Near-Optimal Mobile Crowdsensing: Design Framework and Algorithms

Haoyi Xiong

THESE DE DOCTORAT TELECOM SUDPARIS - INSTITUT MINES-TELECOM
EN CO-ACCREDIATION AVEC L'UNIVERSITE PIERRE ET MARIE CURIE - PARIS 6

Spécialité: Informatique

Ecole doctorale informatique, télécommunications et électronique

Présentée par

Haoyi Xiong

Pour obtenir le grade de
DOCTEUR DE TSP-UPMC

Quasi-Optimal Mobile Crowdsensing: Cadre de Conception et Algorithmes

Soutenue le 22/Jan/2015

devant le jury composé de :

Directeur de thèse :

Mme.	Monique Becker	Professeur HDR, TELECOM SudParis, France

Rapporteurs :

M.	Hervé Rivano	Chargé de Recherches HDR, INRIA, France
M.	Stéphane Galland	Professeur HDR, UTBM , France

Examinateurs :

M.	Pierre Sens	Professeur HDR, Université de Paris 6, France
Mme.	Véronique Vecque	Professeur HDR, Supélec, France
M.	Steven Martin	Professeur HDR, Université de Paris 11, France
Mme.	Lila Boukhatem	Maître de conférences HDR, Université de Paris 11, France (Invité)
M.	Daqing Zhang	Professeur, TELECOM SudParis, France (Encadrant)
M.	Vincent Gauthier	Maître de conférences, TELECOM SudParis, France (Encadrant)

Thèse numéro : 2015n°2015TELE0005

Doctor of Philosophy (Ph.D) Thesis
TELECOM SudParis & University of Paris 6

Specialization: Computer Science and Engineering

Presented by

Haoyi Xiong

Submitted for the partial requirement of
Doctor of Philosophy
from
TSP-UPMC

Near-Optimal Mobile Crowdsensing: Design Framework and Algorithms

Defense at 22/Jan/2015

Jury Members :

Director of Thesis :

Ms.	Monique Becker	Professeur HDR, TELECOM SudParis, France

Reporters :

M.	Hervé Rivano	Chargé de Recherches HDR, INRIA, France
M.	Stéphane Galland	Professeur HDR, UTBM , France

Examiners :

M.	Pierre Sens	Professeur HDR, Université de Paris 6, France
Mme.	Véronique Vecque	Professeur HDR, Supélec, France
M.	Steven Martin	Professeur HDR, Université de Paris 11, France
Mme.	Lila Boukhatem	Maître de conférences HDR, Université de Paris 11, France (Invité)
Mr.	Daqing Zhang	Professeur, TELECOM SudParis, France (Encadrant)
Mr.	Vincent Gauthier	Maître de conférences, TELECOM SudParis, France (Encadrant)

Thèse numéro : 2015n°2015TELE0005

Abstract

Nowadays, there is an increasing demand to provide real-time environment information such as air quality, noise level, traffic condition, etc. to citizens in urban areas for various purposes. The proliferation of sensor-equipped smartphones and the mobility of people are making *Mobile Crowdsensing* (MCS) an effective way to sense and collect information at a low deployment cost. In MCS, instead of deploying static sensors in urban areas, people with mobile devices play the role of mobile sensors to sense the information of their surroundings and the communication network (3G, WiFi, etc.) is used to transfer data for MCS applications.

Typically, an MCS application (or task) not only requires each participant's mobile device to possess the capability of *receiving sensing tasks, performing sensing and returning sensed results* to a central server, it also requires to *recruit participants*, *assign sensing tasks* to participants, and *collect sensed results* that well represents the characteristics of the target sensing region. In order to recruit sufficient participants, the organizer of the MCS task should consider *energy consumption* caused by MCS applications for each individual participant and the *privacy issues*, further the organizer should give each participant a certain amount of *incentives* as encouragement. Further, in order to collect sensed results well representing the target region, the organizer needs to ensure the *sensing data quality* of the sensed results, e.g., the accuracy and the spatial-temporal coverage of the sensed results.

With the *energy consumption*, *privacy*, *incentives*, and *sensing data quality* in mind, in this thesis we have studied four optimization problems of mobile crowdsensing and conducted following four research works:

- **EEMC -** In this work, the MCS task is splitted into a sequence of sensing cycles, we assume each participant is given an equal amount of incentive for joining in each sensing cycle; further, given the target region of the MCS task, the MCS task aims at collecting an expected number of sensed results from the target region in each sensing cycle. Thus, in order to minimize the total incentive payments and the total energy consumption of the MCS task while meeting the predefined data collection goal, we propose **EEMC** which intends to select a minimal number of *anonymous participants* to join in each sensing cycle of the MCS task while ensuring an minimum number of participants returning sensed results.

- **EMC^3 -** In this work, we follow the same sensing cycles and incentives assumptions/settings from **EEMC**; however, given a target region consisting of a set of subareas, the MCS task in this work aims at collecting sensed results covering each subarea of the target region in each sensing cycle (namely *full coverage* constraint). Thus, in order to minimize the total incentive payments and the total energy consumption of the MCS task under the full coverage constraint, we propose **EMC^3** which intends to select a minimal number of *anonymous*

participants to join in each sensing cycle of the MCS task while ensuring at least one participant returning sensed results from each subarea.

- **CrowdRecruiter -** In this work, we assume each participant is given an equal amount of incentive for joining in *all* sensing cycles of the MCS task; further, given a target region consisting of a set of subareas, the MCS task aims at collecting sensed results from a predefined percentage of subareas in each sensing cycle (namely *probabilistic coverage* constraint). Thus, in order to minimize the total incentive payments the probabilistic coverage constraint, we propose **CrowdRecruiter** which intends to recruit a minimal number of participants for the whole MCS task while ensuring the selected participants returning sensed results from at least a predefined percentage of subareas in each sensing cycle.

- **CrowdTasker -** In this work, we assume each participant is given a varied amount of incentives, according to the number of sensing cycles that the participant joins in; further we define a novel sensing data quality metrics based on both the number of subareas covered by sensed results and the number of sensed results in each subarea (namely *overall coverage quality*). Thus, in order to maximize the overall coverage quality with a fixed amount of budget for incentive payment, we propose **CrowdTasker** which intends to optimally recruit a set of participants and determine in which sensing cycles each selected participant can join in the MCS task while ensuring the total incentive payment not exceeding the budget.

Each above work intends to study one practical optimization problem of mobile crowdsensing with specific *incentive*, *energy consumption*, *privacy* and *sensing data quality* settings/objectives. Evaluations with a large-scale real-world dataset show that our proposed *EEMC* EMC^3, CrowdRecruiter and CrowdTasker outperform heuristic methods and other baseline approaches.

Résumé

Aujourd'hui, il y a une demande croissante de fournir les informations d'environnement en temps réel tels que la qualité de l'air, le niveau de bruit, état du trafic, etc. pour les citoyens dans les zones urbaines à des fins diverses. La prolifération des capteurs de smartphones et la mobilité de la population font des Mobile Crowdsensing (MCS) un moyen efficace de détecter et de recueillir des informations à un coût faible de déploiement. En MCS, au lieu de déployer capteurs statiques dans les zones urbaines, les utilisateurs avec des périphériques mobiles jouent le rôle des capteurs de mobiles à capturer les informations de leurs environement, et le réseau de communication (3G, WiFi , etc.) pour le transfert des données pour MCS applications.

En général, l'application MCS (ou tâche) non seulement exige que chaque participant de périphérique mobile de posséder la capacité de réception missions de télédétection, de télédétection et de renvoi détecté résultats vers un serveur central, il exige également de recruter des participants, attribuer de télédétection tâches aux participants, et collecter les résultats obtenues par télédétection ainsi que représente les caractéristiques de la cible zone de détection . Afin de recruter un nombre suffisant de participants, l'organisateur d'une MCS tâche devrait considérer la consommation énergétique causée par MCS applications pour chaque participant et les questions de protection dans la vie privée, l'organisateur doit donner à chaque participant un certain montant des incitations comme un encouragement. En outre, afin de recueillir les résultats obtenues par télédétection et représentant la région cible, l'organisateur doit s'assurer que les données de télédétection qualité des résultats obtenues par télédétection, p. ex., la précision et la spatio-temporelle la couverture des résultats obtenues par télédétection.

Avec la consommation d'énergie, la protection de la vie privée, les mesures d'incitation, de télédétection et qualité des données à l'esprit, dans cette thèse nous avons étudié quatre problèmes d'optimisation de mobile crowdsensing et menées après quatre travaux de recherche:

- EEMC - dans le cadre de ce travail, la tâche deMCS est divisé en une séquence de cycles de détection, nous supposons que chaque participant est donnée une quantité égale de stimulant pour rejoindre dans chaque cycle de télédétection; de plus, étant donné la région cible du MCS tâche, la tâche de MCS vise à recueillir le nombre prévu de télédétection résultats de la région cible dans chaque cycle de télédétection. Ainsi, afin de réduire au minimum les totaux paiements d'incitation et la consommation totale d'énergie de la tâche de MCS tout en réunion les données prédéfinies collection objectif, nous proposons EEMC qui a l'intention de sélectionner un nombre minimal de participants anonymes de se joindre à chaque cycle de détection de la MCS tâche tout en assurant un nombre minimal de participants retour résultats détectée.

- EMC3 - dans le cadre de ce travail, nous avons suivi les mêmes cycles de

détection et des incitations hypothèses/paramètres de EEMC; toutefois, étant donné une région cible composée d'un ensemble de sous-zones, la tâche de MCS dans ce travail vise à collecter détecté résultats couvrant chaque sous-zone de la région cible dans chaque cycle de détection (à savoir la pleine couverture contrainte). Ainsi, afin de réduire au minimum les totaux paiements d'incitation et la consommation totale d'énergie de la tâche de MCS sous la couverture totale contrainte, nous proposons EMC3 qui a l'intention de sélectionner un nombre minimal de participants anonymes à se joindre à chaque cycle de détection du MCS tâche tout en assurant au moins un participant retour détecté les résultats de chaque sous-zone.

- CrowdRecruiter - dans le cadre de ce travail, nous supposons que chaque participant est donnée une quantité égale de stimuler pour rejoindre dans tous les cycles de détection du bac de ramassage tâche; de plus, étant donné une région cible composé d'un ensemble de sous-zones, la tâche de MCS vise à recueillir des résultats détectée par un pourcentage prédéfini de sous-zones dans chaque cycle de détection (à savoir la couverture probabiliste contrainte). Ainsi, afin de réduire les totaux paiements d'incitation la couverture probabiliste contrainte, nous proposons CrowdRecruiter qui envisage de recruter un nombre minimal de participants pour l'ensemble tâche de MCS tout en assurant les participants sélectionnés retour détecté résultats d'au moins un pourcentage prédéfini de sous-zones dans chaque cycle de télédétection.

- CrowdTasker - dans le cadre de ce travail, nous supposons que chaque participant est donnée une quantité variable d'incitations, en fonction du nombre de cycles de détection que le participant se joigne à; de plus, nous nous définir un roman de détection des données métriques de qualité repose à la fois sur le nombre de sous-zones couvertes par télédétection résultats et le nombre de résultats détectée dans chaque sous-zone (c-à-d couverture globale qualité). Ainsi, afin de maximiser la couverture globale de qualité avec un montant fixe de budget de paiement incitatif, nous proposons CrowdTasker qui a l'intention de recruter de façon optimale l'ensemble des participants et de déterminer à qui la télédétection cycles chaque participant sélectionné peut se joindre au MCS tâche tout en assurant le total paiement incitatif dépassant pas le budget.

Chaque travail ci-dessus se propose d'étudier une pratique problème d'optimisation de mobile crowdsensing avec incitation spécifiques, de la consommation d'énergie, la protection de la vie privée et des données de télédétection paramètres qualité/objectifs. Les évaluations avec une grande échelle le monde réel dataset montrent que notre projet EEMC EMC3, CrowdRecruiter CrowdTasker et surpasser les méthodes heuristiques et d'autres approches de base.

Acknowledge

To whom it may concern,

Many thanks for paying attention to my thesis. Doing a PhD is a long and wonderful journey of mine, where I moved to France and left my dad and mom, where I met my wife and turned myself from a naive boy to a (maybe still naive) husband, where I gradually turned myself from an engineer to a researcher. All these stuffs impact my life.

First of all, I gratefully thank following persons who helped me during my PhD — my supervisor Prof. Daqing Zhang, my thesis director Prof. Monique Becker, my co-advisor Prof. Vincent Gauthier, the secretary of our department Madame Francoise Abad, and my (former) colleagues including Mr. Dingqi Yang, Mr. Leye Wang, Miss Xiao Han, Prof. Chao Chen, Dr. Lin Sun, Mr. Longbiao Chen, and Dr. Xiaoping Che. All help from them is valuable and should be highly appreciated. Moreover, I would like to appreciate the collaborators of my research including Prof. Guanling Chen from University of Massachusetts at Lowell, Dr. Jie Zhu at Intel Corporation Santa Clara, and Prof. Hakima Chaouchi Prof. J. Paul Gibson at Insitut Mines-Télécom SudParis. Furthermore, I would like to appreciate the jury members of my PhD defense committee including Prof. Veronique Véque, Prof. Lila Boukhatem, Prof. Rivano Hervé, Prof. Stéphane Galland, Prof. Pierre Sens, and Prof. Steve Martin, who allocate me a slice of their busy hours so as to help me validate my PhD research.

Personally, I believe my thesis is largely contributed by my family. My father Mr. Chunjie Xiong one of the chief electrical engineers (professor-rank) at China Metallurgical Group Corporation with 30 year's experience in designing sensor systems for automated factories taught me programming when I was twelve. My Mom Madame Ming Li a certificated cost appraiser and investment control engineer taught me a lot in budget planning and how to make a decision. All their professionals brought me fundamentals of doing research in computer science and electrical engineering. More important, from our first glance at Lille Europe Gare to my defense today, my wife Sijia Yang is always by my side. From my first paper accepted to my thesis written up, She had given me whatever she could give to support my PhD research, take care of my life, and balance my life-work style. I hope we could love and be together forever.

Finally, I am pretty sure that I miss some important people I should thank. Thus, I would like to say "Thank You!" to all people surrounding me or having contacted me. It must be the LORD who leads me here or there to know you and get help from you. Thank You!

Yours faithfully,

Haoyi XIONG

Table of contents

1 Introduction **17**
- 1.1 Background . . . 17
- 1.2 Research Motivations and Contributions . . . 19
- 1.3 Organization of this Thesis . . . 23

2 State of the Arts **25**
- 2.1 MCS Applications and Frameworks . . . 25
- 2.2 MCS Energy Consumption . . . 26
- 2.3 MCS Energy-saving Strategies . . . 27
- 2.4 MCS Incentive Models . . . 27
- 2.5 MCS Sensing Data Quality Metrics . . . 28
- 2.6 MCS Participant Selection and Task Assignment . . . 28
- 2.7 Human Mobility Prediction for MCS . . . 29

3 *EEMC*: Energy Efficient Mobile Crowdsensing with Anonymous Participants **31**
- 3.1 Introduction . . . 32
  - 3.1.1 Proposed Research: Assumptions, Objectives and the Example 32
  - 3.1.2 Research Challenges and Our Contributions . . . 35
  - 3.1.3 Comparison with the Most Related Work . . . 37
- 3.2 Problem Formulation . . . 38
- 3.3 *EEMC* Framework and Skeleton Algorithm . . . 39
  - 3.3.1 Phase I - Candidate User Identification based on Call Prediction 39
  - 3.3.2 Phase II - Two-step Decision Making Process for Task Assignment 40
- 3.4 Next-Call Prediction Model based on Accumulated Call Traces . . . 42
  - 3.4.1 Probabilsitic Model of Phone Calls . . . 43
  - 3.4.2 Parameter Estimation using Accumulated Traces . . . 43
- 3.5 Adaptive Pace Controller for Task Assignment . . . 44
  - 3.5.1 Adaptive Pace Control for Task Assignment . . . 44
  - 3.5.2 Probability Estimation for Adaptive Pace Control . . . 44
- 3.6 Near-Optimal Decision Maker for Task Assignment . . . 45
  - 3.6.1 Identifying *Future-surer Candidates* . . . 45
  - 3.6.2 Estimating if the Missing Number of Sensed Results can be returned from Future-surer Candidates and Potential Returners 46
  - 3.6.3 Near-optimal Task Assignment Decision Making . . . 47
- 3.7 Experimental Setups . . . 47
  - 3.7.1 Baseline Methods and Parameter Settings . . . 47
  - 3.7.2 Dataset and Experiment Setups . . . 48
- 3.8 Evaluation Results . . . 49

3.8.1 Performance Comparison . . . 50
3.8.2 Cold-start Performance . . . 51
3.8.3 Case Study and Analysis . . . 52
3.8.4 Energy Conservation Comparison . . . 53
3.9 Discussion . . . 54

4 EMC^3: Energy Efficient Data Transfer for Mobile Crowdsensing under Full Coverage Constraint 57
4.1 Introduction . . . 58
4.2 Problem Statement . . . 61
4.3 EMC^3 Framework and Core Algorithms . . . 62
4.3.1 Call/Mobility Prediction . . . 63
4.3.2 Overall Task Assignment Pace Control . . . 64
4.3.3 Sub-optimal Task Assignment Decision Making . . . 66
4.4 Evaluation . . . 68
4.4.1 Baseline Methods and Parameter Settings . . . 69
4.4.2 Dataset and Experiment Setups . . . 69
4.4.3 Performance Evaluation . . . 72
4.4.4 Energy Conservation Comparison . . . 75
4.4.5 Real-time Performance Analysis . . . 77
4.5 Discussion . . . 79

5 CrowdRecruiter: Selecting Participants for Piggyback Crowdsensing under Probabilistic Coverage Constraint 81
5.1 Introduction . . . 82
5.2 CrowdRecruiter: System Overview . . . 85
5.2.1 Participant Selection Problem in CrowdRecruiter . . . 85
5.2.2 Overall Design of CrowdRecuiter . . . 86
5.3 Core Algorithms of CrowdRecruiter . . . 88
5.3.1 Call/Mobility Prediction . . . 88
5.3.2 Utility Calculation of Each Combined Set . . . 88
5.3.3 Coverage Probability Vector Calculation . . . 89
5.3.4 Algorithm Analysis . . . 89
5.4 Evaluation . . . 91
5.4.1 Baseline Methods . . . 91
5.4.2 Dataset and Experiment Setups . . . 92
5.4.3 Number of Participants Comparison . . . 94
5.4.4 Selection Process Comparison . . . 94
5.4.5 Participant Selection Overlaps . . . 94
5.4.6 Performance Evaluation and Comparison . . . 95
5.4.7 Combine Participants from Each Cycle . . . 97
5.5 Discussion . . . 98

6 CrowdTasker: Maximizing Coverage Quality under Incentive Budget Constraint 101
6.1 Introduction . . . 102
6.2 CrowdTasker System Overview . . . 107
6.2.1 Task Allocation Problem in CrowdTasker . . . 107
6.2.2 Overall Design of CrowdTasker . . . 108
6.3 Core Algorithms and Analysis . . . 110
6.3.1 Call/Mobility Prediction . . . 110
6.3.2 Utility Calculation . . . 110
6.3.3 Coverage Quality Estimation . . . 111
6.3.4 Algorithm Analysis . . . 112
6.4 Evaluation . . . 114
6.4.1 Baselines for Evaluation . . . 114
6.4.2 Dataset for Evaluation . . . 114
6.4.3 Coverage Quality Comparison under Budget Constraint . . . 115
6.4.4 Spatial Distribution of the Sensor Readings . . . 116
6.4.5 Computation Time of CrowdTasker . . . 116
6.5 Discussion . . . 117

7 Conclusion 119
7.1 Summary . . . 119
7.1.1 Summary of *EEMC* . . . 120
7.1.2 Summary of EMC^3 . . . 120
7.1.3 Summary of CrowdRecruiter . . . 120
7.1.4 Summary of CrowdTasker . . . 121
7.2 Future Work . . . 121

A Low-complexity Algorithms and Proofs 125
A.1 Algorithms and Proofs from Chapter 3 and 4 . . . 125
A.1.1 Low-complexity Algorithms for $P_{fulfill}$ Computation . . . 125
A.1.2 Low-complexity Algorithm for $P^*_{fulfill}$ . . . 126
A.2 Algorithms and Proofs from Chapter 5 . . . 126
A.2.1 Low-complexity Algorithms for $COV_i(\mathbb{S})$ Computation . . . 126
A.2.2 *Proof–Utility*($\mathbb{S}$) is an submodular function . . . 127
A.3 Algorithms and Proofs from Chapter 6 . . . 128
A.3.1 Low-complexity Algorithms for $CQE(\mathbb{X})$ Computation . . . 128
A.3.2 *Proof–*$CQE(\mathbb{X})$ is an submodular function . . . 128
A.3.3 *Proof*–The total Base/Bonus incentive payment is an submodular function over $\mathbb{X}$ . . . 129

B Curriculum Vitae and Research Publications **131**
B.1 Curriculum Vitae 131
B.2 Research Publications 131
B.2.1 Published or Accepted Papers 131
B.2.2 Under Reviewing/Revision 132

List of Tables

2.1 Energy Cost of Sensors and Sensing Tasks 26
2.2 Energy Cost of Data Transfer: the specific energy consumption depends on the waiting time, buffer size or bandwidth 26

3.1 Symbols and Definitions 38
3.2 Variables used in *EEMC* Algorithms 42
3.3 Data Transfer Energy Consumption Estimation 53
3.4 Energy Consumption Comparison: 3G-based vs Parallel+3G-based (P+3G) vs *EEMC* vs Pace vs Greedy 54

4.1 Performance Comparison based on Residential District Traces: EMC^3 vs Pace vs Greedy 74
4.2 Energy Consumption Computation Models 76
4.3 Energy Consumption Comparison: 3G-based vs Parallel+3G-based (P+3G) vs EMC^3 vs Pace vs Greedy 77
4.4 EMC^3 Average Response Time and the Estimated Maximum Throughput 78

5.1 Number of Selected Participants (CR. refers to *CrowdRecruiter* in all Tables and Figures) 90
5.2 Computation Time Comparison (in seconds, `Phase I`: *Data Preparation and User Call/Mobility Profiling*, `Phase II`: *Iterative Participant Selection Process*) . . 96
5.3 The Combined Number of Selected Participants using **BUSINESS+RESIDENTIAL** Datasets 98

6.1 Computation Time Comparison (in seconds, $B = 30000$, $E = 5$, $b_a = 50$ and $b_o = 1$) 117

List of Figures

1.1 The Four-stage Life Cycle of Mobile Crowdsensing Process 20

3.1 The Use Case of Abidjan's CBD Area 34
3.2 The Two-phase Task Assignment Framework 36
3.3 An Example: Estimating the Parameter with U_i's Accumulated Call Traces 43
3.4 The Example of $P\{X_{k,t}(A_k - R_k) = N\}$ Computing (Best Viewed in Digital Format) 45
3.5 Statistics of Evaluation Traces in D4D Data Set 48
3.6 Comparison of Task Assignments and Returned Participants: EEMC vs Pace vs Greedy 51
3.7 Number of Task Assignments and Returned Participants in Cold Start Period 51
3.8 Number of Task Assignments and Returned Participants varying with Time in the Cycle of $10:00 - 12:00$, 15 Dec 2011 (Best Viewed in Color) 52

4.1 Cell Towers in the Abidjan CBD Area 59
4.2 The EMC^3 Framework 62
4.3 Statistics of CBD Call Traces 69
4.4 Statistics of Abidjan Residential District Call Traces (Best Viewed in Digital Form) 71
4.5 Number of Task Assignments and Returned Participants (CBD Traces) 71
4.6 Number of Covering Participants (CBD Traces) 72
4.7 Number of Covering Participants (Residential District Traces, $N_e =$ 250 and 500) 73
4.8 Task Assignment Process in the Case Study 75

5.1 Cell Towers in the Downtown of Abidjan City 84
5.2 The CrowdRecruiter Framework 86
5.3 Max/Min/Average Coverage of Cell Towers based on the Three Regions and Settings 93
5.4 Selection Process: $\min_{0 \le i < N}\{COV_i(\mathbb{S})\}$ over $|S|$ 95
5.5 Percentage of Shared Participants among Different Methods (Best Viewed in Digital Format) 96
5.6 Temporal Coverage Ratio of Cell Towers in BUSINESS, RESIDENTIAL and MERGED Regions 97

6.1 PCS Task Allocation and Execution 104
6.2 Target region in the Downtown of Abidjan City 105

6.3 The CrowdTasker Framework 108
6.4 Coverage Quality Comparison in the BUSINESS and MERGED Regions (Best viewed with 300% zoom-in) 113
6.5 Spatial Distribution of Sensor Readings in three Regions (B=30000, $E = 5$, $b_a = 50$ and $b_o = 1$) 115

Chapter 1

Introduction

Contents

1.1 Background 17
1.2 Research Motivations and Contributions 19
1.3 Organization of this Thesis 23

1.1 Background

Mobile Crowdsensing (MCS) — a term coined by *Ganti et al.* [1] — is becoming increasingly popular as the number of mobile devices equipped with sensors (including phones, tablets, media players, games and leisure/sports electronic devices) shows dramatic growth. Facilitated by the widespread adoption of sensor-equipped smartphones, MCS has been successfully adopted to enable an ever-increasing number of sensing applications, ranging from highway congestion detection [2] to social trend understanding [3] and urban noise pollution/air quality monitoring [4, 5]. A main area of research in this field is concerned with enabling distributed monitoring applications that do not rely on a dedicated sensor network infrastructure; but where the crowdsensing communication is facilitated by an already existing network between devices (e.g., mobile phones) that are participating in the sensing tasks [6].

Mobile Crowdsensing with Mobile Phone Digital Footprints - In MCS, there are two main players: *MCS organizer* who is the person or organization coordinating the sensing task, and *MCS participants* who are the mobile users involved in the sensing task. To facilitate the mobile crowdsensing with the sensor-enriched mobile phones, the MCS organizer usually requires each MCS participant uploading the digital footprints generated by their mobile phones. For example, an MCS application intends to monitor the air quality of a big city with a large group of mobile phone users. Every hour the MCS application collects one sensor reading from each MCS participant and also fetches each user's real-time GPS position. After collecting the sensed result and the GPS data from each MCS participant, the application maps the air quality sensor reading to each corresponding GPS point on the Google map, so as to draw the "big picture" of air quality in the city. Specifically, following three types of mobile phone digital footprints have been widely studied:

- *Sensor Readings* - A mainstream smartphone might be commonly equipped with multiple sensors including *accelerometers*, *barometers*, *compasses*, *temperature sensors*, and *magnetic field sensors* [7, 8]. Furthermore, *digital cameras* [9], *microphones* [10], *ear-phones* [4], *wireless antennas* [11] and other devices equipped in the smartphone could be used as sensors for many crowdsensing applications. A comprehensive survey on mobile phone sensors and their applications to mobile sensing is [12]

- *Mobility Traces* - The commonly-seen smartphone mobility traces include *GPS trajectories* [13], *cellular trajectories* [14], *call detailed records* [15], *WiFi access point* and *Bluetooth contact traces* [16]. Combining the mobility traces of users with sensor readings, MCS applications can map the sensor readings onto the geographic map and future illustrate the *spatial coverage* of the MCS data collection. For example, [4] leverages a large group of participants in order to monitor the noise pollution in each street of a city; it continously senses each participant's surrounding noise using the ear-phone of smartphone while tracking each participant' mobility using GPS; further, with the GPS mobility traces, the application maps each collected noise result to street where the result is collected, so as to get the street-level noise map.

- *Smartphone App Usage Records* - Smartphone App Usage records including *phone call logs* [15], *email sending/receiving logs* [17], *Google map usage logs* [18], and etc. are frequently used to understand users' app usage behavioral patterns and further predict users' future app usage. With the predicted future app usage, [18, 19] proposes the *piggyback crowdsensing mechanism* to reduce the energy consumption caused by the MCS applications through performing MCS task in parallel with users' smartphone app usages e.g., uploading sensed results while a user placing a 3G call could reduce 75% energy consumption in MCS data transfer [20].

The Objective of Mobile Crowdsensing - Though the most MCS applications can be viewed as a process of collecting digital footprints from mobile users, the objectives of each MCS application is quite different with others, considering the requirements of specific sensing applications. For each MCS task, the organizer needs to specify the *target sensing area*, which often consists of a set of subareas. The organizer also needs to specify the *sensing duration* (e.g. 10 days), which is usually divided into equal-length sensing cycles (e.g. each cycle lasts for an hour). The objective of an MCS task is typically to collect certain environment data from mobile crowd in the target area in each sensing cycle, with the goal of collecting high quality sensed results and supporting the specific environmental monitoring applications. Taking a one-week urban air quality monitoring MCS task as an example, the MCS organizer first divides the whole area into $1km^2$ grid cells and then splits the one-week MCS sensing time into a sequence of one-hour sensing cycles [21], where the application aims at collecting at least one sensed result from each grid cell in each sensing cycle.

The Process of Mobile Crowdsensing - While the objectives of mobile crowdsensing might be different due to the various goals/settings for data collection, the design of MCS applications usually follows a similar paradigm. In general, a mobile crowdsensing application usually consists of *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. [22] divides the life cycle of mobile crowdsensing process into four phases: Task Creation, Task Assignment, Individual Task Execution and Crowd Data Integration, as shown in Fig. 1.1. The key functionalities of each phase 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 phase is *task assignment* - recruiting participants and 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 phase 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.

1.2 Research Motivations and Contributions

With respect to the aforementioned objectives and the process of mobile crowdsensing, our research are based on following well-justified observations:

Observation I. Users' willingness of MCS participation - It is clear that user participation is necessary for successful mobile crowdsensing. However, *three main factors* are known to compromise the users' willingness to become part of a crowd:

- *Privacy -* Due to the privacy concerns, a user may not be willing to participate in all MCS tasks and may wish to anonymize herself in each MCS task in which her device participates. To ensure privacy and, as a consequence, to encourage participation, there must be no way to link a participant to her records in previous MCS tasks.

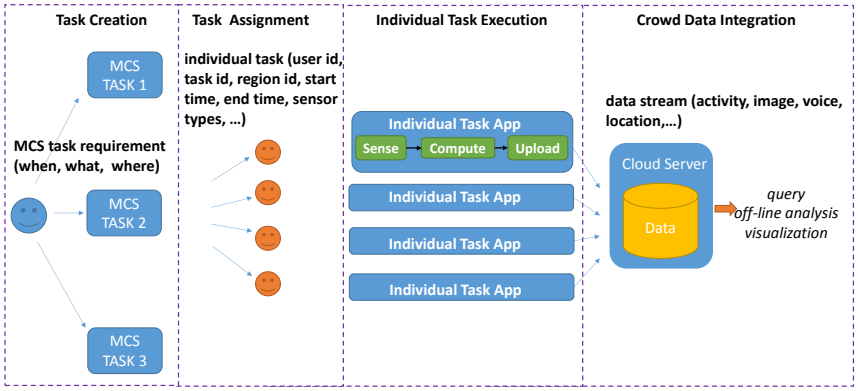


Figure 1.1: The Four-stage Life Cycle of Mobile Crowdsensing Process

- *Energy Consumption* - The energy consumption of MCS on mobile devices may drain the battery and as such might discourage user participation. The energy consumption of an MCS task can be viewed locally by each individual crowd member, or globally from the point of view of the whole crowd. *Individual Energy Consumption* is concerned with the energy consumed by the MCS task in the battery of each individual participant's mobile device; and this depends on the way that the MCS task executes on the device. The *Overall Energy Consumption* is concerned with the total energy consumed by all crowd members.

- *Incentive* - In addition to ensuring mobile users to save energy in MCS, one effective way to encourage mobile users' participation in MCS task is to provide incentives (e.g., money, 3G internet bandwidth, etc.) to each user. Typically, each selected participant is offered a certain amount of money as incentive and thus the MCS organizer needs to prepare a budget equal to the total incentives paid to all participants in each MCS task.

Observation II. Efficiency and the effectiveness of MCS task- While the MCS participants care more about the *energy* consumed for participating the MCS task and the *incentives* received from the task participation, the MCS organizer concerns more about the *quality* of data collected from the MCS task and the *total incentives* paid to all participants.

- *Sensing Data Quality* - Generally, an MCS task might want to collect the sensing data that well represents the characteristics of the target sensing region. Thus, the sensing data quality of an MCS task could be characterized in two aspects:

1. *The accuracy of sensed results* - Supposing there exists noise in each individual sensed result [23] (e.g., the sensing deviation of air quality sensors), it might need to collect *multiple sensed results* from the target region in order to estimate the accurate results. For example, in order to estimate the accurate air quality index of a street, an MCS application collects sensed results from at least 10 MCS participants in the street every hour and estimates the accurate result by averaging all collected results.
2. *The coverage of the sensed results* - Rather than the accuracy of each individual sensed result, the MCS organizer also concerns if the sensed results collected by the participants could fully or partially cover the target region spatially and temporally. For example, an air quality monitoring MCS application needs to collect air quality sensor data from each street of Paris every hour, so as to monitor the air quality of the whole city.

From above two aspects, we can conclude that the sensing data quality of an MCS task might be associated to the *number of sensed results* collected from the target region and the *spatial-temporal coverage* of sensed results over the target region and sensing time slots.

- *Total Incentive Payment* - It is also obvious that the more total incentives paid, the higher MCS sensing data quality achieved. With the sensing data quality quality and total incentive payment issues in mind, the MCS organizer might either aim to

  1. Maximize the overall MCS sensing data quality with a fixed amount of incentive budget, or
  2. Minimize the total incentive payment while ensuring the collected sensed results meeting a predefined sensing data quality.

Our Contribution - In the research being presented, we are motivated to propose MCS framework which addresses the aforementioned *concerns* from both MCS organizers and MCS participants, through reduction of energy consumption of individual crowd members, and effectively allocating incentives to the crowds while optimizing the MCS sensing data quality. Further, we aim to achieve this goal without sacrificing the privacy requirement. With respect to aforementioned motivations, this thesis includes following four contributions:

1. **EEMC -** In this contribution, we first propose an energy-efficient Piggyback Crowdsensing mechanism reducing energy consumption of MCS data transfer by receiving task assignment and returning sensed results in parallel with two 3G calls [20]. Further, assuming each assigned participant would be paid *an equal-mount incentive*, *EEMC* assigns MCS tasks to *a minimal number of anonymous participants* while ensuring *a predefined number of assigned participants* returning sensed results in a specific time-frame, so as to guarantee the data collection

from a minimum number of participants in the target region also minimizing *overall energy consumption* and *the total incentive payment*. Evaluations with a large-scale real-world phone call dataset show that our proposed *EEMC* framework outperforms the baseline approaches, and it can reduce overall energy consumption in data transfer by 54% - 66% when compared to the 3G-based solution.

2. **EMC^3 -** While *EEMC* reduces individual energy consumption and minimizes overall energy consumption/total incentive payment under a simple sensing data quality constraint (i.e., the minimum number of sensed results required in each cycle), this contribution aims at studying an novel MCS task assignment framework under an more complex data quality constraint—i.e., full spatial-temporal coverage constraint. In this contribution, EMC^3 reduces the individual energy consumption caused by MCS data transfer by leveraging the two-call-based piggyback crowdsensing mechanism of *EEMC*. Further, given the target region divided into *subareas*, EMC^3 assigns MCS tasks to *a minimal number of anonymous participants* while ensuring *at least one sensed result being returned from each subarea* in a specific time-frame, in order to minimize the *overall energy consumption* and *the total incentive payment* under *full coverage constraint*. Specifically, EMC^3 incorporates novel pace control and decision making mechanisms for task assignment, leveraging participants' current call, historical call records as well as predicted future calls and mobility, in order to ensure the expected number of participants to return sensed results and fully cover the target area, with the objective of assigning a minimal number of tasks. Extensive evaluation with a large-scale real-world dataset shows that EMC^3 assigns much less sensing tasks compared to baseline approaches, it can save 43%-68% energy in data transfer compared to the traditional 3G-based scheme.

3. **CrowdRecruiter -** While *EEMC* and EMC^3 intend to assign MCS task to a minimal number of participants during the MCS task (i.e., *online task assignment*), this contribution studies an *offline participant selection* problem, where prior to the MCS task a minimal number of participants are firstly selected from volunteers, then during the MCS task each selected participant is required to join all MCS sensing cycles while *ensuring the spatial coverage of the selected participants meeting predefined coverage requirement*. In this contribution, we introduce a novel participant selection framework, named CrowdRecruiter. CrowdRecruiter operates on top of energy-efficient Piggyback Crowdsensing (PCS) task model proposed by [18], minimizes the overall incentive payments by selecting a small number of participants while still satisfying probabilistic coverage constraint. In order to achieve the objective when piggybacking crowdsensing tasks with phone calls, CrowdRecruiter first predicts the call and coverage probability of each mobile user based on historical records. It then efficiently computes the joint coverage probability of multiple users as

a combined set and selects the near-minimal set of participants, which meets coverage ratio requirement in each sensing cycle of the PCS task. We evaluated CrowdRecruiter extensively using a large-scale real-world dataset and the results show that the proposed solution significantly outperforms three baseline algorithms by selecting 10.0% - 73.5% fewer participants on average under the same probabilistic coverage constraint.

4. **CrowdTasker -** While CrowdRecruiter intends to select a minimal number of participants for joining in all sensing cycles of the MCS task while meeting the probabilistic coverage constraint, this contribution proposes a novel PCS task allocation framework—CrowdTasker, which selects one group of participants for each sensing cycle of the MCS task, in order to maximize the overall MCS data quality while satisfying the incentive budget constraint. In order to achieve this goal, CrowdTasker first predicts the call and mobility of mobile users based on their historical records. With a flexible incentive model and the prediction results, CrowdTasker then selects a set of users in each sensing cycle for PCS task participation, so that the resulting solution achieves near-maximal coverage quality without exceeding incentive budget. We evaluated CrowdTasker extensively using a large-scale real-world dataset and the results show that CrowdTasker significantly outperformed three baseline approaches by achieving 3% - 60% higher coverage quality.

1.3 Organization of this Thesis

The rest of thesis is organized as:

- **Chapter 2** gives a comprehensive survey on the state-of-the-art of mobile crowdsensing, including the related work of (a) recent MCS applications and frameworks, (b) energy consumption measurement for MCS applications, (c) energy-saving strategies for MCS applications, (d) MCS incentive model, (e) MCS sensing data quality, (f) MCS participant selection and task assignment, and (g) mobility prediction techniques applied to MCS.

- **Chapter 3**, **Chapter 4**, **Chapter 5** and **Chapter 6** present our work of *EEMC*, EMC^3, CrowdRecruiter and CrowdTasker respectively, where we introduce (a) the motivating example of each framework and the most closest related work, (b) research assumptions/objectives and problem formulation, (c) detailed framework/algorithm designs, and (d) evaluation results using the real-world datasets.

- **Chapter 7** discusses several open issues of the four MCS frameworks and concludes this thesis, where we address several future directions of MCS research in our viewpoints.

Chapter 2

State of the Arts

Contents

2.1 MCS Applications and Frameworks 25
2.2 MCS Energy Consumption 26
2.3 MCS Energy-saving Strategies 27
2.4 MCS Incentive Models 27
2.5 MCS Sensing Data Quality Metrics 28
2.6 MCS Participant Selection and Task Assignment 28
2.7 Human Mobility Prediction for MCS 29

2.1 MCS Applications and Frameworks

There has been much recent research leading to the development of many different mobile crowdsensing applications and services; for example: automated recognition of human activities and context using sensor data [24], automated modeling of location characteristics [25] and linking such location semantics to user profiles [26], mapping network cells to geographic locations [27], social interaction and collective behavior sensing [28, 29], mobile object discovery [30] in urban areas, and road traffic/public transport monitoring [31, 32].

To support the above-mentioned applications, many different mobile crowdsensing frameworks [33, 34, 35, 36] have been proposed. For example, [35] designs a framework to deploy MCS applications on mobile devices in order to scale the MCS system; [33] proposes a framework selecting the MCS participants from volunteers before MCS task execution, where the participant selection is based on mobility data mining and reputation modeling for volunteers; [36] introduces CAROMM – an MCS data collection framework based on mobile data mining in order to reduce the data transmission for results uploading, while maintaining the accuracy of collected results; and [34] further develops CAROMM and provides a real-time context-aware MCS framework delivering integrated sensed results to MCS end-users. [37] has presented a rapid prototyping framework called "Madusa" for mobile crowdsensing.

The proposed framework structures mobile crowdsensing into three main stages — "recruiting-sensing-uploading".

2.2 MCS Energy Consumption

In this section, we mainly introduce the research work measuring the energy consumption of mobile phone for MCS applications. The energy cost for a mobile device to perform a sensing task can be generally divided into three parts: for sensing, computation and data transfer. In our research we particularly focuses on the energy consumption caused by following two parts:

Sensing Task	Sensors (frequency, duty cycle)	Energy (J)	Total Energy (J)
Human Activity Monitoring	Accelerometer (160Hz, 10%)	0.66	1.43
	Microphone (1Hz, 50%)	0.755	
	Compass (1Hz, 10%)	0.015	
	Pressure (1Hz, 100%)	0.0006	
Environment Monitoring	Temperature (1Hz, 20%)	0.0012	0.3
	Microphone (1Hz, 20%)	0.3	

Table 2.1: Energy Cost of Sensors and Sensing Tasks

Energy Consumption in MCS Sensing - The power of sensors, including accelerometer, pressure, temperature, microphone and compass sensors, equipped by the mainstream mobile phones are also covered by Table 2.1. The instrumental results listed in Table 2.1 is measured by work [8, 38, 39]. Particularly, we take care of the sensor energy consumption under various frequency and duty cycles settings, so as to succeed different sensing tasks, e.g., environmental monitoring and human activity recognition.

Type	Connection (**J**)	Data Transfer (**mJ/byte**)
3G (UTMS)	12.0	0.04-0.16 download 0.09-0.3 upload
SMS (SS7)	2.0	3.0
WIFI	5.0-12.0	0.01
2G (GSM/GPRS)	4.0	0.036

Table 2.2: Energy Cost of Data Transfer: the specific energy consumption depends on the waiting time, buffer size or bandwidth

Energy Consumption in MCS Data Transferring - In Table 2.2, we discuss the energy consumption of data transfer, including the cost of connection establishment, data uploading/downloading, connection maintenance and tail, by using the network of 2G, 3G, WIFI and SMS (SS7). We take the energy consumption to establish, to maintain and to end a connection into account as "connection" in the table. All above measurement and instrumental results are investigated from the

work [17, 40, 41]; and interested readers are encouraged to see also in these papers. Since the payload of data uploading/downloading in MCS, including datagrams for both the command word of task assignment and sensory data result, is quite small. Therefore, no matter which data transfer method of 3G, GSM, WIFI or SMS (SS7) is employed, the MCS data transfer of a few bytes [20, 42] might cost most energy in connection including connection establishment, maintenance and tail.

2.3 MCS Energy-saving Strategies

As the energy cost for a mobile device to perform a sensing task can be generally divided into three parts: for sensing, computation and data transfer, we hereby introduce the MCS energy-saving strategies in following three categories:

Saving Energy in MCS Sensing - To reduce the energy cost for sensing, there are many proposals ranging from the adoption of low power sensors [43, 10], adaptive sensor schedulers [44], to using sensing data predictors [32, 45].

Saving Energy in MCS Computing - To save the energy cost for computing, mobile sensing systems have turned towards using low power processors [46], and reducing computation workloads by leveraging energy efficient sensing data processing algorithms [47, 48] or offloading mechanisms [13].

Saving Energy in MCS Data Transfer - To reduce the energy cost for data transfer, three lines of research have been conducted

- Using low power wireless communication [49, 50, 51] can directly reduce the energy consumption of data transfer.
- Using mobile nodes as relays [49, 52] to carry and forward data between sensing devices and the server can save energy, since multi-hop relaying may still cost less than uploading data directly to the server.
- Transferring less sensing data can also save energy. The compression of sensing data [53] can reduce the data size directly. Further, strategies exist for minimizing data transfer by communicating only unpredictable data, while inferring the predictable data [54]. These methods may consume more energy during computation; so they require a careful trade-off to make the whole system more energy-efficient.

Finally, energy harvesting mobile sensing systems [55] have been studied to function with battery-free platforms.

2.4 MCS Incentive Models

Previous research work about MCS incentives has leveraged game theory and auction mechanisms to analyze the optimal payment to be offered by the MCS organizer to

participants, and to find the best compromise between participants' and organizer's profit (i.e. the utility function in game theory) [56, 57]. As an alternative to monetary reward, some approaches offer other incentives such as service time [58] and coupons [59]. In general, these approaches assume the users' cost to finish a task to be known in advance, and this cost follows some specific probability distribution in their simulation experiments.

2.5 MCS Sensing Data Quality Metrics

The straight-forward way of measuring the MCS sensing data quality is to use spatial-temporal coverage [60, 61, 62, 63, 64, 65].The work of both full coverage [60, 61] and partial coverage [62, 63, 65] has been studied. [60, 61] uses the full coverage as the constraint of sensing data quality for MCS data collection; both of them aim to collect at least one result returned from each subarea of the target region. [62] is the first to propose to use the *probabilistic coverage* as the MCS sensing data quality, where the author defines the probabilistic coverage as the percentage of subareas covered by the sensed results in each sensing cycle. [65] defines a novel type of partial coverage metrics—*opportunistic coverage*, which uses the distribution of time duration between each two consequent sensed results obtained in each subarea as the MCS sensing data quality. All these spatial-temporal coverage metrics are associated to the number of sensed results obtained, the number of subarea covered by the sensed results, and the number of sensing cycles that each subarea of the target region are covered.

Rather than using spatial-temporal coverage as the MCS sensing data quality metrics, Krause *et al.* [66, 67] propose to use the observation certainty to measure the quality of sensed results obtained in participatory sensing. Authors assume the noise exists in the obtain sensor data (namely *observations*) and further assume such noise follows certain stochastic process (e.g., Gaussian) in spatial and temporal domain. In this way, this work quantify the MCS sensing data quality as the overall predictive variance [67] of the collected sensor data.

2.6 MCS Participant Selection and Task Assignment

While the MCS participants care about the *energy* consumed for participating the MCS task and the *incentives* received from the task participation, the MCS organizer concerns more about the *sensing coverage* of data collected from the MCS task and the *total incentives* paid to all participants. Thus, many previous work studies the algorithms/frameworks, selecting participants from volunteers and assigning MCS tasks to participants subject to *energy consumption*, *total incentive payment* and *sensing coverage* objectives/constraints.

In order to minimize the overall energy consumption of an MCS task under MCS data quality constraint, the research objective becomes keeping the energy consump-

tion of each mobile device low and finding the minimal number of participants while ensuring a predefined MCS data quality e.g., full or partial coverage of the target region. In [68, 69], the authors introduce the notion of virtual sensors which intend to collaboratively infer sensing values to reduce physical and redundant sensing, they propose spatial and temporal coverage metrics for balancing the overall energy consumption and data quality. In [70], Musolesi *et al.* present several techniques to optimize the information uploading process for continuous sensing, they also consider the coverage and overall energy consumption in MCS. Sheng *et al.* [71] propose a mechanism to reduce the overall energy consumption in mobile crowdsensing by optimizing the schedule of each sensing device, collaboratively all the mobile devices could fully cover the target region with minimal sensing energy.

In order to maximize the overall sensing data quality of the MCS task under the total incentive payment constraint. Reddy *et al.* [72, 33] first study the research challenge of participant recruitment in participatory sensing, they propose a coverage-based recruitment strategy to select a predefined number of participants so as to maximize the spatial coverage. More recently, Singla *et al.* [73] proposes a novel adaptive participant selection mechanism for maximizing spatial coverage under total incentive constraint in community sensing with respect to privacy. Also in [74], Cardone *et al.* develop a Mobile Crowdsensing platform, where a simple participant selection mechanism is proposed to maximize the spatial coverage of crowdsensing with predefined number of participants.

Whilst above work attempts at maximizing the MCS data quality under the budget constraint, two recent MCS frameworks [62, 75] are proposed to minimize the total incentive payments while ensuring the MCS task meeting the coverage constraints. First authors attempt to use a mobility model to predict mobile users' future locations. Based on the predicted results they aim to select a minimal number of mobile users, expecting to cover a certain percentage of the target area in the next timeslot. However, both [62, 75] focus on the mobility model and coverage probability prediction. They assume that each user's historical locations are known and the time slot for mobility prediction is short, as both methods make decisions in each step in order to select new users based on the coverage probability estimation.

2.7 Human Mobility Prediction for MCS

A variety of schemes that address the problem of prediction of user location have been studied. In general, they fall into the schemes based on *individual mobility patterns* and *collective mobility patterns*.

Predictor based on Individual Mobility Patterns - These schemes take advantage of the temporal and spatial regularities that are exhibited in the individual's mobility patterns. The prediction schemes based on *markov models*, especially those based on the *higher-order markovian model* [76] are considered as the state-of-the-art in the practical predictor design [77], since it takes the probable locations for next

movement and the temporal order of movements into account. Besides, some of other schemes foresee user location by detecting periodic patterns in user traces. The predictability of prediction schemes based on individual's mobility patterns is limited, around 90% in the theoretical upper bound [78].

Predictor based on Collective Mobility Patterns - In recent years, many hybrid user location prediction schemes leveraging the *collective mobility patterns* have been studied. They postulate that user movement is driven by social-tie [79], involving the social community identification, and the prediction based on the community attraction to users. As a typical example, CMM [80] leveraged user friendship to cluster users as communities, and then decided user next location by community attraction. Calabrese *et al.* [81] introduced the first predictor fusing the *collective behaviors* and *individual mobility patterns* of mobile phone users. It employs a prediction scheme based on the periodicity of the individual's mobility pattern, and then uses the collective geographical preferences to refine the prediction result.

Chapter 3

EEMC: Energy Efficient Mobile Crowdsensing with Anonymous Participants

Contents

3.1 Introduction 32
3.1.1 Proposed Research: Assumptions, Objectives and the Example . . 32
3.1.2 Research Challenges and Our Contributions 35
3.1.3 Comparison with the Most Related Work 37
3.2 Problem Formulation 38
3.3 *EEMC* Framework and Skeleton Algorithm 39
3.3.1 Phase I - Candidate User Identification based on Call Prediction . 39
3.3.2 Phase II - Two-step Decision Making Process for Task Assignment 40
3.4 Next-Call Prediction Model based on Accumulated Call Traces 42
3.4.1 Probabilsitic Model of Phone Calls 43
3.4.2 Parameter Estimation using Accumulated Traces 43
3.5 Adaptive Pace Controller for Task Assignment 44
3.5.1 Adaptive Pace Control for Task Assignment 44
3.5.2 Probability Estimation for Adaptive Pace Control 44
3.6 Near-Optimal Decision Maker for Task Assignment 45
3.6.1 Identifying *Future-surer Candidates* 45
3.6.2 Estimating if the Missing Number of Sensed Results can be returned from Future-surer Candidates and Potential Returners . . . 46
3.6.3 Near-optimal Task Assignment Decision Making 47
3.7 Experimental Setups 47
3.7.1 Baseline Methods and Parameter Settings 47
3.7.2 Dataset and Experiment Setups 48
3.8 Evaluation Results 49
3.8.1 Performance Comparison 50

3.8.2 Cold-start Performance 51
3.8.3 Case Study and Analysis 52
3.8.4 Energy Conservation Comparison 53
3.9 Discussion 54

3.1 Introduction

As has been introduced in Chapter 1.2, while MCS participants concerns more on *individual energy consumption* caused by MCS, *privacy*, and *incentives* received for the participation, the MCS organizer focuses more on *sensing data quality* and *total incentive payment* of the MCS task. Thus, we intends to study an MCS framework *reducing* energy consumption of each individual participant, *minimizing* the overall energy consumption and incentive payments of the whole MCS task while *meeting* the predefined sensing data quality *goals*.

Particularly, this work studies a novel type of MCS task that aims to collect sensing results from *a specified number of participants* in the target region within a certain time duration. For example, the air quality of the central business district in Abidjan City is monitored by an MCS application, which collects forty samples of air quality sensed by different participants in the district every two hours. Each of the MCS participants *receives* a sensing task assignment, then *executes* it, and finally *returns* the sensing results. As a consequence, the air quality result sensed by participants in the most recent two hour period can be used to estimate and update the aggregated air quality index.

With above settings and objectives in mind, we are motivated to reduce individual energy consumption caused by MCS data transfer leveraging the low-power data transfer mechanism, minimize the overall energy consumption/total incentive payments of the complete MCS task, through the minimization of the total number of participants assigned with the MCS task. Further, we aim to achieve this goal without sacrificing the anonymity requirement of participants.

3.1.1 Proposed Research: Assumptions, Objectives and the Example

In terms of energy conservation of MCS applications on mobile device, three main components — *data transfer* [82, 83, 84, 54], *sensing* [45, 13] and *computation* [47, 46] — have been the focus of study. Different from the existing work in energy-efficient mobile crowdsensing mechanisms (or frameworks) [75, 61, 68, 43], this work aims at designing a novel energy-efficient mobile crowdsensing framework (named *EEMC*) which addresses three aspects of the problem in an innovative manner. The mechanism will 1) minimize overall energy consumption due to data transfer, 2) guarantee that the required number of sensor results will be returned during each

cycle, and 3) maintain the anonymity of users who have participated at any point in the lifetime of the crowdsensing activity. Our research is based on a number of well-justified assumptions:

1. *Connection Setup Cost, and Energy Conservation in MCS Data Transfer* - Recent studies on energy consumption in a range of different devices note that a smartphone, operating on a 3G network, typically needs to consume "12 Joules before the first byte can be sent" [42, 85]. The energy consumption for small data transfer (less than 10KB) is mainly concerned with establishing (and closing) the 3G connection, and is also fixed around 12 Joules [17]. This is coherent with our previous study [19] on air quality sensing, where we observed that when task assignments and the results of the common MCS tasks are relatively simple and the transferred data is quite small ($\leq$ 10KB), then the energy consumption of data transfer to receive a task assignment and returning the result is also fixed at approximately 12 Joules.

2. *Parallel Transfer and Energy-efficient MCS Data Transfer* - If a mobile phone receives the task assignment and uploads the sensed result in parallel with the user's regular phone calls, then the additional energy consumed in data transfer for an MCS task would be significantly reduced thanks to reuse of the already established 3G connections [51, 19]. This type of technique — that piggybacks data over connections established by voice calls or other 3G mobile applications — is known commonly as *Parallel Transfer*. Taking the Nokia N95 phone as an example, a 3G data connection typically consumes around 12 Joules (which is consistent with our first assumption), while the additional energy incurred when piggybacking a data packet of 10KB over a 3G call is around 2.5 Joules (which corresponds to a 75% - 90% reduction in energy consumption). As an interesting comparison, sensing the noise with a microphone in the same phone requires about 1 Joule in order to get a valid sample [39].

3. *Receive-Sense-Return Cycles and Delay-tolerant MCS* - To support MCS applications, many different task assignment schemes [33, 34, 35, 36, 37] have been proposed. All these schemes structure mobile crowdsensing applications (on mobile devices) into three main stages "Receive—Sense—Return" (or "recruiting—sensing—uploading" in [37]). In the first stage, the mobile device receives task assignment from the central server, then executes the sensing task during the second stage, and returns the sensed results back to the central server in the third and final stage. A wide range of MCS tasks (a good example is the previously mentioned air quality sensing application) can be completed successfully, provided all mobile devices can go through these three stages within a specified time-frame (delay) for each single task [86].

4. *Two-call-based MCS Mechanism for Cyclic Sensing Tasks* - Considering the *delay tolerant nature* of many MCS tasks, it is a reasonable assumption that

(a) CBD Area of Abidjan (b) An Example of Sequential Task Assignment

Figure 3.1: The Use Case of Abidjan's CBD Area

we can divide an MCS task into equal-length (*Receive-Sense-Return*) cycles. In each sensing cycle, the central server attempts to collect sensing results from a required number of participants. With *parallel transfer* in mind, we can significantly reduce energy consumption in data transfer of a sensing cycle if we are able to assign sensing tasks to the mobile phone users who will place (make or receive) *two or more phone calls in the cycle*. These users receive task assignments and return their sensed results piggy-backing the data transfer on top of the calls through the *parallel transfer* approach.

In summary, to enable energy efficient mobile crowdsensing with *Two-call-based MCS Mechanism*, our initial research makes the assumptions that:

- Each MCS task lasts for a limited duration and involves a series of sensing cycles;
- All participants receive task assignments and return sensing results, only when they are involved in calls;
- In each cycle, a participant will be assigned with tasks no more than once;
- Due to privacy concerns, all participants will be anonymized for each MCS task in such a way that we cannot link any participant to records of her previous MCS tasks.

Based on the above assumptions, our research proposes an MCS task assignment mechanism which meets two objectives:

1. *to ensure the required number of participants returning the sensing results within the cycle*, and
2. *to minimize the number of redundant task assignments.*

To further illustrate the proposed *research assumptions and objectives*, let us reconsider the aforementioned air quality sensing use case. An environmental NGO in Ivory Coast, with the help of a local telco, launches an air quality monitoring MCS task in Abidjan City's CBD region where 25 cell towers are installed (see also in Fig. 6.2). In order to provide the timely air quality sensed results to the citizens of Abidjan city, the MCS task is designed to update the air quality reading once every 2 hours (i.e., a sensing cycle lasts for 2 hours). In order to provide reliable measures, the application is designed to secure the data collection from a minimum number (e.g., 80) of mobile users in the target area per sensing cycle. In order to facilitate the task assignment, as shown in Fig. 3.1b, *EEMC* is deployed on a central server which continuously monitors all mobile users' calls in the target region, analyses the call activities of MCS participants, and decides, for each incoming call, if a participant placing (making or receiving) the call should be assigned with a sensing task. Please note that, only when a participant makes/receives a phone call in the target region can she receive the task assignment or return the sensed result. In this way, tasks are assigned in a *sequential manner* as new calls are established, tasks assigned and sensed results returned.

3.1.2 Research Challenges and Our Contributions

In order to achieve the proposed research objectives and validate them through a realistic use case, we address the following key ***technical challenges***:

- *Next-call Prediction for the new arrival caller/callee based on accumulated call traces* - It is not possible to know in advance which of the crowdsensing participants will be involved in (two) phone calls during a particular sensing cycle. Thus, we need an effective method for predicting possible participation based on the participant's previous call history. However, due to the anonymization requirements, we cannot link a user with her phone call records during previous MCS tasks. Thus, there needs to be a method to predict the future phone call patterns of users using their accumulated history restricted to the current task.

- *Dynamically decide whether further task assignment is needed* - No method for call prediction can be perfect. As a consequence, tasks may be assigned to participants (based on their predicted call patterns), who fail to be involved in the minimum 2 calls required for the *"receive"* and *"return"* stages in the sensor cycle. To mitigate this problem, we propose assigning redundant tasks in such a way that the required number of results will always be returned even if individual participant's call behaviour is not as predicted. To avoid energy waste, the redundant task assignments should be *as few as possible*. The key decision that has to be made is concerned with how to update task assignments (if it all) when a new call is established during a single cycle.

- *Current Calling User vs. Future Users? a non-trivial trade-off* - Simple analysis

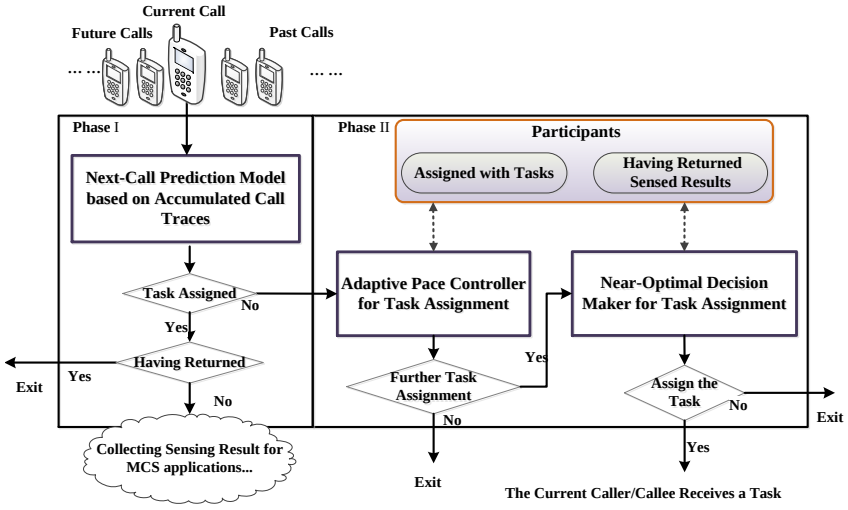


Figure 3.2: The Two-phase Task Assignment Framework

would suggest that it is a good strategy to assign a task to any user who has just established a call (caller or callee), provided that they have not already been assigned a task and provided that further task assignments are needed. However, this may not be a good approach if this user has a low chance of being involved in a second call before the current cycle is complete. The decision should not be made in a `local manner` — it is better to compare the probability of the user meeting the 2-call per cycle requirement with the global probability set of meeting the same requirement for all other crowd members (i.e., the participants having not placed any calls in the current cycle but with higher probabilities of placing two calls before the end of the cycle).

In this work, we propose a two-phase approach (illustrated by the process shown in Figure 3.2) in order to address the above-mentioned challenges. Consider the situation where a user is making or receiving a phone call, our first phase queries and updates her mobile phone call traces, and *identifies* whether she is a *candidate for task assignment* based on *phone call prediction*. In the second phase, with a user for whom we haven't yet assigned any task in the current cycle, a *two-step* decision making process is proposed to determine whether or not we should assign a task to her; where the first step (using the *Adaptive Pace Controller for Task Assignment* component) decides *if further task assignments are needed based on tasks already assigned and the participants having already returned their sensing results*, and the second step (using *Near-Optimal Decision Maker for Task Assignment*) decides *if the current caller/callee should receive the task assignment through comparison with potential callers/callees in the time remaining of the current cycle*. The detailed

contributions of this work are:

1. Firstly, motivated by saving energy in data transfer of MCS tasks for both individual participants and the whole crowd, we propose a novel mobile crowdsensing framework *EEMC* leveraging both the *parallel transfer* mechanism and the *Receive-Sense-Return cycle* pattern, whilst also respecting the requirement for anonymity. Further, we investigate and formulate the technical problem inside *EEMC*—a task assignment decision making problem—with *minimal number of task assignments* as the goal and the *predefined number of returned sensed results* as the constraint. To the best of our knowledge, this is the first work which addresses the issue of energy-efficient MCS data transfer in the proposed way.

2. Secondly, we develop a two-step decision making process, and algorithms, to control the task assignments. When the proposed algorithm makes decision on task assignment, it considers four types of participants: 1) the calling user, 2) the participants already assigned with tasks, 3) the participants having already returned sensing results, and 4) the future users who are (potentially) going to make two phone calls. Though this algorithm is designed for *EEMC*, other MCS frameworks with a similar optimization goal – but which do not assume that each assigned participant will return his/her sensed result – can also benefit from application of the algorithm.

3. Thirdly, we evaluate *EEMC* on the D4D dataset [15] containing 4-month call detail records of Ivory Coast citizens. The result shows that *EEMC* can guarantee that the required number of participants return their sensing results whilst making fewer redundant task assignments than the baseline schemes. When we consider overall energy consumption in data transfer for MCS applications, such as air quality or noise monitoring at the Abidjan CBD area, compared to the traditional 3G-based scheme the reduction is quite significant. In our case study, *EEMC* reduces energy consumption in data transfer by approximately 75% for a specific participant, with a global reduction of 54% - 67% for the whole crowd.

3.1.3 Comparison with the Most Related Work

Regard the state-of-the-art discussed in the Chapter 2, we sort the most related work of our study into following three categories:

1. *Using low power wireless communication as energy-saving strategy for MCS data transfer* - The most related work is [49, 50]. Our research follows this approach by leveraging the parallel transfer with voice call [51] as a low power communication method.

2. *Task assignment mechanism minimizing overall energy consumption and total incentive payment under the sensing data quality constraint* - The most related

Symbols	Definitions
t_0	The starting time of an MCS task;
T	The duration of a sensing cycle;
N_e	The expected number of returned participants
k	The index of a specific cycle;
t	The elapsed time during cycle k, where $t \in [t_0 + (K-1)T, t_0 + K * T)$;
A_k	The set of participants already assigned with tasks in the cycle k;
R_k	The set of participants having already returned sensing results k;

Table 3.1: Symbols and Definitions

work is [33, 34, 35, 36]. Different from all previous work, which assumes that each assigned participant would return sensed results, *EEMC* assumes that assigned participants may not be able to return sensed results. This is a much more realistic assumption as it can, amongst other things, cope with a common scenario of a participating user's phone being turned off in the middle of a cycle (perhaps due to the battery losing charge). In order to manage this more realistic model of the crowd of user participants, a more complex allocation algorithm based on redundancy needs to be used. However, redundancy increases energy consumption. Thus, the research challenge is to have a "fault tolerant" allocation mechanism which attempts to minimize the number of *redundant task assignments*.

3. *Validation and Experiments* - The validation approaches used in previous papers use either *small scale* real-world data or a large scale *simulated* data set. We argue that there are *weaknesses* in both these types of evaluation approaches; and we adopt a *large-scale real-world* approach using the mobile phone dataset D4D to verify the effectiveness of our proposed algorithms.

3.2 Problem Formulation

An MCS task consists of a sequence of sensing cycles — assumed to be of the same length/frequency — with each cycle requiring a predefined number of sensing data to be collected. This expected number is the most important target in data collection as sensing data processing can be compromised if insufficient updated data is available. For simplicity, we assume that the expected number of sensing data requirement is constant throughout the task, and between cycles.

In this work, the MCS tasks are treated as independent of each other in order to respect the privacy protection policy. Individual calling history information of mobile users should not be shared amongst MCS tasks. However, during an individual MCS task, the calling history of a different group of users can be recorded, but the record

will expire when the MCS task ends. In order to collect a set of sensing data from a single mobile user in one cycle it is necessary and sufficient that the user be involved in two calls: one call for assigning a task from the server and the other for returning sensing data. Also, no mobile user in a sensing cycle can be assigned the task of collecting sensing data more than a single time. With these conditions in mind, we formally formulate the problem as follows.

Given an MCS task with starting time t_0, sensing cycle T, and the expected number of sensing data N_e from a sensing cycle, we record the time-stamps and participants making/receiving phone calls from t_0. We denote A_k as the set of mobile users who have been assigned with sensing tasks since the start of cycle k, and R_k as the set of mobile users who have returned sensing results, where R_k is always a subset of A_k. Every time a participant makes/receives a phone call in the sensing cycle k, our problem is to decide whether to assign a task to the participant. *The goal of task assignments* is to:

$$\text{minimize } |A_k|, \text{ subject to } |R_k| \geq N_e$$

by the end of cycle k. It should be noted that, as we cannot know in advance who is going to place another call, we cannot statically optimize the task assignment process. Therefore, the dynamic decision making for task assignments is based on a phone call history and prediction model. In this way, our research decomposes the original task assignment problem into two sub-problems: *phone call prediction*, and the *task assignment decision making based on the prediction.*

3.3 *EEMC* Framework and Skeleton Algorithm

As shown in Fig. 3.2, *EEMC* consists of two main phases: *Candidate User Identification based on Call Prediction* and *Two-step Decision Making Process for Task Assignment*. These two phases are designed to solve the *two sub-problems* for task assignment decision making, respectively. In the rest of this section, we will briefly describe each of the two phases.

3.3.1 Phase I - Candidate User Identification based on Call Prediction

Given an incoming call, *Phase I* of *EEMC* first checks if the caller is in the MCS participant list. If so, it will update the call traces of the *current caller*, and identify *if the current caller is a candidate for task assignment* through predicting her future calls. Phase I has a simple design to be implemented as a single core functional module:

- **Next-Call Prediction Model based on Accumulated Call Traces.** With historical call traces of the current caller as the input, a Predictive Model estimates the probability of the user placing another phone call in the *remaining time* (from the current time to the end of cycle).

If the *current caller* has a high probability of placing another call and has not yet received any task assignment in the current cycle, then *EEMC* deems that the user is a suitable candidate for task assignment and goes to the second step for task assignment decision making. If the *current caller* has received the sensing task assignment but hasn't returned the sensed result, then *EEMC* collects the sensed result from her. If she has already returned the sensed result or is not in the selected MCS participant list, then *EEMC* skips the call and exits the assignment process.

3.3.2 Phase II - Two-step Decision Making Process for Task Assignment

Given the *current caller* who has been identified as a candidate for task assignment (by Phase I), *Phase II* firstly decides 1) *if* EEMC *need make further task assignment(s)* and, if so, then 2) it decides if current caller should receive the task assignment. The Phase II design is based on two functional modules, one for each step of the decision making process:

- **Adaptive Pace Controller for Task Assignment.** Given the list of *participants already assigned* (A_k) and the list of *participants already returned* (R_k), *EEMC* estimates the probability of having a *missing number* ($N_e - |R_k|$) of potential returners ($A_k - R_k$) placing another call before the end of current sensing cycle. If the probability is higher than the given *success probability* P_s, then we decide the *tasks already assigned* are able to ensure *the expected number* of participants returning and *further task assignments are not needed immediately.* If the probability is lower than the given success probability, then *EEMC* goes to the second step for decision making of task assignment.

- **Near-optimal Decision Maker for Task Assignment.** Given the state and history of all known participants, *EEMC* identifies the *future candidate users* who haven't placed any call in the current cycle but who are likely to place two calls before the end of current cycle. Then, from this set of future users, *EEMC* predicts the users who have higher probability of placing two calls than the *current caller* placing another call. (We name this set the ***future-surer candidates***). With the sets of *potential returners* and *future-surer candidates* as inputs, *EEMC* estimates the probability of having a *missing number* of participants – from the two input sets — returning the sensed results. If the probability is higher than the given threshold (P_s), then there exists a sufficient number of better candidates in future; and *EEMC* skips the *current caller* and leaves the sensing task to future candidates. If the probability is lower than the given threshold, then *EEMC* assigns the sensing task to the *current caller.*

With the two steps described above, *EEMC* assigns tasks to the participants who have the "*higher probabilities*" of placing another call to return their sensing results, and stops making further task assignment immediately when it predicts the tasks already

Algorithm 1: The Skeleton of EEMC Algorithm

Input : M, k, A_k, R_k,U_i,c_j,t,$S_{k,t}$, S_1,...S_{k-1}, and P_s
Output: {true,false}–Assign or Not

```
1  begin
       /* Phase I: Candidate User Identification based on Call Prediction        */
2      update_Call_Model(U_i, t, k); // Predictive Model based on Accumulated Call
           Traces
3      if U_i ∈ A_k then
4          if U_i ∉ R_k then
5              collect_Sensing_Result(U_i, R_k);
6          end
7          return false;
8      end
       /* Phase II: Two-step Decision Making Process for Task Assignment         */
9      if |R_k| < N_e then
           // Step 1:  Pace Controller for Task Assignment
10         P_fullfill ← prob_fulfill(A_k, R_k, N_e, t);
11         if P_fullfill < P_s then
               // Step 2:  Near-Optimal Decision Maker for Task Assignment
12             if k ≤ M then
                   // Cold-start
13                 if P_{k,t}{x_i ≥ 1} > P_{k,t}{x_i ≥ 0} then
14                     A_k ∪ {U_i} → A_k;
15                     return true;
16                 else
17                     return false;
18                 end
19             else
                   // future-surer user based selection
20                 FS_{U_i} ← futureSurer(U_i, S_1..S_{k-1}, S_{k,t});
21                 P*_fullfill ← prob*_fulfill(A_k, R_k, FS_{U_i}, N_e, t);
22                 if P*_full < P_s then
23                     A_k ∪ {U_i} → A_k;
24                     return true;
25                 else
26                     return false;
27                 end
28             end
29         end
30     end
31 end
```

Symbols	Definitions
$S_{k,t}$	The set of participants who make/receive phone calls from the start of cycle k to t, where $t \in [t_0 + (k-1) * T, t_0 + k * T)$;
S_k	The set of participants who make/receive phone calls throughout the whole cycle k;
$C_{i,k,t}$	The number of calls made/received by user U_i from the start of cycle k to t, where $t \in [t_0 + (k-1) * T, t_0 + k * T)$;
$C_{i,k}$	The number of calls made/received by user U_i throughout the whole cycle k;
M	The MCS task consists of M cycles in a day;
$P_{k,t}\{x_i = n\}$	The probability of U_i making/receiving n calls from time t to the end of cycle k, where $t \in [t_0 + (k-1) * T, t_0 + k * T)$;
FS_{U_i}	The set of future-surer users of U_i, where U_i makes/receives a phone call at t of cycle k, $\forall U_j \in FS_{U_i}, P_{k,t}\{x_j \geq 2\} > P_{k,t}\{x_i \geq 1\}$;
$P_{fulfill}$	the probability of having at least a *missing number*$(N_e - \|R_k\|)$ of *potential returners* placing another call before the end of cycle;
$P^*_{fulfill}$	the probability of having at least a *missing number*$(N_e - \|R_k\|)$ of sensed results returned from *potential returners* and *future-surer candidates* $((A_k - R_k) \cup FS_{U_i})$;

Table 3.2: Variables used in *EEMC* Algorithms

assigned can secure the expected number of participants returning. Heuristically, the proposed method can minimize the total number of task assignments.

Following the above-mentioned two-phase framework, we design and implement the task assignment algorithm of *EEMC*. The skeleton of the *EEMC* algorithm is shown in Algorithm 1, where the variables are defined in Table 3.1 and 3.2. We will describe each module in the design of the *EEMC* algorithm in the following sections.

3.4 Next-Call Prediction Model based on Accumulated Call Traces

EEMC predicts the call of a mobile user dependent upon the periodicity of past calls in recorded call traces. Assume an MCS task parts one day into M sensing cycles. Given a sensing cycle k and the elapsed time t, we build a user U_i's call model in cycle k by mining U_i's call traces (including time-stamps and cell tower ids) in *corresponding cycles* of previous days. For instance, to predict the call of a user in the current sensing cycle from 08:00 to 10:00, all her past call records throughout the same period 08:00-10:00 will be adopted. Note that the calls made/received by U_i in the current cycle are likewise incorporated for her call prediction.

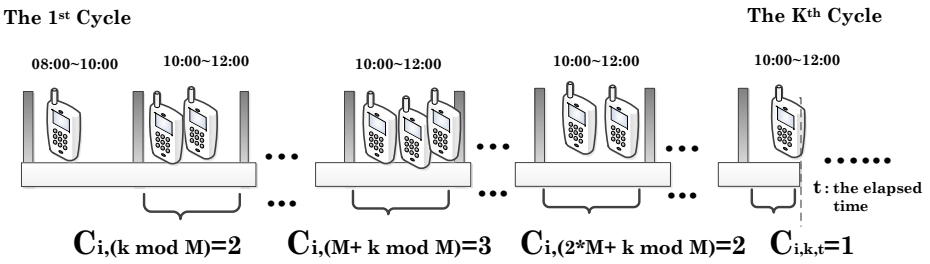


Figure 3.3: An Example: Estimating the Parameter with U_i's Accumulated Call Traces

3.4.1 Probabilsitic Model of Phone Calls

Assuming the call sequence follows an inhomogeneous Poisson process [87, 88], then the probability of a user U_i placing n phone calls from instant t to the end of cycle k can be modeled as:

$$P_{k,t}\{x_i = n\} = (\lambda_{i,k,t}\frac{\Delta t}{T})^n * e^{-\lambda_{i,k,t}\frac{\Delta t}{T}}/n!$$

where $\Delta t = (t_0 + K * T) - t$ denotes the *remaining time* from instant t to the end of the cycle, T is the sensing cycle duration, and $\lambda_{i,k,t}$ refers to the Poisson intensity.

3.4.2 Parameter Estimation using Accumulated Traces

According to the Poisson law and maximum likelihood estimation (MLE) [89], when $k \leq M$ the Poisson intensity $\lambda_{i,k,t} = C_{i,k,t}$ refers to the number of calls made/received by U_i from the start of cycle k to time t; when $k > M$, $\lambda_{i,k,t}$ is estimated as the average number of phone calls that a user U_i has placed in previous corresponding cycles, specifically it is modeled as:

$$\lambda_{i,k,t} = \frac{\sum_{1 \leq k' \leq \lfloor k/M \rfloor} C_{i,(k'*M+k \bmod M)} + C_{i,k,t}}{\lceil k/M \rceil} \tag{3.1}$$

where $C_{i,(k'*M+k \bmod M)}$ $(1 \leq k' \leq \lfloor k/M \rfloor)$ refers to the number of phone calls made/received by U_i in all previous *corresponding cycles* of cycle k (cycle k is included). For example, as shown in Figure 3.3, the sensing cycle k is from 10:00 to 12:00 in the fourth day of the MCS task. Then, $C_{i,k \bmod M} = 2$, $C_{i,M+k \bmod M} = 3$ and $C_{i,2M+k \bmod M} = 2$ stand for the numbers of phone calls made/received by U_i during the corresponding cycles in the first, second and third day respectively. As only one phone call has been made/received by U_i from the start of cycle k to the elapsed time t, *EEMC* counts the number of phone calls made in current cycle as $C_{i,k,t} = 1$. Thus, in this example, the Poisson intensity of U_i in the sensing cycle k is estimated to be $\lambda_{i,k,t} = \frac{(2+3+2)+1}{4} = 2$.

3.5 Adaptive Pace Controller for Task Assignment

In this section, we would like to introduce: 1) the *adaptive pace control mechanism* for task assignment, 2) the probability estimation used in *adaptive pace control mechanism* (i.e., estimating if the *missing number* of sensed results can be returned from *potential returners*), and 3) a *low-complexity algorithm* to reduce the time consumption of the probability estimation in the Adaptive Pace Controller.

3.5.1 Adaptive Pace Control for Task Assignment

Given the set of *potential returners* $(A_k - R_k)$, the *missing number* of sensed results $(N_e - |R_k|)$ and the instant time (t) in cycle k, we estimate:

- $P_{fulfill}$—*the probability of having at least* $(N_e - |R_k|)$ *potential returners placing another call before the end of cycle* k.

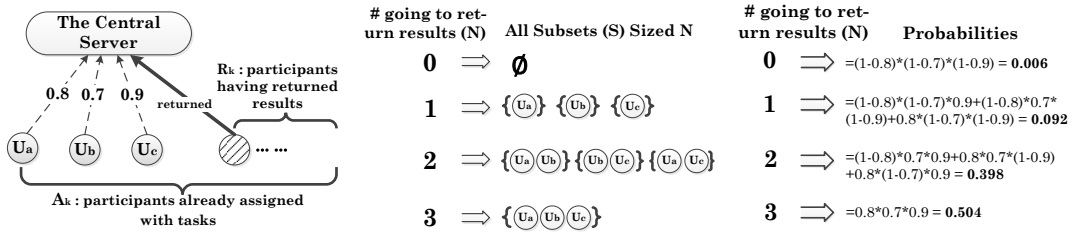
With $P_{fulfill}$ defined and the success probability threshold P_s given, *EEMC* controls the task assignment in a straight-forward way—if $P_{fulfill} \geq P_s$ then further task assignments are not needed immediately and *EEMC* stops making further task assignments; if $P_{fulfill} < P_s$ then further task assignments are still needed and *EEMC* moves to the next step for task assignment decision making (please see also in the pseudo code between line 9-11 of Algorithm 1). In this way, the key is to calculate $P_{fulfill}$.

3.5.2 Probability Estimation for Adaptive Pace Control

In order to estimate $P_{fulfill}$, we first define $P\{X_{k,t}(A_k - R_k) = N\}$ as the probability of having N out of $|A_k - R_k|$ *potential returners* placing at least another call before the end of cycle k, where $N \leq |A_k - R_k|$. To calculate this probability, we need to first enumerate all possible subsets of N participants from $A_k - R_k$. For each subset of N participants, we need to calculate the probability of having N participants placing at least another single call before the end of current cycle. Finally, as with the example shown in Figure 3.4, $P\{X_{k,t}(A_k - R_k) = N\}$ provides an estimation of the sum of probabilities for all possible subsets, and it is calculated as specified in Equation 3.2.

$$P\{X_{k,t}(A_k - R_k) = N\} = \sum_{\forall s \subset (A_k - R_k)}^{|s|=N} \prod_{\forall U_m \in s} P_{k,t}\{x_m \geq 1\} \prod_{\forall U_m \in A_k - R_k - s} (1 - P_{k,t}\{x_m \geq 1\}) \tag{3.2}$$

In this way, $P_{fulfill}$ is estimated as the sum of $P\{X_{k,t}(A_k - R_k) = N\}$, where N is an integer ranging from the *missing number* of sensed result $(N_e - |R_k|)$ to the total



(a) The example of A_k and R_k (b) Enumeration of Subsets (c) Computation of Probabilities

Figure 3.4: The Example of $P\{X_{k,t}(A_k - R_k) = N\}$ Computing (Best Viewed in Digital Format)

number of *potential returners* ($|A_k - R_k|$) (see Equation 3.3).

$$P_{fulfill} = \begin{cases} 0, & |A_k| < N_e \\ \sum\limits_{N \geq N_e - |R_k|}^{N \leq |A_k - R_k|} P\{X_{k,t}(A_k - R_k) = N\}, & |A_k| \geq N_e \end{cases} \tag{3.3}$$

Please note that, when the number of participants already assigned is less than the expected number of sensed results (i.e., $|A_k| < N_e$) then it is not possible to collect the pre-defined number of sensed results, thus $P_{fulfill} = 0$. For the low-complexity $P_{fulfill}$ calculation, please refers to **Appendix A.1.1**.

3.6 Near-Optimal Decision Maker for Task Assignment

Given the incoming call from one of the MCS participants and previous call records, the key algorithms of this step include 1) identifying all *future-surer candidates*, 2) estimating if the *missing number* of sensed results can be returned from *future-surer candidates* and *potential returners*, and 3) the Near-Optimal task assignment decision making.

3.6.1 Identifying *Future-surer Candidates*

Given the current caller U_i, we consider U_m as a *future-surer candidate* if:

- U_m has placed calls in previous *corresponding cycles* but hasn't placed any call in the current cycle, i.e., $U_m \in S_1 \cup S_2 \cdots \cup S_{k-1} - S_{k,t}$, **and**
- U_m has a higher probability of placing at least two calls than U_i placing at least another call, i.e., $P_{k,t}\{x_m \geq 2\} > P_{k,t}\{x_i \geq 1\}$.

Putting all the *future-surer candidates* together with regard to U_i, they are denoted as FS_{U_i}.

Algorithm 2: Identifying Future-Surer Candidates

Input : $S_1, S_2 \ldots, S_{k-1}, S_{k,t}$ and U_i
Output: FS_{U_i} : the set of future-surer users for U_i

```
FS_{U_i} ← ∅;
for U_l ∈ S_1 ∪ S_2 ··· ∪ S_{k-1} − S_{k,t} do
    if P_{k,t}{x_l ≥ 2} > P_{k,t}{x_i ≥ 1} then FS_{U_i} ∪ {U_l} → FS_{U_i}
end
return FS_{U_i};
```

3.6.2 Estimating if the Missing Number of Sensed Results can be returned from Future-surer Candidates and Potential Returners

Given the set of *future-surer candidates* FS_{U_i}, the set of *potential returners* $(A_k - R_k)$, and the *missing number* of sensed results $(N_e - |R_k|)$, we estimate $P^*_{fulfill}$ as the probability of having at least the *missing number* of sensed results $(N_e - |R_k|)$ returned from the *potential returners* and *future-surer candidates* $((A_k - R_k) \cup FS_{U_i})$ before the end of cycle k. Apparently the estimation of $P^*_{fulfill}$ depends on the probability of each U_m returning the sensed results $(U_m \in (A_k - R_k) \cup FS_{U_i})$ before the end of cycle k, each U_m's returning probability can be computed using Equation 3.4.

$$P'_{k,t}(U_m) = \begin{cases} P_{k,t}\{x_m \geq 1\}, U_m \in (A_k - R_k) \\ P_{k,t}\{x_m \geq 2\}, U_m \in FS_{U_i} \end{cases} \tag{3.4}$$

In the case of $U_m \in (A_k - R_k)$ (belonging to the *potential returner set*), $P'_{k,t}(U_m)$ is modeled as the probability of U_m placing at least another call before the end of cycle k. In the case of $U_m \in FS_{U_i}$ (belonging to the *future-surer candidate set*), then $P'_{k,t}(U_m)$ is modeled as the probability of U_m placing at least two calls before the end of cycle k. Given each user U_m's returning probability $P'_{k,t}(U_m)$, similar to the estimation of $P_{fulfill}$ in Equation 3.3, $P^*_{fulfill}$ can be computed using Equations. 3.5 and 3.6, where $\mathrm{P}\{X^*_{k,t}(FS_{U_i} \cup (A_k - R_k)) = N\}$ refers to the probability of N sensed results being returned from *future-surer candidates* and *potential returners*.

$$P^*_{fulfill} = \begin{cases} 0 & , |A_k \cup FS_{U_i}| < N_e \\ \sum\limits_{N \geq N_e - |R_k|}^{N \leq |(A_k - R_k) \cup FS_{U_i}|} P\{X^*_{k,t}(FS_{U_i} \cup (A_k - R_k)) = N\}, & |A_k \cup FS_{U_i}| \geq N_e \end{cases} \tag{3.5}$$

$$P\{X^*_{k,t}(FS_{U_i} \cup (A_k - R_k)) = N\} = \sum_{\forall s \subset (A_k - R_k) \cup FS_{U_i}}^{|s|=N} \prod_{\forall U_m \in s} P'_{k,t}(U_m) \times \prod_{\forall U_m \notin s} (1 - P'_{k,t}(U_m)) \tag{3.6}$$

For the low-complexity $P^*_{fulfill}$ calculation, please refers to **Appendix A.1.2**.

3.6.3 Near-optimal Task Assignment Decision Making

With $P^*_{fulfill}$ computed and the threshold P_s, *EEMC* assigns a task to the current caller (U_i) if $P^*_{fulfill}$ is lower than P^*_s. The pseudo code of Near-Optimal task assignment decision making is shown in lines 12-28 of Algorithm 1.

Please note that, according to our proposed *Future-surer Candidates Identification* listed in Algorithm 2, it is impossible to discover any *future-surer candidates* in the sensing cycles of the first day in an MCS task (i.e., $k \leq M$). Thus, there needs a method to ***cold-start*** the proposed *Near-optimal Decision Maker* in the first day of an MCS task. Rather than comparing the current caller with potential users in the future, we propose a method to make the task assignment decision making based on the *current caller's* next-call probability alone. As shown in lines 12-19, when $k \leq M$, this step decides to assign a task to the current caller U_i, if $P_{k,t}\{x_i \geq 1\} > P_{k,t}\{x_i = 0\}$. If U_i doesn't have a higher probability of placing another call before the end of cycle, then this step skips the current caller.

3.7 Experimental Setups

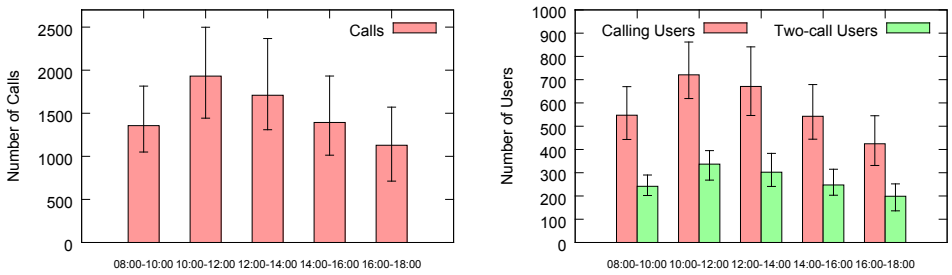
In this section, we introduce two baselines for comparison with *EEMC*, then present an overview of our dataset and experiment configuration.

3.7.1 Baseline Methods and Parameter Settings

In this section, we present the configurations and setups of our proposed baselines.

- ***Greedy -*** The most obvious method for task assignment to ensure a predefined number of sensed results is the ***Greedy method***, which assigns the sensing task to each new calling participant, until the expected number of sensed results are returned (i.e. until $|R_k| = N_e$). This baseline method provides an upper bound of total task assignments to ensure that the expected number of participants return data.

- ***Pace -*** As there is a delay between task assignment to a participant and the return of the sensed result from the participant (through making another



(a) Number of Calls

(b) Number of Calling Users and Two-call Users

Figure 3.5: Statistics of Evaluation Traces in D4D Data Set

call), redundant tasks could be assigned when the expected number of results have been returned. Indeed, if the expected number of returned results can be predicted in advance, based on our proposed *Adaptive Pace Controller* module, the task assignment process could terminate earlier to avoid some unnecessary task assignment. The task assignment strategy leveraging the *adaptive pace controller for task assignment* module is defined as ***Pace-controller-based method*** (or ***Pace*** in short).

The comparison between **Greedy** and **Pace** shows whether our proposed *Pace controller* can stop making further task assignments when the tasks already assigned are sufficient to guarantee the expected number of participants returning. Furthermore, compared to the Pace method, *EEMC* assigns tasks considering not only participants with tasks already assigned, but also the future callers/receivers. Thus, the comparison between **Pace** and *EEMC* demonstrates the improved performance of our proposed *Optimal Task Assignment Decision Making* method with respect to the minimization of the total number of task assignments. In all experiments, we set the threshold P_s=99.99% for the evaluation of *Pace controller-based* baseline and *EEMC*.

3.7.2 Dataset and Experiment Setups

The "Data for Development" (D4D) project collected 4-months of Call Detail Records (CDR) from Orange Telecom subscribers in the Ivory Coast, nationwide. Each CDR record includes the calling time, the cellular tower where the call was made/received, and the identifier of the mobile phone user. The D4D data set has been split into consecutive two-week periods. In each time period, 50,000 users are *randomly selected* from all subscribers in the Ivory Coast. All selected users are assigned with anonymized identifiers. Thus in this study, we assume that each MCS task lasts for two weeks. For each participant, we can retrieve her call traces in the current MCS task but cannot link to her previous records. As we discussed in Section 3.1, the

mobile phone users inside the D4D data set perfectly satisfy the privacy constraints for MCS participants. The detailed experiment settings are as follow:

1. *Sensing Cycles* - We evaluate *EEMC* when monitoring the CBD of Abidjan (shown in Fig. 6.2) from Monday to Friday every week (holidays excluded). Each sensing cycle lasts two hours; and we sense only in the working hours from 08:00 to 18:00 of a day. Thus, we split a working day into 5 equal-length sensing cycles (i.e. 8:00-10:00, ..., 16:00-18:00).

2. *Participants* - In every two-week period, 2000-3000 mobile phone users recorded in our dataset would place phone calls in the target area (i.e., approximately 0.3% local mobile subscribers living in the target area). We assume them to be participants in our MCS task. To further introduce the call behaviors of these participants, we count the numbers of phone calls, calling participants and frequent users (those with two or more phone calls in a sensing cycle). The average/minimum/maximum numbers of these are shown in Fig. 3.5. It shows that 1) on average, 1200-2000 calls will be received/made in the target region per sensing cycle, 2) on average, no more than half the calling participants (i.e., approximately 200 participants) will place another call in a sensing cycle, and 2) at least 136 users will place two or more phone calls in a sensing cycle.

3. *The Expected Number of Sensed Results* - Consequently, we cannot ensure the expected number of participants returning in each of sensing cycles, if we expect more than 136 participants to return. Thus, for our experiments, we set the expected number of returned participants in each cycle N_e to be evenly distributed from 10 to 130, i.e., $N_e = 10, 20, 30, ...130$.

In the following sections, we will introduce the evaluation results based on the experiment setups specified above.

3.8 Evaluation Results

In this section, we present and compare the evaluation results of *EEMC*, Pace and Greedy methods:

1. In section 3.8.1, we show the overall performance comparison of *EEMC*, Pace and Greedy, including the average/maximal/minimal number of task assignments and returned participants in each sensing cycle.

2. In section 3.8.2, we extract and present the performance of *EEMC* at the cold start period.

3. In section 3.8.3, we examine in detail the execution of the three algorithms on a subset of the experimental data in order to illustrate their behaviors. Through

a case study of *EEMC*, Pace and Greedy, we analyse how *EEMC* assigns tasks step by step in a sensing cycle.

4. In section 3.8.4, we estimate how much energy our proposed *EEMC* scheme can save in data transfer, compared to the commonly-seen 3G-based MCS schemes.

The results above will combine to show the excellence of *EEMC* with respect to minimizing the total number of task assignments and saving overall energy consumption whilst guaranteeing the expected number of participants returning results.

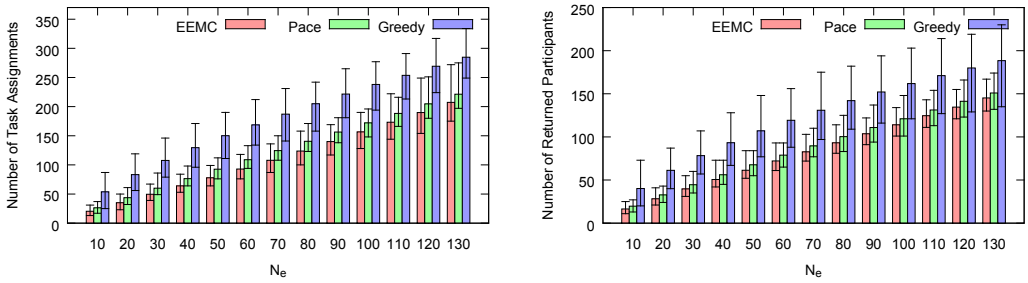
3.8.1 Performance Comparison

In Figure 4.5, we present the average/ minimal/ maximal numbers of task assignments and returned participants for *EEMC*, Pace and Greedy in each sensing cycle with varied N_e (10 to 130).

1. **Number of Returned Participants.** The primary constraint of our work is to ensure the expected number of participants returning their sensing results. Figure 4.5b shows that, for either *EEMC*, Pace or Greedy, the minimal number of returned participants in each sensing cycle is equal to or greater than the expected number (N_e). It means, with any of these methods, the MCS tasks can be successfully fulfilled in each of sensing cycles. However, in all the cases the number of returned results is bigger than the expected number N_e, even though the number of returned results for *EEMC* is 3.8% - 17% less than Pace and 23% - 59% less than Greedy on average.

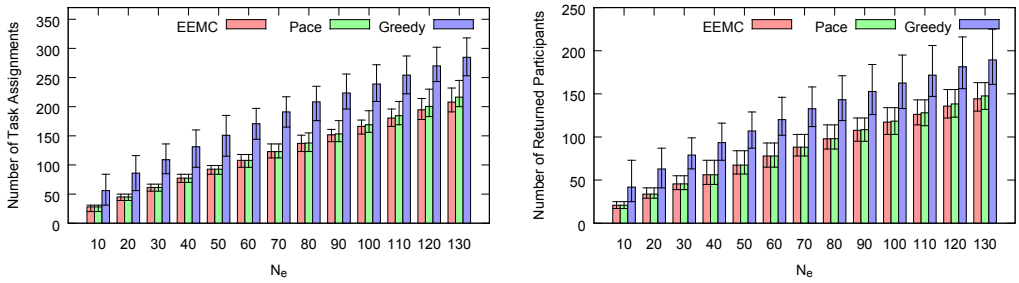
2. **Number of Task Assignments.** Furthermore, the optimization goal of *EEMC* is to minimize the total number of task assignments. Figure 4.5a shows clearly that *EEMC* assigns less tasks to participants than Pace and Greedy. On average, *EEMC* reduces task assignments by 6%-23% when compared to Pace, and it reduces task assignments by 27%-62% when compared to the Greedy method.

For the Greedy method, it is obvious that the delay between the task assignment to the participant (*who returns the* N_e^{th} *sensed result in this cycle*) and the return of the sensed result causes a large number of redundant task assignments and unnecessary returned results; while the Pace method may assign tasks to the participants not placing another call in the sensing cycle, which leads to high redundant task assignments. In contrast, for *EEMC*, the reason for the redundant task assignment is mainly due to the inaccurate call prediction with limited number of call traces. However, in terms of the number of task assignments and returned results, *EEMC* still outperforms all other methods in all conditions. In summary, we can conclude that the overall performance of *EEMC* is the best among the three schemes. It ensures data collection from the expected number (10–130) of participants and assigns the minimal number of redundant tasks among all evaluated schemes.



(a) Task Assignments (b) Returned Participants

Figure 3.6: Comparison of Task Assignments and Returned Participants: EEMC vs Pace vs Greedy



(a) Task Assignments (b) Returned Participants

Figure 3.7: Number of Task Assignments and Returned Participants in Cold Start Period

3.8.2 Cold-start Performance

As discussed in Section 3.6.3, *EEMC* needs to cold-start its Near-Optimal decision making module in the first day of every MCS task (namely *cold-start periods*). Figure 3.7a illustrates the number of task assignments per cycle in the cold-start periods, while Figure 3.7b presents the number of returned participants. During the cold-start periods, *EEMC* slightly outperforms Pace but performs worse than the average in normal periods. It is because the Near-Optimal decision making module assigns tasks to callers with "*maximal probabilities*" to return their sensing results after the cold-start period. Pace also performs worse during the cold-start periods, due to the inaccuracy of probability estimation at the beginning of MCS tasks.

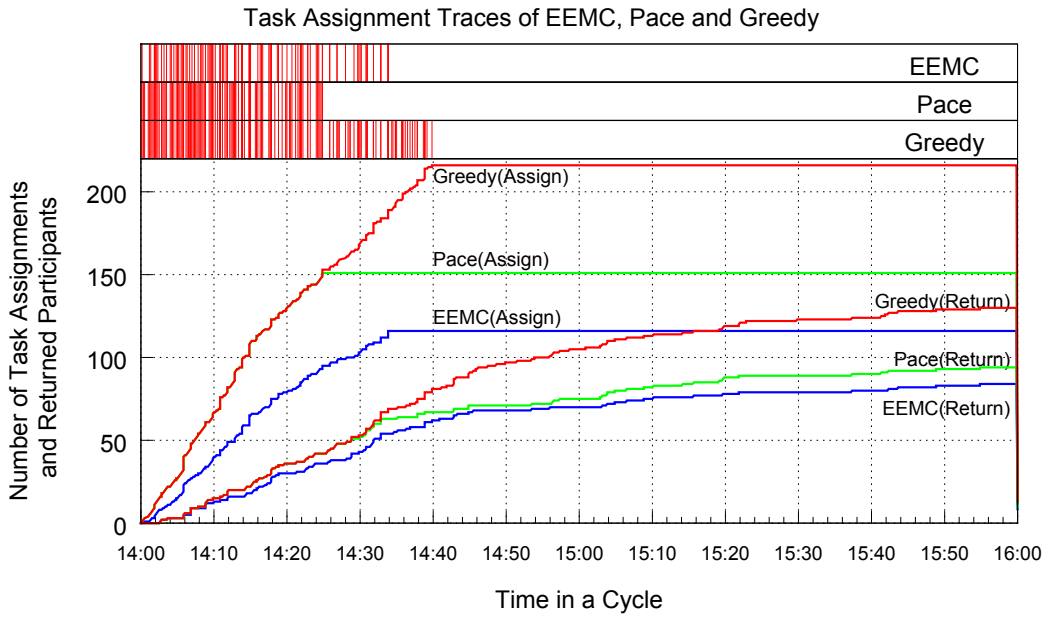


Figure 3.8: Number of Task Assignments and Returned Participants varying with Time in the Cycle of $10:00-12:00$, 15 Dec 2011 (Best Viewed in Color)

3.8.3 Case Study and Analysis

To verify whether each proposed algorithm works as designed using the real-world data sets, we investigate how *EEMC* assigns tasks inside a single (typical) sensing cycle. We choose the sensing cycle of $14:00-16:00$, 15 Dec 2011 for the case study and set the expected number of returned participant as 80 (i.e., $N_e = 80$). Please note that this sensing task is not in the cold start period.

In Figure 3.8, we count the number of task assignments and returned participants varying against time inside the chosen sensing cycle and visualize the process of task assignments. We evaluate all three schemes, observing that:

- Comparing Greedy with Pace, Pace assigns tasks to the same calling participants as Greedy but stops assigning new tasks at 14:24 when 42 participants return their sensed results, while Greedy keeps assigning new tasks until 14:39 when a total of 80 participants return their sensed results. The Pace method stops 15 minutes earlier than the Greedy method, which causes 65 less redundant task assignments and 36 less unnecessary returned results. Such improvement is contributed by our proposed *Adaptive Pace Controller* which stops assigning a new task when it estimates that the tasks already assigned are enough to fulfill the minimum requirement.

- Comparing *EEMC* with Pace, *EEMC* gives up assigning tasks to calling partic-

Table 3.3: Data Transfer Energy Consumption Estimation

Schemes	Energy Consumption
3G-based scheme	$N_e * (12 + 12) = 24 * N_e$
Parallel+3G-based scheme	$N_e * (3 + 12) = 15 * N_e$
EEMC Pace and Greedy	$\|A_k\| * 3 + \|R_k\| * 3$

ipants even in the beginning of the cycle; because it predicts there are sufficient number of future users who have higher probabilities to place two calls before the end of current cycle. We can see that *EEMC* holds the tasks and leaves them to *future-surer users*. In this way, *EEMC* stops making new task assignments later (at 14:33, when 54 participants return) but assigns less tasks (35 less) than the Pace to fulfill the task. Since *EEMC* always choose the users with higher probabilities to place two calls, it can guarantee the expected number of participants returning after assigning a smaller number of tasks.

Our analysis suggests that it is reasonable to conclude that all three algorithms in our comparison work as designed on the real world data sets.

3.8.4 Energy Conservation Comparison

With the number of task assignments and returned results obtained, it becomes possible to estimate the energy consumption of *EEMC* and corresponding baselines. In this section, we would like to estimate how much energy our proposed *EEMC* scheme can save in data transfer, compared to the following schemes:

- ***3G-based Scheme***: receives the task assignment by establishing a new 3G connection, and returns the sensed results by establishing another 3G connection.

- ***Parallel+3G-based Scheme***: receives a task assignment when the participant places a phone call through parallel data transfer, and returns the sensed results by establishing a new 3G connection.

These two schemes do not need redundant task assignments (i.e., both methods can secure N_e participants returning their sensed results through assigning tasks to N_e participants), since all the participants can return the sensed results via a new 3G connection by using these two schemes. Table 4.2 lists the overall energy consumption estimation formulas in data transfer for all the schemes; and these formulas are based on:

1. the common observations reported by existing literature [17, 51, 42, 85] measuring on the energy consumption of N95 and Android phones, and

Table 3.4: Energy Consumption Comparison: 3G-based vs Parallel+3G-based (P+3G) vs *EEMC* vs Pace vs Greedy

N_e	**3G (J)**	**P+3G (J)**	***EEMC* (J)**	**Pace (J)**	**Greedy(J)**
10	240	150	110.37	138.00	**281.48**
20	480	300	190.18	229.32	**433.75**
30	720	450	268.15	313.75	**557.88**
40	960	600	343.77	397.35	**668.28**
50	1200	750	417.66	480.78	**771.35**
60	1440	900	494.98	563.03	863.82
70	1680	1050	571.48	642.29	953.82
80	1920	1200	650.74	722.37	1040.85
90	2160	1350	730.73	801.59	1120.88
100	2400	1500	811.95	879.31	1199.57
110	2640	1650	893.13	958.64	1274.27
120	2880	1800	972.88	1037.97	1347.80
130	3120	1950	1057.31	1116.76	1419.49

2. the assumption that the data packets for task assignment or sensed results are small (less than 10KB each).

Considering the MCS applications such as air quality monitoring and environment noise monitoring, this assumption is reasonable and the energy estimated using the formula could serve as a reference for comparison purposes.

Table 4.3 shows each scheme's average energy consumption per sensing cycle as N_e varies. *EEMC* outperforms all the other schemes. Specifically, it can save 54%–66% energy compared to the 3G-based scheme; It can save 26%–46% energy compared to the Parallel+3G-based scheme. Note that these evaluations are based on small number of expected sensed results (i.e., $N_e \leq 130$). If an MCS task needs more participants to collect sensed data and there are more sensing cycles per day, the total energy saving will be much more significant. Interestingly, if we compare the *EEMC*, Pace, Greedy with the Parallel+3G-based scheme, we can see that *EEMC* outperforms all the other schemes in all the conditions, but the Greedy method consumes more energy than the Parallel+3G-based scheme when $N_e < 60$. In summary, all the evaluation results show the effectiveness of *EEMC* in saving energy consumption in data transfer for both individual participants and the whole crowds.

3.9 Discussion

In this section, we discuss issues which are not reported or addressed in this work due to space and time constraint; these issues are planned for ongoing and future work.

Prediction and Parameter Adaption: The performance of *EEMC* depends

on the accuracy of call prediction and the parameter setting used in the algorithm. In this study we currently use a simple prediction algorithm and a fixed set of parameter settings in all the sensing cycles, in future work we plan to study adaptive task assignment pace control and decision making strategies, and design advanced call/mobility prediction methods.

Two-call-based Data Transfer: Our research assumes a participant needs two calls to receive task assignment and return her sensed result. This assumption is made because being involved in a call risks interfering with sensing; a good example of this is if the sensors are measuring noise. However, many sensor tasks can be safely carried out during a call; and in such case only one call is likely to be needed.

Sensing Coverage: In this research, we have not proposed any techniques to consider the coverage of mobile crowdsensing. In our future work, we will study the coverage of users by obtaining their mobility traces.

Aggregating Multiple Energy-efficient Strategies: In addition to piggybacking 3G data transfer over 3G calls, other data transfer methods, e.g., transferring data via WiFi, also consumes less energy when compared to common 3G-based solutions. Furthermore, there exist a wide range of techniques, such as adopting low-power consumption sensors or energy-efficient sensing techniques, that can save energy in the MCS tasks. In our future work, we intend to study an integrated MCS framework aggregating multiple energy-saving strategies to minimize the energy consumption in a holistic manner.

Energy Consumption vs Battery Life: For a smartphone, the energy consumption to receive a task assignment and return the sensed result is no more than 0.7% of its battery's energy reserve capacity (e.g., Nokia N95 with 950 mAh battery). However, even given this small percentage, our proposed energy saving mechanism can have a significant impact on individual users. For example, suppose active participants are selected for 5 cycles a day, *EEMC* can save 2.6% of battery usage, which is enough to answer the last call of an individual user before battery drain or to put the phone in standby for one more hour.

Fairness in allocation of tasks: Users may be more motivated to join the sensing crowd if they know that energy resources are used fairly. In other words, that tasks are distributed as equally as possible amongst the crowdsensing members. They may also consider it unfair if they are allocated tasks when their mobile phone batteries are below a certain threshold value.

Chapter 4

EMC^3: Energy Efficient Data Transfer for Mobile Crowdsensing under Full Coverage Constraint

Contents

4.1 Introduction 58
4.2 Problem Statement 61
4.3 EMC^3 Framework and Core Algorithms 62
4.3.1 Call/Mobility Prediction 63
4.3.1.1 Modeling Call Patterns 64
4.3.1.2 Modeling Mobility Patterns 64
4.3.2 Overall Task Assignment Pace Control 64
4.3.2.1 Estimating $P_{fulfill}$ 65
4.3.2.2 Estimating P_{cover_l} 66
4.3.3 Sub-optimal Task Assignment Decision Making 66
4.3.3.1 Identifying future-surer candidates 66
4.3.3.2 Estimating if the Missing Number of Sensed Results can be returned from Future-surer Candidates and Potential Returners 67
4.3.3.3 Estimating if all Desired Cell Towers can be covered by Future-surer Candidates and Potential Returners 67
4.3.3.4 Sub-optimal Task Assignment Decision Making 68
4.4 Evaluation 68
4.4.1 Baseline Methods and Parameter Settings 69
4.4.2 Dataset and Experiment Setups 69
4.4.3 Performance Evaluation 72
4.4.3.1 Performance Comparison based on CBD Traces 72

4.4.3.2 Performance Comparison based on Residential District Traces . . . 73
4.4.3.3 Case Study and Analysis . . . 74
4.4.4 Energy Conservation Comparison . . . 75
4.4.5 Real-time Performance Analysis . . . 77
4.5 Discussion . . . 79

4.1 Introduction

The previous work *EEMC* studies an mobile crowdsensing framework that intends to assign sensing tasks to a minimal number of participants, while ensuring at least a predefined number of participants returning sensed results from the target region in each sensing cycle. However, in terms of sensing data quality, ensuring a minimum number of participants returning sensed results might not be a good sensing data quality criterion, especially for full-coverage-constrained MCS applications where the target region is divided into a set of subarea and the MCS application is required to collect at least one sensed result from each subarea in each sensing cycle.

In this work, we propose EMC^3—an energy-efficient mobile crowdsensing framework *reducing* individual energy consumption caused by MCS data transfer, reducing the total incentive payment and overall energy consumption by *minimizing* the number task assignments, while *ensuring* at least a predefined number of participants returning sensed results and *at least one sensed result returned from each subarea* of the target region in each sensing cycle.

In order to save the energy of the individual MCS data transfer, EMC^3 adopts the piggybacked energy-efficient MCS data transfer strategy proposed in *EEMC*. Then, this research is based on following assumptions and settings:

- Only when a user places calls, the device could receive sensing task assignment; Only when another call comes before the end of cycle, it could return sensed results to the server (in this work we use the *mobile device*, *mobile user* and *participant* interchangeably);
- In each cycle, one participant can receive task assignment and upload results at most once;
- In the starting cycle of each MCS task, due to user identity anonymization, there are no historical call or mobility traces for any user from other or previous MCS tasks. All users only accumulate call and mobility traces within one MCS task.

Based on the above assumptions and observations, the research objective of this work is to fulfill the following three goals in each cycle of the MCS task:

- *Ensure an expected number of participants returning the sensed results.*

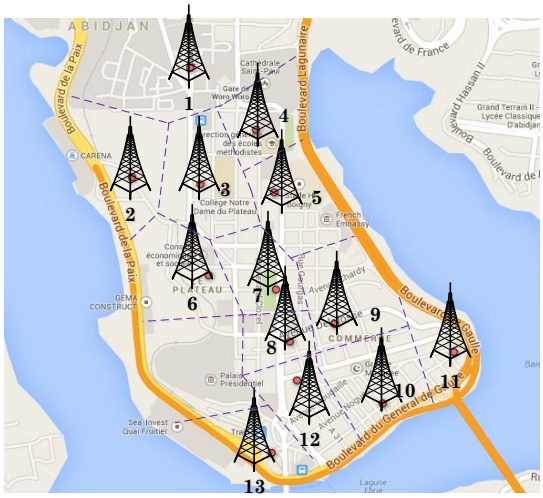


Figure 4.1: Cell Towers in the Abidjan CBD Area

- *Make sure that the returned sensed results fully cover the target sensing area.*
- *Minimize the number of total task assignments to reduce overall energy consumption.*

To further clarify the research goals, let's consider the following use case: *In the CBD area of Abidjan city in Cote d'Ivoire (around 7 km^2), there are 13 cellular towers installed in the 3G network as shown in Fig. 1. The city government, with the help of a telecom operator, launches a series of MCS tasks leveraging the 3G cellular network infrastructure. One of the MCS tasks is air quality monitoring in the CBD area, it requires to update the air quality to the citizens of Abidjan once every 2 hours (cycle) and the task lasts for 2 weeks. In order to provide reliable measurements, the application needs to get sensed results from at least 40 mobile users, covering all 13 cellular towers in each cycle.* Please note that, in the considered use case and the rest of this work, we use cell towers as the coverage metrics, primarily due to two reasons: 1) The cell tower IDs of mobile phones are accessible in call logs, even though the cell tower is not the right coverage metrics for many MCS applications, the mobile phone call logs with cell tower as coverage metrics are used to illustrate the basic idea of handling coverage constraint problem in MCS applications; 2) For MCS applications such as urban air quality monitoring [21], noise level monitoring [90], etc., covering all the cell towers in a given region ensures that each part of the given area is measured with certain guarantee, even though the sampled granularity in terms of cell tower may not be the best choice. If it could be characterized more precisely, the proposed approach could be easily adapted.

With the above research goals and use case, the key issues in designing the MCS framework include:

1) Identify "candidate users" who might place two or more calls, and predict which subarea each user might cover in each MCS cycle. As only the users placing two calls

can fulfill sensing task using *parallel transfer*, and some candidate users must cover the low-density call subarea, thus it's necessary to choose the right candidates based on call and mobility prediction of the current caller, to minimize redundant task assignments. Apparently, assigning sensing task to users placing one call in a cycle or to candidate users only from high-density call subareas would lead to redundant task assignments, causing big overall energy consumption.

2) Given the arrived call sequence at certain instant of an MCS cycle, estimate if the number of users assigned could expect the predefined number of returned results and cover all the subareas. As the users receiving task assignment need to wait till the next call to return sensed results, thus there is a delay between assigning tasks and receiving the expected number of results from the target area, so it's necessary to make predictions and stop unnecessary task assignments.

3) If further task assignments are still needed to achieve the goal of getting expected number of returned results and full coverage, we need to decide if the sensing task should be assigned to the current one or the future candidates. As there are more valid candidates than needed and candidates from low-density call subareas might appear late in one cycle, we should decide the task assignment based on whether the current candidate or future candidates have higher probability of meeting the three goals.

4) Ensure the goals to be met despite the time-varying and inaccurate nature of all probability estimations. As the candidate user selection and task assignment are all based on future call and mobility predictions, while all those predictions are based on probability estimations which might not be accurate. For example, both the future call and mobility predictions are based on the historical traces, in the first MCS cycle, the prediction accuracy for both call and mobility could be very low, this will definitely cause sub-optimal decisions, leading to redundant task assignments. Fortunately, the estimation of all parameters is carried out with each incoming call, with continuous monitoring and adjustment, the system is designed to adapt itself to get both the expected number of sensed results and the full coverage, filtering out a lot of unnecessary task assignments.

In summary, the main contributions of this work are:

1) We formulate the problem of energy saving in data transfer of MCS tasks for both individual and all participants, with consideration of privacy issue as well as full coverage constraint. To the best of our knowledge, this is the first work addressing this issue. In particular, we propose to leverage the *parallel transfer* and *delay-tolerant mechanism* to achieve the energy saving purpose in MCS applications.

2) We develop a *three-step decision making process* and the related algorithms for effective task assignment in MCS applications. Specifically, we first *identify "candidate users"* who might place two or more calls and predict which subarea they might cover in each MCS cycle; Then we *judge if sufficient task assignments have been made* by considering if the number of assigned users could expect to return a pre-defined number of sensed results and cover all the target area; Finally, we *decide if*

a new sensing task should be assigned to the current one or a future candidate, based on whether the current candidate or the future candidate has higher probability of meeting the three goals.

3) Through extensive evaluation of our proposed algorithms on the real world dataset D4D [15], which contains 4-month call records of 50, 000 users from Cote d'Ivoire, we verify that our proposed MCS framework EMC^3 can ensure the expected number of participants returning their sensed results with full coverage and much less redundant task assignments than baseline approaches. Through leveraging parallel transfer over 3G calls, EMC^3 reduces around 75% energy consumption in data transfer for a returned participant and 43% - 68% overall energy consumption in data transfer for MCS applications, such as air quality or noise monitoring at the Abidjan CBD area, compared to the traditional 3G-based scheme.

4.2 Problem Statement

With the observations, assumptions and research goals elaborated in the introduction, the essence of the research problem of this work is to determine if a task assignment should be made, given an incoming call and historical call and location traces of a specific MCS task, in order to obtain a pre-defined number of returned sensed results with minimum number of task assignments under the full coverage constraint and given assumptions. While the number of task assignments and returned results are easy to count, we need to define what the full coverage of target area means.

In this work, we say that a cell tower is covered by a user when she places a call receiving a task assignment or returning sensed results to the server in the cell tower. If a user places one call in one cell tower for receiving a task assignment and another call in another cell tower for returning the sensed results, then these two cell towers are said to be covered by the user. Hence, the full coverage means that all the cell towers in the target area should be covered by at least one call for receiving task assignment or returning sensed results. Please note that the cell towers traversed by the user between the two calls are not counted in this work. In the rest of this work, we name the participant who covers a cell tower as the ***covering participant*** for the cell tower. With all the above definitions, we formally formulate the MCS task assignment problem in EMC^3 as follows:

Given an MCS task with starting time t_0, sensing cycle duration T, the expected number of collected sensing data N_e from each sensing cycle, and a cover region pre-defined by a set of cell towers T_{WR}; Given the incoming call and all previous call traces (including the time stamped calls and cell towers associated) in the MCS task, the elapsed time t in current cycle k, we denote A_k as the set of participants who have been assigned with sensing tasks since the start of cycle k, R_k as the set of participants who have returned sensed results, and $cover_k$ as the set of cell towers that have been covered, where apparently R_k is a subset of A_k and $cover_k$ is a subset of T_{WR}. Our problem is to decide if an MCS sensing task should be assigned to the

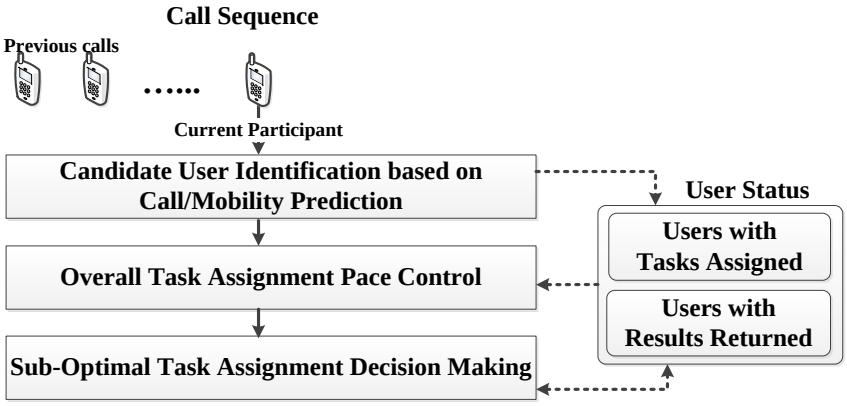


Figure 4.2: The EMC^3 Framework

current caller, with the objective to

$$\text{minimize } |A_k|, \text{ subject to } |R_k| \geq N_e \textbf{ and } cover_k \equiv T_{WR}$$

by the end of cycle k. It is worth noting that we do not know when and where a participant would place a phone call in advance but there are sufficient number of calls covering the cell towers of the target area.

4.3 EMC^3 Framework and Core Algorithms

EMC^3 follows a centralized MCS system approach where a central server continuously monitors all the participants' calling activities in the target region and decides if a user should receive sensing task assignment for each incoming call. As shown in Fig. 5.2, EMC^3 consists of three main components, i.e., *candidate user identification*, *task assignment pace control*, and *sub-optimal task assignment decision making*; These three functional modules correspond to the *three-step working process* of EMC^3, respectively. In addition to the three functional components, EMC^3 takes the previous call traces (including the current cycle and previous cycles) as input, it also keeps the user list with task assignments as well as the user list with sensed results returned for *task assignment pace control* and *sub-optimal task assignment decision making*. In the following, we will briefly describe each of the three functional components:

Candidate User Identification based on Call/Mobility Prediction. Given an incoming call, the *candidate user identification* module first updates the call records for the user. Based on the user's historical time-stamped call and location records, the module can predict the probability of having future calls and the associated cell towers before the end of the cycle. If the user has a high probability of placing another call in the desired cell towers, and she hasn't received any task

assignment in the current cycle, then she is considered as a *candidate user* for further task assignment (go to next step for task assignment pace control). Otherwise, EMC³ either collects sensed data from her (in case she received task assignment but hasn't returned results in the same cycle) or ignores her to take care of the next call.

Overall Task Assignment Pace Control. This module controls if further task assignment is still needed to fulfill the goal of getting expected number of sensed results from all the cell towers. For this purpose, the module first counts the number of *returned users* and their *covered cell towers*, collects the user list who have got task assignment but haven't returned sensed results (defined as *potential returners*), and computes their probability of returning the *missing number* of sensed results in the desired cell towers. If the number of *returned users* reaches the pre-defined value and the *returned users* fully cover all the cell towers, then the task assignment process of the current MCS cycle stops; If previous task assignments can expect to return the pre-defined number of results covering all cell towers, then no immediate assignment is needed in order to avoid redundant task assignment. If previous task assignments cannot ensure the return of expected number of results or the full coverage, then further task assignment is still needed (goes to next step for task assignment decision making).

Sub-optimal Task Assignment Decision Making. Given the incoming call and previous call records, if the task assignment pace control module informs that further task assignment is still needed, then this module decides if the current caller/receiver should be assigned with a sensing task in order to meet the three research goals. In order to make an `optimal` (`sub-optimal`) decision, this module counts the number of *returned users* and the *covered cell towers*, collects the *potential returner* list, and predicts the *future frequent callers* who haven't placed phone calls but would have higher probability of making at least two calls than the current caller making another call (defined as *future-surer candidates*). With the returned user list, *potential returner* list and *future-surer candidate* list, the module estimates if the last two sets of users can expect to return the *missing number* of pre-defined sensed results in the *desired cell towers*. If the probability is very high, then the task assignment is skipped for current caller and left to future users; If the last two user lists cannot ensure to get the *missing number* of sensed results in the required cell towers, the sensing task is assigned to the current caller, indicating that the current user is among the most potentially frequent callers.

In the following, we introduce the core algorithms used in the three components in detail.

4.3.1 Call/Mobility Prediction

We predict the call/mobility of a user based on the periodicity of previous calls and locations in historical call traces. Suppose an MCS task splits one day into M sensing cycles. Given a sensing cycle k and the elapsed time t, we model a user U_i's

call/mobility pattern in cycle k by using U_i's phone call traces (including time-stamps and cell tower ids) in ***corresponding cycles*** of previous days. For example, to predict the call/mobility pattern of a user in current sensing cycle from 08:00 to 10:00, we will use all her previous call records during the same period 08:00-10:00. Note that the calls placed by U_i in a current cycle are also included for her call/mobility prediction.

4.3.1.1 Modeling Call Patterns

Assume the call sequence follows an inhomogeneous Poisson process [87], then the probability of a user U_i to place n phone calls from instant t to the end of cycle k can be modeled as:

$$P_{k,t}\{x_i = n\} = (\lambda_{i,k,t}\frac{\Delta t}{T})^n * e^{-\lambda_{i,k,t}\frac{\Delta t}{T}}/n! \tag{4.1}$$

where $\Delta t = (t_0 + K * T) - t$ denotes the *remaining time* from instant t to the end of the cycle, T is the sensing cycle duration, and $\lambda_{i,k,t}$ refers to the Poisson intensity, which is estimated as the average number of phone calls that a user U_i has placed in previous corresponding cycles, specifically it is modeled as:

$$\lambda_{i,k,t} = \frac{\text{Number of calls of } U_i \text{ in perviois corresponding cycles}}{\lceil k/M \rceil}$$

4.3.1.2 Modeling Mobility Patterns

Given previous call records at sensing cycle k, a participant U_i, a set of cell towers T_{WR} and a cell tower $c_j \in T_{WR}$, we define U_i's future presence probability in cell tower c_j as the the ratio between the number of U_i's historical calls at corresponding cycles in cell tower c_j and the total number of calls at corresponding cycles, i.e.,:

$$D_k(i,j) = \frac{\text{Number of calls of } U_i \text{ in the corresponding cycles in } c_j}{\text{Number of calls of } U_i \text{ in the corresponding cycles}}$$

If the given participant U_i hasn't placed any call in the corresponding cycles, then $D_k(i,j) = 0, \forall c_j \in T_{WR}$.

4.3.2 Overall Task Assignment Pace Control

Given the list of *potential returners* $(A_k - R_k)$, the *missing number* of sensed results $(N_e - |R_k|)$ and the instant time (t) in cycle k, we estimate

- $P_{fulfill}$: the probability of having at least $(N_e - |R_k|)$ *potential returners* placing another call before the end of cycle k.

Given the list of *potential returners* $(A_k - R_k)$, a *desired cell tower* $c_l \in (T_{WR} - cover_k)$ and the instant time (t), we estimate

Algorithm 3: Pace Control Mechanism

```
if |R_k| < N_e OR cover_k ≠ T_WR then
    computing P_fulfill
    computing P_cover_l, for ∀c_l ∈ (T_WR − cover_k)
    if P_fulfill < P_G1 OR ∃c_l ∈ (T_WR − cover_k), P_cover_l < P_G2 then
        Goto Next Step for Further Task Assignment;
    end
    else
        No Need for Further Task Assignment;
    end
end
else STOP;
```

- P_{cover_l}: the probability of having at least one *potential returner* placing another call to cover the cell tower c_l before the end of cycle k.

With $P_{fulfill}$ and P_{cover_l} defined, EMC^3 controls the pace of task assignment using the pseudo code in Algorithm 3, where P_{G1} and P_{G2} are two given thresholds. In this way, the key is to calculate $P_{fulfill}$ and P_{cover_l}.

4.3.2.1 Estimating $P_{fulfill}$

First, we define $P\{X_{k,t,1}(A_k - R_k) = N\}$ as the probability of having N out of $|A_k - R_k|$ *potential returners* placing at least another call before the end of cycle k, where $N \leq |A_k - R_k|$ (see Eq. 4.2). In this way, $P_{fulfill}$ is estimated as the sum of $P\{X_{k,t,1}(A_k - R_k) = N\}$, where N is an integer ranging from the *missing number* of sensed result ($N_e - |R_k|$) to the total number of *potential returners* ($|A_k - R_k|$) (see Eq. 4.3).

$$P\{X_{k,t,1}(A_k - R_k) = N\} = \sum_{\forall s \subset A_k - R_k}^{|s|=N} \prod_{\forall U_m \in s} P_{k,t}\{x_m \geq 1\} \prod_{\forall U_m \in A_k - R_k - s} (1 - P_{k,t}\{x_m \geq 1\}) \tag{4.2}$$

$$P_{fulfill} = \begin{cases} 0, & |A_k| < N_e \\ \sum_{N \geq N_e - |R_k|}^{N \leq |A_k - R_k|} P\{X_{k,t,1} \quad (A_k - R_k) = N\}, & \\ & |A_k| \geq N_e \end{cases} \tag{4.3}$$

Please note that, when the number of participants already assigned is less than the expected number of sensed results – i.e., $|A_k| < N_e$, then it is not possible to collect the

pre-defined number of sensed results, thus $P_{fulfill} = 0$. Considering the complexity of $P_{fulfill}$ estimation, we propose an algorithm to reduce the computation complexity and time as shown in Appendix A.1.1.

4.3.2.2 Estimating P_{cover_l}

First, we define $COV_{k,t}(m,l)$ as the probability of a given *potential returner* U_m ($U_m \in A_k - R_k$) covering a given *uncovered cell tower* c_l ($c_l \in T_{WR} - cover_k$) before the end of cycle k. Assume U_m received the task assignment in cell tower c_{assign} ($c_{assign} \in T_{WR}$), apparently there are two possible cases: one is $c_{assign} = c_l$, the other is $c_{assign} \neq c_l$. In the case of $c_{assign} = c_l$, $COV_{k,t}(m,l)$ is equal to the probability of U_m placing at least another call before the end of cycle k (in arbitrary cell tower T_{WR}). In the case of $c_l \neq c_{assign}$, $COV_{k,t}(m,l)$ is equal to the probability of U_m placing another call in cell tower c_l before the end of cycle k. Hence we have:

$$COV_{k,t}(m,l) = \begin{cases} P_{k,t}\{x_m \geq 1\}, & c_l = c_{assign} \\ P_{k,t}\{x_m \geq 1\} * D_k(m,l), & c_l \neq c_{assign} \end{cases} \tag{4.4}$$

where $P_{k,t}\{x_m \geq 1\}$ denotes the probability of U_m placing at least another call before the end of cycle k, and $D_k(m,l)$ is the probability of U_m appearing in cell tower c_l. With the above definition of $COV_{k,t}(m,l)$, P_{cover_l} can be calculated using Eq. 4.5 below [62]:

$$P_{cover_l} = 1 - \prod_{\forall U_m \in A_k - R_k} (1 - COV_{k,t}(m,l)) \tag{4.5}$$

4.3.3 Sub-optimal Task Assignment Decision Making

Given the incoming call and previous call records, the key algorithms of this step include 1) identifying all *future-surer candidates*, 2) estimating if the *missing number* of sensed results can be returned from *future-surer candidates* and *potential returners*, 3) estimating if all *desired cell towers* can be covered by *future-surer candidates* and *potential returners*, and 4) sub-optimal task assignment decision making.

4.3.3.1 Identifying future-surer candidates

Given the current caller U_i, we consider U_m as a *future-surer candidate* if:

- U_m has placed calls in previous *corresponding cycles* but hasn't placed any call in current cycle, **and**
- U_m has a higher probability of placing at least two calls than U_i placing at least another call, i.e., $P_{k,t}\{x_m \geq 2\} > P_{k,t}\{x_i \geq 1\}$, **or** U_m has placed more calls in any desired cell tower ($T_{WR} - cover_k$) than U_i.

Putting all the *future-surer candidates* together with regard to U_i, they are denoted as FS_{U_i}.

4.3.3.2 Estimating if the Missing Number of Sensed Results can be returned from Future-surer Candidates and Potential Returners

Given the set of *future-surer candidates* FS_{U_i}, the set of *potential returners* $(A_k - R_k)$, and the *missing number* of sensed results $(N_e - |R_k|)$, we estimate $P^*_{fulfill}$ as the probability of having at least the *missing number* of sensed results $(N_e - |R_k|)$ returned from the *potential returners* and *future-surer candidates* $((A_k - R_k) \cup FS_{U_i})$ before the end of cycle k. Apparently the estimation of $P^*_{fulfill}$ depends on the probability of each U_m returning the sensed results $(U_m \in (A_k - R_k) \cup FS_{U_i})$ before the end of cycle k, each U_m's returning probability can be computed using Eq. 4.6.

$$P'_{k,t}(U_m) = \begin{cases} P_{k,t}\{x_m \geq 1\}, U_m \in (A_k - R_k) \\ P_{k,t}\{x_m \geq 2\}, U_m \in FS_{U_i} \end{cases} \tag{4.6}$$

In the case of $U_m \in (A_k - R_k)$ (belonging to the *potential returner set*), $P'_{k,t}(U_m)$ is modeled as the probability of U_m placing at least another call before the end of cycle k. In the case of $U_m \in FS_{U_i}$ (belonging to the *future-surer candidate set*), then $P'_{k,t}(U_m)$ is modeled as the probability of U_m placing at least two calls before the end of cycle k. Given each user U_m's returning probability $P'_{k,t}(U_m)$, similar to the estimation of $P_{fulfill}$ in Eq. 4.3, $P^*_{fulfill}$ can be computed using Eqs. 4.7 and 4.8.

$$P^*_{fulfill} = \begin{cases} 0, & |A_k \cup FS_{U_i}| < N_e \\ \sum\limits_{N \geq N_e - |R_k|}^{N \leq |(A_k - R_k) \cup FS_{U_i}|} P\{X_{k,t,2} \quad (FS_{U_i}) + X_{k,t,1}(A_k - R_k) = N\}, & \\ & |A_k \cup FS_{U_i}| \geq N_e \end{cases} \tag{4.7}$$

$$P\{X_{k,t,2}(FS_{U_i}) + X_{k,t,1}(A_k - R_k) = N\} = \sum_{\forall s \subset (A_k - R_k) \cup FS_{U_i}}^{|s|=N} \prod_{\forall U_m \in s} P'_{k,t}(U_m) \times \prod_{\forall U_m \notin s} (1 - P'_{k,t}(U_m)) \tag{4.8}$$

Considering the complexity of $P^*_{fulfill}$ estimation, we use the same algorithm as shown in Appendix A.1.2 to reduce the computation time.

4.3.3.3 Estimating if all Desired Cell Towers can be covered by Future-surer Candidates and Potential Returners

Given a *desired cell tower* c_l ($c_l \in (T_{WR} - cover_k)$), the set of U_i's *future-surer candidates* (FS_{U_i}), and the set of *potential returners* $(A_k - R_k)$, we define $P^*_{cover_l}$: the probability of cell tower c_l to be covered by at least one participant from the set of *potential returners* and *future-surer candidates* $((A_k - R_k) \cup FS_{U_i})$. Apparently the estimation of $P^*_{cover_l}$ depends on the probability of each U_m $(U_m \in (A_k - R_k) \cup FS_{U_i})$

covering the given cell tower c_l before the end of cycle k, the probability of each U_m's covering c_l can be computed using Eq. 4.9.

$$COV^*_{k,t}(m,l) = \begin{cases} COV_{k,t}(m,l), & U_m \in (A_k - R_k) \\ P_{k,t}\{x_m \geq 2\} * \quad (1-(1-D_k(m,l))^2), & \\ & U_m \in FS_{U_i} \end{cases} \tag{4.9}$$

In the case of $U_m \in (A_k - R_k)$ (belonging to the *potential returner set*), $COV^*_{k,t}(m,l)$ is the same as $COV_{k,t}(m,l)$. In the case of $U_m \in FS_{U_i}$ (belonging to the *future-surer candidate set*), then $COV^*_{k,t}(m,l)$ is modeled as the probability of U_m placing at least two calls (at least one of the first two calls placed in cell tower c_l), before the end of cycle k . Given the probability of each user U_m covering cell tower c_l – i.e., $COV^*_{k,t}(m,l)$, similar to the estimation of P_{cover_l} in Eq. 4.5, $P^*_{cover_l}$ can be computed using Eq. 4.10.

$$P^*_{cover_l} = 1 - \prod_{U_m \in (A_k - R_k) \cup FS_{U_i}} (1 - COV^*_{k,t}(m,l)) \tag{4.10}$$

4.3.3.4 Sub-optimal Task Assignment Decision Making

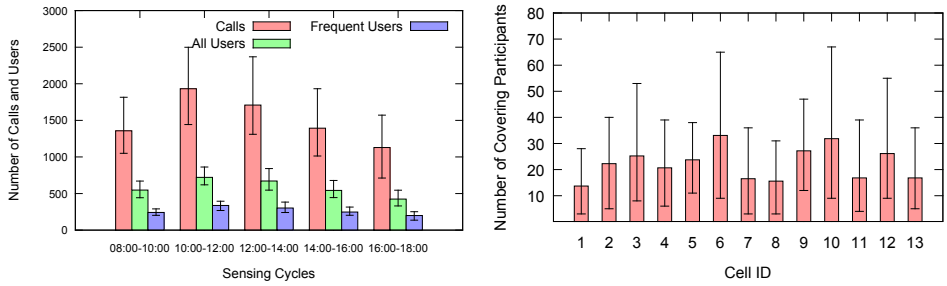
With $P^*_{fulfill}$, $P^*_{cover_l}$ computed and two thresholds P_{G1}, P_{G2} given, EMC^3 assigns a task to the current caller (U_i) if $P^*_{fulfill}$ is lower than P_{G1}, or there exists any cell tower $c_l \in (T_{WR} - cover_k)$ having $P^*_{cover_l}$ lower than P_{G2}. The pseudo code of sub-optimal task assignment decision making is shown in Algorithm 4.

Algorithm 4: Sub-optimal Task Assignment Decision Making Mechanism

```
computing P*_fulfill
computing P*_cover_l, for ∀c_l ∈ (T_WR − cover_k)
if P*_fulfill < P_G1 OR ∃c_l ∈ (T_WR − cover_k), P*_cover_l < P_G2 then
    Assign the sensing task to U_i;
end
else
    Not Assign;
end
```

4.4 Evaluation

In this section, we will report the evaluation results using the large-scale real-world call traces to verify the effectiveness for our proposed method in reducing energy consumption in data transfer for MCS tasks. We first introduce two baseline methods and the parameter settings for evaluating EMC^3 briefly. Second, we present two D4D



(a) Calls, Users and Frequent Users (b) Covering Participants

Figure 4.3: Statistics of CBD Call Traces

phone call traces and the basic experiment settings. Then, the detailed evaluation results of EMC^3 with respect to the two baseline methods are presented and compared. Finally, based on the known methods in energy consumption estimation, the EMC^3 and other relevant schemes are compared in terms of energy consumption for MCS data transfer.

4.4.1 Baseline Methods and Parameter Settings

In our evaluation, we provide two baseline methods with respect to EMC^3:

1. ***Greedy***: assigning the sensing task to each new calling user, till the expected number of sensed results are returned and all the cell towers are covered.
2. ***Pace Control based Method (Pace)***: leveraging our proposed *task assignment pace control* mechanism. If the pace control mechanism decides that further task assignment is still needed and the current caller is new in this cycle, it assigns the sensing task to the current caller.

Apparently, Greedy method is the baseline which can show the upper bound for the maximum number of task assignment and returned results, it can also provide ground truth for coverage. Compared to the Greedy method, the Pace method can show the effectiveness of pace control mechanism in reducing the redundant task assignment. The comparison between EMC^3 and Pace method shows the effectiveness of *sub-optimal task assignment decision making* mechanism in determining if the current or future callers are better candidates for task assignment, in order to avoid redundancy in task assignments. In all the experiments, we set the thresholds $P_{G1} = 99.99\%$ and $P_{G2} = (99.99\%)^{1/|T_{WR}|}$ for evaluating EMC^3 as well as Pace control based method.

4.4.2 Dataset and Experiment Setups

The dataset we use in this research is the D4D dataset, which contains 50,000 users' phone call traces (each call records includes user id, call time, and cell tower) in four

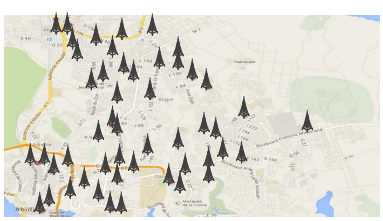
months from Cote d'Ivoire (where 2000 cell towers are installed). Specifically, the 50,000 users are re-selected randomly from all the mobile users every 2 weeks with anonymized user ids. Thus in this study, we assume that each MCS task lasts for two weeks accordingly. Further more, we split the 4-month data traces into eight two-week slots, with each two-week slot corresponding to one MCS task. And every MCS task executes five cycles every working day from 8:00 to 18:00, with each cycle lasting for two hours (i.e. 8:00-10:00, ..., 16:00-18:00). We extract the phone call records from the CBD area (named "Plateau") and a high-end residential district (named "Cocody") of Abidjan city, and use these two call traces for evaluation:

CBD Traces - As shown in Fig. 6.2, the *first* target region for the MCS task execution is assumed to contain 13 cell towers in the CBD of Abidjan city. For each MCS task (two-week period), about 2000 - 3000 users [1] have been found placing calls in the target region. These users are considered as the crowdsensing participants. In order to have the ground truth about the CBD region in D4D dataset, we show the number of calls, calling users, as well as the *frequent users* (placing at least two calls in a cycle) in each sensing cycle in Fig. 4.3a. Because the minimum number of frequent users in these cycles is 101, we thus set the expected number of returned participants (N_e) from 30 to 100. For coverage, we show the Max/Min/Avg[2] number of all covering participants found from the datasets in each cell tower per sensing cycle in Fig. 4.3b, extracting from the 4-month dataset. It can be seen from Fig. 4.3b that each cell tower can be covered by at least 3 participants per cycle, which means that the full coverage constraint is supported by the ground truth of the D4D dataset.

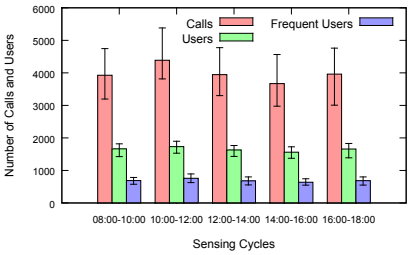
Residential District Traces - As shown in Fig. 4.4a, the *second* target region for the MCS task execution is assumed to contain 50 cell towers in an upmarket residential area (around 40 km^2) of Abidjan city. For each MCS task (two-week period), about 7000 - 8000 users have been found placing calls in the target region. In order to get the ground truth about the call traces, we show the number of calls, calling users, as well as the *frequent users* in each sensing cycle in Fig. 4.4b. Because the minimum number of frequent users in these cycles is 560, we thus set the expected number of returned participants $N_e = 250$ and $N_e = 500$ respectively. For coverage, we show the Max/Min/Avg number of all covering participants found from the traces in each cell tower per sensing cycle in Fig. 4.4c. It can been seen that each cell tower can be covered by at least 4 participants per cycle, which means the call traces of Residential District can also meet the full coverage constraint. Obviously, the Residential District Traces contain more call records from more people in a larger area.

[1] As a reference, there are about 7.2 million inhabitants in Abidjan, where around 75% inhabitants are mobile phone users [91].

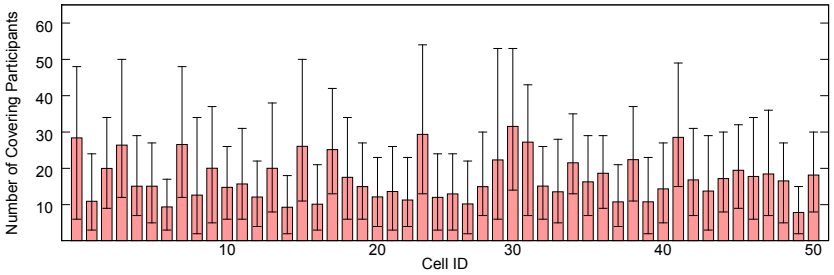
[2] In this work, we name "Max/Min/Avg" as "Maximum/Minimum/Average" in short.



(a) Cell Tower Distribution

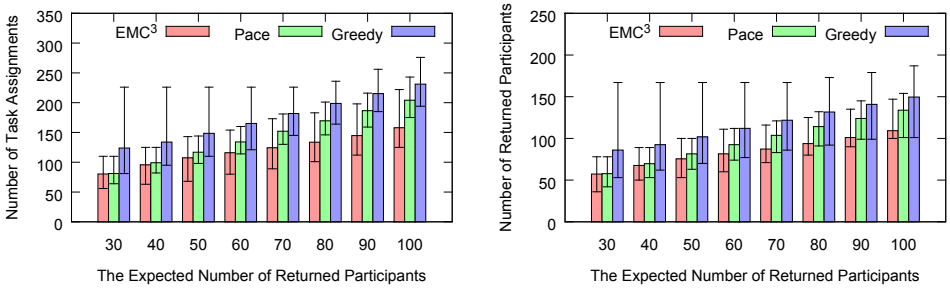


(b) Calls, Users and Frequent Users



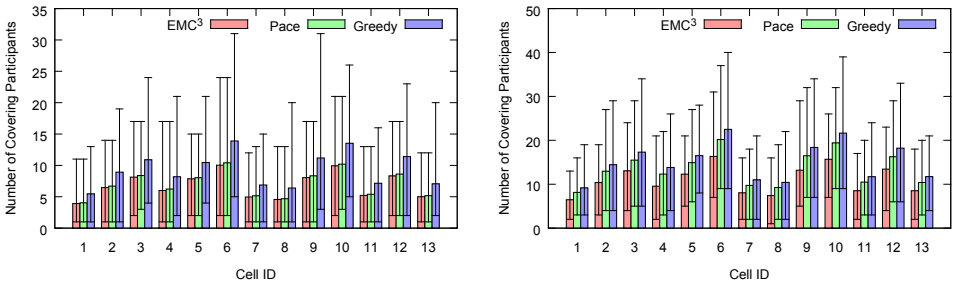
(c) Covering Participants

Figure 4.4: Statistics of Abidjan Residential District Call Traces (Best Viewed in Digital Form)



(a) Task Assignments (b) Returned Participants

Figure 4.5: Number of Task Assignments and Returned Participants (CBD Traces)



(a) $N_e = 40$ (b) $N_e = 100$

Figure 4.6: Number of Covering Participants (CBD Traces)

4.4.3 Performance Evaluation

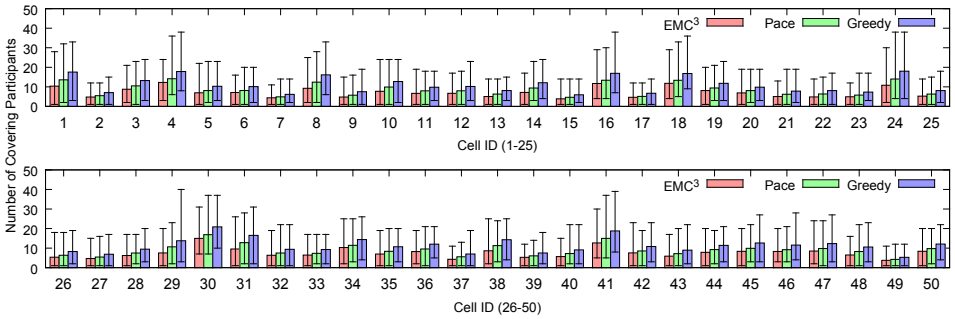
In this part, we first compare the performance of EMC^3, Pace and Greedy methods in terms of number of task assignments, number of returned results, and coverage; Then we use an example to explain why the proposed EMC^3 outperforms Pace and Greedy method.

4.4.3.1 Performance Comparison based on CBD Traces

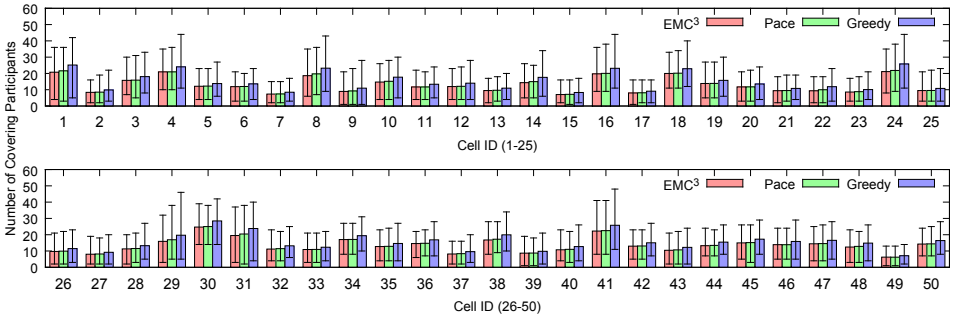
In Fig. 4.5, we present the Max/Min/Avg number of task assignments and returned participants for the three methods under the same MCS setting, when the expected number of returned results N_e is set to vary from 30 to 100 based on CBD Traces. In order to show the coverage of the three methods, we show the Max/Min/Avg number of covering participants in each cell tower under the same MCS setting with $N_e = 40$ and 100, respectively in Fig. 4.6. Due to the space limit, we only select the evaluation results with $N_e = 40$ and 100. From the evaluation results shown in Fig. 4.5 and Fig. 4.6, we observe that:

Task Assignments. Fig. 4.5a shows clearly that EMC^3 assigns fewer tasks to participants than Pace and Greedy. On average, EMC^3 reduces 1%-23% task assignments compared to Pace, and it also reduces 27%-35% task assignments compared to Greedy method.

Returned Participants. Fig. 4.5b shows, even in the worst case, all EMC^3, Pace and Greedy are able to collect sensed results from more than N_e participants. However, in all the cases the number of returned results is bigger than the expected number N_e, even though the number of returned results for EMC^3 is 1% - 18% fewer than Pace and 26% - 33% fewer than Greedy on average. For the Greedy and Pace methods, it's easy to understand that the big number of returned results are due to the highly redundant task assignments. For EMC^3, the reason for the big number of task assignment and returned results is mainly due to the inaccurate call/mobility prediction with limited number of call traces.



(a) $N_e = 250$



(b) $N_e = 500$

Figure 4.7: Number of Covering Participants (Residential District Traces, $N_e = 250$ and 500)

Coverage. Fig. 4.6 shows that any of the 13 cell towers can be covered by at least one participant with these three methods. Specifically, some cell towers have more participants than the others in a cycle. Interestingly, the distribution of covering participants in different cell towers remains more or less the same when N_e varies, and it's similar to the natural distribution of covering participants shown in Fig. 4.3b.

4.4.3.2 Performance Comparison based on Residential District Traces

In Table 4.1, we present the performance comparison between EMC^3 and baselines using Residential District Traces. We count the average/minimum/maximum number of task assignments and returned participants. It is obvious that EMC^3 outperforms Pace and Greedy— EMC^3 reduces $8\% - 18\%$ task assignments compared to Pace and reduces $24\% - 36\%$ task assignments compared to Greedy; and the number of returned participants by EMC^3 is $5\% - 15\%$ and $17\% - 33\%$ less than Pace and Greedy respectively. Furthermore, in terms of coverage, all 50 cell towers are fully covered by these three methods in every sensing cycle with all N_e settings (please see also Fig. 4.7, where the Max/Min/Avg number of covering participants in each cell

Table 4.1: Performance Comparison based on Residential District Traces: EMC^3 vs Pace vs Greedy

Schemes	Task Assignments			Returned Participants		
	Avg.	Min.	Max.	Avg.	Min.	Max.
$N_e = 250$						
EMC^3	446.6	310	979	297.2	250	574
Pace	541.5	387	1015	352.2	262	609
Greedy	703.5	578	1122	445.5	369	714
$N_e = 500$						
EMC^3	821.4	695	994	507.9	500	574
Pace	887.8	756	1120	535.6	502	714
Greedy	1075.2	967	1194	615.5	536	718

tower under the setting of $N_e = 250$ is shown); and the observation about the covering participants distribution is quite similar to that obtained from our experiment based on CBD call traces. From the above evaluation results, we can see that EMC^3 performs consistently better than the two baseline approaches in terms of task assignment while all the methods can achieve the goal of full coverage and collecting the predefined number of sensed results, when the target area and number of participants are different.

4.4.3.3 Case Study and Analysis

In order to gain more insights about the observed phenomena, we would like to show the actual task assignment process using the three methods and the Residential District call traces in sensing cycle 16:00-18:00, on 14 December 2011, where N_e is set to 250. Fig. 5.4 shows the actual task assignment traces of EMC^3, Pace and Greedy in the top part of the diagram (with the total number of task assignments $|A_k|$ listed), the number of covered cell towers in the middle ($|cover_k|$), and the number of returned results in the bottom (R_k). From Fig. 5.4, we can observe the detailed differences among EMC^3, Pace and Greedy methods, including:

Pace vs Greedy: In the beginning of the cycle, both Pace and Greedy methods assign sensing tasks to each new caller/receiver, but Pace stops assigning tasks at 16:22 when only 112 participants return their sensed results and 46 cell towers are covered. Obviously, Pace method doesn't make any further task assignments if the assigned participants are estimated to meet the requirements of covering all cell towers and collecting an expected number of sensed results. Greedy, however, stops making new task assignment at 16:39 when a total of 250 participants return their sensed results and all 50 cell towers are covered. The Pace method stops 17 minutes earlier than the Greedy method, which causes 233 less redundant task assignments and 141 less unnecessary returned results. Apparently, it's all due to the pace control mechanism

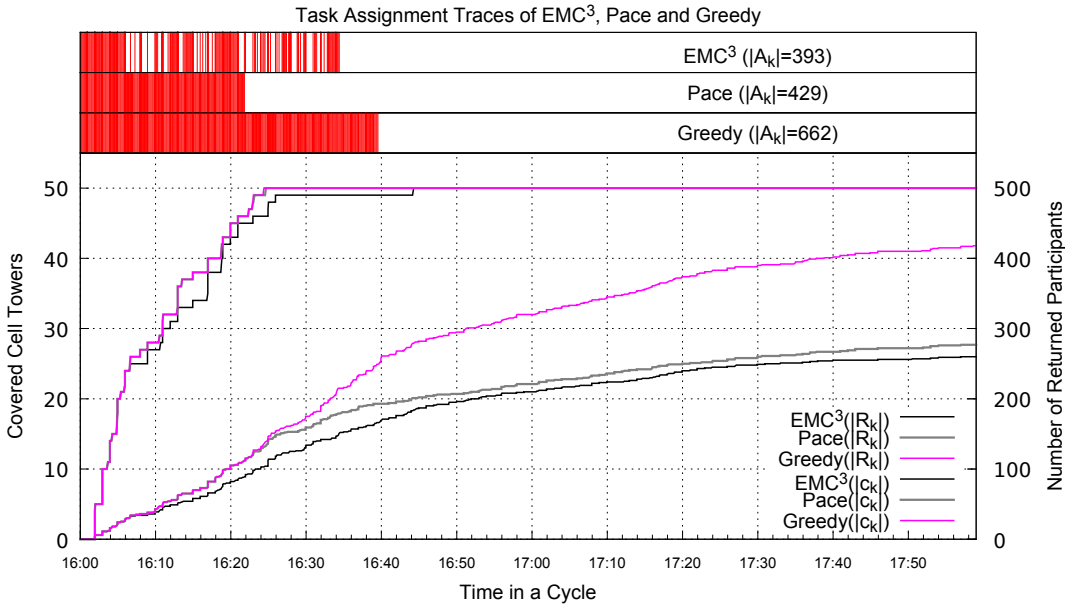


Figure 4.8: Task Assignment Process in the Case Study

based on future call/mobility prediction.

EMC^3 vs Pace: In the beginning of the cycle, also EMC^3 assigns sensing tasks to each new caller/receiver like Pace, because the number of task assignments is much lower than the expected number of returned results N_e (250). But with the number of task assignments and returned results increasing, EMC^3 begins to select only "*frequent callers/receivers*" who have high probability of covering "*desired cell towers*" for task assignment, while Pace continues to assign sensing tasks to each new caller/receiver until the number of task assignments made is estimated to ensure receiving the expected number of sensed results covering all cell towers. In the case of Fig. 5.4, Pace stops assigning tasks at 16:22 when 112 participants return their sensed results and 46 cell towers are covered. EMC^3, however, stops making new task assignment at 16:34 when a total of 152 participants return their sensed results and 49 cell towers are covered. The EMC^3 method stops 12 minutes later but assigns 36 less redundant tasks than Pace. Apparently, EMC^3 outperforms Pace because of its *decision making* mechanism for task assignment, which is based on the prediction of call/mobility for both participants with task assigned and future callers/receivers.

4.4.4 Energy Conservation Comparison

After getting the number of task assignments and returned results with EMC^3, we would like to estimate how much energy our proposed EMC^3 scheme can save in data

Table 4.2: Energy Consumption Computation Models

Schemes	Energy Consumption
3G-based scheme	$N_e * (12 + 12) = 24 * N_e$
Parallel+3G-based scheme	$N_e * (3 + 12) = 15 * N_e$
EMC^3, Pace and Greedy	$\|A_k\| * 3 + \|R_k\| * 3$

transfer, comparing to the following 3G-based MCS schemes:

1. ***3G-based Scheme***: receives the task assignment by establishing a new 3G connection, and returns the sensed results by establishing another 3G connection.

2. ***Parallel+3G-based Scheme***: receives a task assignment when the participant places a phone call through parallel data transfer, and returns the sensed results by establishing a new 3G connection.

Because no redundant task assignment is needed to collect sensed results with the above two schemes, we thus assume that only N_e participants from the 13 cell towers are selected to perform the MCS sensing task. Based on the literature [92] about the mobile phone energy consumption estimation method, we model the overall energy consumption in data transfer for MCS tasks by using the formulas listed in Table 4.2. Here the energy consumption estimation is based on the setting of Nokia N95 and the simple assumption that the data packets for task assignment or sensed results are small (less than 10KB each). Considering the MCS applications such as air quality monitoring and environment noise monitoring, this assumption is reasonable and the energy estimated using the formula could serve as a reference for comparison purpose.

Table 4.3 shows each scheme's average energy consumption per sensing cycle with varied N_e settings for both CBD and Residential District Traces. As can be seen from Table 4.3, EMC^3 can save 43% - 68% energy on average, compared to the 3G-based scheme; It can save 8% - 48% energy compared to the Parallel+3G-based scheme. Till now we are assuming that the number of expected sensed results is small, if the MCS application needs to recruit hundreds of participants and collect sensed data for many cycles a day, then the total energy saving would be significant. Interestingly, if we compare the EMC^3, Pace, Greedy with the Parallel+3G-based scheme, we can see that EMC^3 outperforms all the other schemes in all the conditions, but the Greedy method consumes more energy than the Parallel+3G-based scheme when $N_e < 50$ (using CBD traces). In summary, all the above evaluation and analytical results show the effectiveness of EMC^3 in reducing the energy consumption in data transfer for both individual and all MCS participants.

Table 4.3: Energy Consumption Comparison: 3G-based vs Parallel+3G-based (P+3G) vs EMC^3 vs Pace vs Greedy

N_e	**3G (J)**	**P+3G (J)**	**EMC^3 (J)**	**Pace (J)**	**Greedy(J)**
CBD Traces					
30	720	450	*412.55*	416.47	**629.77**
40	960	600	*489.64*	506.36	**679.33**
50	1200	750	*548.57*	594.95	**751.11**
60	1440	900	*592.20*	679.57	831.69
70	1680	1050	*634.71*	766.97	910.92
80	1920	1200	*682.34*	850.90	991.24
90	2160	1350	*737.41*	931.62	1068.25
100	2400	1500	*801.48*	1013.90	1142.43
Residential District Traces					
250	6000	3750	*2231.4*	2681.1	3447
500	12000	7500	*3897.9*	4270.2	5072.1

4.4.5 Real-time Performance Analysis

As the decision for task assignment should be made immediately when a participant places/receives a phone call, in this section we would like to investigate if the proposed EMC^3 algorithm can be executed in the real-time setting. Thus, we first compute EMC^3's *response time*—i.e., the duration from the initial of a call (from/to a participant in the target region) to the time when the decision of task assignment is made; and then, based on the computed *response time*, we estimate EMC^3 *maximum throughput* [93]—i.e., the maximum number of mobile users allowed in the MCS system. We carry out experiments using a common laptop with an Intel Core i7-2630QM Quart-Core CPU and 8G memory. EMC^3 algorithm is implemented with the Java SE platform and is running on a Java HotSpot(TM) 64-Bit Server VM.

In order to compute the *response time* and *maximum throughput* in the realistic deployment condition, we build an EMC^3 simulator consisting of two phases:

1. *Filter* - When a mobile user makes/receives a phone call, the system checks if the call is made/received in the target region and if the user is in the list of MCS participants; and all these operations are implemented as a simple DB query based on an embedded database. In this phase, EMC^3 identifies participants in the target region from all calls; and if the calling user is not a participant or the call is not made/received in the target region, then EMC^3 filters out the call immediately.

2. *Process* - Given a participant making/receiving a phone call, this phase executes the EMC^3 three-step task assignment decision making algorithm to decide if the participant should receive a task assignment.

Table 4.4: EMC³ Average Response Time and the Estimated Maximum Throughput

	Response Time (10^{-3} sec.)		Max. Throughput (calls/sec.)	
N_e	filter	process	filter	process
CBD Traces				
30	0.0076	1.7795	131578.95	561.96
40	0.0078	1.9925	128205.13	501.88
50	0.0079	2.6816	126582.28	372.91
60	0.0080	3.1746	125000.00	315.00
70	0.0080	3.9928	125000.00	250.45
80	0.0080	5.0457	125000.00	198.19
90	0.0080	5.9189	125000.00	168.95
100	0.0080	6.9771	125000.00	143.33
Residential District Traces				
250	0.0413	268.2682	24213.08	3.73
500	0.0417	475.6196	23980.82	2.10

Table 6.1 presents the average *response time* and the estimated *maximum throughput* in both phases based on different call traces and MCS task settings. Even when EMC³ is used to monitor the Residential District, it only spends averagely no more than 0.0417 milliseconds [3] in the "*filter*" phase, which means EMC³ is able to handle 23980 calls every second. As a reference, according to the D4D dataset[4], we estimate there are approximately 1800 calls made/received by all mobile phone users from the whole Cote d'Ivoire every second. Furthermore, EMC³ requires averagely 0.475 seconds to complete the three-step task assignment decision making process using the Residential District traces where $N_e = 500$, which means EMC³ is able to make decision for 2.1 incoming calls from MCS participants every second under the given condition, where *Pace* on average requires 0.076 seconds to complete the task assignment decision making process and is able to handle 13 incoming calls per second under the same setting. As a reference, even in the busiest time slot (i.e., 10:00-12:00 in working days) of Residential District, there are 0.77 calls averagely made/received by MCS participants every second. All above estimation shows that, with a high-performance server EMC³ can easily support an larger target region than either CBD area or Residential District in real-time; and the response time of EMC³ can be controlled to a certain value if each server is in charge of a fixed geographical area.

[3]The time consumed in communication and networking has not been taken into account here; because actually EMC³ is assumed to be deployed on telecom operator's network.

[4]D4D dataset contains call traces of 0.3% randomly-sampled nationwide mobile phone population; and the average number of calls per second in D4D dataset is 5.4 calls/sec. Thus, we estimate the the average number of calls per second as 5.4/(0.3%)=1800 calls/sec.

4.5 Discussion

In this section, we discuss issues which are not reported or addressed in this work, these could be added to our future work.

Cold Start Problem: In the first day of an MCS task, as there are no historical call records due to the privacy consideration, the prediction for *frequent callers* and *future surer candidates* won't be accurate. Thus EMC^3 has the "cold start" problem which makes it perform the same as Pace method in sensing cycles of the first day, gradually with the accumulation of historical call records, EMC^3 performs better and better. The detailed evaluation results are not reported here due to space limit, but will be reported in future work.

Prediction and Parameter Adaption: As the performance of EMC^3 depends on the prediction accuracy of call/mobility prediction and the parameter setting used in the algorithm, in this study we currently use a simple prediction algorithm and a fixed set of parameter setting in all the sensing cycles, in future work we plan to study adaptive task assignment pace control and decision making strategies, and design advanced call/mobility prediction methods.

Sensing Coverage: Due to the limitation of the D4D dataset, we can only measure one's coverage at the cell tower level; and the cell towers traversed by users between two calls are not accessible in this work. Apparently, if the user's mobility traces can be obtained continuously at fine granularity, we might consider the coverage of users more precisely.

Aggregating Multiple Energy-efficient Strategies: In addition to piggybacking 3G data transfer over 3G calls or data packets, other data transfer methods, e.g., transferring data via WiFi/Bluetooth, also consumes less energy in data transfer, compared to common 3G-based solutions. Besides, there exist a wide range of techniques, such as adopting low-power consumption sensors or energy-efficient sensing techniques, that can save energy in the MCS tasks. In our future work, we intend to study an integrated MCS framework aggregating multiple energy-saving strategies to minimize the energy consumption in a holistic manner.

Enabling Ultra-large Scale Crowdsensing: The evaluation result shows that EMC^3 is able to handle a large area–with tens of cell towers installed and thousands of participants making/receiving phone calls–in the real-time, while securing the data collection from hundreds of participants and under the full coverage constraint. When nation scale crowdsensing is needed, we can just divide the whole nation into multiple sub-areas and deploy multiple EMC^3 servers to manage each sub-area collaboratively. Apparently, in this way, EMC^3 can scale easily without performance issues.

Chapter 5

CrowdRecruiter: Selecting Participants for Piggyback Crowdsensing under Probabilistic Coverage Constraint

Contents

5.1 Introduction 82
5.2 CrowdRecruiter: System Overview 85
5.2.1 Participant Selection Problem in CrowdRecruiter 85
5.2.2 Overall Design of CrowdRecuiter 86
5.3 Core Algorithms of CrowdRecruiter 88
5.3.1 Call/Mobility Prediction 88
5.3.2 Utility Calculation of Each Combined Set 88
5.3.3 Coverage Probability Vector Calculation 89
5.3.4 Algorithm Analysis 89
5.4 Evaluation 91
5.4.1 Baseline Methods 91
5.4.2 Dataset and Experiment Setups 92
5.4.3 Number of Participants Comparison 94
5.4.4 Selection Process Comparison 94
5.4.5 Participant Selection Overlaps 94
5.4.6 Performance Evaluation and Comparison 95
5.4.6.1 Computation Time Analysis 95
5.4.6.2 Analysis of Selected Participants 96
5.4.7 Combine Participants from Each Cycle 97
5.5 Discussion 98

5.1 Introduction

In this Chapter, we introduce our third MCS framework CrowdRecruiter, which is different with *EEMC*(introduced in Chapter 3) and EMC^3 (introduced in Chapter 4) in following ways:

- *Probabilistic Sensing Coverage* - While EMC^3 is designed to collect sensed results fully covering the target region, CrowdRecruiter uses a novel sensing coverage metrics namely *Probabilistic Coverage*. For many MCS applications, such as environment monitoring, full coverage is not always required. It is often sufficient to ensure a high ratio of spatial coverage in a specified time frame and get an idea of the situations in most places that people frequently visit. Thus, given the target region consisting of a set of subareas, CrowdRecruiter aims to collect sensed results covering a *predefined percentage* of subareas.

- *One-call-based Piggyback Crowdsensing Mechanism* - Energy consumption is known to be one of the key factors compromising the user's willingness for MCS task participation. While *EEMC* and EMC^3 adopt a *two-call-based MCS mechanism* for energy-efficient MCS data transfer, CrowdRecruiter leverages Piggyback Crowdsensing Task Model proposed in [18], where the energy consumption caused by MCS data transfer, sensing and computing can be reduced by piggybacking MCS sensing, computing and data transfer jobs over the *smartphone app opportunities*. It is shown in [19, 86] that sensing the air quality and uploading sensed results in parallel with a 3G call can reduce about 75% of energy consumption in data transfer compared to the 3G-based solution, while piggyback sensing scheme can significantly reduce the energy consumed by sensors and microprocessors when performing MCS tasks [18].

- *Participant Recruitment* - While *EEMC* and EMC^3 decide if to assign an MCS task to each mobile user during the MCS process, CrowdRecruiter intends to *recruit* a group of participants from all volunteering mobile users, prior to the MCS process, where each recruited participant is required to join in all sensing cycles of the whole MCS process. Further, assuming that each recruited participant is paid an *equal-mount of incentive*, CrowdRecruiter needs to select a minimal number of participants while ensuring a predefined percentage of subareas being covered by the selected participant in each sensing cycle, in order to minimize the overall incentive payment under the probabilistic coverage constraint.

To show the key concepts and ideas of the PCS applications with CrowdRecruiter, a motivating example is given as follows.

An environment NGO plans to monitor the air quality for citizens in Abidjan City, Cote d'Ivoire, updating the air quality index every hour during daytime. With the help of a telecom operator, the NGO makes an agreement with around 5000 smartphone

users, who are willing to be selected for an one-week-long air quality sensing trial and install a PCS application on their own smartphones. There are 131 cell towers in the Abidjan downtown area as shown in Fig. 6.2. For the purpose of this trial we divide each working day into 10 sensing cycles (08:00–18:00) and each sensing cycle lasts for one hour. In order to minimize the total cost of the crowdsensing task while ensuring the sensing quality for the one week trial, NGO hopes to select a minimal set of users from the 5000 candidates, who are able to place 3G calls at 90% of the 131 cell towers in each sensing cycle. In such a way, each selected mobile user could sense the air quality and upload the air quality information of the cell tower when the 3G call is placed at certain cell tower, and the combined set of users can cover 90% of the 131 cell towers in all sensing cycles. To facilitate the selection of the minimal set of users, one week's call and mobility records of the 5000 candidates (including the time stamp and cell tower ID for each call) before the trial are made available for NGO by the telecom operator. After the minimal set of mobile users are selected according to their historical call/mobility traces, each selected participant would receive a fixed sum of incentives from NGO and activate the PCS application on their mobile phones. The PCS application with the PCS task engine [18] will sense and upload air quality data when the participant places a 3G call at a new cell tower in each one-hour time frame throughout the trial period. With the piggybacking mechanism, each participant is expected to consume a small amount of energy for the PCS task, and the total incentive cost for the whole PCS task is also maintained minimal.

Note that, in the considered use case and the rest of this work, we use cell towers as the coverage metrics, primarily due to two reasons: 1) The cell tower IDs of mobile phones are accessible in call logs, even though the cell tower is not the right coverage metrics for many MCS applications, the mobile phone call logs with cell tower as coverage metrics can be used to illustrate the core idea of handling coverage constraint problem in MCS applications; 2) For MCS applications, such as urban air quality monitoring, covering a high percentage of cell towers in a given region ensures that the most part of the given area is measured, even though the sampled granularity in terms of cell tower may not be the best choice. If it could be characterized more precisely, the proposed approach could be easily adapted.

From the above use case, it can be seen that the objective of the research work is to select a minimal set of participants for the PCS task while ensuring a predefined cell tower coverage in each sensing cycle, with following two assumptions.

1. Only the selected participants are involved in the sensing and uploading task. The mobile device of each selected user performs sensing and data uploading task only when the participant places (makes/receives) a 3G call at a new cell tower in each sensing cycle.

2. All the mobile users agreed to participate in the PCS task make their historical call/mobility traces available to facilitate the participant selection. Only the historical call traces in the recent week are provided to NGO, while other

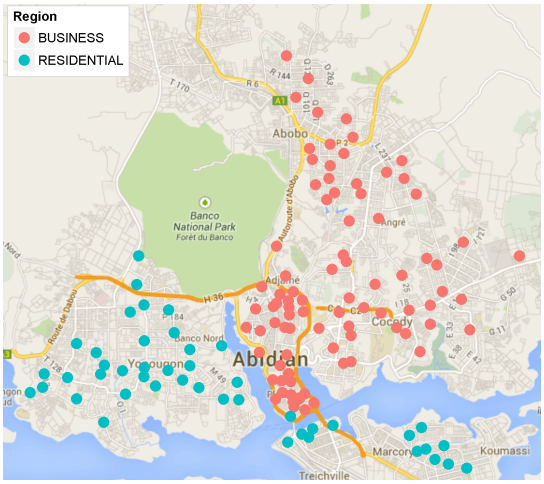


Figure 5.1: Cell Towers in the Downtown of Abidjan City

personal information is only known to the telecom operator.

In order to solve the above research problem, there are at least three challenges in the PCS system design:

1) Estimating the spatial coverage probability of selected participants based on their call/mobility traces in each sensing cycle. Since we only have user's historical call activities and mobility traces, and the call/mobility pattern will change in the PCS deployment week, we thus have to find a way to predict the call/mobility pattern of each selected user accurately. Even with inaccurate prediction results, we need further to characterize the spatial coverage probability of each participant and estimate if the joint spatial coverage of a set of selected participants meets the predefined probability threshold.

2) Lowering the complexity and increasing the speed of search for the minimal set of participants meeting the probabilistic coverage requirement. A brute-force approach for searching the minimal set of participants is to enumerate all the possible combinations from 1 to k participants (out of 5000 users), where k is the minimal number that ensures that one of the combined set with k participants could meet the probabilistic coverage constraint in each sensing cycle. This search problem, however, is NP hard in nature [94]. Thus it is necessary to develop a fast approximation algorithm to find a near-minimal set of participants meeting the coverage constraint.

3) Setting the user selection metrics and stopping criterion for the near-optimal participant set search. A common approach to search for the near-minimal set of participants is the greedy algorithm [94]. First the best user according to a certain coverage metric is selected into the solution. Then one more user out of the unselected candidates is combined with the already selected participants. Among all the combinations, the set with the highest coverage metrics is selected as the best set. If the lowest coverage probability of the best set across all sensing cycles is larger

than the required threshold, the near-minimal set of participants is found and the participant selection process terminates. Otherwise, another user needs to be added to the selected set until the above lowest coverage probability condition holds true. However, how to combine the coverage probability of multiple users and measure which user set has higher coverage probability are no-trivial, as these metrics might affect which user will be selected as part of the participant set and thus determine how many users will be included in the final participant set.

With the abovementioned research objective and challenges, the main contributions of this work are:

1) We formulate the problem of selecting minimal number of participants in piggyback crowdsensing (PCS) under probabilistic coverage constraint, with consideration of both total energy consumption and incentives paid in a PCS task. To the best of our knowledge, this is the first work addressing the participant selection issue in the context of PCS, where we select participants according to their historical call/mobility pattern and leveraging the call opportunities of mobile users to sense and upload data for crowdsensing task.

2) In order to select the minimal set of participants under the coverage constraint, we propose a two-phase participant selection framework named *CrowdRecruiter*. It takes a novel approach to measure the coverage probability of multiple users as a combined set and selects the minimal set of participants. Theoretical analysis shows that the proposed approximation algorithm can achieve globally near-optimality with low computational complexity.

3) We evaluate our proposed algorithms with the real world dataset D4D[1] [15], which contains 4-month call records of 50, 000 users from Cote d'Ivoire. We verify that the proposed algorithm performs better than three baseline approaches, using the call records of two separate regions in Abidjan. Specifically, CrowdRecruiter selects 10.0%-73.5% fewer participants on average than the baseline approaches, under the same coverage constraint.

5.2 CrowdRecruiter: System Overview

In this section, we formulate the participant selection problem in CrowdRecruiter and describe the proposed framework to solve this problem.

5.2.1 Participant Selection Problem in CrowdRecruiter

As PCS has provided an energy-efficient task model, the primary objective of CrowdRecruiter is to minimize total incentive payments while meeting a predefined coverage constraint. In the proposed model, a PCS task may run over a period of time (e.g., a week) and consist of multiple sensing cycles, such as 10 one-hour cycles per day from 08:00–18:00. We consider that a cell tower t is *covered* in a sensing cycle

[1]D4D Dataset —`http://www.d4d.orange.com/en/home`

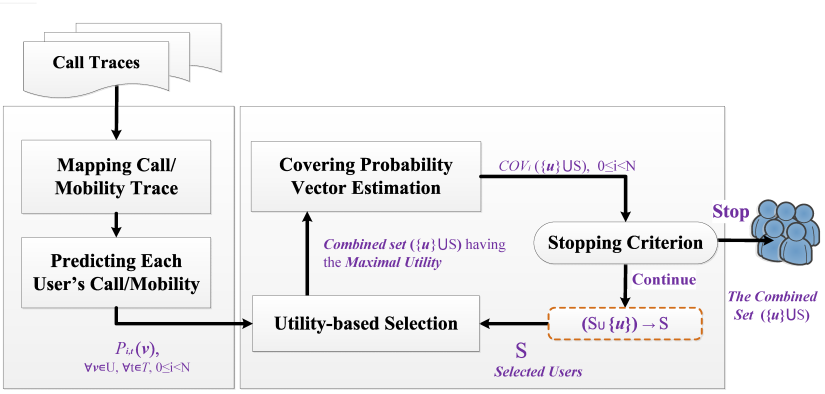


Figure 5.2: The CrowdRecruiter Framework

i if a participant places a call at t in i. Note that if a participant places multiple calls at different cell towers in i, all these cell towers are said to be covered in i. Thus the goal for CrowdRecruiter is to select a minimal number of participants from a set of volunteering mobile users, given their historical call and location traces, in order to meet the spatial-temporal coverage constraint that specifies the percentage of covered cell towers in the target area should be equal to or greater than a *coverage threshold* during all sensing cycles. With these definitions, we formally formulate the participant selection problem in CrowdRecruiter as follows.

Given a set of volunteering mobile users $\mathbb{U}$, a target area consisting of a set of cell towers T, the call traces of all users in $\mathbb{U}$ (including the time stamped calls and associated cell towers), we denote $\mathbb{S}$ as the set of participants selected from $\mathbb{U}$ (i.e., $\mathbb{S} \subset \mathbb{U}$), and $c_i(\mathbb{S})$ as the set of cell towers being covered in the i^{th} sensing cycle by $\mathbb{S}$. The problem is then to find $\mathbb{S}$ as a subset of $\mathbb{U}$, with the objective to

$$\text{minimize } |\mathbb{S}|, \text{ subject to } \frac{|c_i(\mathbb{S})|}{|T|} \geq R_{atio} \text{ and } 0 \leq i < N$$

where N is the total number of sensing cycles for the PCS task. It is worth noting that we cannot foreknow when and where a participant will place a phone call during the PCS task (i.e., $c_i(\mathbb{S})$ is unknown when we select participants). Thus we decompose the participant selection problem into two sub-problems: *call/mobility prediction*, and *participant selection based on the prediction*.

5.2.2 Overall Design of CrowdRecuiter

CrowdRecruiter follows a centralized participant selection approach, where a central server collects and stores the volunteering mobile users' historical call traces in the target area, and the server selects participants from all volunteering mobile users before the PCS task execution. Only selected participants are requested to perform

sensing tasks and return sensed results in each sensing cycle. Considering the *two sub-problems* in participant selection, we hereby propose a two-phase approach. Given the historical call traces of all users, the *first phase predicts each user's call/mobility during the PCS task*; and the *second phase selects participants* based on the prediction results. The framework is shown in Fig.5.2 and works as follows.

Phase I - Data Preparation and User Call/Mobility Profiling. Given the call traces of all volunteering mobile users, this phase computes the *call/mobility profile* of each user–i.e., *probability of each user placing at least one call at a particular cell tower in a given sensing cycle.* Specifically, CrowdRecruiter computes the profile of each user with following two steps:

- **Mapping Call/Mobility Traces -** Given the historical call/mobility traces of all users, this step maps each user's historical call/mobility traces onto N sensing cycles. Then it counts $\lambda_{u,i,t}$–the average number of calls placed by each user ($u \in \mathbb{U}$) at each cell tower ($t \in T$) in each sensing cycle ($0 \leq i < N$);
- **Predicting each User's Call/Mobility -** Given $\lambda_{u,i,t}$, this step estimates $P_{i,t}(u)$–the probability of the user ($u \in \mathbb{U}$) placing at least one call at each cell tower ($t \in T$) during each sensing cycle ($0 \leq i < N$).

Phase II - Iterative Participant Selection Process. Given the call/mobility profile of each user, we propose an iterative process to select participants incrementally:

- The algorithm first picks up the *single user having the maximal utility among all volunteering users* and selects that user into the solution, where the utility is defined formally in the next section;
- The algorithm then selects an *unselected user having the maximal utility when combing with the selected users* and adds that user into the solution;
- The algorithm keeps selecting and adding new participants until the selected participants could cover a predefined percentage of cell towers in every sensing cycle.

Specifically, an iteration consists of following three steps:

- **Utility-based Selection -** Given the full set of volunteering users ($\mathbb{U}$) and the set of *selected participants* ($\mathbb{S}$), this step first combines each unselected user ($\forall u \in \mathbb{U} \backslash \mathbb{S}$) with the selected participants in order to generate a combined set–i.e., $\{u\} \cup \mathbb{S}, \forall u \in \mathbb{U} \backslash \mathbb{S}$; second it calculates the utility of each combined set (i.e., $Utility(\{u\} \cup \mathbb{S})$); and then it selects the combined set having the *maximal utility* and keeps it as the newly selected set of participants for next iteration.
- **Covering Probability Vector Calculation -** Given the combined set (e.g., $\{u\} \cup \mathbb{S}$) having the maximal utility, this step computes a vector of probabilities, where each element of the vector is the probability of the combined set meeting the predefined coverage ratio in a specific sensing cycle.

- **Stopping Criterion -** Given the vector of probabilities based on the combined set with the maximal utility, this step first finds the minimal probability in the vector and compares the it to a given *stopping threshold*. If the minimal probability is greater than or equal to the stopping threshold, it returns the combined set as the final selected user set of the algorithm. Otherwise it uses the combined set as the selected participants and starts next iteration.

5.3 Core Algorithms of CrowdRecruiter

In this section, we introduce the core algorithms of Call/Mobility Prediction, Utility Calculation and Covering Probability Vector Estimation.

5.3.1 Call/Mobility Prediction

Assuming the call sequence follows an inhomogeneous Poisson process [87], the probability of a user u to place n phone calls at cell tower $t(t \in T)$ in sensing cycle $i(0 \leq i < N)$ can be modeled as:

$$p_{i,t}(u,n) = \lambda_{u,i,t}^{n} * e^{-\lambda_{u,i,t}}/n! \tag{5.1}$$

where $\lambda_{u,i,t}$ refers to the Poisson intensity, which is estimated as the average number of calls that u has placed at t in the historical traces corresponding to the sensing cycle i. For example, to estimate $\lambda_{u,i,t}$ for sensing cycle i from 08:00 to 09:00, we will count the average number of calls placed by u at t during the same period 08:00-09:00 in the historical records. Therefore, we can estimate the probability of user u placing at least one call during i through t as Eq. 5.2.

$$P_{i,t}(u) = \sum_{n=1}^{+\infty} P_{i,t}(u,n) = 1 - e^{-\lambda_{u,i,t}} \tag{5.2}$$

5.3.2 Utility Calculation of Each Combined Set

Given each combined set ($\mathbb{S} \cup \{u\}$) of participants, this algorithm computes the utility of the combined set ($Utility(\mathbb{S} \cup \{u\})$). Specifically, we define the utility of a combined set as the *expectation of cell towers being covered by the combined set in all sensing cycles*. We compute the utility as:

$$Utility(\mathbb{S} \cup \{u\}) = \sum_{0 \leq i < N} \sum_{t \in T} \mathbb{Q}_{i,t}(\mathbb{S} \cup \{u\}), \tag{5.3}$$

where $\mathbb{Q}_{i,t}(\mathbb{S} \cup \{u\})$ refers to the probability of a given cell tower t being covered by the combined set during sensing cycle i, and the probability is estimated as:

$$\mathbb{Q}_{i,t}(\mathbb{S} \cup \{u\}) = 1 - \prod_{\forall \upsilon \in \mathbb{S} \cup \{u\}} (1 - P_{i,t}(\upsilon)) \tag{5.4}$$

Given the utility of each combined set, CrowdRecruiter picks the best set having the maximal utility and continues for next iteration until the stopping criterion is met. Our theoretical analysis in **Algorithm Analysis** section shows our approach's approximation to an optimal solution.

5.3.3 Coverage Probability Vector Calculation

Given the combined set ($\mathbb{S} \cup \{u\}$) with the maximal utility, this algorithm computes a vector of probabilities, where each element of the vector is the probability of at least a predefined percentage (R_{atio}) of cell towers being covered by ($\mathbb{S} \cup \{u\}$) in the corresponding sensing cycle. Eq. 5.5 gives the formula to estimate the probability at the i^{th} sensing cycle.

$$COV_i(\mathbb{S} \cup \{u\}) = \sum_{T_c \subseteq T}^{|T_c| \geq \tau} \prod_{\forall t \in T_c} \mathbb{Q}_{i,t}(\mathbb{S} \cup \{u\}) * \prod_{\forall t' \in T \setminus T_c} (1 - \mathbb{Q}_{i,t'}(\mathbb{S} \cup \{u\})) \tag{5.5}$$

where $\tau = \lceil |T| * R_{atio} \rceil$ refers to the minimum number of cell towers that should be covered in every sensing cycle, T_c is a subset of T, referring to a combination of cell towers that should be covered (i.e., $|T_c| \geq \tau$). Considering the computational complexity of Eq. 5.5, we introduce an algorithm to reduce the cost in **Appendix A.2.1**.

5.3.4 Algorithm Analysis

In this section, we analyze the proposed algorithms. First, we propose a brute force approach that can find optimal solution to the participant selection problem. Second, we analyze the time complexity of getting the optimal solution using the brute force approach. Finally, we show how the proposed algorithm could approximate the optimal solution but with low computational complexity.

Intuitively, it is easy to think of a brute force approach as follows. *Suppose there exists an algorithm, given a number $k \leq |\mathbb{U}|$, being capable of enumerating all possible combinations of k users and further finding the combination S_k with the maximal coverage of cell towers in all sensing cycles. Given this algorithm, an ideal solution is to run this algorithm from $k = 1, 2, 3 \ldots$, until it finds $S_{k'}$ making a predefined number of cell towers being covered in each cycle.* The resulting $S_{k'}$ should be the optimal solution.

It is, however, impossible to get the optimal solution using the brute force approach in polynomial time. The total number of k-user combinations inside $\mathbb{U}$ is $\frac{|\mathbb{U}|!}{(|\mathbb{U}|-K)!*k!}$, which grows combinatorially when the number of users ($|\mathbb{U}|$) increases. For example, there are totally $3.0 \times e^{+64}$ combinations for picking 50 users from 1000 users. Thus we need a solution that approximates the optimal result but with low computational complexity.

Table 5.1: Number of Selected Participants (CR. refers to *CrowdRecruiter* in all Tables and Figures)

(a) **BUSINESS** Region

Task	CR.	MaxMin	MaxCom	MaxCov
$R_{atio} = 95\%$				
1	523	601	810	1309
2	510	576	822	1266
3	414	475	748	2130
4	704	753	1424	1965
avg.	537.8	601.3	951	1667.5
$R_{atio} = 85\%$				
1	261	289	413	525
2	257	274	390	454
3	198	255	379	455
4	318	354	506	822
avg.	258.5	293	422	564

(b) **RESIDENTIAL** Region

Task	CR.	MaxMin	MaxCom	MaxCov
$R_{atio} = 95\%$				
1	501	598	911	2112
2	512	638	714	2251
3	508	624	768	1701
4	500	631	631	1576
avg.	505.3	622.8	756	1910
$R_{atio} = 85\%$				
1	257	315	373	585
2	261	326	345	717
3	268	320	355	618
4	243	350	275	551
avg.	257.3	327.8	337	617.8

(c) **MERGED** Region

Task	CR.	MaxMin	MaxCom	MaxCov
$R_{atio} = 95\%$				
1	766	859	1239	2368
2	753	818	1113	2385
3	688	760	1109	2479
4	898	1012	1703	2537
avg.	776.3	862.3	1291	2442.3
$R_{atio} = 85\%$				
1	320	419	547	902
2	329	348	599	892
3	285	305	465	959
4	353	437	620	1236
avg.	321.8	377.3	557.8	997.3

The proposed CrowdRecruiter approach adopts a simple iterative process based on Greedy Algorithms. In the worst case, the algorithm runs $|\mathbb{U}| * (|\mathbb{U}| + 1)/2$ iterations (all users being selected), in order to get the solution. In the best case, the algorithm needs to run $|\mathbb{U}|$ iterations (i.e., selecting a single user meeting the goal). Further, the utility function we defined is a non-negative/non-decreasing submodular set function (proof in **Appendix A.2.2**). According to the theory of submodular function maximization [95], this greedy participant selection process gets a *Near-Optimal* solution in maximizing the utility function with a constant error bound. Suppose S_{k^c} is the $k-$user combination selected by CrowdRecruiter and $S_{k'}$ is the optimal solution having the maximal utility among all $k-$user combinations. There exists $Utility(S_{k^c}) \geq (1-1/e) * Utility(S_{k'}) \approx 0.63 * Utility(S_{k'})$ [95]. As a reference, supposing our algorithm has selected 10 users with the expectation of covering 63 cell towers, the optimal solution among all enumerated 10-user combinations could cover no more than 100 cell towers in expectation. In this way, when CrowdRecruiter finds a set of users being able to meet the predefined coverage ratio in all sensing cycles, the set of users should be near-minimal. For the theoretical treatment of this approximation, the readers are encouraged to find more details in [94].

5.4 Evaluation

In this section, we report the evaluation results using large-scale real-world call traces to verify the effectiveness of CrowdRecruiter's participant selection algorithms for PCS tasks. We first introduce three baseline methods for evaluation. Then we present three D4D phone call traces and the basic experiment settings. Finally, the detailed evaluation results of CrowdRecruiter with respect to the three baseline methods are presented and compared.

5.4.1 Baseline Methods

In our evaluation, we provide three baseline methods with different utility functions from CrowdRecruiter, but all of them share the same iteration process and stopping criterion.

1. **MaxMin -** Given each combined set ($\{u\} \cup \mathbb{S}$), MaxMin computes the utility as the minimum probability among all sensing cycles, i.e., $\min_{0 \leq i < N}\{COV_i(\{u\} \cup \mathbb{S})\}$. MaxMin then picks the combined set with maximum utility as the selected set in each iteration. Intuitively, the MaxMin algorithm tries to maximize the minimal probability when adding the next participant to the selected set in order to make the proposed stopping criterion being achieved as fast as possible. In a sense, MaxMin method aims to select the minimal number of participants by improving the minimal coverage probability of the selected set in each iteration, while CrowdRecruiter intends to achieve the same objective by improving the overall coverage probability of the selected set in each iteration.

2. **MaxCom -** The basic idea of MaxCom is to select the next participant who best complements with the selected set of participants in terms of coverage probability. Given the selected users and unselected users, MaxCom first computes the difference between the predefined probabilistic coverage and the coverage of selected users, obtaining an error matrix corresponding to user's call/mobility profile. Subsequently the MaxCom algorithm selects the unselected user having the most similar call/mobility profile to the error matrix. Finally MaxCom combines the user with the selected users as the combined set for further computation. MaxCom is implemented based on the idea proposed by [33].

3. **MaxCov -** The basic idea of MaxCov is to simply select the next participant who covered the most cell towers in the historical call traces, among all the unselected mobile users.

5.4.2 Dataset and Experiment Setups

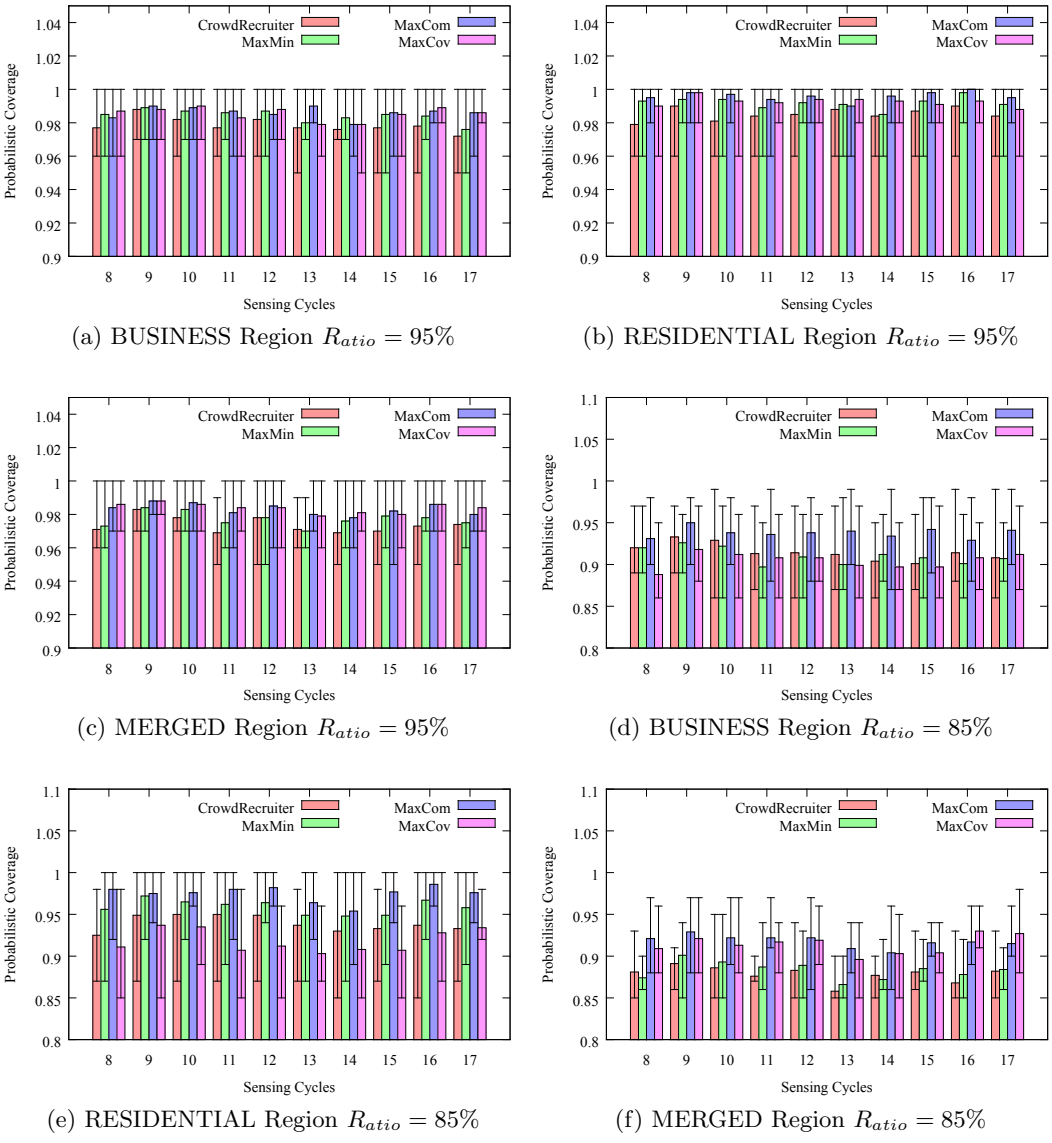
The dataset we used in evaluation is the D4D dataset [15], which contains 50,000 users' phone call traces (each call records includes user id, call time, and cell tower) from Cote d'Ivoire. All these users are re-selected randomly every 2 weeks with anonymized user ids. Thus in this study, we design experiments based on such two-week periods. The call traces in the first week were used for participant selection, and we simulated the spatial-temporal coverage of selected participants using call traces in the second week. Specifically, we extract the call traces of two connected regions in four two-week periods and build the following three datasets for our evaluation:

- **BUSINESS -** a commercial center of the city where 86 cell towers having been installed and around 7945-8799 mobile phone users placing phone calls in any two-week period.

- **RESIDENTIAL -** a residential area where 45 cell towers having been installed and around 6034-6627 mobile phone users placing phone calls in any two-week period.

- **MERGED -** combined area of both BUSINESS and RESIDENTIAL regions where 131 cell towers having been installed and around 11363-12049 unique mobile phone users placing phone calls in any two-week slot.

We used the four periods' call traces to simulate four PCS tasks, each lasting for 2 weeks. We assume that each PCS task executes 5 days per week, 10 *sensing cycles* every working day from 8:00 to 18:00, with each *sensing cycle* lasting for 1 hour (i.e. 8:00-09:00, ..., 17:00-18:00). In all experiments, we set the *stopping threshold* in *stopping criterion* using an empirical value of $(99.99\%)^{1/(|T|*N)}$ for evaluating CrowdRecruiter as well as other three baselines.



(a) BUSINESS Region $R_{atio} = 95\%$

(b) RESIDENTIAL Region $R_{atio} = 95\%$

(c) MERGED Region $R_{atio} = 95\%$

(d) BUSINESS Region $R_{atio} = 85\%$

(e) RESIDENTIAL Region $R_{atio} = 85\%$

(f) MERGED Region $R_{atio} = 85\%$

Figure 5.3: Max/Min/Average Coverage of Cell Towers based on the Three Regions and Settings

5.4.3 Number of Participants Comparison

In Table 5.1, we present the performance comparison on number of selected participants between CrowdRecruiter and baselines. It is clear that CrowdRecruiter outperforms MaxMin, MaxCom and MaxCov methods in all PCS tasks—On average, CrowdRecruiter selects 10.0% - 21.5% fewer participants compared to MaxMin, selects 23.7% - 43.5% fewer participants compared to MaxCom, and selects 54.2% - 73.5% fewer participants compared to MaxCov.

In terms of *coverage*, we show the Max/Min/Average percentage of cell towers being covered by the selected participants for each sensing cycle in Figure 5.3. For all sensing cycles, the required percentage (i.e., 95% and 85%) of cell towers are covered by the selected participants for all four methods without significant differences.

5.4.4 Selection Process Comparison

Here we show and compare the participant selection process of the top three selection methods, using BUSINESS region and $R_{atio} = 95\%$ as an example. Figure 5.4 illustrates the variation of the minimal coverage among all sensing cyles ($min_{0 \leq i < N} COV_i(S)$) over the number of already selected participants ($|S|$) using CrowdRecruiter, MaxMin and MaxCom, where we can observe how the minimal coverage probability evolves:

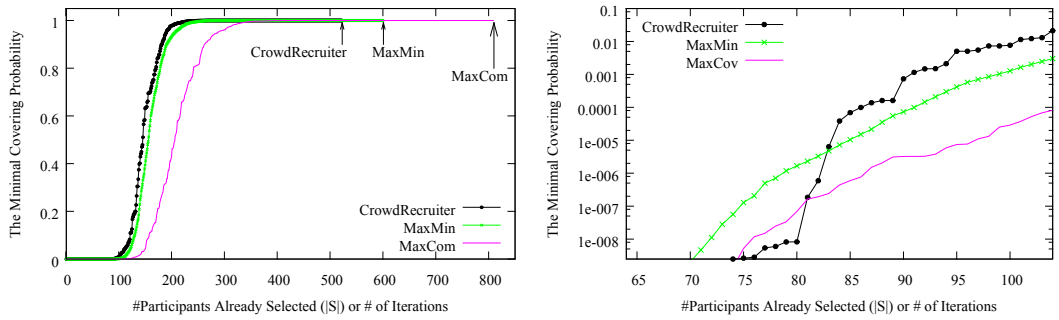
1. In Figure 5.4a we can see that CrowdRecruiter makes $min_{0 \leq i < N} COV_i(S)$ grow fastest of all three methods. At the tail of the curve, CrowdRecruiter makes the $Min_{0 \leq i < N} COV_i(S)$ converge to the predefined threshold with the smallest number of selected participants.

2. From the zoom-in Figure 5.4b, we can find MaxMin and MaxCom had higher $min_{0 \leq i < N} COV_i(S)$ than CrowdRecruiter when $|S| < 80$. However, CrowdRecruiter outperforms other two methods in maximizing $min_{0 \leq i < N} COV_i(S)$ when $|S| \geq 83$.

Based on above two observations, we conclude that, though CrowdRecruiter is not designed to optimize $min_{0 \leq i < N} COV_i(S)$, it can approximate to the optimal solution and perform the best among all these methods.

5.4.5 Participant Selection Overlaps

Now we compare the participants selected by all these algorithms for BUSINESS region under $R_{atio} = 95\%$ and 85%. We count the number of common participants shared by any two of algorithms. Figure 5.5 shows a matrix diagram corresponding to the percentage of common participants selected by both $Algorithm_x$ and $Algorithm_y$ inside $Algorithm_x$. For example, Figure 5.5a shows that on average 72% of CrowdRecruiter's selected participants are shared with MaxMin when $R_{atio} = 95\%$, while



(a) Participant Selection Process (b) Zoom-in $65 \leq |S| \leq 105$

Figure 5.4: Selection Process: $\min_{0 \leq i < N}\{COV_i(\mathbb{S})\}$ over $|S|$

on average 64% of MaxMin's selected participants are shared with CrowdRecruiter under the same settings. From Figure 5.5a, we can see that CrowdRecruiter shares a large number of selected participants with MaxMin (72%), MaxCom (65%) and MaxCov (61%) methods. These results show that the bigger the number of shared participants between CrowdRecruiter and the baseline method, the better the baseline method performs. A similar phenomenon can be observed in Figure 5.5b under the coverage constraint $R_{atio} = 85\%$.

5.4.6 Performance Evaluation and Comparison

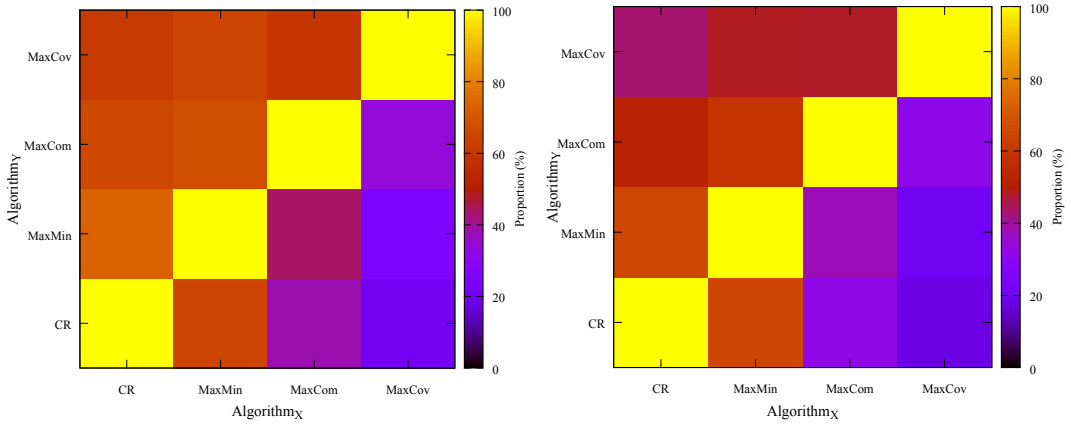
In this section we investigate how the proposed algorithms perform when applying to a large region and two connected sub-regions. Specifically, we evaluate and compare the performance of all participant selection algorithms using the BUSINESS, RESIDENTIAL and MERGED datasets, where the region for MERGED dataset is just the sum of two connected sub-areas BUSINESS and RESIDENTIAL. We measure the computation time and examine the selected participants for these three regions. We carried out experiments using a laptop with an Intel Core i7-2630QM Quart-Core CPU and 8GB memory. CrowdRecruiter and baseline algorithms were implemented with the Java SE platform on a Java HotSpot(TM) 64-Bit Server VM.

5.4.6.1 Computation Time Analysis

Table 6.1 presents the average *time* consumed in each phase using RESIDENTIAL (45 cell towers), BUSINESS (86 cell towers) and MERGED (131 cell towers) datasets with constraint $R_{atio} = 95\%$. For any region, CrowdRecruiter consumes significantly less time than MaxMin while needing similar time as MaxCom and MaxCov. The time consumption on MERGED dataset (47.6 sec.) is slightly longer than the sum of BUSINESS and RESIDENTIAL (31.6 sec.). However, the total time consumption of

Table 5.2: Computation Time Comparison (in seconds, `Phase I`: *Data Preparation and User Call/Mobility Profiling*, `Phase II`: *Iterative Participant Selection Process*)

	CR.	MaxMin	MaxCom	MaxCov
BUSINESS Region				
Phase I	4.9	5.0	4.9	4.9
Phase II	16.4	1350.4	25.9	3.7
RESIDENTIAL Region				
Phase I	3.7	3.6	3.6	3.7
Phase II	6.6	234.9	15.0	2.3
MERGED Region				
Phase I	10.2	11.1	9.8	10.3
Phase II	37.4	4611.2	56.9	12.5



(a) $R_{atio} = 95\%$ (b) $R_{atio} = 85\%$

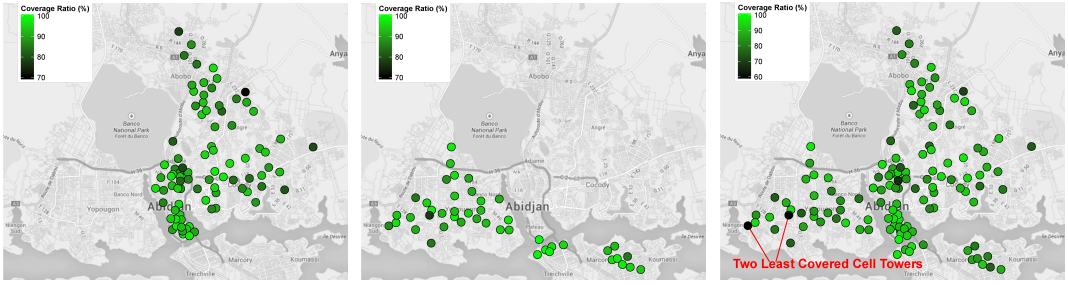
Figure 5.5: Percentage of Shared Participants among Different Methods (Best Viewed in Digital Format)

CrowdRecruiter on MERGED dataset is no more than 1 minute, while MaxMin may need to spend more than 1 hour to find the solution.

5.4.6.2 Analysis of Selected Participants

Now we investigate the question whether it is beneficial to select participants in a large combined region or in two sub-regions with CrowdRecruiter. We compare the combined results based on BUSINESS and RESIDENTIAL datasets to the results based on MERGED dataset, in terms of number of selected participants and spatial-temporal coverage.

Table 5.3 presents the combined number of selected participants where a participant selected for both regions is counted only once. Clearly the combined number of selected participants for all methods is significantly larger than the number of participants selected for MERGED region as shown in Table 5.1c. That means, all these



(a) BUSINESS 85% (b) RESIDENTIAL 85% (c) MERGED 85%

Figure 5.6: Temporal Coverage Ratio of Cell Towers in BUSINESS, RESIDENTIAL and MERGED Regions

methods may select more participants if they apply to subareas of a target region, leading to less computation time but more incentive payments.

As the coverage ratio specified in this work is not 100%, it is conceivable that some cell towers may have low temporal coverage or zero coverage (e.g., not covered in any sensing cycle). Thus we would like to examine the temporal coverage of the cell towers using the three datasets when Ratio = 85%. As shown in Figure 5.6, while most cell towers can be covered in more than 80% sensing cycles when using the BUSINESS, RESIDENTIAL and MERGED datasets, the two least covered cell towers using three datasets fall into the tower id = 724 and id = 646 in the MERGED region, where both cell towers were still covered in 59% of the sensing cycles.

These results suggest the tradeoff between incentive cost and computation time when deciding whether to employ *divide and conquer* selection strategy. When the computation complexity of participant selection is too high in a large area, it can be reduced by selecting a less optimal set of participants in multiple small sub-areas and paying more incentives.

5.4.7 Combine Participants from Each Cycle

CrowdRecruiter has adopted a *per-task selection* approach, optimizing the selected participants for all the sensing cycles. An alternative approach is to select participants for each cycle and combine them into the solution for the task. Here we evaluate the algorithms of CrowdRecruiter under such *per-cycle selection* settings (denoted as *CrowdRecruiter-A*) using the MERGED dataset, and compare the result to that of CrowdRecruiter's *per-task selection* approach. The key findings are summarized as follows:

1. Under the *per-cycle-selection* setting, on average 321 participants are selected for each sensing cycle by CrowdRecruiter-A, where more than 65% of the selected participants for each sensing cycle are also selected by CrowdRecruiter (in *per-*

Table 5.3: The Combined Number of Selected Participants using **BUSINESS+RESIDENTIAL** Datasets

Task	CR	MaxMin	MaxCom	MaxCov
$R_{atio} = 95\%$				
1	1007	1180	1678	3066
2	1009	1197	1502	3194
3	906	1084	1484	3390
4	1178	1354	1999	3201
avg.	1025	1203.8	1665.8	3212.8
$R_{atio} = 85\%$				
1	515	602	774	1036
2	515	596	730	1114
3	462	570	729	1001
4	553	699	775	1280
avg.	511.3	616.8	752	1107.8

task-selection setting).

2. Combining the selected participants for all sensing cycles, on average 1491 participants are selected by CrowdRecruiter-A over the four PCS tasks—i.e., 92.1% more participants than CrowdRecruiter. Specifically, 95.0% participants selected by CrowdRecruiter also appear in the combined result of CrowdRecruiter-A; while more than 50% participants in the combined result of CrowdRecruiter-A are not selected by CrowdRecruiter.

3. The participants selected by CrowdRecruiter-A for any two different sensing cycles are mostly different, only around 30% selected participants are shared.

It can be seen that even though CrowdRecruiter-A selects fewer participants for each sensing cycle, the total number of selected participants for the whole PCS task is much bigger than that selected by CrowdRecruiter.

5.5 Discussion

In this section, we discuss issues that are not reported or addressed in this work, which can be added to our future work.

Redundant Cell Tower Coverage: CrowdRecruiter participants return sensed data by piggybacking over their phone calls. If a participant places multiple calls at a cell tower in one sensing cycle, the CrowdRecruiter client on the phone will only sense and upload during the first call to prevent redundancy. In one sensing cycle, however, a cell tower may still be covered by multiple participants if they all place a call. This redundant coverage usually is not a problem and for some applications it may even be desirable to gather multiple samples in one area. By analyzing the CrowdRecruit

results on MERGED dataset, on average a cell tower gets 2.0 and 3.2 samples from different participants in every sensing cycle with 85% and 95% coverage, respectively. For comparison, if we simply select all users as participants, a cell tower would receive 18 samples on average and 79 samples in maximum.

Call/Mobility and Coverage Prediction: The actual coverage of CrowdRecruiter depends on the set of selected participants and the accuracy of call/mobility prediction. Even though the simple prediction techniques work well in CrowdRecruiter for participant selection, we intend to further improve the coverage evaluation and call/mobility prediction methods in future work. Note that if the actual coverage with historical traces of all volunteers is less than the target coverage, CrowdRecruiter may not return a solution as it will stop when all users have been selected. In such cases, the MCS organizer needs to either adjust the desired coverage constraint, or to recruit more volunteers.

Sensing Coverage and Privacy: Due to the limitation of the D4D dataset, we can only measure the sensing coverage at the cell tower level. As the CrowdRecruiter's approach is general, if the user's mobility traces can be obtained continuously at fine granularity [96], we could support the PCS applications meeting the coverage requirement also at fine granularity. This, however, leads to privacy concerns. Currently CrowdRecruiter uses historical call/mobility traces to derive predictive models. One way to reduce the privacy threats is to only provide predictive models, rather than raw traces, to CrowdRecruiter, as supplied by the mobile operators. Or the CrowdRecruiter client software running on the user's device can capture raw data, but only upload predictive models for participant selection.

Leveraging Multiple Piggyback Sensing Opportunities: In addition to piggyback sensing tasks over mobile phone calls, other piggyback methods also exist. For instance, executing sensing tasks in parallel with Google Map usage also reduces energy consumption when performing PCS tasks [18]. We plan to study the participant selection that leverages multiple piggyback sensing opportunities in a holistic manner as many predictive models, such as for app usage [97], already exist.

Different Incentive Payment Models: In this works we adopt the payment model where each participant receives a fixed amount of incentive through the whole task period. Each participant is requested to sense and upload data for a PCS task in every sensing cycle. In other MCS applications, different incentive payment models may be needed. For instance, a per-job payment may be more engaging for some PCS tasks that require more effort (e.g., taking a picture or recording an audio clip) [25]. As a future work, we plan to study different selection strategies that are suitable for those payment models.

Using Real Sensing Datasets: For many MCS applications (e.g., moving object searching [30] and tracking [98]), the spatial coverage might not be the appropriate constraint required. In our future work, other performance measures need to be derived according to the requirements of MCS applications and sensing datasets.

Chapter 6

CrowdTasker: Maximizing Coverage Quality under Incentive Budget Constraint

Contents

6.1 Introduction . . . 102

6.2 CrowdTasker System Overview . . . 107

6.2.1 Task Allocation Problem in CrowdTasker . . . 107

6.2.2 Overall Design of CrowdTasker . . . 108

6.2.2.1 Predicting each user's call/mobility using the historical call/mobility traces . . . 108

6.2.2.2 Selecting the participants and determine in which sensing cycles the participants are allocated sensing tasks using an Iterative Greedy Process . . . 109

6.3 Core Algorithms and Analysis . . . 110

6.3.1 Call/Mobility Prediction . . . 110

6.3.2 Utility Calculation . . . 110

6.3.2.1 The $Utility_1$ Calculation . . . 110

6.3.2.2 The $Utility_n$ Calculation ($n \geq 2$) . . . 111

6.3.3 Coverage Quality Estimation . . . 111

6.3.4 Algorithm Analysis . . . 112

6.4 Evaluation . . . 114

6.4.1 Baselines for Evaluation . . . 114

6.4.2 Dataset for Evaluation . . . 114

6.4.3 Coverage Quality Comparison under Budget Constraint . . . 115

6.4.4 Spatial Distribution of the Sensor Readings . . . 116

6.4.5 Computation Time of CrowdTasker . . . 116

6.5 Discussion . . . 117

6.1 Introduction

While Chapter 5 introduces CrowdRecruiter framework leveraging the energy-efficient piggyback crowdsensing (PCS) task model and aiming to minimize the total incentive payment under probabilistic coverage constraint, in this chapter we propose an novel PCS task allocation framework, CrowdTasker, intending to maximize the overall coverage quality under the fixed incentive budget constraint. Instead of characterizing the MCS sensing data quality using the probabilistic coverage, CrowdTasker leverages a novel coverage quality measurement considering the *number of sensed results* obtain in each subarea/time-slot and the *spatial-temporal coverage*; further, rather than rewarding each participant an equal-amount of incentives, CrowdTask adopts a *flexible incentive model*. More specific, Our research is motivated by following observations:

1. **Data Quality of MCS Tasks.** For each MCS task (including PCS), the organizer needs to specify the target sensing area, which often consists of a set of subareas. The organizer also needs to specify the sensing duration (e.g. 10 days), which is usually divided into equal-length sensing cycles (e.g. each cycle lasts for an hour). The objective of an MCS task is typically to collect certain environment data from mobile crowd in the target area in each sensing cycle, with the goal of ensuring certain data quality in each sensing cycle. In order to ensure data quality, a common approach is to collect more than one reading from each subarea, so that the actual value of each subarea can be deduced from multiple sensor readings. Taking a one-week urban air quality monitoring MCS task as an example, the MCS organizer first divides the whole area into $1km^2$ grid cells and then splits the one-week MCS sensing time into a sequence of one-hour sensing cycles [21]. If we request to sense one reading with one mobile device from each grid cell, the reading might be inaccurate due to various reasons related to the sensing device or sensing condition. If we request to sense more readings from several mobile phones in each grid cell, the deduced value from multiple readings can better characterize the status of the grid cell. However, if we increase the number of sensing readings in each grid cell above a certain number, the data quality of the deduced value might not increase anymore. Therefore, the data quality of the MCS task is associated with the number of sensor readings in each grid cell, but will saturate when the number of sensor readings reach a certain threshold [99].

2. **Coverage Quality of MCS Tasks.** As data quality is associated with the number of sensor readings in each grid cell, it is thus influenced by mobility of mobile users and the sensing coverage in each sensing cycle. In order to quantify the data quality in MCS, we propose to use *Coverage Quality* as the sensing metrics in this work. The coverage quality of each subarea is characterized by the number of sensor readings obtained in each sensing cycle when the number is smaller than a threshold, and it remains constant when the number of sensed

readings exceeds the threshold. The coverage quality of the whole area is the sum of the coverage quality of all subareas. For example, if the coverage quality threshold is set to 3 (i.e. 3 sensor readings are desired in each subarea and each sensing cycle) for an air quality monitoring trial, the MCS task is said to have the coverage quality of 0, 1, 2, 3, 3 in a certain subarea when it receives 0, 1, 2, 3, 4 sensor readings, respectively. Namely, the data quality will not increase when more than 3 sensor readings are obtained in each subarea. If an MCS task is designed for a target region consisting of five subareas, with coverage quality of 0, 1, 2, 3 and 3 respectively, then the *overall coverage quality* of the MCS task is 0 + 1 + 2 + 3 + 3 = 9. In the rest of this work, we will use the overall coverage quality to characterize the data quality of an MCS task, and set the *overall coverage quality* as the *optimization goal*.

3. **Incentive Model and Total Budget.** In addition to ensuring mobile users to save energy in MCS, one effective way to encourage mobile users' participation in MCS task is to provide incentives (e.g., money, 3G internet bandwidth, etc.) to each user. Typically, each selected participant is offered a certain amount of money as incentive and thus the MCS organizer needs to prepare a budget equal to the total incentives paid to all participants in each MCS task. With the coverage quality and total budget in mind, the MCS organizer needs to select participants with the objective of either

   - *minimizing* the total budget while *ensuring the coverage quality*, or
   - *maximizing* the coverage quality with *a fixed budget*.

   Instead of providing each participant an equal amount of incentive, it is reasonable to give more incentives to active participants if they are requested to collect sensor readings in more sensing cycles. Thus we adopt a more flexible incentive model that consists of the following two components:

   - *Base incentive* - a fixed incentive paid to each selected participant (e.g., $50),
   - *Bonus incentive* - a varying incentive proportional to the number of sensing cycles assigned (e.g., $1 bonus for participating in one sensing cycle).

   For example, for the participant shown in Fig. 6.1 who is involved in three sensing cycles in a PCS task, she would be given $50+$1*3 =$53. In the context of a PCS task for a target sensing region and the given sensing duration, we would like to address the task allocation problem in order to maximize the coverage quality of the PCS system with a fixed amount of incentives.

Motivating Example – The basic idea of CrowdTasker can be illustrated by the following example. *With the help of a telecom operator, an environment NGO plans to monitor the air pollution for citizens in Abidjan City, Cote d'Ivoire, updating the*

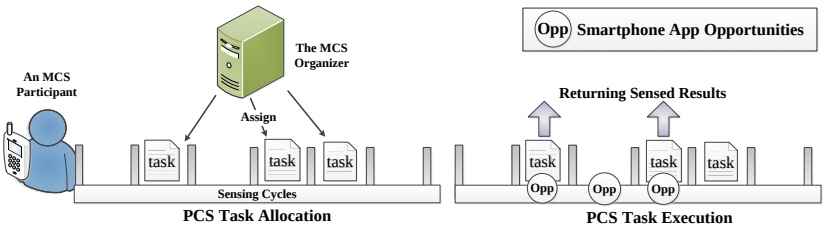


Figure 6.1: PCS Task Allocation and Execution

air pollution index every hour during daytime with a total budget of 30000 euros. For the purpose of air quality sensing, as shown in Fig. 6.2, the NGO splits the urban area (about 100km^2 with 131 cell towers installed) into 131 subareas around each cell tower, where the size of each subarea is less than 1km^2 ; then the NGO divides each working day into 10 sensing cycles (08:00–18:00) and each sensing cycle lasts for one hour. To accurately deduce the air quality index, the NGO aims to collect 3 sensor readings from each subarea per sensing cycle. In each subarea, the coverage quality is counted as 0, 1, 2, and 3 if the MCS task collects 0, 1, 2, and 3 readings, respectively, and the coverage quality remains 3 if more than 3 sensor readings are collected. The overall coverage quality of the target region is the sum of coverage quality in each cell tower.

Through the telecom operator, the NGO makes an agreement with 10000 smartphone users, who are willing to participate in a five-day air quality sensing trial (i.e., 50 cycles in total) and to install a PCS application [18] on their smartphones. According to the agreement, ***(a)*** *a five-day's call and mobility records of the 10000 candidates (including the time stamp and cell tower ID for each call) before the trial are made available to the NGO by the telecom operator;* ***(b)*** *the NGO will provide each selected MCS participant a base incentive of 50 euros and a bonus incentive of 1 euro for each assigned sensing cycle;* ***(c)*** *the PCS application will sense and upload air quality data when the selected participant places a 3G call at a new subarea in each assigned sensing cycle.*

Thus each selected participant could receive 51 to 100 euros in the five-day sensing trial, depending on how many sensing cycles they are assigned sensing tasks. Given the budget of 30000 euros, the MCS organizer can recruit 300 to 588 participants and each selected participant could be assigned MCS task for 1–50 sensing cycles, thus the best solution to the task allocation with the fixed budget of 30000 euros is to find the best user combination and each selected user's best cycle combination, to maximize the coverage quality across all 50 sensing cycles throughout the five-day PCS trial.

Technical Challenges. Given the above use case and research objectives, there are at least three research challenges in the PCS system design:

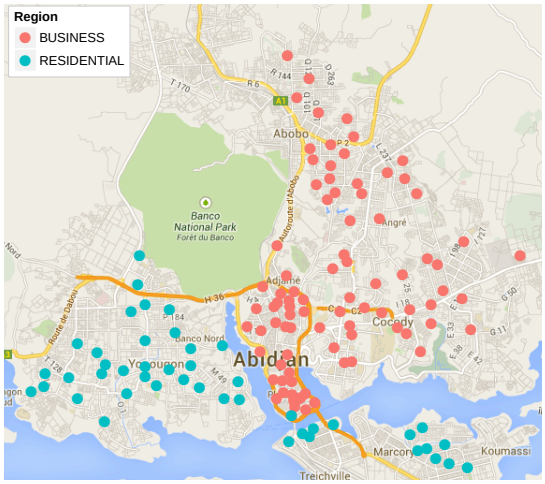


Figure 6.2: Target region in the Downtown of Abidjan City

1) Predicting each user's call/mobility based on their historic call/mobility traces and estimating the coverage quality of a selected set of participants with sensing tasks allocated in different sensing cycles. Since we only have user's historical call records and mobility traces, and the call/mobility pattern will change in the PCS deployment week, we thus have to find a way to predict the call/mobility pattern of each user accurately. Even with inaccurate prediction results, we need to characterize the spatial coverage probability of each participant and estimate the joint coverage quality of the selected set of participants with sensing tasks allocated in different sensing cycles.

2) Lowering the complexity of task allocation in order to achieve near-maximal estimated coverage quality under the budget constraint. Considering the motivating example, a brute-force approach of PCS task allocation is to first enumerate all possible user combinations, where each user combination is a user set with 300 to 588 users out of 10000 candidates; then for each user set combination, the algorithm further enumerates all possible user-cycle combinations (user number ranging from 300 to 588, cycle number ranging from 1 to 50) for task allocation in different sensing cycles. Finally for each user-cycle combination (a set of participants with sensing tasks allocated in a set of sensing cycles), the brute-force algorithm estimates the overall coverage quality and the overall incentive payment, and the set satisfying the budget constraint while achieving the maximal estimated coverage quality is selected as the optimal set. This search problem, however, is NP hard in nature [100, 94]. Thus it is necessary to develop a fast approximation algorithm to search the near-optimal combination set achieving near-maximal coverage quality with the given budget.

3) Designing a task allocation process which can approximate the "real cost" of each participant and search the near-optimal set of user-cycle

combinations according to the estimated coverage quality and cost. A common approach for searching the near-optimal user-cycle combination set is the greedy algorithm [101], which incrementally adds new user to the selected set of participants and searches the best user-cycle combination in terms of estimated coverage quality, where each user-cycle combination refers to allocating a sensing task to a certain user in a certain sensing cycle. Specifically, in each iteration of the greedy search, each unselected user-cycle combination is combined with already selected ones. And among all the combined sets, the set with the highest *Coverage Quality Improvement per Incentive Cost* is selected as the best set. If the given budget is used up by the selected set, the near-optimal combination set is said to be found and the greedy search process terminates. Otherwise another user-cycle combination is added in the selected set until the budget is fully utilized. The real incentive cost of each user, however, depends on how many cycles he/she gets assigned with sensing tasks. As we cannot foreknow this in the process, it is hard to compute the "real cost" of each user-cycle combination. Therefore, we need to design a task allocation process which can iteratively approximate the "real cost" of each participant and select the near-optimal set of user-cycle combinations according to the estimated coverage quality and cost.

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

1) We formulated the problem of maximal-coverage-quality task allocation in piggyback crowdsensing (PCS) given the budget constraint, with a flexible incentive model. To the best of our knowledge, this is the first work addressing the task allocation issue in the context of PCS, where we optimally select participants and assign sensing tasks to participants in different sensing cycles according to the predicted call/mobility pattern and leverage the call opportunities of participants to sense and upload data for crowdsensing task.

2) In order to maximize the coverage quality with a fixed amount of incentives, we proposed a two-phase task allocation framework named CrowdTasker. It takes a novel approach to search user-cycle combination set, achieving near-maximal coverage quality under the budget constraint. Theoretical analysis shows that the proposed search algorithm can achieve the near-optimality with low computational complexity.

3) We evaluated our proposed algorithms with the real world dataset D4D [15], which contains 4-month call records of 50,000 users from Cote d'Ivoire. We show that the proposed framework performed better than three baseline approaches, using the call records of two separate regions in Abidjan. Specifically, CrowdTasker achieved 3.0%-60% higher coverage quality on average than the baseline approaches, under the same budget constraint.

6.2 CrowdTasker System Overview

In this section, we formulate the task allocation problem and present the CrowdTasker framework in detail.

6.2.1 Task Allocation Problem in CrowdTasker

In CrowdTasker, assuming (1) a PCS task runs over a period of time (e.g., a week) and each day is comprised of 10 one-hour sensing cycles from 08:00–18:00; (2) The target region of the PCS task consists of a set of subareas, with each subarea around a cell tower; (3) Each selected participant produces one sensor reading in a sensing cycle i if a participant places a call at the corresponding cell tower t in i. Note that if a participant places multiple calls at different cell towers in i, all these cell towers receive a copy of sensor reading from the user in i; (4) CrowdTasker computes coverage quality of the subarea t in cycle i as the minimum between the number of *expected sensor readings* E (i.e., the threshold) and *that of returned sensor readings*, while the *overall coverage quality* is the sum of coverage quality in all subareas across all sensing cycles.

Then, the goal for *task allocation* is to select a number of participants from the volunteering mobile users, and determines in which sensing cycles each selected participant is assigned the PCS task, in order to maximize the overall coverage quality with the given budget. With these definitions, we formulate the task allocation problem in CrowdTasker as follows.

Given a fixed budget for overall incentive payments B, the Base incentive b_a and Bonus incentive b_o, a set of volunteering mobile users $\mathbb{U}$, a target area consisting of a set of cell towers T, the call traces of all users in $\mathbb{U}$ (including the time stamps and associated cell towers of their calls), we denote $\mathbb{S}$ as the set of participants selected from $\mathbb{U}$ (i.e., $\mathbb{S} \subseteq \mathbb{U}$). For each selected participant $\forall u \in \mathbb{S}$, we further denote C_u as a set of cycles assigned to u for PCS task participation (e.g., $C_u = \{0, 2, ...\}$), and $N_{u,i,t}$ as the number of calls made by u at cell tower t in cycle i. The problem is then to find $\mathbb{S}$ as a subset of $\mathbb{U}$ and for $\forall u \in \mathbb{S}$ to assign a subset of sensing cycles C_u, with the objective to

$$max \sum_{0 \le i < I} \sum_{t \in T} min\{\sum_{u \in \mathbb{S}} min\{N_{u,i,t}, 1\} * \mathbb{A}(C_u, i), E\}$$
$$subject\ to\ |\mathbb{S}| * b_a + \sum_{u \in \mathbb{S}} |C_u| * b_o \le B$$

where I is the total number of sensing cycles for the PCS task; $\mathbb{A}(C_u, i)$ is a binary function identifying if the participant u is assigned the PCS task in cycle i. Specifically if $i \in C_u$ then $\mathbb{A}(C_u, i) = 1$, if $i \notin C_u$ then $\mathbb{A}(C_u, i) = 0$. It is worth noting that we cannot foreknow when and where a participant will place a phone call during the PCS task, i.e., $N_{u,i,t}$ is unknown when we select participants.

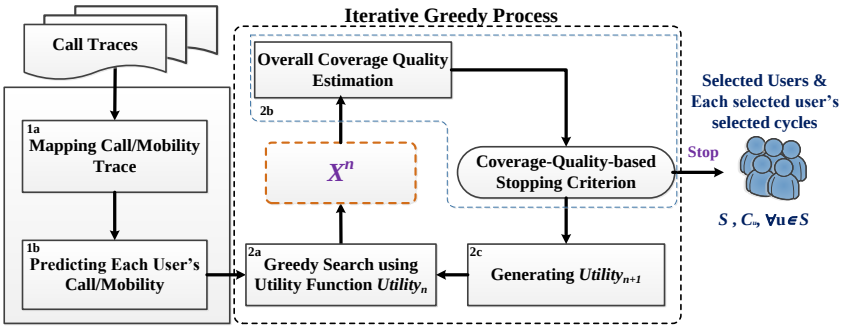


Figure 6.3: The CrowdTasker Framework

6.2.2 Overall Design of CrowdTasker

CrowdTasker follows a centralized task allocation approach, where a central server collects and stores the volunteering mobile users' historical call traces in the target area, and the server selects participants from all volunteering users ($\mathbb{S} \subseteq \mathbb{U}$) and assigns tasks to each participant in a set of sensing cycles (C_u for each $\forall u \in \mathbb{S}$) before the PCS task execution. Only selected participants are needed to perform sensing tasks, and each selected participant returns sensor readings only in the assigned sensing cycles when a phone call is made. In order to solve the above task allocation problem, CrowdTasker employs a two-phase solution. In *Phase 1*, it predicts each user's call/mobility in the trial stage, using the historical call and mobility traces of all users. In *Phase 2*, it incrementally selects participants and assigns sensing tasks to each participant in different sensing cycles based on the prediction results, the estimated coverage quality and incentive cost. The framework is shown in Fig.6.3 and works as follows.

6.2.2.1 Predicting each user's call/mobility using the historical call/mobility traces

Given the call traces of all volunteering mobile users, this phase computes the *call/mobility profile* of each user–i.e., *probability of each user placing at least one call at a particular cell tower in a given sensing cycle*. Specifically, CrowdTasker computes the profile of each user with following two steps:

1a. Mapping Call/Mobility Traces - Given the historical call/mobility traces of all users, this step maps each user's historical call/mobility traces onto I sensing cycles and T cell towers. Then it counts $\lambda_{u,i,t}$–the average number of calls placed by each user ($u \in \mathbb{U}$) at each cell tower ($t \in T$) in each sensing cycle ($0 \leq i < I$);

1b. Predicting each User's Call/Mobility - Given $\lambda_{u,i,t}$, this step estimates $P_{i,t}(u)$–the probability of the user ($u \in \mathbb{U}$) placing at least one call at each cell tower ($t \in T$) during each sensing cycle ($0 \leq i < I$).

6.2.2.2 Selecting the participants and determine in which sensing cycles the participants are allocated sensing tasks using an Iterative Greedy Process

Given the call/mobility profile of each user and the overall incentive budget B, we propose an *Iterative Greedy Process* that can approximate the "real incentive cost" of each participant and search the near-optimal set of user-cycle combinations according to the estimated coverage quality and cost. In order to estimate the "real incentive cost", Phase II first uses the given budget and a greedy search process with *a utility function considering only the estimated coverage quality (namely $Utility_1$)* to select a set of user-cycle combinations (namely $\mathbb{X}^1$), then roughly estimates the incentive cost of each user-cycle combination using the selected set. With the estimated incentive cost, Phase II generates a new Utility function *considering both estimated coverage quality and cost (namely $Utility_2$)*; then the greedy search process is repeated with $Utility_2$ to re-select a new set of user-cycle combinations (namely $\mathbb{X}^2$). In this way, Phase II repeats the process of estimating the incentive cost using the last selected set and searching a new set (i.e., $\mathbb{X}^n$ and $n = 2, 3, 4...$) with the estimated coverage quality and cost (i.e., using $Utility_n$ and $n = 2, 3, 4...$), until the near-optimal combination set is obtained. Specifically, each iteration of the *Iterative Greedy Process* consists of following three steps:

2a. Greedy User-Cycle Combination Set Search - Given the full user set $\mathbb{U}$ and all sensing cycles $0 \leq i < I$, the algorithm combines each volunteering user with each sensing cycle so as to get the complete set of *user-cycle combinations* i.e., $\mathbb{COM} = \{\langle u, i\rangle | \forall u \in \mathbb{U}, 0 \leq i < I\}$, where $\langle u, i\rangle$ refers to the user-cycle combination of user u in cycle i. With the total Budget B and the Utility function $Utility_n$ $(n = 1, 2, 3 \dots)$, the algorithm selects a set of user-cycle combinations incrementally, where:

- The algorithm first selects a single user-cycle combination $\langle u, i\rangle \in \mathbb{COM}$ having the maximal utility (using $Utility_n$) and adds the combination into solution i.e., $\{\langle u, i\rangle\} \rightarrow \mathbb{X}^n$;
- The algorithm then selects one unselected user-cycle combination $\langle v, j\rangle \in \mathbb{COM} \backslash \mathbb{X}^n$ having the maximal utility when combining with $\mathbb{X}^n$ using $Utility_n$, and adds the combination into solution i.e., $\{\langle v, j\rangle\} \cup \mathbb{X}^n \rightarrow \mathbb{X}^n$;
- The algorithm calculates the *remaining budget* as $B_R = B - b_a * |S| - |\mathbb{X}^n| * b_o$ where S is the set of all participants appeared in $\mathbb{X}^n$. Then the algorithm keeps selecting another unselected user-cycle combination in each iteration until the remaining budget is not enough to select one more user-cycle combination, i.e., $B_R < b_o$ **or** $b_o \leq B_R < b_a + b_o$ when $\forall u \in S$ and $0 \leq i < I$ there $\exists \langle u, i\rangle \in \mathbb{X}^n$.

The algorithm finally obtains a set of user-cycle combinations (i.e., $\mathbb{X}^n$) with the given budget B and $Utility_n$.

2b. Overall Coverage Quality Estimation and the Stopping Criterion - Given the set of selected user-cycle combinations $\mathbb{X}^n$ from **2a**, the algorithm estimates the overall coverage quality $CQE(\mathbb{X}^n)$, based on prediction results. Then the algorithm compares $CQE(\mathbb{X}^n)$ to the overall coverage quality of previous iteration $CQE(\mathbb{X}^{n-1})$. If $CQE(\mathbb{X}^n) \leq CQE(\mathbb{X}^{n-1})$ then the algorithm returns $\mathbb{X}^{n-1}$; otherwise the algorithm continues for the next step.

2c. Generating new Utility Function - Given the selected user-cycle combinations $\mathbb{X}^n$, the algorithm computes a new Utility function $Utility_{n+1}$ based on newly estimated incentive cost for the next iteration of 2a.

After the Iterative Greedy Process terminates, all users $\mathbb{S}$ appeared in $\mathbb{X}^{n-1}$ from **2b** (where $\forall\langle u, i\rangle \in \mathbb{X}^{n-1}, \exists u \in \mathbb{S}$) are selected as participants and each selected participant $u \in \mathbb{S}$ is allocated sensing tasks in sensing cycles C_u (where $C_u = \{i | 0 \leq i < I$ and $\exists\langle u, i\rangle \exists \mathbb{X}^{n-1}\}$).

6.3 Core Algorithms and Analysis

In this section, we introduce the core algorithms of Call/Mobility Prediction, Utility Calculation and Coverage Quality Estimation.

6.3.1 Call/Mobility Prediction

Assuming the call sequence follows an inhomogeneous Poisson process [87], the probability of a user u to place *at least one phone call* at cell tower $t (t \in T)$ in sensing cycle $i (0 \leq i < N)$ can be modeled as:

$$P_{i,t}(u) = 1 - e^{-\lambda_{u,i,t}} \tag{6.1}$$

where $\lambda_{u,i,t}$ refers to the Poisson intensity, which is estimated as the average number of calls that u has placed at t in the historical traces corresponding to the sensing cycle i. For example, to estimate $\lambda_{u,i,t}$ for sensing cycle i from 08:00 to 09:00, we will count the average number of calls placed by u at t during the same period 08:00-09:00 in historical traces.

6.3.2 Utility Calculation

We now describe two types of utility functions $Utility_1$ and $Utility_n$ $(n \geq 2)$. $Utility_1$ is used for the first iteration of the Iterative Greedy Process, and a new utility function $Utility_n$ $(n \geq 2)$ is generated for each consecutive iteration.

6.3.2.1 The $Utility_1$ Calculation

Given the set of incrementally selected user-cycle combinations $\mathbb{X}^1$ in the first iteration of *Iterative Greedy Process* ($\mathbb{X}^1 = \emptyset$ for initialization the greedy search process). The

utility for adding a user-cycle combination $\langle v, i\rangle$ combining with $\mathbb{X}^1$ is calculated as:

$$Utility_1(\langle v, j\rangle|\mathbb{X}^1) = CQE(\langle v, j\rangle \cup \mathbb{X}^1) - CQE(\mathbb{X}^1) \tag{6.2}$$

where $CQE(\mathbb{X}^1)$is the estimated coverage quality of $\mathbb{X}^1$, and $CQE(\langle v, j\rangle \cup \mathbb{X}^1)$ is the estimated coverage quality of the combined set merging $\langle v, j\rangle$ and $\mathbb{X}^1$. Intuitively $Utility_1$ is the coverage quality improvement after adding $\langle v, j\rangle$ into $\mathbb{X}^1$.

6.3.2.2 The $Utility_n$ Calculation ($n \geq 2$)

During n^{th} iteration of the Iterative Greedy Process, given the selected set of user-cycle combinations $\mathbb{X}^n$, the algorithm computes the utility for adding each user-cycle combination $\langle v, j\rangle$ to the selected set $\mathbb{X}^n$ as:

$$Utility_n(\langle v, j\rangle|\mathbb{X}^n) = \frac{CQE(\langle v, j\rangle \cup \mathbb{X}^n) - CQE(\mathbb{X}^n)}{cost_n\langle v, j\rangle} \tag{6.3}$$

where $cost_n\langle v, j\rangle$ is the *modular incentive cost* [102] of the user-cycle combination $\langle v, j\rangle$. Intuitively $Utility_n$ is the coverage quality improvement over the incentive cost of allocating a sensing task to a specific user in a specific sensing cycle. $cost_n\langle v, j\rangle$ is computed as:

$$\begin{aligned} cost_n\langle v, j\rangle = C(\mathbb{X}^{n-1}) + \sum_{\langle u,i\rangle \in \{\langle v,j\rangle\} \setminus \mathbb{X}^{n-1}} (b_a + b_o) - \\ \sum_{\langle u,i\rangle \in \mathbb{X}^{n-1} \setminus \{\langle v,j\rangle\}} [C(\mathbb{X}^{n-1}) - C(\mathbb{X}^{n-1} \setminus \{\langle u, i\rangle\})] \end{aligned} \tag{6.4}$$

where $\mathbb{X}^{n-1}$ is the user-cycle combination set selected in the $n-1^{th}$ iteration of Iterative Greedy Process. The cost function $C(\mathbb{X}) = b_a * |S| + b_o * |\mathbb{X}|$ is the total budget of the user-cycle combination set $\mathbb{X}$, where S is the set of participants appeared in $\mathbb{X}$.

6.3.3 Coverage Quality Estimation

Given a set of user-cycle combinations $\mathbb{X}$, which consists of selected participants $\mathbb{S}$ and each selected participant u's sensing cycles C_u for PCS task participation, the coverage quality of $\mathbb{X}$ is

$$\begin{aligned} CQE(\mathbb{X}) = \sum_{0 \leq i < I} \sum_{\tau \in T} \sum_{\forall U \subseteq \mathbb{S}} min\{|U|, E\} * \prod_{\forall u \in U} \\ (P_{i,u}(t) * \mathbb{A}(C_u, i)) * \prod_{\forall v \in \mathbb{S} v \notin U} (1 - P_{i,v}(t) * \mathbb{A}(C_v, i)) \end{aligned} \tag{6.5}$$

where U refers to the set of participants probably returning their sensor readings in cycle i and cell tower t, E refers to the threshold of sensor readings expected to receive

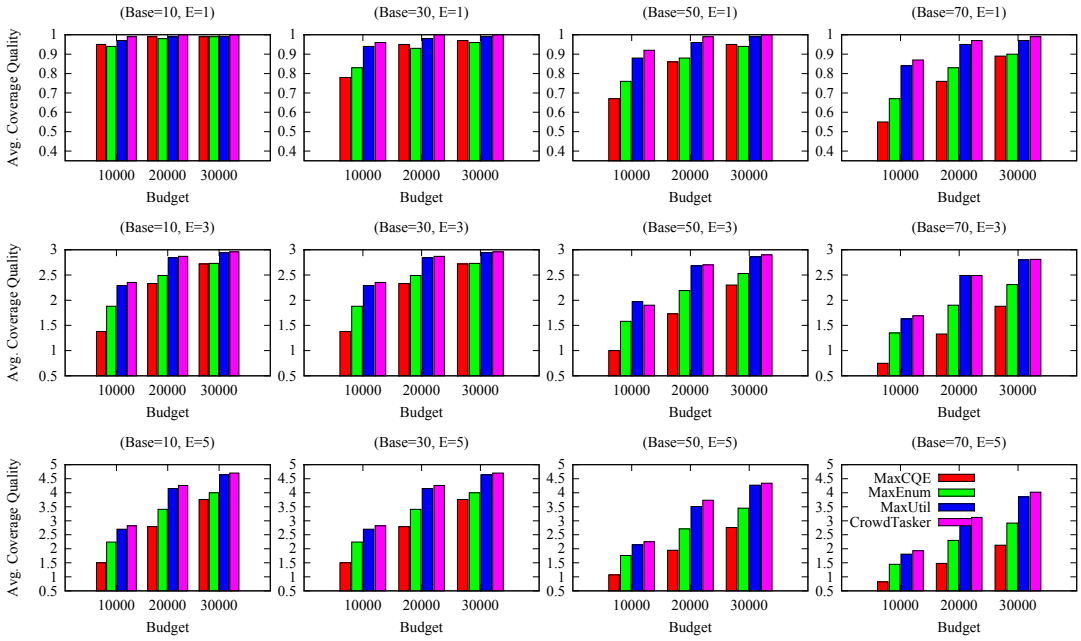
in each subarea/cycle, and the function $\mathbb{A}(C_u, i)$ is defined in Section 6.2.1. To solve Eq. 6.5, we implemented a low complexity algorithm for Eq. 6.5 computation using Probability Generating Function [103].

6.3.4 Algorithm Analysis

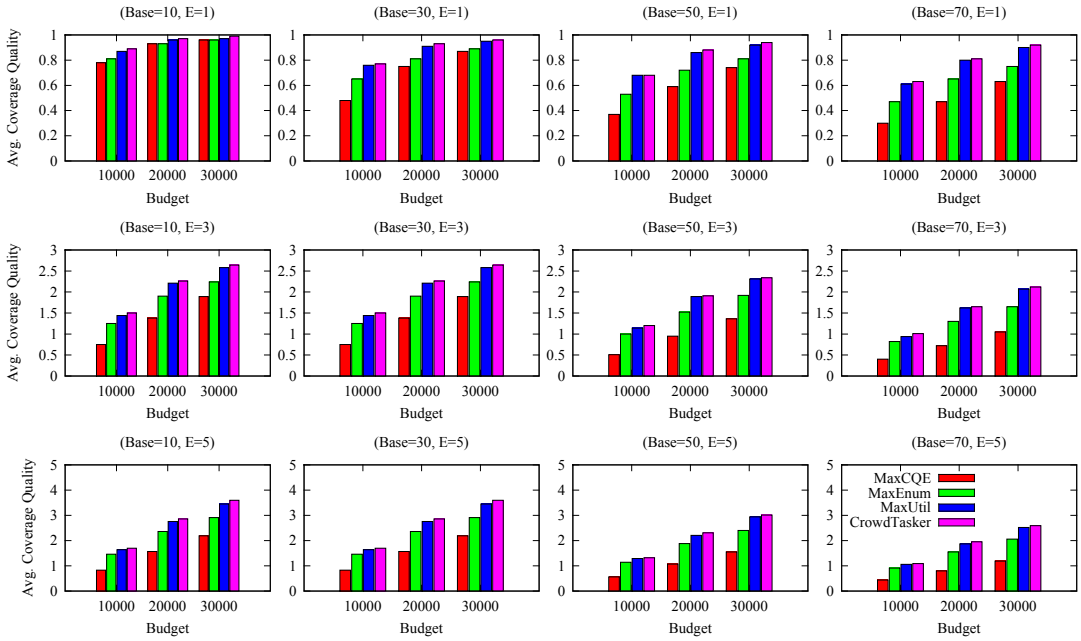
In this section, we first analyze a brute force approach that can find optimal solution of the task allocation problem. We then comparatively show that CrowdTasker achieves near-optimal solution with much lower computational complexity.

Intuitively, a brute force approach can enumerate all possible combinations of k users where $\frac{B}{b_a+b_o*I} \leq k \leq \frac{B}{b_a+b_o}$ *(I is the total number of sensing cycles). Given each user-combination* $S \in \mathbb{U}$*, the algorithm enumerates the cycles of each user in each user-combination, and finds each user's cycle-combination* $(C_u, \forall u \in S)$ *in order to maximize the estimated coverage quality while ensuring the overall incentive payment not exceeding the budget B. Among all user-combinations, the algorithm selects the user-combination* S^* *and the corresponding cycle-combinations* $(C^*_u, \forall u \in S^*)$ *having the maximal estimated coverage quality.* The resulting S^* and $C_u, \forall u \in S^*$ should be the optimal solution for task allocation. It is, however, impossible to get the optimal solution using this brute force approach in polynomial time. The total number of k-user combinations inside $\mathbb{U}$ is $\frac{|\mathbb{U}|!}{(|\mathbb{U}|-k)!*k!}$, which grows combinatorially when the number of users ($|\mathbb{U}|$) increases. The total number of cycle-combinations of a k-user combination is 2^{k*I}, which grows exponentially when the number of cycles (I) increases. As a reference, in our motivating example, there are $1.0 \times e^{491}$ user-combinations for picking 388 users from 5000 users and 1.12×10^{15} possible cycle-combinations for each of the 388 users. Thus we need a solution that approximates the optimal result but with lower computational complexity.

CrowdTasker adopts the *Iterative Greedy Process* with Nested-Loops. In our experience which was also demonstrated in [104], the outer loop typically runs 5–7 iterations in the worst case. The inner-loop (i.e. step 2a in III.B) runs $\frac{|\mathbb{U}|^2*I+1}{2} * I$ iterations in the worst case. In the best case, the algorithm needs to run $|\mathbb{U}| * I * 2$ iterations where the outer loop runs two iterations and the inner loop runs $|\mathbb{U}| * I$ times (i.e., selecting a single cycle of a user). Both $CQE(\mathbb{X})$ and $C(\mathbb{X})$ are submodular functions over $\mathbb{X}$, as proved in **Appendix A.3.1 and A.3.2**. According to the theory of submodular function maximization under the submodular kapsack constraint [104], CrowdTasker can guarantee to get a *Near-Optimal* solution with $(\alpha, 1-e^{-1})$-approximation bound ($\alpha \approx \frac{b_a+bo}{b_a}$) when maximizing CQE with the given budget. For example, given the Base/Bonus incentive settings $b_a = \$50$ and $b_o = \$1$, supposing with \$10000 budget the optimal solution obtained by the brute-force enumeration algorithm achieves the totally coverage quality of 1000 in expectation, then CrowdTasker with $\$10000 * \frac{50+1}{50} = \$10200$ budget can achieve at least a coverage quality of 630.



(a) BUSINESS Region (the legend in the right bottom figure)



(b) MERGED Region (the legend in the right bottom figure)

Figure 6.4: Coverage Quality Comparison in the BUSINESS and MERGED Regions (Best viewed with 300% zoom-in)

6.4 Evaluation

In this section, we report the evaluation results using large-scale real-world call traces to verify the effectiveness of CrowdTasker's task allocation algorithms for PCS tasks. We first introduce three baseline methods. Then we present the three D4D phone call traces collected from three regions of different sizes and the experiment settings. Finally, the detailed evaluation results of CrowdTasker with respect to the three baseline methods under different incentive settings and coverage quality thresholds are presented.

6.4.1 Baselines for Evaluation

We provide three baseline task allocation methods using the greedy and partial enumeration for comparative studies.

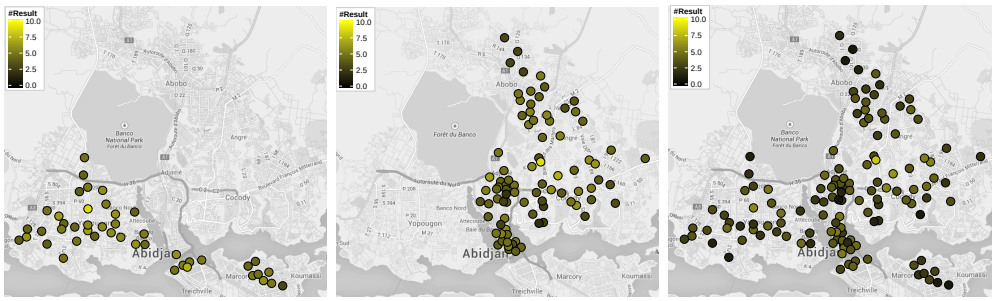
1) **MaxCQE -** This method adopts the same Greedy User-Cycle Combination Set search algorithm (**2a.** of CrowdTasker): adding a user-cycle combination in each iteration and using the same stopping criterion but with a different utility function. In each iteration, given an unselected user-cycle combination $\langle u, i\rangle$, the selected set $\mathbb{X}$, MaxCQE calculates the utility as the coverage quality improvement of adding $\langle u, i\rangle$ to the selected set $\mathbb{X}$, namely $CQE(\langle u, i\rangle \cup \mathbb{X})$.

2) **MaxUtils -** This method uses the same Greedy User-Cycle Combination Set search algorithm as MaxCQE but with a different utility function $\frac{CQE(\langle u,i\rangle \cup \mathbb{X}) - CQE(\mathbb{X})}{C(\langle u,i\rangle \cup \mathbb{X}) - C(\mathbb{X})}$, where $\mathbb{X} = \{\langle v, j\rangle | \forall v \in \mathbb{S}, \forall j \in C_v\}$, $C(\mathbb{X})$ is specified in Equation 6.4 and refers to the total incentive of $\mathbb{X}$. The utility function of MaxUtils is defined as the ratio of the coverage quality improvement and the total incentive difference of adding the new user-cycle combination.

3) **MaxEnum -** Rather than selecting an unselected cycle of a user in each iteration, MaxEnum uses a greedy algorithm to select an unselected participant in each iteration. In each iteration, MaxEnum first enumerates all possible cycle sets of each user, and selects each user's best cycle set (e.g., the cycle set $C_v^{\#}$ for user v) having the maximal utility $\frac{CQE(\mathbb{C}_v^{\#} \cup \mathbb{X}) - CQE(\mathbb{X})}{b_a + b_o * |\mathbb{C}_v^{\#}|}$, where $\mathbb{C}_v^{\#} = \{\langle v, j\rangle | \forall j \in C_v^{\#}\}$ and the utility stands for the "Performance/Cost" ratio (coverage quality improvement versus the cost) of adding the user v with the cycle set $C_v^{\#}$. Then among all unselected users, it selects/adds the user (with the selected cycle set) having the maximal utility. This algorithm continues selecting/adding users (with the cycle sets) one by one until the remained budget is less than $b_a + b_o$, and the participants (with cycles) already selected are returned as the result for task allocation.

6.4.2 Dataset for Evaluation

The dataset we used in evaluation is the D4D dataset [15], which contains 50,000 users' phone call records (each call record includes user id, call time, and cell tower)



(a) RESIDENTIAL Region (b) BUSINESS Region (c) MERGED Region

Figure 6.5: Spatial Distribution of Sensor Readings in three Regions (B=30000, E = 5, $b_a = 50$ and $b_o = 1$)

from Cote d'Ivoire. All these users are re-selected randomly every 2 weeks with anonymized user ids and totally 10 two-week periods of call records are stored in the dataset. In each two-week period, our experiment uses the call/mobility traces in the first week for task allocation, and we tested the coverage quality of selected participants with assigned cycles using call/mobility traces in the second week. Specifically, we extracted the call/mobility traces of two connected regions–**BUSINESS** (86 cell towers with 7945 mobile users in the call records), **RESIDENTIAL** (45 cell towers with 6034 users), and a merged region containing call/mobility traces from above two regions–**MERGED** (131 cell towers with 11363 users), as shown in Fig. 6.2. We further assume that each PCS task executes for 5 days from Monday to Friday in a week, runs 5 *sensing cycles* every working day from 8:00 to 18:00, with each *cycle* lasting 2 hours (i.e. 8:00-10:00, ..., 16:00-18:00). Thus each PCS trial consists of 25 sensing cycles.

6.4.3 Coverage Quality Comparison under Budget Constraint

In Fig. 6.4, we present the average coverage quality in each sensing cycle and each cell tower of the four methods under the same budget/incentive settings, when:

- The Bonus incentive is fixed to $b_o = 1$, while the base incentive is set to B_a =10, 30, 50 and 70;
- The total amount of incentive budget is set to B =10000, 20000 and 30000;
- The coverage quality threshold in each cell tower/sensing cycle is set to E =1, 3 and 5.

Note that the average coverage quality could not be bigger than E, as the maximal coverage quality of each cell tower/cycle is E. Due to the space limit, we only show the evaluation results with BUSINESS and MERGED regions for the two-week period

from Dec. 12, 2011 to Jan. 01, 2012. From the coverage quality comparisons shown in Fig. 6.4, we can observe that:

- *In all the cases CrowdTasker outperformed the three baselines under the same budget constraint.* Specifically, CrowdTasker achieved on average 60% higher coverage quality than MaxCQE, 18% higher than MaxEnum, and 3% higher than MaxUtils. The evaluation results based on RESIDENTIAL region shows similar results.
- *Using any of these four methods, higher average coverage quality (per cell tower/cycle) can be achieved in the BUSINESS region than that in the MERGED region under the same budget and incentive settings.* When using CrowdTasker, the coverage quality in each cell tower/cycle of the BUSINESS region is on average 21% higher than that in the MERGED region under all incentive/budget settings. Note that BUSINESS region is a subset of the MERGED region with fewer cell towers installed. Thus, it is reasonable to expect that under the same budget constraint, CrowdTasker could achieve higher coverage quality on average in small target region (e.g. BUSINESS) than that in big target region (e.g. MERGED).

6.4.4 Spatial Distribution of the Sensor Readings

After evaluating the performance of CrowdTasker and three baselines from coverage quality perspectives, we evaluate the spatial distribution of sensor readings using CrowdTasker with the target regions of different size. In Fig. 6.5, we present the average number of sensor readings returned from each cell tower in each sensing cycle using CrowdTasker, using the dataets from the BUSINESS, RESIDENTIAL and MERGED regions, with the same setting $B = 30000$, $E = 5$, $b_a = 50$ and $b_o = 1$. From Fig. 6.5, we can see that *when using CrowdTasker, the sensor readings are uniformly distributed across cell towers in any of the three regions.* While the coverage quality threshold in each cell tower/cycle is set to $E = 5$, the experiment shows that each cell tower gets on average 5.3, 4.5 and 3.3 sensor readings using datasets from RESIDENTIAL, BUSINESS and MERGED regions, respectively. Further the standard deviation is 0.98, 1.1 and 1.2 for three regions, respectively. This suggests that each cell tower can get a comparable number of sensor readings in any of the three regions using CrowdTasker.

6.4.5 Computation Time of CrowdTasker

In this section, we evaluate the computation time of CrowdTasker and three baseline methods, and show how fast each method could complete the task allocation process. We carried out experiments using a laptop with an Intel Core i7-2630QM Quart-Core CPU and 8GB memory. CrowdTasker and baseline algorithms were implemented with the Java SE platform on a Java HotSpot^{TM} 64-Bit Server. Table 6.1 presents

Table 6.1: Computation Time Comparison (in seconds, $B = 30000$, $E = 5$, $b_a = 50$ and $b_o = 1$)

Regions	CrowdTasker	MaxCQE	MaxEnum	MaxUtils
RESIDENTIAL	929.9	113.1	83.5	408
BUSINESS	3022.4	168.7	224.2	786.1
MERGED	4112.3	217.9	276.1	843.3

the average *time* consumed using RESIDENTIAL (45 cell towers), BUSINESS (86 cell towers) and MERGED (131 cell towers) datasets with the setting $B = 30000$, $E = 5$, $b_a = 50$ and $b_o = 1$. From Table 6.1, we can see that though CrowdTasker took longer time than other three methods, the total computation time of CrowdTasker on MERGED dataset was less than 70 minutes. As the task allocation process is done off-line and the sequential algorithm was run on a laptop, shorter computation time can be easily achieved by running on a more powerful computer or using parallel algorithms.

6.5 Discussion

In this section, we discuss issues that are not reported or addressed in this work due to space and time constraint, which could be added to or explored in our future work.

Using MaxUtils for $Utility_1$ calculation: In some cases MaxUtils performed as well as CrowdTasker. It is reasonable to think whether CrowdTasker could be further improved when using the Utility function of MaxUtils for $Utility_1$ calculation (i.e., using MaxUtils to initialize the iterative greedy process). Our experiment, however, found this solution did not obtain a better result because the coverage quality of the *second selected set* of user-cycle combinations did not improve i.e., $CQE(\mathbb{X}^2) \leq CQE(\mathbb{X}^1)$ when replacing $Utility_1$ with the Utility function of MaxUtils.

Call/Mobility Prediction and Privacy: The actual coverage quality achieved by CrowdTasker depends on the set of selected participants, each participant's selected cycles for PCS task participation, and the accuracy of call/mobility prediction. While the simple prediction techniques worked well in CrowdTasker for task allocation, we intend to further improve the coverage quality estimation and call/mobility prediction methods in future work. To obtain the call/mobility prediction models, currently CrowdTasker collects, stores and analyzes the raw call/mobility traces of mobile users. This, however, leads to privacy issues. One way to reduce the privacy threats is to only provide predictive models, rather than raw traces, to CrowdTasker, as supplied by the mobile operators. Or the CrowdTasker client software running on the user's device can capture raw data, but only upload predictive models for task allocation.

Coverage Quality Metrics and Incentive Models: Due to the limitation of the D4D dataset, we can only measure the sensing coverage at the cell tower level. If the user's mobility traces can be obtained continuously at a finer granularity [96],

CrowdTasker is still applicable as it is a general approach. Further in this work we adopt the Base/Bonus incentive model where each participant receives both Base–a fixed amount of incentive through the whole task period and Bonus–a fixed amount of incentive for MCS task participation in each sensing cycle. Each participant is only requested to sense and upload data for a PCS task in the assigned sensing cycles. In other MCS applications, different incentive models may be needed. For instance, a reputation-based incentive model may be more engaging for some PCS tasks that give each participant different incentives according to his/her trustworthiness [105]. As future work, we plan to study different task allocation strategies that are suitable for those incentive models.

Leveraging Multiple Piggyback Opportunities: In addition to piggyback sensing tasks over mobile phone calls, other piggyback methods also exist. For instance, executing sensing tasks in parallel with Google Map usage also reduces energy consumption when performing PCS tasks [18]. We plan to study the participant selection that leverages multiple piggyback sensing opportunities in a holistic manner as many predictive models, such as for app usage [97], already exist.

Chapter 7

Conclusion

Contents

7.1 Summary 119
7.1.1 Summary of *EEMC* 120
7.1.2 Summary of EMC^3 120
7.1.3 Summary of CrowdRecruiter 120
7.1.4 Summary of CrowdTasker 121
7.2 Future Work 121

7.1 Summary

In this Thesis, we studied the fundamental question

How can we design a mobile crowdsensing application, in order to collect high quality sensor data as energy-efficiently and cost-effectively as possible?

Most previous approaches that addressed this question have relied upon partial concerns of mobile crowdsensing design, e.g., considering only one or few designing issues among *energy consumption, incentive-based encourage, privacy, overall sensing data quality* and *total incentive payment*, without taking all these five factors into account.

In this Thesis, we presented four novel frameworks for mobile crowdsensing, considering all aforementioned issues, and with different optimization objectives/constraints (e.g., maximizing sensing data quality under budget constraint, minimizing overall energy consumption under sensing data quality constraint, and etc.), so as to meet the requirements of practical MCS applications. In order to reduce energy consumption of each participant, the frameworks are proposed to leverage various novel energy-saving strategies like parallel data transfer and piggybacked sensing task model. In order to select MCS participants and assign MCS task precisely, subject to different objectives and constraints, we design several participant selection/task allocation algorithms adopting the sequential decision making, and combinatorial optimization techniques for these frameworks.

In following sections, we will briefly summarize the key contributions presented in this Thesis.

7.1.1 Summary of *EEMC*

In Chapter 3, we have presented *EEMC*— a framework to enable energy-efficient mobile crowdsensing, where the goal is to reduce energy consumption in data transfer for both individual participants and the whole crowds while securing the sensed result collection from a minimum number of participants within a specific timeframe (namely a *sensing cycle*). The proposed framework embeds several mechanisms from existing work such as *parallel transfer* and *cycle-based delay-tolerant participatory sensing* into a novel *Two-call-based MCS data transfer scheme*, which is capable of reducing energy consumption in data transfer for individual device by 75% compared to the common 3G-based schemes. In order to reduce overall energy consumption for the whole crowds, we propose a two-step task assignment decision making algorithm to avoid redundant task assignments. Evaluations with a large-scale real-world dataset show that: the proposed algorithm constantly outperforms baseline approaches in terms of task assignment; and *EEMC* can reduce overall energy consumption in data transfer by 54%–66% when compared to the 3G-based schemes.

7.1.2 Summary of EMC^3

In Chapter 4, we have investigated the problem of reducing energy consumption of both individual user and all participants in data transfer caused by task assignment and data collection of MCS tasks, considering the user privacy issue, minimal number of task assignment requirement and sensing area coverage constraint. This problem is motivated by the needs of encouraging more mobile users to participate in urban-scale crowdsensing applications. To address the problem, we propose a novel MCS framework called EMC^3, leveraging a proposed delay-tolerant MCS setting, the parallel transfer technique, and a three-step process for task assignment. Evaluations with a large-scale real-world dataset show that our proposed EMC^3 framework outperforms the baseline approaches, and it can reduce 43%-68% overall energy consumption in data transfer compared to the 3G-based solution.

7.1.3 Summary of CrowdRecruiter

In Chapter 5, we proposed a novel participant selection framework, named *CrowdRecruiter*, for Piggyback Crowdsensing (PCS), which intends to minimize the total incentive payments by selecting a small number of participants while satisfying a predefined coverage constraint. The PCS was adopted to reduce energy consumption of individual mobile device, by exploiting call opportunities to perform sensing tasks and return sensed results. In order to select the minimal set of participants under probabilistic coverage constraint, CrowdRecruiter first predicts the call and coverage probability of each mobile user, then proposes a utility function to measure the joint coverage probability of multiple users, and finally deploys a low-complexity but effective algorithm to incrementally select the participants. Evaluations with a large-scale

real-world dataset show that our proposed CrowdRecruiter outperforms three baseline approaches, and on average it selects 10.0%–73.5% fewer participants compared to baseline approaches under the same probabilistic coverage constraint.

7.1.4 Summary of CrowdTasker

In Chapter 6, we proposed a novel task allocation framework, *CrowdTasker*, for Piggyback Crowdsensing (PCS). CrowdTasker is designed to maximize the overall coverage quality across all sensing cycles with a fixed budget by selecting a number of participants and determining in which sensing cycles each selected participant is needed for the PCS task participation. The PCS was adopted to reduce energy consumption of individual mobile device, by exploiting call opportunities to perform sensing tasks and upload sensed data. In order to allocate PCS task maximizing the coverage quality while satisfying the budget constraint, CrowdTasker first predicts the coverage probability of each mobile user, then performs a near-optimal participant/cycle task allocation search algorithm with low computational complexity. Theoretical analysis proves that CrowdTasker can achieve near-optimality, evaluations with a large-scale real-world dataset show that CrowdTasker outperformed three baseline approaches, and on average it achieved 3%–60% higher coverage quality compared to baseline approaches under the same budget constraint.

7.2 Future Work

The long-term goal of our research is pushing at the frontier of the techniques about mobile crowdsensing, especially in situations where a large group of participants are needed to distributedly collect sensor data in a large target region. With respect to this research goal, I plan to continue designing novel MCS frameworks and applications for urban environmental monitoring.

In my future work, I will try to answer following particular questions:

- How can we determine, which sensing data quality metrics (e.g., spatial-temporal coverage, number of samples, confidential level, and etc.) should be used in each practical MCS application, in order to delivery accurate sensed result to the end-users?

- How can we determine, which type of incentives (e.g., money) and how much incentives should be paid to each participant, in order to encourage their participation in both psychological and economical aspects?

- How can we make use of mobile users' historical digital footprints (e.g., mobility traces), in order to better understand each user's behavioral/mobility patterns but without scarifying users' privacy?

- How can we build mobile crowdsensing frameworks and applications, which optimally recruit participants and allocate sensing tasks, subject to the various sensor data collection goals/constraints, addressing the energy, incentive, sensing data quality and privacy issues? Is there any theoretical guarantee for the performance of data collection, even in the worst-case?

In order to answer these questions, we might need to solve quite a lot technical challenges, bring together geographical information processing, human factors of computing, privacy protecting, machine learning, optimal decision making and other sensor network areas. Hereby, we need to identify the next steps of future research and the directions along the way, some of which I outline in the following.

1. *Characterizing the target region using Sparse and Partial Observations* - Collecting sensor data fully covering the target region or covering the most part of the target region usually costs so much (e.g., total energy and incentive). Recent studies in compressive sensing and spatial correlation shows it is possible to recover the sensed results of the whole target region, through collecting a few sensed results that sparsely cover the target region. Exactly, we have already started studying a novel sparse sampling strategy [106] that intends to collect sensor data from a minimal number of subareas while inferring sensor data of the rest subareas with high accuracy.

2. *Making trade-off among Incentives, Privacy and Energy consumption* - Recent studies in incentive pricing mechanism [56, 57] show that there might exists an equilibrium price satisfying both the MCS organizer and each MCS participant, according to the cost (energy consumption, mobile phone usage, risky of privacy leakage, and etc.) of each participant obtaining a sensed result and the economical value of the sensed result. In the future research, we plan to study the incentive payment mechanisms making trade-off among incentives, privacy and energy consumption, considering the both psychological and economical aspects.

3. *Online human behavior/mobility learning using partial and incremental traces* - In this thesis, we use users' historical call/mobility traces to learn human mobility patterns and further allocate sensing tasks according to the patterns. In the practical MCS applications, however, there might not be able to collect users' complete historical traces for a long time. Thus, there needs a method to get user's real-time mobility and behavioral data, further aggregate the data newly arrived with the traces already collected, in order to obtain the incremental traces. Further the method should be able to learn users' behavioral/mobility patterns through mining the incremental mobility/behavioral traces.

4. *Optimal participant selection and task allocation subject to complex MCS data collection objectives/constraints* - In this thesis, we study several optimization

algorithms for optimal participant selection and task allocation, subject to some specific MCS sensing data quality and incentive objectives/constraints. Our future work plans to study a general optimization framework that is able to handle more complex objectives/constraints.

I believe that these directions might pose great potentials for academic research as well as for building MCS systems and applications that will have great real-world influence with significant benefits to our society.

Appendix A

Low-complexity Algorithms and Proofs

Contents

A.1 Algorithms and Proofs from Chapter 3 and 4 125
A.1.1 Low-complexity Algorithms for $P_{fulfill}$ Computation 125
A.1.2 Low-complexity Algorithm for $P^*_{fulfill}$ 126
A.2 Algorithms and Proofs from Chapter 5 126
A.2.1 Low-complexity Algorithms for $COV_i(\mathbb{S})$ Computation 126
A.2.2 *Proof–Utility*($\mathbb{S}$) is an submodular function 127
A.3 Algorithms and Proofs from Chapter 6 128
A.3.1 Low-complexity Algorithms for $CQE(\mathbb{X})$ Computation 128
A.3.2 *Proof–*$CQE(\mathbb{X})$ is an submodular function 128
A.3.3 *Proof*–The total Base/Bonus incentive payment is an submodular function over $\mathbb{X}$ 129

A.1 Algorithms and Proofs from Chapter 3 and 4

A.1.1 Low-complexity Algorithms for $P_{fulfill}$ Computation

As the computation complexity of enumerating all subsets from a *n-length* set is $O(2^n)$, it is very time-consuming to solve Equation 3.2 through a subset-enumeration algorithm. For example, there are $2^{50} = 1.126 \times e^{15}$ subsets in a set with 50 elements. To reduce the computation complexity of $P_{fulfill}$ in Equation 3.2, we proposes an algorithm with $O(n^2)$ complexity. According to the Probability Generating Function Theory [103], $P\{X_{k,t}(A_k - R_k) = N\}$ is **equivalent** to the coefficient of z^N in the following polynomial over z:

$$\prod_{U_m \in A_k - R_k} (z * P_{k,t}\{x_m \geq 1\} + (1 - P_{k,t}\{x_m \geq 1\})) \tag{A.1}$$

Finally, we can resolve the above polynomials and calculate all necessary coefficients by using algorithm 5.

Algorithm 5: Computing Coefficients

Input : A_k, R_k, and $P_{k,t}\{x_m = n\}$
Output: coeffs– the array of coefficients

```
1  begin
       /* initiate the coefficients of polynomial.                    */
2      coeffs← NEW_ARRAY_OF_SIZE(1);
3      coeffs[0]← 1;
       /* Cumulative Product of Binomials                             */
4      for 0 ≤ m < |A_k − R_k| do
5          new_length←LENGTH_OF(coeffs)+1;
6          new_coeffs←NEW_ARRAY_OF_SIZE(new_length);
7          for 0 ≤ i <LENGTH_OF(coeffs) do
8              new_coeffs[i] += coeffs[i] * (1-P_{k,t}{x_m ≥ 1});
9              new_coeffs[i+1] += coeffs[i] * P_{k,t}{x_m ≥ 1};
10         end
11         coeffs←new_coeffs;
12     end
13     return coeffs;
14 end
```

A.1.2 Low-complexity Algorithm for $P^*_{fulfill}$

Similar to Equation 3.2, $P\{X^*_{k,t}(FS_{U_i} \cup (A_k - R_k)) = N\}$ is **equivalent** to the coefficient of z^N in polynomial:

$$\prod_{U_m \in (A_k - R_k) \cup FS_{U_i}} (z * P'_{k,t}(U_m) + 1 - P'_{k,t}(U_m)) \tag{A.2}$$

Obviously, all coefficients in Equation A.2 can be resolved by an algorithm similar to Algorithm 5 under $O(n^2)$ complexity.

A.2 Algorithms and Proofs from Chapter 5

A.2.1 Low-complexity Algorithms for $COV_i(\mathbb{S})$ Computation

The overall computation complexity of this approach should be $O(\sum_{k=\tau}^{|T|} \frac{|T|!}{(|T|-k)!*k!} \times k) = O(\sum_{k=\tau}^{|T|} \frac{|T|!}{(|T|-k)!*(k-1)!})$, where $\sum_{k=\tau}^{|T|} \frac{|T|!}{(|T|-k)!*k!}$ is the number of cell tower combinations ($k \in [\tau, |T|]$ refers to the size of each combination), and the complexity of probability computation for a k-size combination is $O(k)$. However the overall computation complexity is unacceptable, since the number of cell tower combinations grows combinatorially when the size of T increases. For example, given an overall set

of 120 cell towers, there are $\frac{120!}{(120-100)!*100!} \approx 1.1 \times 10^{+10}$ cell tower combinations, each of which consists of 100 cell towers.

In order to simplify the calculation of Eq. 5.5, we propose a fast algorithm based on Probability Generating Function Theory [103]. Specifically, we compute $COV_i(\mathbb{S} \cup \{u\})$ as:

$$COV_i(\mathbb{S} \cup \{u\}) \equiv \sum_{k=\tau}^{|T|} \text{coeff}_i(k, \mathbb{S} \cup \{u\}), \tag{A.3}$$

where $\text{coeff}_i(k, \mathbb{S} \cup \{u\})$ denotes the coefficient of z^k in the following polynomial over z:

$$\prod_{t \in T} (z * \mathbb{Q}_{i,t}(\mathbb{S} \cup \{u\}) + (1 - \mathbb{Q}_{i,t}(\mathbb{S} \cup \{u\}))) \tag{A.4}$$

Note that using a classic polynomial production algorithm [107], we can resolve the polynomial in Eq. A.5 and calculate all necessary coefficients with $O(|T|^2)$ complexity.

A.2.2 *Proof*–$Utility(\mathbb{S})$ is an submodular function

First, we prove $Utility(S)$ is an *non-negative/non-decreasing function* over S and a simple proof is as follows.

Proof - Since $\forall S \subseteq \mathbb{U}$ and $\forall u \in S$ there exists $0 \leq P_{i,t}(u) \leq 1$, we can conclude $\mathbb{Q}_{i,t}(S) \geq 0$. Further $\forall u' \in \mathbb{U} \backslash S$ there exists $\mathbb{Q}_{i,t}(S \cup \{u'\}) = 1 - (1 - \mathbb{Q}_{i,t}(S)) * (1 - P_{i,t}(u')) \geq \mathbb{Q}_{i,t}(S)$, we can conclude $\mathbb{Q}_{i,t}(S)$ a non-decreasing set function. Finally, as $Utility(S)$ is the sum of $\mathbb{Q}_{i,t}(S)$, the utility function is a non-negative/non-decreasing function. □

Second, we prove $Utility(S)$ is a *sumbodular set function* and a simple proof is as follows.

Proof - $\forall S \subseteq \mathbb{U}$ and $\forall u', u'' \in \mathbb{U} \backslash S$, there exists

$$\begin{aligned}
&\mathbb{Q}_{i,t}(S \cup \{u', u''\}) - \mathbb{Q}_{i,t}(S \cup \{u'\}) \\
&= \prod_{u \in S} (1 - P_{i,t}(u)) * (1 - P_{i,t}(u')) * (1 - P_{i,t}(u'')) \\
&\leq \prod_{u \in S} (1 - P_{i,t}(u)) * (1 - P_{i,t}(u')) \\
&= \mathbb{Q}_{i,t}(S \cup \{u'\}) - \mathbb{Q}_{i,t}(S)
\end{aligned}$$

$\mathbb{Q}_{i,t}(S)$ thus is a submodular set function, according to the definition of submodularity [95]. Further, since $Utility(S)$ is the sum of $\mathbb{Q}_{i,t}(S)$, $Utility(S)$ is an submodular set function as well. □

A.3 Algorithms and Proofs from Chapter 6

A.3.1 Low-complexity Algorithms for $CQE(\mathbb{X})$ Computation

The straightforward solution to Eq. 6.5 is to first, enumerate all possible user combinations from all selected participants in $\mathbb{S}$, to compute the probability of each user combination returning sensed result in each sensing cycle and each cell tower (e.g, $P_{U,i,t}$ for the user combination U in cell tower t and cycle i) and further compute the expected coverage quality of this combination (e.g., $min\{|U|, E\} * P_{U,i,t}$), and then to sum the expected coverage quality of each user combination in each cell tower and each sensing cycle as the result. The overall computation complexity of this approach should be $O((2^{|\mathbb{S}|} - 1) * |\mathbb{S}| * |T| * I)$, where $2^{|\mathbb{S}|} - 1$ is the number of user combinations and $|\mathbb{S}|$ refers to the complexity of probability computation. However the overall computation complexity is unacceptable, since the number of user combinations grows exponentially when the size of $\mathbb{S}$ increases. For example, given an overall set of 100 selected users, there are $2^{100} - 1 = 1.27 * e^{+30}$ user combinations. In order to simplify the calculation of Eq. 6.5, we propose a fast algorithm based on Probability Generating Function Theory [103]. Specifically, we compute $CQE(\mathbb{X})$ as:

$$CQE(\mathbb{X}) \equiv \sum_{0 \le i < I} \sum_{\tau \in T} \sum_{0 \le l < |\mathbb{S}|} min\{|l|, E\} * \text{coeff}(i, t, l, \mathbb{S})$$

where $\forall \langle u, i \rangle \in \mathbb{X}$ there exists $\exists u \in \mathbb{S}$ and $\exists i \in C_u$, and $\text{coeff}(i, t, l, \mathbb{S})$ denotes the coefficient of z^l in the following polynomial over z:

$$\prod_{u \in \mathbb{S}} (z * P_{i,u}(t) * \mathbb{A}(C_u, i)) + (1 - P_{i,u}(t) * \mathbb{A}(C_u, i))) \tag{A.5}$$

Note that using a classic polynomial production algorithm [107], we can resolve the polynomial in Eq. A.5 and calculate all necessary coefficients with $O(|\mathbb{S}|^2)$ complexity. Thus the overall computational complexity of this algorithm is $O(|T| * I * |\mathbb{S}|^2)$.

A.3.2 *Proof*–$CQE(\mathbb{X})$ is an submodular function

Proof I - $CQE(\mathbb{X})$ is an submodular function: For each cell tower t and cycle i, given a set of users U_i assigned MCS task in the cycle i, and the function $g_{i,t}(U_i) = \sum_{U \in U_i} min\{|U|, E\} * \prod_{\forall u \in U} P_{i,u}(t) * \prod_{\forall v \in U_i, v \notin U} (1 - P_{i,v}(t))$ estimating the coverage quality achieved by users U_i in the cycle i and cell tower t, we can simply prove that $g_{i,t}(U \cup \{u, v\}) - g_{i,t}(U \cup \{u\}) \le g_{i,t}(U \cup \{u\}) - g_{i,t}(U)$ where u and v are two users assigned with cycle i; thus we say $g_{i,t}(U)$ is a submodular function over the set of users assigned with cycle i. Further $CQE(\mathbb{X})$ is the linear sum of $g_{i,t}(U)$ over each sensing cycle i and each cell tower t. Thus, we can conclude that $CQE(\mathbb{X})$ is ansubmodular function. □

A.3.3 *Proof*–The total Base/Bonus incentive payment is an submodular function over $\mathbb{X}$

Proof II - $C(\mathbb{X})$ is an submodular function: Given any user-pair-set $\mathbb{X}$, and two user-cycle-pairs $\langle u,i\rangle, \langle v,j\rangle$ ($\langle u,i\rangle \neq \langle v,j\rangle$, $\langle u,i\rangle \notin \mathbb{X}$, $\langle v,j\rangle \notin \mathbb{X}$), we prove $C(\mathbb{X})$ as a submodular function as follows. If there exists another user-cycle-pair of user u in $\mathbb{X}$ then $C(\mathbb{X}\cup\{\langle u,i\rangle,\langle v,j\rangle\})-C(\mathbb{X}\cup\langle v,j\rangle) = C(\mathbb{X}\cup\langle u,i\rangle)-C(\mathbb{X}) = b_o$; else if u is a new user in $\mathbb{X}$ and $u \neq v$ then $C(\mathbb{X}\cup\{\langle u,i\rangle,\langle v,j\rangle\})-C(\mathbb{X}\cup\langle v,j\rangle) = C(\mathbb{X}\cup\langle u,i\rangle)-C(\mathbb{X}) = b_o+b_a$; else if u is a new user in $\mathbb{X}$ and $u = v$ then $C(\mathbb{X}\cup\{\langle u,i\rangle,\langle v,j\rangle\})-C(\mathbb{X}\cup\langle v,j\rangle) = b_o < C(\mathbb{X}\cup\langle u,i\rangle) - C(\mathbb{X}) = b_o + b_a$. Thus, $C(\mathbb{X}\cup\{\langle u,i\rangle,\langle v,j\rangle\}) - C(\mathbb{X}\cup\langle v,j\rangle) \leq C(\mathbb{X}\cup\langle u,i\rangle) - C(\mathbb{X})$ and we can conclude $C(\mathbb{X})$ is a submodular set function. □

Appendix B

Curriculum Vitae and Research Publications

B.1 Curriculum Vitae

Haoyi Xiong was born in Wuhan, China. From Sept 2011 to present, he is a PhD student supervised by Prof. Monique Becker, Prof. Daqing Zhang and Dr. Vincent Gauthier at Institut Mines-Télécom/TELECOM SudParis and Université Pierre et Marie Curie (Paris VI) . He received his M.Sc in Information Technology from the Hong Kong University of Science and Technology in 2010, and B.Eng in Electrical Engineering and Automation from Huazhong University of Science and Technology in 2009. His research interests include mobile crowdsensing, participatory sensing, and human mobility data mining. He has served as a TPC-member and an external reviewer for IEEE I-SPAN'14, IEEE WCNC-IOT'14, IEEE UIC '13–14, and IEEE CPSCom'13, and as a reviewer for Journals including ACM TIST, IEEE ComMag, and Springer PUC.

B.2 Research Publications

B.2.1 Published or Accepted Papers

1. **Haoyi Xiong**, Daqing Zhang, Guanling Chen, Leye Wang, and Vincent Gauthier, *CrowdTasker*: Maximizing Coverage Quality under Budget Constraint. *IEEE International Conference on Pervasive Computing and Communications (PerCom'15), to appear.*

2. **Haoyi Xiong**, Daqing Zhang, Leye Wang and Hakima Chaouchi, EMC^3: Energy-efficient Data Transfer in Mobile Crowdsensing under Full Coverage Constraint, 2014. *IEEE Transactions on Mobile Computing (TMC), Preprint Online.*

3. **Haoyi Xiong**, Daqing Zhang, Leye Wang, Paul Gibson and Jie Zhu, EEMC: Enabling Energy-Efficient Mobile Crowdsensing with Anonymous Participants, 2014. *ACM Transactions on Intelligent System and Technology (TIST), to appear.*

4. Daqing Zhang, **Haoyi Xiong**, Leye Wang and Guanling Chen, CrowdRecruiter: Selecting Participants for Piggyback Crowdsensing under Probabilistic Coverage Constraint, 2014. *ACM International Joint Conference on Pervasive and Ubiquitous Computing (Ubicomp'14), Seattle, WA.*

5. **Haoyi Xiong**, Daqing Zhang, Daqiang Zhang, Vincent Gauthier, Kun Yang and Monique Becker, MPaaS: Mobility Prediction as a Service in Telecom Cloud, *Information Systems Frontiers, Vol 16, pp 59–75, 2014, Springer.*

6. **Haoyi Xiong,** Leye Wang and Daqing Zhang, EEMC: An Energy-Ef?cient Mobile Crowdsensing Mechanism by Reusing Call/SMS Connections, *D4D Data Challenge, The Third International Conference on the Analysis of Mobile Phone Datasets (NetMob'13), Massachusetts, USA, 2013.*

7. **Haoyi Xiong**, Daqing Zhang, Daqiang Zhang and Vincent Gauthier, Predicting Mobile Phone User Locations by Exploiting Collective Behavioral Patterns, *The 9th IEEE Conference on Ubiquitous Intelligence and Computing (UIC'12), Fukuoka, Japan, 2012. (**Best Paper Award**).*

8. Daqing Zhang, Leye Wang, **Haoyi Xiong** and Bin Guo, 4W1H in Mobile Crowd Sensing. *IEEE Communications Magazine, 2014, IEEE.*

9. Daqiang Zhang, Daqing Zhang, **Haoyi Xiong**, Laurence T. Yang and Vincent Gauthier, NextCell: Predicting Location Using Social Interplay from Cell Phone Traces, *IEEE Transactions on Computers, preprint, IEEE.*

10. Daqiang Zhang, Daqing Zhang, **Haoyi Xiong**, Ching-Hsien Hsu, Athanasios Vasilakos, BASA: Building Mobile Ad-Hoc Social Networks on Top of Android, *IEEE Network Magazine, Vol.28, pp 4–9, 2014, IEEE.*

11. Daqiang Zhang, Min Chen, Mohsen Guizani, **Haoyi Xiong**, and Daqing Zhang, Mobility Prediction in Telecom Cloud Using Mobile Calls, *IEEE Wireless Communication Magazine, Vol 21, pp 26–32, 2014, IEEE*

12. Leye Wang, Daqing Zhang and **Haoyi Xiong**, effSense: Energy-Efficient and Cost-Effective Data Uploading in Mobile Crowdsensing, *Workshop on Pervasive Urban Crowdsensing Architecture and Applications (PUCAA'13) with Ubicomp'13.*

B.2.2 Under Reviewing/Revision

1. Leye Wang, Daqing Zhang, Animesh Pathak, Chao Chen and **Haoyi Xiong**, CCS-TA: Toward Online Task Allocation in Mobile Compressive Crowdsensing, *Submitted.*

2. Leye Wang, Daqing Zhang, Zhixian Yan, **Haoyi Xiong** and Bin Xie, effSense: A Novel Mobile Crowdsensing Framework for Energy-Efficient and Cost-Effective Data Uploading, *Submitted*.

3. Leye Wang, Daqing Zhang, **Haoyi Xiong** and J. Paul Gibson, ecoSense: Minimize Participants' Total 3G Data Cost in Mobile Crowdsensing Using Opportunistic Relays, *Submitted*.

Bibliography

[1] R.K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine*, 49:32–39, 2011.

[2] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 2009 ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[3] K.K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P.J. Rentfrow. Metis: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications. In *Proceedings of the 2013 IEEE Conference on Pervasive Computing and Communications*, volume 18, page 22, 2013.

[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 *Proceedings of the 2010 ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 105–116. ACM, 2010.

[5] P. Dutta, P.M. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff. Common sense: participatory urban sensing using a network of handheld air quality monitors. In *Proceedings of the 2009 ACM conference on embedded networked sensor systems*, pages 349–350. ACM, 2009.

[6] Bin Guo, Zhiwen Yu, Daqing Zhang, and Xingshe Zhou. From participatory sensing to mobile crowd sensing. In *Proceedings of the 2014 IEEE Pervasive Computing and Communication Workshops, to appear*. IEEE, 2014.

[7] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *Proceedings of the 2012 International Symposium on Wearable Computers*, pages 17–24. IEEE, 2012.

[8] B. Priyantha, D. Lymberopoulos, and J. Liu. Littlerock: Enabling energy-efficient continuous sensing on mobile phones. *IEEE Pervasive Computing*, 10(2):12–15, 2011.

[9] Jingtao Wang, Shumin Zhai, and John Canny. Camera phone based motion sensing: interaction techniques, applications and performance study. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 101–110. ACM, 2006.

[10] E.C. Larson, T.J Lee, S. Liu, M. Rosenfeld, and S.N. Patel. Accurate and privacy preserving cough sensing using a low-cost microphone. In *Proceedings*

of the 2011 ACM International Conference on Ubiquitous Computing, pages 375–384. ACM, 2011.

[11] Arsham Farshad, Mahesh K Marina, and Francisco Garcia. Urban wifi characterization via mobile crowdsensing. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–9. IEEE, 2014.

[12] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[13] H.S. Ramos, T. Zhang, J. Liu, N.B. Priyantha, and A. Kansal. Leap: a low energy assisted gps for trajectory-based services. *Proceedings of the 2011 ACM International Conference on Ubiquitous Computing*, pages 335–344, 2011.

[14] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[15] V.D. Blondel, M. Esch, C. Chan, F. Clerot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki. Data for development: the d4d challenge on mobile phone data. *CoRR*, abs/1210.0137, 2012.

[16] L. Vu, Q. Do, and K. Nahrstedt. Jyotish: A novel framework for constructing predictive model of people movement from joint wifi/bluetooth trace. In *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications*, pages 54–62. IEEE, 2011.

[17] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 2009 ACM SIGCOMM conference on Internet Measurement Conference*, pages 280–293. ACM, 2009.

[18] Nicholas D Lane, Yohan Chon, Lin Zhou, Yongzhe Zhang, Fan Li, Dongwon Kim, Guanzhong Ding, Feng Zhao, and Hojung Cha. Piggyback crowdsensing (pcs): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 7. ACM, 2013.

[19] H. Xiong, L. Wang, and D. Zhang. Eemc: An energy-efficient mobile crowdsensing mechanism by reusing call/sms connections. In *Proceedings of the 2013 Conference on the Analysis of Mobile Phone Datasets*, pages 323–329. MIT, 2013.

[20] J.K. Nurminen and J. Nöyränen. Parallel data transfer with voice calls for energy-efficient mobile services. *Proceedings of the 2009 Internatioan Conference on Mobile Wireless Middleware, Operating Systems, and Applications*, pages 87–100, 2009.

[21] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.

[22] Daqing Zhang, Leye Wang, Haoyi Xiong, and Bin Guo. 4w1h in mobile crowd sensing. *Communications Magazine, IEEE*, 52(8):42–48, 2014.

[23] Andreas Krause, Eric Horvitz, Aman Kansal, and Feng Zhao. Toward community sensing. In *Proc. ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[24] N.D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A.T. Campbell, and F. Zhao. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 2011 ACM International Conference on Ubiquitous Computing*, pages 355–364. ACM, 2011.

[25] Y. Chon, N.D. Lane, F. Li, H. Cha, and F. Zhao. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proceedings of the 2012 International Conference on Ubiquitous Computing, ACM*, 2012.

[26] S. Isaacman, R. Becker, R. Cáceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky. Identifying important places in people?s lives from cellular network data. In *Proceedings of the 2011 International Conference on Pervasive Computing*, pages 133–151. Springer, 2011.

[27] M. Ficek, N. Clark, and L. Kencl. Can crowdsensing beat dynamic cell-id? In *Proceedings of the 2012 International Workshop on Sensing Applications on Mobile Phones*, page 10. ACM, 2012.

[28] 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 *Proceedings of the 2011 Annual International Conference on Mobile Computing and Networking*, pages 73–84. ACM, 2011.

[29] J. Zheng and L.M. Ni. An unsupervised framework for sensing individual and cluster behavior patterns from human mobile data. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 153–162. ACM, 2012.

[30] H. Weinschrott, J. Weisser, F. Durr, and K. Rothermel. Participatory sensing algorithms for mobile object discovery in urban areas. In *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications*, pages 128–135. IEEE, 2011.

[31] 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 2010 International Conference on Mobile systems, Applications, and Services, pages 123–136. ACM, 2010.

[32] Y. Jiang, D. Li, G. Yang, Q. Lv, and Z. Liu. Deliberation for intuition: a framework for energy-efficient trip detection on cellular phones. In *Proceedings of the 2011 ACM International Conference on Ubiquitous Computing*, pages 315–324. ACM, 2011.

[33] S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. In *Proceedings of Pervasive*, pages 138–155. 2010.

[34] P.P. Jayaraman, A. Sinha, W. Sherchan, S. Krishnaswamy, A. Zaslavsky, P.D. Haghighi, S. Loke, and M.T. Do. Here-n-now: A framework for context-aware mobile crowdsensing. In *Proceedings of the 2012 International Conference on Pervasive Computing*, 2012.

[35] Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan. Lowering the barriers to large-scale mobile crowdsensing. In *Proceedings of the 2013 International Workshops on Mobile Computing Systems and Applications*, 2013.

[36] W. Sherchan, P.P. Jayaraman, S. Krishnaswamy, A. Zaslavsky, S. Loke, and A. Sinha. Using on-the-move mining for mobile crowdsensing. In *Proceedings of the 2012 IEEE Conference on Mobile Data Management*, pages 115–124. IEEE, 2012.

[37] M-R. Ra, B. Liu, T.F. La Porta, and R. Govindan. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 2012 International Conference on Mobile Systems, Applications, and Services*, pages 337–350. ACM, 2012.

[38] N. Priyantha, D. Lymberopoulos, and J. Liu. Eers: Energy efficient responsive sleeping on mobile phones. In *Proceedings of the 2010 International Workshop on Sensing for App Phones*, 2010.

[39] Y. Wang, J. Lin, M. Annavaram, Q.A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of The 2009 International Conference on Mobile Systems, Applications, and Services*, pages 179–192. ACM, 2009.

[40] G.P. Perrucci, F.H.P. Fitzek, and J. Widmer. Survey on energy consumption entities on the smartphone platform. In *Proceedings of the 2011 IEEE Vehicular Technology Conference (VTC Spring)*, pages 1–6. IEEE, 2011.

[41] A. Rice and S. Hay. Decomposing power measurements for mobile devices. In *Proceedings of the 2010 IEEE Conference on Pervasive Computing and Communications*, pages 70–78. IEEE, 2010.

[42] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J.P. Singh. Who killed my battery?: analyzing mobile browser energy consumption. In *Proceedings of the 2012 international conference on World Wide Web*, pages 41–50. ACM, 2012.

[43] 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 *Proceedings of the 2012 ACM international conference on Ubiquitous computing*, 2012.

[44] M.B. Kjærgaard, S. Bhattacharya, H. Blunck, and P. Nurmi. Energy-efficient trajectory tracking for mobile devices. In *Proceedings of the 2011 ACM International Conference on Mobile systems, Applications, and Services*, pages 307–320. ACM, 2011.

[45] D. Gordon, J. Czerny, T. Miyaki, and M. Beigl. Energy-efficient activity recognition using prediction. In *Proceedings of the 2012 International Symposium on Wearable Computers*, pages 29–36. IEEE, 2012.

[46] M-R. Ra, B. Priyantha, A. Kansal, and J. Liu. Improving energy efficiency of personal sensing applications with heterogeneous multi-processors. In *Proceedings of the 2012 ACM International Conference on Ubiquitous Computing*. ACM, 2012.

[47] D. Chu, N.D. Lane, T.T Lai, C. Pang, X. Meng, Q. Guo, F. Li, and F. Zhao. Balancing energy, latency and accuracy for mobile sensor data classification. In *Proceedings of the 2011 ACM International Conference on Embedded Networked Sensor Systems*, pages 54–67. ACM, 2011.

[48] K. Frank. *Adaptive and Tractable Bayesian Context Inference for Resource Constrained Devices*. PhD thesis, Waterford Institute of Technology, 2011.

[49] D. Puccinelli, S. Giordano, M. Zuniga, and P.J. Marrón. Broadcast-free collection protocol. In *Proceedings of the 2011 ACM International Conference on Embedded Networked Sensor Systems*, pages 29–42. ACM, 2012.

[50] J. Brown, J. Finney, C. Efstratiou, B. Green, N. Davies, M. Lowton, and G. Kortuem. Network interrupts: supporting delay sensitive applications in low power wireless control networks. In *Proceedings of the 2007 ACM workshop on Challenged networks*, pages 51–58. ACM, 2007.

[51] J.K. Nurminen. Parallel connections and their effect on the battery consumption of a mobile phone. In *Proceedings of the 2010 IEEE Consumer Communications and Networking Conference*, pages 1–5. IEEE, 2010.

[52] B. Pásztor, M. Musolesi, and C. Mascolo. Opportunistic mobile sensor data collection with scar. In *Proceedings of the 2007 IEEE Conference on Mobile Ad-hoc and Sensor Systems*, pages 1–12. IEEE, 2007.

[53] E. Soroush, K. Wu, and J. Pei. Fast and quality-guaranteed data streaming in resource-constrained sensor networks. In *Proceedings of the 2008 ACM International Symposium on Mobile Ad-hoc Networking and Computing*, pages 391–400. ACM, 2008.

[54] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A.T Campbell. Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones. *Pervasive Computing*, pages 355–372, 2010.

[55] J. Smith, A. Sample, P. Powledge, S. Roy, and A. Mamishev. A wirelessly-powered platform for sensing and computation. *Proceedings of the 2006 International Conference on Ubiquitous Computing*, pages 495–506, 2006.

[56] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 173–184. ACM, 2012.

[57] Boi Faltings, Jason Jingshi Li, and Radu Jurca. Incentive mechanisms for community sensing. *IEEE Transactions on Computers*, page 1, 2013.

[58] Tie Luo and Chen-Khong Tham. Fairness and social welfare in incentivizing participatory sensing. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, pages 425–433. IEEE, 2012.

[59] Andreas Albers, Ioannis Krontiris, Noboru Sonehara, and Isao Echizen. Coupons as monetary incentives in participatory sensing. In *Collaborative, Trusted and Privacy-Aware e/m-Services*, pages 226–237. Springer, 2013.

[60] 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*, to appear.

[61] X. sheng, J. Tang, and W. Zhang. Energy-efficient collaborative sensing with mobile phones. In *Proceedings of the 2012 IEEE Conference on Computer Communications*. IEEE, 2012.

[62] 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 the 2011 IEEE Conference on Sensing, Communication and Networking*, pages 134–142. IEEE, 2011.

[63] Daqing Zhang, Haoyi Xiong, Leye Wang, and Guanling Chen. Crowdrecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 703–714. ACM, 2014.

[64] Y Chon, N.D Lane, Y Kim, F Zhao, and H Cha. Understanding the coverage and scalability of place-centric crowdsensing. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 3–12. ACM, 2013.

[65] D Zhao, H Ma, and L Liu. Energy-efficient opportunistic coverage for people-centric urban sensing. *Wireless Networks*, pages 1–16.

[66] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, volume 7, pages 1650–1654, 2007.

[67] Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9:2761–2801, 2008.

[68] D. Philipp, J. Stachowiak, P. Alt, F. Dürr, and K. Rothermel. Drops: Model-driven optimization for public sensing systems. In *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications*, volume 18, page 22, 2013.

[69] H. Weinschrott, F. Durr, and K. Rothermel. Streamshaper: Coordination algorithms for participatory mobile urban sensing. In *Proceedings of MASS*, pages 195–204. IEEE, 2010.

[70] 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 *Proceedings of Pervasive*, pages 355–372. January 2010.

[71] X. Sheng, J. Tang, and W. Zhang. Energy-efficient collaborative sensing with mobile phones. In *Proceedings of INFOCOM*, 2012.

[72] S Reddy, K Shilton, J Burke, D Estrin, M Hansen, and M Srivastava. Using context annotated mobility profiles to recruit data collectors in participatory sensing. In *Location and Context Awareness*, pages 52–69. Springer, 2009.

[73] A Singla and A Krause. Incentives for privacy tradeoff in community sensing. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[74] G Cardone, L Foschini, P Bellavista, A Corradi, C Borcea, M Talasila, and R Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. *Communications Magazine, IEEE*, 51(6), 2013.

[75] S. Hachem, A. Pathak, and V. Issarny. Probabilistic registration for large-scale mobile participatory sensing. In *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications*, volume 18, page 22, 2013.

[76] F. Lassabe, P. Canalda, P. Chatonnay, F. Spies, N.M.D. Center, and D. Charlet. Predictive mobility models based on kth markov models. In *IEEE Int. Conf. on pervasive services*, pages 303–306, 2006.

[77] L. Song, D. Kotz, R. Jain, and X. He. Evaluating location predictors with extensive wi-fi mobility data. In *in Proc. of INFOCOM 2004.*, pages 1414–1424. IEEE, 2004.

[78] C. Song, Z. Qu, N. Blumm, and A-L. Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[79] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *in Proc. of the 17th ACM Conf. on KDD*, pages 1082–1090, San Diego, CA, USA, 2011.

[80] M. Musolesi and C. Mascolo. A community based mobility model for ad hoc network research. In *in Proc. of the 2nd Intl. Workshop on Multi-hop Ad Hoc Networks: From Theory To Reality*, pages 31–38, Florence, Italy, 2006. ACM.

[81] F. Calabrese, G. Di Lorenzo, and C. Ratti. Human mobility prediction based on individual and collective geographical preferences. In *in Proc. IEEE Intl. Conf. on Intelligent Transportation Systems, Madeira Island, Portugal*, 2010.

[82] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *Proceedings of the 2012 ACM Conference on Embedded Networked Sensor Systems*, pages 1–14. ACM, 2012.

[83] D. Puccinelli and S. Giordano. Connectivity and energy usage in low-power wireless: An experimental study. In *Proceedings of PerCom Workshops*, pages 590–595. IEEE, 2013.

[84] D. Akimura, Y. Kawahara, and T. Asami. Compressed sensing method for human activity sensing using mobile phone accelerometers. In *Proceedings of the 2012 Networked Sensing Systems Conference*, pages 1–4. IEEE, 2012.

[85] R. Pease. What is killing smartphones? bbc - future - technology, 2013.

[86] L. Wang, D. Zhang, and H. Xiong. effsense: energy-efficient and cost-effective data uploading in mobile crowdsensing. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 1075–1086. ACM, 2013.

[87] 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*, 102(480):1185–1198, 2007.

[88] Y-B. Lin. Reducing location update cost in a pcs network. *IEEE/ACM Transactions on Networking (TON)*, 5(1):25–33, 1997.

[89] C. Gourieroux, A. Monfort, and A. Trognon. Pseudo maximum likelihood methods: applications to poisson models. *Econometrica: Journal of the Econometric Society*, pages 701–720, 1984.

[90] Tong Liu, Yu Zheng, Lubin Liu, Yanchi Liu, and Yanmin Zhu. Methods for sensing urban noises. Technical Report MSR-TR-2014-66, May 2014.

[91] Infoasaid. Telecommunications overview of cote d'ivore, 2013.

[92] J.K. Nurminen. Parallel connections and their effect on the battery consumption of a mobile phone. In *Proceedings of CCNC*, pages 1–5, January 2010.

[93] Leonard Kleinrock. *Queueing systems: volume 2: computer applications*, volume 82. John Wiley & Sons New York, 1976.

[94] Christos H. Papadimitriou and Yaron Singer. Budget feasible mechanisms. *CoRR*, abs/1002.2334, 2010.

[95] P R Goundan and A S Schulz. Revisiting the greedy approach to submodular set function maximization. *Optimization online*, pages 1–25, 2007.

[96] D T Wagner, A Rice, and A R Beresford. Device analyzer: Large-scale mobile data collection. In *Workshop on Big Data Analytics*, 2013.

[97] C Zhang, X Ding, G Chen, K Huang, X Ma, and B Yan. Nihao: A predictive smartphone application launcher. In *The 4th International Conference on Mobile Computing, Applications, and Services (MobiCase)*, pages 294–313. Springer, 2012.

[98] R Lange, F Durr, and K Rothermel. Efficient tracking of moving objects using generic remote trajectory simplification. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 829–831. IEEE, 2010.

[99] F Murena. Measuring air quality over large urban areas: development and application of an air pollution index at the urban area of naples. *Atmospheric Environment*, 38(36):6195–6202, 2004.

[100] O Shehory and S Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1):165–200, 1998.

[101] B P Gerkey and M J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[102] R Iyer and J A Bilmes. Submodular-bregman and the lova _sz-bregman divergences with applications. In *Advances in Neural Information Processing Systems*, pages 2942–2950, 2012.

[103] Crispin W Gardiner et al. *Handbook of stochastic methods*, volume 3. Springer Berlin, 1985.

[104] R K Iyer and J A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444, 2013.

[105] Y Zhang and M v.d.S. Reputation-based incentive protocols in crowdsourcing applications. In *INFOCOM, 2012 Proceedings IEEE*, pages 2140–2148. IEEE, 2012.

[106] Leye Wang, Daqing Zhang, Animesh Pathak, Chao Chen, and Haoyi Xiong. Ccs-ta: Toward online task allocation in mobile compressive crowdsensing. Technical report, September 2014.

[107] D J Eck. Introduction to programming using java, 2006.