le budget sont les deux préoccupations principales qui, cependant, ont des conflits intrinsèques. Par exemple, pour parvenir à une meilleure qualité de la tâche, cela nécessite plus de budget; ainsi, trouver le compromis entre la qualité et le budget est une question vitale pour que les organisateurs puissent effectuer des tâches MCS satisfaisantes. La problématique de traitement approprié des préoccupations des participants et des organisateurs, au cours du processus MCS, a engendré un intérêt croissant d'une grande communauté de recherche de nos jours.

Cette thèse a pour objectif de répondre aux préoccupations des participants et des organisateurs en proposant deux catégories de mécanismes pour l'exécution des tâches MCS. La première catégorie porte sur le **téléchargement montant collaboratif des données** *(collaborative data uploading*), dans lequel les participants s'associent mutuellement, grâce à des rencontres opportunistes, dans le processus de téléchargement des données, afin d'économiser la consommation d'énergie, le coût des données mobiles, etc.

La deuxième catégorie porte sur **la collecte clairsemée des données mobiles** (*sparse mobile crowdsensing*). Afin de réduire les coûts de détection, tels que l'énergie et la motivation, tout en conservant une qualité satisfaisante des données, nous proposons ce mécanisme permettant la sélection intelligente d'une petite partie de la zone cible pour la détection, tout en inférant les données de la zone non contrôlée restante avec une grande précision.

Le **téléchargement montant collaboratif des données** regroupe deux contributions : ***effSense*** et ***ecoSense***.

***effSense*: Energy-efficient and Cost-effective Collaborative Data Uploading**

La consommation d'énergie et le coût des données sont deux préoccupations importantes pour les participants MCS. Les chercheurs ont développé plusieurs approches pour réduire l'énergie et/ou le coût des données afin d'attirer les adhésions à la collecte participative. Les travaux existants supposent essentiellement que les données détectées doivent être envoyées à un serveur central dès qu'elles sont générées. En fait, certaines tâches de collecte de données mobiles n'exigent pas nécessairement le téléchargement des données détectées en temps réel. Pour les tâches qui n'exigent pas un téléchargement des données captées en temps réel (*delay-tolerant* crowdsensing task), nous avons conçu un procédé de téléchargement collaboratif de données (désigné par ***effSense***) tirant parti des réseaux hétérogènes (par exemple, 3G, WiFi et Bluetooth) et des collaborations d'utilisateurs (relais de données), afin de permettre (1) aux utilisateurs sans plan de données ou n'étant pas prêts à utiliser leur plan de données pour les tâches de collecte (appelés utilisateurs sans plan de données) pour réduire le coût des données en transmettant des données à une passerelle Bluetooth ou d'autres téléphones mobiles rencontrés (plutôt que via le réseau 3G), et (2) aux utilisateurs disposant d'un plan de données (appelés utilisateurs avec plan de données) à consommer moins d'énergie lors du téléchargement de données.

En classifiant les utilisateurs mobiles en deux groupes, avec différents objectifs d'optimisation: utilisateurs sans plan de données (réduction des coûts de données) par rapport aux utilisateurs avec plan de données (réduction de la consommation d'énergie), nous proposons de modifier le modèle de téléchargement de données à temps réel pour disposer d'un certain délai avec des cycles de téléchargement fixes. De cette façon, le téléchargement de données captées au cours des tâches de collecte devient tolérant au délai et les données ne doivent être envoyées au serveur central qu'avant la fin de chaque cycle de téléchargement (plutôt qu'immédiatement après la production). Comme un certain délai est autorisé, les participants mobiles pourraient trouver d'autres participants ou des équipements/réseaux moins coûteux pour relayer les données détectées avant la fin de chaque cycle de téléchargement, de sorte que le coût des données ou la consommation d'énergie soient préservés. Plus précisément, ***effSense*** procure à chaque appareil mobile un système réparti de téléchargement/relais de données lui permettant de décider quand et comment effectuer le téléchargement des données captées, afin de réduire le coût des données pour les utilisateurs sans plan de données et la consommation d'énergie pour les utilisateurs avec plan de données.

effSense est conçu sur la base des observations suivantes:

1) Les utilisateurs sans plan de données peuvent supprimer le coût de téléchargement des données mobiles en utilisant des réseaux sans coût tels que Bluetooth et WiFi. Par exemple, ils peuvent télécharger des données directement vers le serveur, via WiFi, ou les transférer, via Bluetooth, vers un autre appareil en mesure de transmettre des données au serveur sans encourir de frais supplémentaires.

2) Les utilisateurs avec plan de données peuvent réduire la consommation d'énergie dans le téléchargement grâce à des méthodes économes en énergie plutôt que l'établissement d'une nouvelle connexion 3G. Par exemple, la superposition d'une tâche de téléchargement de données sur un appel vocal 3G permet d'économiser 75-90% de

consommation d'énergie. D'autre part, le téléchargement des données via WiFi ou Bluetooth consomme moins d'énergie que via 3G.

Les fonctions clés ayant servi à la conception d'***effSense*** incluent:

1) Identification et prédiction les événements critiques : les événements critiques typiques incluent l'appel vocal, la rencontre avec un autre utilisateur, le passage par une passerelle Bluetooth, la connexion à un point d'accés WiFi, etc. La prédiction des événements critiques pour chaque utilisateur constitue la base d'***effSense*** dans la sélection de la meilleure stratégie de relais des données.

2) Estimation de la consommation d'énergie associée à chaque événement critique : Dans le téléchargement, la consommation d'énergie n'est pas toujours proportionnelle à la taille des données. Par exemple, le téléchargement d'un paquet de données inférieur à 10Ko via 3G consomme environ 12 joules, quelle que soit la taille exacte des données. Conformément à la littérature existante sur la consommation d'énergie des téléphones mobiles, nous avons estimé la consommation d'énergie de divers événements critiques pour différentes tailles de données.

3) Conception d'algorithmes temps réel permettant de décider du téléchargement ou de la conservation des données, à chaque événement. Les algorithmes doivent être légers et exécutés sur chaque téléphone sans engendrer de consommation d'énergie ou de coût de données importants.

En résumé, ***effSense*** apporte les contributions suivantes:

1) A notre connaissance, ceci est le premier travail qui vise à réduire, à la fois, la consommation d'énergie et le coût des données mobiles dans les tâches de collecte des données mobiles en tirant parti des réseaux hétérogènes et des mécanismes de tolérance au délai.

2) Nous avons considéré deux types d'utilisateurs (sans plan et avec plan de données) avec des objectifs différents et proposons une structure de collaboration comprenant deux systèmes de téléchargement de données pour chaque type d'utilisateurs, respectivement: l'un est purement gourmand, l'autre est basé sur les prédictions de mobilité/d'appel. Alors que le premier est très efficace dans le traitement du problème à froid lorsque les historiques d'appels ou les journaux de mobilité des participants ne sont pas disponibles, le second peut atteindre une meilleure performance en tirant parti de la prévision des événements critiques selon les journaux historiques des participants.

3) Nous avons évalué ***effSense*** avec deux bases de données du monde réel - MIT Reality Mining et Nodobo. Les résultats montrent que, par rapport aux méthodes traditionnelles, le volume de téléchargement augmente d'environ 48-52% pour les utilisateurs sans plan des données, sans aucun coût supplémentaire et que la consommation d'énergie est réduite de 55 à 65% pour les utilisateurs avec plan des données.

***ecoSense*: Minimisation des coûts de données grâce au téléchargement collaboratif des données (Collaborative Data Uploading)**

Dans MCS, le remboursement incitatif par les organisateurs des frais associés pour couvrir le coût de téléchargement des données 3G est une stratégie de marketing efficace pour compenser le problème du coût des données des participants. Comme le remboursement du coût des données 3G devrait augmenter le budget d'incitation des organisateurs, en particulier pour les tâches qui ont besoin d'un grand nombre de participants, la réduction du budget de remboursement devient ainsi un problème critique pour les organisateurs. Pour résoudre ce problème, nous avons d'abord étudié les grilles tarifaires des coûts de données 3G. Actuellement, deux modèles de prix sont largement utilisés par la plupart des opérateurs de télécommunications: *Plan de données illimité* (Unlimited data Plan) ou *Pay As You Go*.

Plan illimité de données (UnDP): avec le plan de données illimité, un utilisateur peut transférer un nombre illimité de données pendant une période de temps (habituellement pendant un mois). Le coût d'un plan de données illimité est fixé, par exemple à 7 $ / mois (notée $Price_u$).

Pay As You Go (PAYG): un utilisateur paie le coût des données 3G en fonction de la quantité de données transférées via 3G, par exemple $ 0.1 / Moctets (notée $Price_p$).

Avec les deux plans de prix 3G ci-dessus, la solution simple au remboursement du coût de données 3G est de choisir le bon régime de remboursement pour chaque utilisateur mobile en fonction de la quantité des données téléchargées. Bien que cette méthode d'affectation directe prenne en charge, raisonnablement bien, le téléchargeant les données en temps réel pour de nombreuses tâches MCS tolérant au délai, le budget de remboursement peut encore rester assez élevé. Dans le mécanisme de téléchargement collaboratif de données, les événements suivants peuvent être mis à profit pour réduire le coût des données 3G des participants durant le délai imparti:

1) *Réseau sans coûts* : Un participant PAYAG peut utiliser un réseau sans frais, tel que Bluetooth ou Wi-Fi (par exemple à la maison ou au bureau), pour télécharger des données détectées sur le serveur dans le délai imparti, ce qui réduit son coût de données 3G.

2) *Collaboration des utilisateur*s : Les participants UnDP peuvent aider au réacheminement des données issues des participants PAYG vers le serveur. Ceci réduit les coûts de données 3G pour les participants PAYG, sans augmenter les coûts pour les participants UnDP, en diminuant ainsi le budget de remboursement de l'organisateur.

Sur la base de ces événements, nous incorporons un nouveau module de partition dans le système de téléchargement collaboratif de données, dont le but est de répartir de façon optimale les participants en sous-groupes UnDP ou PAYG, afin de minimiser le budget de remboursement de données 3G de l'organisateur, en profitant au maximum des réseaux sans coût et des collaborations des utilisateurs. Le modèle résultant est appelé **ecoSense**. Notons la différence entre **ecoSense** et **effSense**: Dans **effSense**, un participant disposant d'un plan UnDP est prévu (par exemple, en fonction de préférences de chaque utilisateur) et on ce concentre sur la conception de stratégies d'ajout de données; tandis que dans

ecoSense, en plus de stratégies téléchargeant, un autre problème technique essentiel est de décider si on souhaite attribuer un participant au groupe UnDP ou PAYG.

Dans la conception d'**ecoSense** , deux questions importantes sont considérées:

1) Comment transférer des données lorsque deux participants PAYG se rencontrent?

Si la stratégie d'acheminement est évidente entre un participant PAYG et un participant UnDP (à savoir PAYG → UnDP), elle est beaucoup plus complexe entre deux participants PAYG et aura plus d'incidence sur le budget de remboursement de l'organisateur. Parmi toutes les stratégies possibles, la diffusion (échange permanent des données entre deux participants PAYG) est sensé générer le plus petit budget de remboursement. En effet, suite à la diffusion, si l'un ou les deux participants PAYG rencontrent un participant UnDP ou un réseau sans coût, leurs données pourront être téléchargées sans frais. Cependant, les diffusions pourraient engendrer trop redondance dans les acheminements en déchargeant rapidement les batteries de téléphones mobiles des participants. Bien que notre travail vise la réduction du budget de remboursement de données 3G de l'organisateur, l'économie d'énergie des participants devrait également être prise en compte dans une certaine mesure. Autrement, même si **ecoSense** permet de minimiser le budget de remboursement, la consommation d'énergie des téléphones pourrait être trop élevée, et **ecoSense** devient impraticable. Ainsi, pour étudier le compromis entre le budget de remboursement de données 3G de l'organisateur et la consommation d'énergie des téléphones des participants, nous avons appliqué différentes stratégies de téléchargement/ d'acheminement de données.

2) Comment décider du régime de remboursement de chaque participant - PAYG ou UnDP?

Pour minimiser le budget de remboursement de données 3G de l'organisateur, une autre question clé est de déterminer les participants qui doivent être affectés à chacun des régimes (et pas seulement le pourcentage de participants alloué à chaque régime). Ainsi, l'algorithme de partition des participants doit examiner le profil de mobilité de chaque participant et le volume de données détectées:

•*Schéma de mobilité :* Afin de maximiser les possibilités d'acheminement des données entre les participants PAYG et UnDP, un profilage précis du schéma de mobilité de chaque participant est nécessaire pour décider la catégorie à laquelle on doit l'affecter : PAYG ou UnDP. Intuitivement, les participants «actifs» qui peuvent aider d'avantage d'autres participants à acheminer leurs données doivent être affectés à la catégorie UnDP.

•*Taille des données détectées :* Le volume des données détectées d'un participant devrait également avoir une incidence selon qu'il soit affecté à PAYG ou à UnDP. Généralement, un participant qui télécharge un plus grand volume de données doit être affecté à UnDP.

L'algorithme de partition proposé prédit d'abord le profil de mobilité de chaque participant et estime le volume de données détectées puis utilise un algorithme génétique pour obtenir la répartition des participants dans les groupes PAYG et le UnDP. Cet algorithme est exécuté avant le début de chaque mois (seulement une fois par mois) sur le serveur de l'organisateur MCS.

En résumé, **ecoSense** apporte les contributions suivantes:

1) A notre connaissance, ceci est le premier travail visant à minimiser le budget de remboursement de données 3G de l'organisateur en tirant parti des réseaux hétérogènes (par exemple 3G, Bluetooth, WiFi) et des collaborations entre utilisateurs dans le MCS.

2) Nous avons proposé un modèle de téléchargement collaboratif des données, appelé **ecoSense**, permettant de minimiser budget de remboursement de données 3G de l'organisateur. **ecoSense** considère deux plans tarifaires 3G pour rembourser les participants - Pay As You Go et la formule illimitée - et propose des stratégies de téléchargement de données pour les participants UnDP et PAYG dans la période de téléchargement imparti. En outre, un algorithme de partition a été conçu pour affecter les participants aux groupes PAYG et UnDP, afin de minimiser le budget de remboursement de données 3G de l'organisateur, en tirant au maximum parti des réseaux sans frais et des collaborations des participants (relais de données).

3) Afin d'évaluer notre modèle, nous avons exploité un ensemble de données réelles issues des bases MIT Reality Mining et SWIM. Les résultats de l'évaluation montrent que **ecoSense** peut réduire le budget de remboursement de données 3G de l'organisateur jusqu'à -50% par rapport à la solution d'affectation directe.

La seconde catégorie de nos travaux sur **la collecte clairsemée des données mobiles** (*sparse mobile crowdsensing*) comporte deux contributions **CCS-TA** et **DUM-εe** :

CCS-TA: Allocation de tâches assurant la qualité pour la collecte clairsemée des données mobiles

Pour obtenir des résultats de collecte de qualité élevée dans les applications MCS, les travaux existants vise généralement à impliquer suffisamment de participants de sorte que leurs données peuvent couvrir la quasi totalité de la zone cible, à savoir, une couverture complète ou une couverture probabiliste élevée.

Plus précisément, en assurant la couverture de chaque sous-zone ou cellule (couverture complète) ou la plupart des cellules (couverture probabiliste) par les participants mobiles, les applications MCS tentent d'obtenir des valeurs de détection précises et fiables dans toute la zone cible, afin d'élaborer une image de détection complète qui répond aux besoins des organisateurs MCS. La relation sous-jacente entre une couverture complète (ou une couverture probabiliste élevée) et la qualité des données est que la couverture complète (ou une couverture probabiliste élevée) peut permettre aux organisateurs de MCS d'obtenir des valeurs de détection représentatives de la zone cible. En substance, pour les organisateurs de MCS, l'exigence de la qualité des données est d'obtenir une valeur de détection raisonnablement précise pour chaque sous-zone (cellule).

La couverture probabiliste complète ou élevée est certes une approche simple pour obtenir des données représentatives dans chaque cellule, mais son inconvénient est que quelque soit le mécanisme d'allocation efficace des tâches utilisé, l'organisateur MCS doit allouer les tâches à presque toutes les cellules, au moins une fois. Cela peut encore engendrer un budget incitatif élevé, en particulier lorsque l'organisateur prévoit de réaliser diverses campagnes MCS à grande échelle.

Afin de réduire davantage le nombre de participants, une autre approche consiste à sélectionner seulement quelques cellules pour la détection de données, tout en déduisant les données associées au reste des cellules de la zone ciblée avec une bonne précision. Cette méthode alternative est en fait réalisable puisque certains types de données environnementales, telles que la température et le bruit, se sont avérées utiles dans le processus d'inférence de bonne qualité en raison des fortes corrélations temporelles et spatiales entre ces données.

Nous proposons alors d'utiliser comme métrique de qualité de données la précision générale des données détectées plutôt que la couverture de la zone de détection. Ainsi, on exploite les corrélations temporelles et spatiales entre les données détectées pour en déduire les données manquantes des cellules non contrôlées. En sélectionnant activement un nombre minimal de cellules pour la répartition des tâches, nous tentons de minimiser le nombre de participants tout en assurant la précision globale de données. Nous nommons « **collecte clairsemée des données mobiles** (Sparse mobile Crowdsensing)», ce nouveau paradigme qui applique des algorithmes d'inférence pour élaborer la couverture complète à partir de la couverture partielle des données détectées partielles.

L'idée de base peut être illustrée par le cas d'utilisation suivant: un organisateur MCS lance une tâche de surveillance de la température de l'environnement dans une zone urbaine cible, qui a déjà été décomposée en cellules selon les exigences de l'organisateur. L'organisateur doit mettre à jour la carte de détection de température complète, une fois toutes les heures (cycle de détection), et dans chaque cycle de détection, l'exigence de qualité des données est que l'erreur absolue moyenne pour toute la zone doit être inférieure à 0,25°C. Dans chaque cycle de détection, pour répondre à cette exigence de qualité des données tout en minimisant le nombre des tâches attribuées, l'organisateur procède à une sélection active d'un sous-ensemble de cellules à solliciter pour la détection physique, à savoir la répartition des tâches aux participants à ces cellules sélectionnées. Sur la base des valeurs de température détectées dans les cellules sélectionnées, les valeurs de température des cellules restantes sont inférées.

Pour accomplir la tâche de **collecte clairsemée des données mobiles,** décrite dans le cas d'utilisation ci-dessus, nous devons aborder les deux défis suivants:

1) Combien et quelles cellules doivent être choisies pour l'allocation des tâches? Dans chaque cycle, afin de minimiser le nombre de tâches assignées, tout en assurant la qualité des données, l'organisateur doit sélectionner un sous-ensemble minimal de cellules pour l'allocation des tâches. Afin de trouver cette combinaison minimale de cellules, nous avons besoin d'identifier les cellules pertinentes dont les valeurs détectées, peuvent servir, au mieux, à déduire les valeurs des autres cellules. Cependant, l'identification des cellules pertinentes d'une manière directe n'est pas triviale car, sans connaître la vraie valeur détectée dans une cellule, il est difficile de prévoir combien cette valeur peut aider à améliorer la précision des données.

2) Comment mesurer quantitativement et estimer la qualité des données en ligne à travers une tâche MCS sans connaître les vraies valeurs de détection des cellules non contrôlées?

Étant donné que les vraies valeurs de détection des cellules non contrôlées sont inconnues, nous ne pouvons pas mesurer l'exactitude des données directement en comparant les

données inférées avec la réalité du terrain inconnu. Nous avons donc besoin d'une méthode pour estimer efficacement une telle précision des données de détection en ligne dans chaque cycle de détection. En outre, comme la précision des données estimées a intrinsèquement un certain écart de la précision réelle, il est pratiquement difficile de garantir strictement que tous les cycles de détection satisfassent l'erreur limite prédéfinie. Par conséquent, nous avons besoin de trouver un moyen pratique pour définir l'exigence de qualité des données associée à une tâche MCS au lieu de régler simplement la limite d'erreur pour chaque cycle de détection.

Selon les objectifs et les défis de recherche mentionnés ci-dessus, les principales contributions de ce travail sont:

1) Nous avons défini une nouvelle métrique pour évaluer la qualité d'une tâche MCS, appelé ***(ε,p)-qualité***, qui considère non seulement l'erreur limite nécessaire ε mais aussi la fraction p de cycles de détection des données de détection qui doivent répondre à cette limite ε. Ainsi, nous proposons d'exploiter les corrélations temporelles et spatiales entre les valeurs de détection issues de cellules différentes afin de réduire de manière significative le nombre de tâches attribuées. A notre connaissance, ceci est la première contribution dans le MCS qui tente de réduire le nombre de cellules sollicitées dans la détection en sélectionnant intelligemment les meilleures cellules à détecter permettant de déduire les valeurs manquantes des cellules restantes, afin de garantir que la précision globale des données est conforme à l'exigence de qualité prédéfinie.

2) Afin de réduire le nombre de cellules de détection requis pour la répartition des tâches dans chaque cycle de détection, tout en ciblant la qualité prédéfinie ***(ε, p)***, nous proposons une architecture de répartition des tâches en ligne à deux phases, appelé **CCS-TA (Allocation de compression CrowdSensing Task)**. Dans la première phase, nous determinons quelle est la meilleure cellule à ajouter pour la détection dans chaque cycle selon des techniques d'apprentissage actives. Après avoir recueilli les données de détectées dans les cellules sélectionnées, nous appliquons ensuite , dans la deuxième phase, une méthode basée sur l'inférence bayésienne pour estimer la précision globale de données en ligne après avoir inféré les données des cellules restantes en utilisant la détection compressive. Sur la base de la précision globale des données estimée, CCS-TA détermine si d'avantage de cellules doivent être sélectionnés pour la répartition des tâches afin d'assurer la qualité prédéfinie ***(ε, p)***.

3) Pour montrer l'efficacité du CCS-TA, nous avons effectué des évaluations approfondies sur des bases de données réelles de surveillance de température et de qualité de l'air. Dans le cas de la surveillance de la température, on constate que CCS-TA a besoin d'attribuer des tâches à seulement 15,5% des cellules en moyenne, ce qui peut assurer une erreur de détection globale inférieure à 0.25◦C dans 95% des cycles, donc satisfaisant la qualité (0.25◦C, 0,95). A titre de comparaison, les approches classiques ont besoin d'allouer plus de 18,0 à 26,5% de tâches afin de garantir le même niveau de qualité.

***DUM-εe*: Protection de l'intimité différentielle pour la collecte clairsemée des données mobiles**

Dans les applications de collecte clairsemée MCS, les participants communiquent non seulement les données détectées, mais aussi leur emplacement et le temps correspondants. Cela peut avoir des conséquences graves sur la vie privée des participants MCS. Connaissant l'emplacement de chaque participant, un adversaire, qui veut exploiter des informations sur le participant, peut organiser un large spectre d'attaques, telles que la surveillance physique, le harcèlement, le vol d'identité, et la violation d'informations sensibles. Ainsi, la confidentialité de la localisation est un aspect essentiel du MCS, puisque les utilisateurs mobiles n'accepteront pas participer à une tâche MCS si leur vie privée est menacée.

La préservation de la vie privée a été largement abordée dans le contexte des systèmes basés sur la localisation. Il existe deux mécanismes généraux pour protéger l'emplacement de la vie privée d'un utilisateur: (i) protéger l'identité de l'utilisateur à travers l'anonymat, de sorte que les traces de localisation ne puissent pas être associés à un utilisateur particulier, et (ii) modifier la localisation réelle de l'utilisateur, à l'aide des mécanismes d'obscurcissement afin de réduire les informations de localisation livrées au fournisseur de services. Dans ce travail, nous nous sommes concentrés sur ce dernier mécanisme de protection de la vie privée.

Quantité de travaux ont été menées sur la protection de la vie privée avec diverses techniques d'obscurcissement. Ces mécanismes de protection visent soit à cacher soit à perturber l'emplacement réel de l'utilisateur en augmentant l'incertitude de sa localisation par un intrus. Le mécanisme le plus populaire est la masquage (cloaking), où l'emplacement de l'utilisateur est représenté sous la forme d'une région masquée (contenant de multiples cellules très réduites) au lieu d'un lieu ou une cellule bien déterminé. Habituellement, le nombre de cellules réduites dans la région masquée est utilisé pour mesurer le niveau de protection de la vie privée. Cependant, un défaut commun des mécanismes de masquage est qu'ils sont sensibles à la connaissance préalable de l'adversaire sur la distribution de l'emplacement de l'utilisateur. Par exemple, si un utilisateur apparaît dans une région masquée contenant une école et un bureau gouvernemental, et l'adversaire sait à l'avance que l'utilisateur est susceptible d'être à l'école (par exemple, l'utilisateur est un étudiant) ; par conséquent, l'adversaire aura élevé une forte présomption que l'utilisateur se trouve à l'école dans le domaine masqué. Ce qui porte atteinte à la protection recherchée par le masquage.

L'intimité différentielle a été proposée afin de remédier à cette lacune due au mécanisme d'obscurcissement relative à la connaissance préalable de l'adversaire. Dans le domaine des bases de données statistiques, son objectif initial est de protéger les données d'un individu lors de la publication des données agrégées (par exemple, comptage et somme) à partir de la base de données. L'intimité différentielle spécifie que la modification des données d'un utilisateur unique aura un effet négligeable sur le résultat de la requête. Cela se fait généralement par ajout d'une quantité contrôlée d'un bruit aléatoire en sortie de la requête. Supposons qu'un adversaire tente de trouver la valeur d'un certain attribut sur un utilisateur (par exemple, l'âge) dans une base de données. Même si l'adversaire dispose déjà des résultats de la requête grâce à l'agrégation des autres valeurs d'attribut, il ne peut pas gagner plus de connaissances sur la valeur d'attribut à partir du résultat de la requête perturbée, quelque soit la connaissance préalable que l'adversaire détient.

En appliquant l'intimité différentielle aux services basés sur la localisation (LBS), Andres et al. ont proposé la géo-indiscernabilité, ce qui donne à un utilisateur *l*-intimité dans une zone circulaire *R* avec un certain rayon. La probabilité de reporter la même localisation obscurcie r_0 à partir de deux localisations réelles dans *R* est similaire (le degré de similitude dépend de *l*). Ainsi, après avoir observé la localisation obscurcie r_0 d'un utilisateur, l'adversaire gagne peu de connaissances supplémentaires sur la localisation au sein de *R* qui produit r_0, indépendamment de la connaissance préalable de l'adversaire sur la distribution de l'emplacement de l'utilisateur. Pour protéger l'intimité de la localisation tout en maintenant la qualité de service dans LBS (telles que les requêtes sur les centres d'intérêt), les deux mécanismes de dissimulation et d'intimité différentielle supposent que la distance entre l'emplacement réel de l'utilisateur et l'emplacement obscurci est petite. Cette hypothèse est bien adaptée pour les LBS car la qualité de la sortie d'une requête basée sur la localisation préservant la vie privée est généralement dégradée avec l'augmentation de la distance entre les localisations obscurcie et réelle.

Cependant, dans a collecte clairsemée MCS, la perte de la qualité des données est affectée par la différence entre les données détectées aux deux emplacements réel et obscurci, en plus de la distance entre ces deux emplacements. Afin de limiter la perte de qualité de données, lorsque les valeurs de détection des deux emplacements se trouvent à proximité, l'emplacement du participant peut être cartographié à un endroit éloigné de l'emplacement réel. Par exemple, lors de la détection de la qualité de l'air dans une ville, deux parcs peuvent avoir des valeurs similaires de qualité de l'air. Si un participant est dans un parc, l'obscurcissement de son emplacement à un autre parc subirait peu de perte de la qualité des données, même si les deux parcs sont éloignés l'un de autre. En raison de cette distinction entre les qualités de LBS et la collecte clairsemée MCS, au lieu d'utiliser directement les mécanismes de LBS, nous avons besoin de reconcevoir les mécanismes d'obscurcissement utilisés dans la collecte clairsemée MCS pour protéger l'intimité de l'emplacement, tout en assurant la qualité des données.

Avec ce concept à l'esprit, nous avons exploré la façon d'équilibrer l'intimité de l'emplacement pour les participants et la qualité globale des données obtenues pour les applications MCS clairsemées. Nous considérons trois éléments clés dans le mécanisme de conception pour la préservation de l'intimité de localisation: les exigences en matière de protection des renseignements personnels du participant, la connaissance préalable de l'adversaire sur la distribution de la position réelle du participant, et la dégradation de la qualité des données provenant de l'obscurcissement des emplacements réels. Nos principales contributions sont les suivantes:

1. Apporter l'intimité différentielle de localisation en introduisant la notion de préservation de vie privée *ε-region-ambiguïté* , afin de réduire ce qu'un adversaire peut apprendre des participants indépendamment de sa connaissance préalable sur la distribution de l'emplacement des participants.

2. Répondre simultanément aux deux objectifs de localisation et d'obscurcissement pour la qualité des données en modélisant l'emplacement sensible à la qualité des données avec l'exigence de préservation de vie privée comme problématique d'optimisation. Nous avons alors proposé un nouveau programme linéaire appelé DUM-εe, qui sélectionne la fonction optimale d'obscurcissement de l'emplacement et réduit la perte de qualité des

données grâce à la Minimisation de l'incertitude des données sous les contraintes de ε-region-ambiguïté et une répartition uniforme des emplacements brouillées. Ainsi, la fonction d'obscurcissement optimale obtenue permet d'assurer l'intimité différentielle de localisation avec un taux de perte considérablement réduit de la qualité des données.

3. Réduire le nombre de contraintes dans DUM-εe de $O(n^3)$ à $O(n^2)$ en proposant une programme linéaire d'approximation rapide (FDUM-εe). Comme le nombre de contraintes affecte à la fois le temps et la complexité de l'espace des techniques de programmation linéaire, FDUM-εe nécessite beaucoup moins de temps de calcul et d'utilisation d'espace de stockage que DUM-εe. Par conséquent, il peut être appliqué dans les tâches de MCS à grande échelle que DUM-εe ne peut pas gérer.

Au mieux de notre connaissance, ce travail est le premier à mettre en œuvre l'intimité différentielle de localisation dans MCS, tout en réduisant la perte de la qualité des données due à l'obscurcissement. Nous avons évalué expérimentalement nos programmes d'optimisation, DUM-εe et FDUM-εe, en utilisant des bases de données du monde réel sur l'environnement (température et humidité) et la surveillance du trafic. Nos résultats montrent que, par rapport aux trois approches de base, DUM-εe peut réduire de 15 à 45% la perte de la qualité des données, en garantissant le même niveau d'intimité différentielle. Comparé à DUM-εe, FDUM-εe peut atteindre une qualité de données similaire (2 à 6% de plus de perte de qualité), alors qu'il nécessite moins de 1% de temps de calcul pour générer la fonction d'obscurcissement.

Enfin, nous avons résumé les apports en termes de téléchargement collaboratif et de collecte clairsemée, et discuté des futures perspectives de recherche sur:

- la façon d'améliorer nos mécanismes pour faire face aux comportements malveillants des participants
- l'adaptation de nos mécanismes à des applications plus innovantes dans les scénarios de la ville intelligente,
- l'intégration de toutes les techniques proposées dans cette thèse dans une plate-forme de MCS unifiée.