6-2017

Recommending personalized schedules in urban environments

Cen CHEN
Singapore Management University, cenchen.2012@phdis.smu.edu.sg

RECOMMENDING PERSONALIZED SCHEDULES IN URBAN ENVIRONMENTS

CEN CHEN

SINGAPORE MANAGEMENT UNIVERSITY

2017

Recommending Personalized Schedules in Urban Environments

by

Cen Chen

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee

Hoong Chuin Lau (Supervisor/Chair)
Professor
Singapore Management University

Shih-fen Cheng (Co-Supervisor)
Associate Professor
Singapore Management University

Pradeep Varakantham
Associate Professor
Singapore Management University

Stephen F. Smith
Professor
Carnegie Mellon University

Singapore Management University
2017

Recommending Personalized Schedules in Urban Environments

Cen Chen

Abstract

In this thesis, we are broadly interested in solving real world problems that involve decision support for coordinating agent movements in dynamic urban environments, where people are agents exhibiting different human behavior patterns and preferences. The rapid development of mobile technologies makes it easier to capture agent behavioral and preference information. Such rich agent specific information, coupled with the explosive growth of computational power, opens many opportunities that we could potentially leverage, to better guide/influence the agents in urban environments.

The purpose of this thesis is to investigate how we can effectively and efficiently guide and coordinate the agents with a personal touch, which entails optimized resource allocation and scheduling at the operational level. More specifically, we look into the agent coordination from three specific aspects with different application domains: (a) crowd control in leisure environments by providing personalized guidance to individual agents to smooth the congestions due to the crowd; (b) mobile crowdsourcing by distributing location-based tasks to part-time crowd workers on-the-go to promote the platform efficiency; (c) workforce scheduling by better utilizing full-time workforce to provide location-based services at customers' homes. For each, we propose models and efficient algorithms, considering agent-level preferences and problem-specific requirements. The proposed solution approaches are shown to be effective through various experiments on real-world and synthetic datasets.

Contents

List of Figures

List of Tables

Acknowledgement

First of all, I am deeply grateful to my primary advisor Professor Hoong Chuin Lau and secondary advisor Associate Professor Shih-fen Cheng for the insightful conversations and useful feedbacks over the years. Their valuable guidance, encouragement, and great considerations have helped shape my interests and ideas, which are great supports for my research and career development.

I am very thankful to my thesis committee members, Associate Professor Pradeep Varakantham and Professor Stephen Smith, for their commitments and constructive comments to improve the thesis. I would like to thank: Professor Archan Misra for his insightful feedbacks and guidance which help shape the idea for Chapter 4 and direct me to this prominent topic; Professor Stephen Smith and Dr. Zachary Rubinstein for the discussions and valuable guidance during my visit at CMU, from which Chapter 5 have benefited a lot.

I would like to thank Living Analytics Research Center for the scholarships and financial supports for my study. I would also like to extend my thanks to Professor Steve Fienberg and Professor Ee-peng Lim for giving me the opportunity to participate in PhD Overseas Training Residency in CMU.

I have been very privileged to collaborate with many great people: Associate Professor Zhiling Guo, Aldy Gunawan, Na Fu, Thivya Kandappu, Nikita Jaiman, Randy Tandriansyah, Yinfei Yang, Forrest Bao, Qiang Qu. Also sincere thanks to all my amazing friends and lab mates in SMU: Larry Lin, Wei Xie, Jing Guo, Peipei Xia, Jingjing Gu, Ying Ding, Yuan Tian, Jing Ren, Wei Gong, Jiali Du and all others with whom I shared my precious graduate life.

Last but not least, I would like to express my deepest gratitude to my family for their unconditional love, support, encouragement, and companionship throughout my Ph.D. journey.

- To my family.

List of Publications

The work discussed in this thesis has directly led to 8 papers. We list these and other papers published during the Ph.D. candidature as follows.

Journal

1. Shih-fen Cheng, **Cen Chen**, Thivya Kandappu, Hoong Chuin Lau, Archan Misra, Nikita Jaiman, Randy Tandriansyah, Desmond Koh. Scalable Urban Mobile Crowdsourcing: Handling Uncertainty in Worker Movement. *ACM Transactions on Intelligent Systems and Technology*, to appear.

2. **Cen Chen**, Shih-Fen Cheng, and Hoong Chuin Lau. Multi-agent orienteering problem with time-dependent capacity constraints, *Web Intelligence and Agent Systems: An International Journal*, 12.4: 347-358, 2014.

Conference and Workshop

1. **Cen Chen**, Zachary Rubinstein, Stephen Smith and Hoong Chuin Lau. Tackling Large-scale Home Health Care Delivery Problem with Uncertainty. *International Conference on Automated Planning and Scheduling (ICAPS-17)*, Pittsburgh, USA, June 2017.

2. **Cen Chen**, Zachary B. Rubinstein, Stephen F. Smith, Hoong Chuin Lau. Achieving Near-optimal Solutions for Large-scale Home Health Care Scheduling Problem. *AAAI Conference on Artificial Intelligence*

(AAAI-17) Workshop on AI and OR for Social Good, San Francisco, February 2017.

3. **Cen Chen**, Shih-Fen Cheng, Hoong Chuin Lau, and Archan Misra. Towards city-scale mobile crowdsourcing: Task recommendations under trajectory uncertainties, *International Joint Conference on Artificial Intelligence (IJCAI-15)*, Beunos Aires, Argentina, July 2015.

4. **Cen Chen**, Shih-Fen Cheng, Archan Misra, and Hoong Chuin Lau. Multi-agent task assignment for mobile crowdsourcing under trajectory uncertainties (Extended Abstract), *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-15)*, Istanbul, Turkey, May 2015.

5. **Cen Chen**, Shih-Fen Cheng, Aldy Gunawan, Archan Misra, Koustuv Dasgupta, and Deepthi Chander. TRACCS: A Framework for Trajectory-Aware Coordinated Urban Crowd-Sourcing, *AAAI Conference on Human Computation and Crowdsourcing (HCOMP-14)*, Pittsburgh, USA, November 2014.

6. **Cen Chen** Shih-Fen Cheng, and Hoong Chuin Lau. The multi-agent orienteering problem (Extended Abstract), *Metaheuristics International Conference (MIC-13)*, Singapore, August 2013.

Related Publications

1. Thivya Kandappu, Nikita Jaiman, Randy Tandriansyah, Archan Misra, Shih-Fen Cheng, **Cen Chen**, Hoong Chuin Lau, Deepthi Chander, and Koustuv Dasgupta. TASKer: Behavioral insights via campus-based experimental mobile crowd-sourcing, *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp-16)*, Heidelberg, Germany, September 2016.

2. Thivya Kandappu, Archan Misra, Shih-Fen Cheng, Hoong Chuin Lau, **Cen Chen**, Nikita Jaiman, Randy Tandriansyah, Koustuv Dasgupta, and Deepthi Chander. Campus-scale mobile crowd-tasking: Deployment and behavioral insights, *ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW-16)*, San Francisco, CA, USA, February 2016.

Other Publications

1. **Cen Chen**, Peilin Zhao, Longfei Li, Jun Zhou, Xiaolong Li and Minghui Qiu. Locally Connected Deep Learning Framework for Industrial-scale Recommender Systems (Poster), *International World Wide Web Conference (WWW-17)*, Perth, April 2017.

2. Yinfei Yang, **Cen Chen**, Minghui Qiu, and Forrest S. Bao. Aspect Extraction from Product Reviews Using Category Hierarchy Information (Short), *European Chapter of the Association for Computational Linguistics (EACL-17)*, Valencia, April 2017.

3. Yinfei Yang, **Cen Chen**, Minghui Qiu, and Forrest S. Bao. Aspect-Based Helpfulness Prediction for Online Product Reviews, *IEEE International Conference on Tools with Artificial Intelligence (ICTAI-16)*, San Jose, November 2016.

4. **Cen Chen**, Zhiling Guo, Shih-fen Cheng, and Hoong Chuin Lau. An Experimental Investigation of Product Competition and Marketing in Social Networks, *China Summer Workshop on Information Management (CSWIM-16)*, Dalian, China, June 2016.

5. Qiang Qu, **Cen Chen**, Christian S. Jensen, Anders Skovsgaard. Space-Time Aware Behavioral Topic Modeling for Microblog Posts, *IEEE Data Engineering Bulletin*, 38(2): 58-67, 2015.

Chapter 1

Introduction

1.1 Background & Motivation

Urbanization is a social phenomenon involving population shifts from rural areas to urban cities. Rapid urbanization[1] improves the vitality of the society, while inevitably leading to high population density in urban cities. As a result of overcrowding, this has led to huge increase in demand vis-à-vis supply of public facilities and services. Main reason for this is due to insufficient expansion space, long construction time and high costs involved for infrastructure expansion. At the same time, urban environments are typically multi-faceted large physical systems, which naturally lacks coordination in the usage of resources. Rapid urbanization, coupled with inefficient usage of the existing facilities and services, essentially may deteriorate people's quality of life in various ways, such as longer waiting times for queues, manpower scarcity for public services and decreased efficiency for city logistics. In response to the rapid urbanization, city planners may invest on building more infrastructures, such as introducing more manpower and providing more public services, which requires tremendous financial budgets. However, an increase in financial spend-

[1]United Nations [94] reported that 54% of world's population live in cities in 2014, and this figure is projected to increase to 66% by 2050.

ing does not always equate to expected outcomes. Rather, greater emphasis should be placed on better coordination mechanisms to make use of the existing infrastructures in order to improve service efficiency. In this research, we are broadly interested in solving the real-world problems that involve decision support for *agent coordination in complex urban settings*, where people are agents exhibiting different human behavior patterns and preferences.

When coordinating agents in urban environments, it is a very important yet challenging task to tailor the schedules/recommendations with agent-specific information. Personalized context-aware recommendation is an emerging trend in our daily life, which has demonstrated its importance in various real-world domains, such as online retailing. A survey reported that 87% of the consumers found themselves "influenced to buy more when retailers personalize" [30]. Better personalization helps to facilitate better adoption of the recommendations, regardless of being the items suggested, the guidance provided, or the schedules generated. Recommendations in the retail industry often considers each person's own preferences independently and others' preferences are solely used as side information to improve the recommendation or for the cold-start situation[2]. However, in generating recommendation or schedule for agents in an urban environment, we need to take into consideration multiple interactive urban dwellers with diverse behavior patterns and preferences, as the decision of one agent may affect the quality of the schedule/recommendation of another. Such dependencies among agents make personalization much harder to achieve in complex urban environments.

Against this backdrop, the objective of this thesis is therefore to investigate *how we can effectively and efficiently coordinate agents with a personal touch*, i.e., taking into consideration individual agent's personal interests as well as their interactions with one another. The rapid development of mobile tech-

[2] "Cold-start problem" is prevalent in recommender systems, which refers to the situation, where we have no clue about one's own preference or information.

nologies makes it easier to capture agent behavioral and preference information nowadays. Such rich agent-specific information opens many opportunities for agent coordination that can be potentially leveraged. How to better make use of such information to guide/influence the agents, is critical in any decision support system that recommends or schedules for agents.

At the high level, agent coordination problems can be classified into two types: with non-cooperative agents and cooperative ones. More specifically, as shown in Figure 4.2, we study the two types of coordination problems with the following application domains.

- For the non-cooperative case, we look into *crowd control* problem in the leisure environments, with the objective of providing personalized guidance to individual agents to reduce the congestions due to the crowd.
- For the cooperative case, we first study the *mobile crowdsourcing* problem, where location-based tasks are centrally distributed to part-time crowd workers to promote task completion. We then explore a mobile version of the *workforce scheduling* problem in the home health care context, with the goal to roster and route the full-time providers to serve customers at their specified locations to improve public service efficiency.

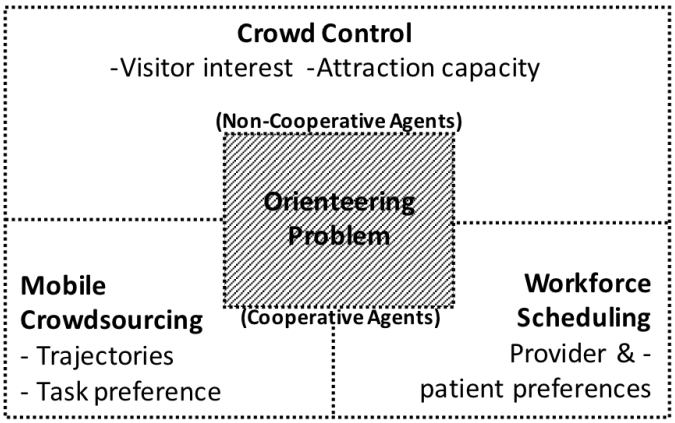


Figure 1.1: Overviews of the problems addressed in the thesis.

Non-cooperative problems usually can be solved using techniques from game

theory, while cooperative ones often employ techniques from the operations research literature.

Mathematically speaking, these problems share some commonality that agents move on a graph to perform some actions and receive rewards for performing these actions. Thus, the routing aspect is important to be incorporated, putting aside the additional problem-specific considerations/constraints. To a large extent, the problems of our concern all share the *Orienteering Problem* as the underlying problem model. Orienteering problems [135, 57] have long been extensively studied to model various problems from transportation and logistics, whose goal is to determine the routes (i.e., a subset of nodes to visit in a sequence) such that the total reward collected is maximized within the given time budget. Hence, we model these problems as variants of the Orienteering Problem.

This thesis focuses on developing *computationally efficient strategies* to address the above-mentioned three coordination problems. The inclusion of personalization makes these problems much harder to solve, due to the additional agent-specific preference constraints. Thus, the challenges lying is to provide scalable mechanisms to deal with the complexity of preference handling for each agent. In this thesis, we intentionally exploit the problem structures, i.e., the potential decomposability that individual agent's preference and orienteering are independent from those of the other agents. Techniques, such as Lagrangian relaxation and sampled fictitious play, are employed to decompose the large problem into much smaller agent-level sub-problems, which essentially shift the computation complexity into individual sub-problems that can be solved efficiently and parallelly.

In the following subsections, we will provide details on these problems.

1.1.1 Crowd Control

The first application domain that motivates this research is the crowd control problem in leisure environments. For service operators, one critical task they need to perform on a daily basis is to provide proper guidance to visitors. Such guidance may be passive, which is delivered via sign board or staff's ground instructions. The guidance can also be active, in which case individual visitors may be guided by following instructions delivered in real time to their mobile devices. As the venue operator needs to watch closely how queues build up at different attractions throughout the day, it needs to generate recommendations for individual visitors, following their respective preferences, while observing capacity constraints for different attractions throughout the day.

In the real world, recommendations for visitors are often generated in a passive manner, with less emphasis to visitor's preferences. For example, Disney employed a FASTPASS system to incentivize visitors to visit popular attractions at later times by giving reservation coupons, which guarantee them immediate access if they visit these attractions at the system-generated appointment return times [44]. However, in this strategy, individual agents are treated the same without considering agent-specific interests and constraints, thus guidance generated might be less appealing for individual agents to follow. And here lies the potential: if we could generate recommendations for visitors actively to control the crowd without sacrificing their visiting experience, it will be a win-win situation for both visitors and park operators.

1.1.2 Mobile Crowdsourcing

The second application domain explored is the mobile crowdsourcing. In the traditional crowdsourcing paradigm, task owners distribute tasks via online platforms to attract crowd workers who would work on them for monetary rewards. Mobile crowdsourcing is a rapidly growing extension to the tradi-

tional crowdsourcing paradigm, characterized by (a) tasks having strong location specificity (i.e., the task requires a crowd worker physically visiting a specific location) and (b) tasks principally requiring the use of mobile devices/smartphones at these locations. A lot of tasks nowadays exhibit clear location characteristics, such as citizen sensing (asking participants to contribute sensor readings such as pollution, congestion, noise level) [64, 145, 139, 146], price checks, store audits (e.g., checking shelves, store displays), logistics (package pickup and delivery) [4], just to name a few.

In most of the existing works [2, 68, 67, 61], tasks are prioritized and assigned to available crowd workers using the knowledge about the worker's current location and availability, which is a myopic strategy that does not consider agent-specific behaviors, e.g., individual agent's movement patterns and preferences, such as the amount of time they are willing to spend, the types of tasks preferred and willingness to accept the tasks assigned. Nowadays, with the mobile crowdsourcing app installed in worker's phone, it's easy for the company to capture workers' trajectories and their preferences. The often predictable movement patterns of the mobile crowd workers clearly present a unique opportunity: instead of letting users individually select tasks in an uncoordinated fashion, a centralized crowd-tasking platform can suggest or recommend tasks in a more efficient way. In other words, we want to guide the crowd workers to perform tasks that align with their daily movement patterns and their own preferences, and at the same time, maximize the task completion rate and the global revenue.

1.1.3 Workforce Scheduling

The last application domain that draws our interest is a mobile version of the workforce scheduling, where service providers are scheduled to service the customers at their specified locations. More specifically, we study the workforce

scheduling for home health care visits. Home health care provides a wide range of health care services that are delivered at patients' homes. Over the past decade, an increasing number of people subscribe to home health care services, especially for patients with chronic conditions. According to the US National Association for Home Care & Hospice [93], roughly 12 million people received home health care services from 33,000 providers in 2008. This number is set to grow rapidly with an increasing aging population: Population Reference Bureau [98] predicts that "the number of Americans above 65 will increase from 46 million in 2016 to over 98 million by 2060". Against the manpower crunch in health care professionals, in-home service providers are under increasing pressure to provide high-quality service at a low cost to an ever growing demand.

For workforce scheduling problems in the home health care context, individual's preferences are commonly addressed, but not comprehensively. Problems in this domain differ substantially, as problems considered often originate from different regions with various requirements and regulations. Time windows, qualifications, and provider availabilities are commonly considered, while other preferences, such as workload fairness, continuity of care, inter-visit temporal dependencies are less incorporated [41]. To close this gap, we aim to address a more realistic problem with a comprehensive set of considerations from the domain, that tightly integrating both patients' and providers' preferences.

1.2 Contributions

To summarize, we make the following contributions in this thesis:

1. What distinguishes the problems studied in this thesis from many agent coordination problems in the literature is that they largely ignore personalization, i.e., neglect the importance of agent behavior and preferences. In this thesis, we study the coordination problems with emphasis on personalization in three less applied domains, under both non-cooperative

and cooperative settings.

2. For crowd control, we introduce a multi-agent version of the Orienteering Problem, where agents are self-interested. In this non-cooperative setting, instead of seeking for a globally optimal plan, we focus on identifying equilibrium solutions where individual agents cannot improve their current utilities by deviation. We first formulate the centralized problem as a variant of the Orienteering Problem. Due to the distributed nature of the problem, we then formulate the problem in the game-theoretic framework and propose to use the sampled fictitious play algorithm with rejection-base sampling to seek for equilibrium solutions.

3. For mobile crowdsourcing, we investigate a trajectory-aware crowdsourcing framework, where a centralized engine recommends tasks for a large pool of workers while taking into account the uncertain movement patterns and individual preferences. We formulate the problem as a variant of the Orienteering Problem. Mathematical formulations are developed, with the objective to find assignments and routes for all workers that maximize the cumulative expected utility. Subsequently, effective heuristics based on Lagrangian relaxation and dual decomposition are presented to scale up the solution approaches to city-scale scenarios.

4. For workforce scheduling, we study a multi-period home health care scheduling problem that generates start-of-the-day schedules. To make the schedules personal and efficient, we look into realistic scenarios with a comprehensive set of considerations, e.g., time windows, continuity of care, workloads, inter-visit temporal dependencies, and especially duration uncertainties. We formulate the problem as a variant of the Orienteering Problem and incorporate the uncertainties by chance constraints. Scalable solution approaches based on Lagrangian relaxation and sample average approximation are then proposed to solve the problem.

1.3 Organization of the Dissertation

The rest of this thesis is organized as follows. Chapter 2 introduces the preliminaries. Chapter 3 elaborates on the crowd control problem in leisure environments. Chapter 4 presents the trajectory-aware mobile crowdsourcing problem. Chapter 5 investigates the home health care scheduling problem in the urban environments. Chapter 6 concludes the thesis and discusses the future work.

Chapter 2

Preliminaries

In this chapter, we present an overview of the Orienteering Problem and Lagrangian relaxation technique.

2.1 Orienteering Problem

The term **orienteering** originates from the cross-country navigation sport where participants have to visit control points with aid of the map and compass within a given time period. The first orienteering event was held in Norway in 1897 [1]. Ever since then, different types of orienteering event have been developed. For example the *classic orienteering event* is to visit every control points in a predefined order and the participant, who completed with the shortest time, wins. Another important type is *score orienteering event*, where participants do not require to visit all controls. Instead, control points are associated with different scores depending on difficulty. The winner of the game will be the participant who collects the highest score within a time budget.

The **Orienteering Problem (OP)**, formally defined by Tsiligirides [128], was motivated by scheduling the score orienteering event in which participants get rewards from visiting a set of control points. OP, also known as the selective traveling salesman problem (TSP), can be used to model a wide variety of real-

world problems like tour planning, route planning for facility inspection and patrolling of security forces in a network. As a generalization of the TSP, it is a notoriously challenging NP-hard problem, that has long been studied since the 1980s. Golden et al. [51] prove the NP-hardness of the OP by reducing the problem to TSP.

OP is defined over a graph $G = (N, E)$, where N is a set of nodes and E denotes a set of edges. A node $i \in N$ can be viewed as an intersection point on the graph, which exhibits either physical or virtual location characteristics. On the road map, a node represents a physical location. On the internet, a node can be a virtual IP address. Each node i is associated with a positive *reward* s_i, which represents the incentive received when visiting the node. An edge $e_{ij} \in E$ is an accessible link between nodes i and j. Each edge is associated with a non-negative *cost*, which is often interpreted as travel time, distance, or petrol consumption. Travel time is commonly used in the OP. We denote the travel time between node i and j as t_{ij}. A node can be visited at most once within the given time budget T_{max}. Given the origin node 1 and the destination node n, the objective of the OP is to determine a route, i.e., a set of sequenced nodes, to visit that maximizes the total accumulative reward collected for the route within the time budget.

2.1.1 Mathematical Formulation

With these notations and assumptions, we can then formulate the OP as an Integer Linear Program (ILP). We first define the following decision variables:

Variables	Descriptions
$x_{ij} \in \{0, 1\}$	set to 1 if node i is visited before node j in the route, and 0 otherwise.
$u_i \in \{1, ..., n\}$	denote the position of node i in the route.

Mathematically, OP is formulated as follows [134]:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} s_i \cdot x_{ij}, \tag{2.1}$$

$$\sum_{t=1}^{T} \sum_{j=1}^{n} x_{1j} = \sum_{t=1}^{T} \sum_{i=1}^{n} x_{in} = 1, \tag{2.2}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{n} x_{id} = \sum_{t=1}^{T} \sum_{j=1}^{n} x_{dj} \leq 1, \qquad \forall d \in 2, ..., n-1, \tag{2.3}$$

$$\begin{cases} 2 \leq u_i \leq n, & \forall i \in 2, ..., n, \\ u_i - u_j + 1 \leq (N-1)\left(1 - x_{ij}\right), & \forall i, j \in 1, ..., n, \end{cases} \tag{2.4}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} \cdot x_{ij} \leq T_{max}, \tag{2.5}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in 1, ..., n. \tag{2.6}$$

Objective (2.1) maximizes the total reward collected. The first set of constraints (2.2) ensures that the route starts at node 1 and ends at node n. Constraints (2.3) guarantee that flows are conserved at all nodes and each node d is visited at most once. Constraints (2.4) specify the visiting sequences of the nodes in the route and eliminate the sub-tours (follow the TSP Miller-Tucker-Zemlin sub-tour elimination constraints [87]). Finally, constraint (2.5) ensures that the route ends before the given time budget T_{max}.

2.1.2 Common OP Variants

The most common variants of the OP include: 1) The Team Orienteering Problem (TOP), in which a group of centrally controlled agents are sent to collect rewards by visiting check points [24, 13], 2) The Orienteering Problem with Time Windows (OPTW), in which service time windows are specified for each node [66], and 3) the combination of the above two variants, i.e., the Team Orienteering Problem with Time Windows (TOPTW) [90].

2.1.3 Applications of OP

OP have been successfully applied in many application domains over the years.

1. Scheduling salesman visits: Tsiligirides [128] presented the problem of traveling salesman visiting a number of cities, with sales at each city known beforehand. As he is constrained by his limited time budget, he has to schedule a route of a subset of cities to maximize his total sales. This is the first orienteering application mentioned in the literature.

2. Goods delivery: Golden et al. [50] introduced an inventory routing problem, where trucks deliver fuels to consumers. As the fuel supply is limited on a daily basis, the company have to decide a subset of customers to serve based on consumer's urgency score, such that customers can maintain their inventories at an adequate level.

3. Network design: Thomadsen and Stidsen [122] applied OP to a single-ring design problem for building telecommunication networks. A ring topology is used to protect breakdown in a telecommunication network. In a ring, the number of nodes and the length of rings are limited due to network delay problem. Each node in a ring is associated with certain revenue. The goal is to design the ring, i.e., a subset of nodes in a sequence, to maximize the revenue.

4. Security surveillance: Wang et al. [138] presented a military surveillance scenario, where a submarine or an unmanned aircraft is involved in surveillance activities. The length of the expedition is often limited by fuel or time constraint and locations are associated with multi-dimensional benefits. The goal is to visit and photograph the best subset of locations.

5. Tour guidance: Another widely studied OP application is in the context of tour guidance. When visitors go to a city, it is often impossible for

them to explore all the attractions within a day. Thus, they have to schedule a route containing the best attractions to visits, that maximize their visiting experience, depending on their personal preferences [131, 114, 138, 130, 111].

2.1.4 Solution Approaches

A large number of OP variants and corresponding algorithms for solving them have been discussed in the literature. Tsiligirides [128] presented an early survey of heuristic methods for OP; this was followed by more recently surveys of Vansteenwegen et al. [134] and Gunawan et al. [57], which provide mathematical formulations and solution approaches for the OP and its related variants.

2.1.4.1 Solution Approaches for OP

Feillet et al. [39] presented an overview of the exact algorithms for solving the OP. Researchers have proposed several exact algorithms, mainly by employing OR techniques, such as branch-and-bound (e.g., Laporte and Martello [77] and Ramesh et al. [100]), branch-and-cut (e.g., Fischetti et al. [42] and Gendreau et al. [47]), cutting-plane by Leifer and Rosenwein [78], and branch-and-price by Feillet et al. [39]. Heuristics algorithms have also been discussed in literature to tackle OP, such as Monte Carlo technique by Tsiligirides [128], local search based methods (e.g., Golden et al. [50, 52], Ramesh and Brown [101], and Chao et al. [23]), tabu search heuristics (e.g., Gendreau et al. [48], Keller [71], and Liang et al. [80]), artificial neural network by Wang et al. [137], genetic algorithm by Tasgetiren [119], and ant colony optimization (ACO) approach by Liang et al. [80].

More recent approaches for solving the OP are ACO and variable neighborhood search (VNS) based approaches proposed by Schilde et al. [111], particle

swarm optimization based algorithms by Sevkli and Sevilgen [112], Multi-Level VNS by Liang et al. [81], Greedy Randomized Adaptive Search Procedure (GRASP) with path relinking by Campos et al. [19], and the latest algorithm proposed by Marinakis et al. [85], which is a combination of GRASP, evolutionary algorithm and two local search procedures.

2.1.4.2 Solution Approaches for OP Variants

TOP is an extension of OP where the goal is to plan a set of routes for all the members of the team that maximizes the total rewards collected by the team within the time limit T_{max}[24]. TOP is well-studied, and many researchers have proposed either exact solution approaches (e.g., Butt and Cavalier [18], Tang and Miller-Hooks [118], and Boussier et al. [13]) or heuristic approaches (e.g., Chao et al. [24], Tang and Miller-Hooks [118], Archetti et al. [3], and Ke et al. [69]). Most recent research efforts on TOP have been on the development of efficient and effective heuristics, such as greedy randomized adaptive search procedure by Souffriau et al. [115], guided local search approach by Vansteenwegen et al. [132], memetic algorithm by Bouly et al. [12], population based meta-heuristics (e.g., [92], Dang et al. [32], and Dang et al. [34]), and a Pareto mimic algorithm proposed by Ke et al. [70]. A few recent exact approaches was discussed by Dang et al. [33] using a branch-and-cut algorithm, followed by Keshtkaran et al. [72] employing a branch-and-price approach.

Compared to OP and TOP, both their time window extensions, i.e., OPTW and TOPTW, received much less attention in the early year. OPTW was first solved by Kantor and Rosenwein [65], using an insertion heuristic. Exact algorithms were mainly developed by Righini and Salani [104, 105] using dynamic programming. Special cases of OPTW were discussed by Mansini et al. [84], where start and end nodes are the same, and by Bar-Yehuda et al. [7], where all nodes have the same scores and nodes are positioned in a line or with asymmetric distances. More recent approaches for solving the OPTW are proposed

by Duque et al. [35] with a pulse algorithm and Gunawan et al. [54] using Iterated Local Search (ILS) technique.

Majority of the research efforts of TOPTW are solely on the development of heuristics, e.g., ACO (e.g., Montemanni and Gambardella [90] and Gambardella et al. [46]), ILS by Vansteenwegen et al. [133], hybridized evolutionary LS algorithm by Labadie et al. [74], LP-based Granular VSN by Labadie et al. [75], hybrid algorithm based on GRASP and ILS by Souffriau et al. [116], hybrid LS and Simulated Annealing(SA) based approaches by Hu and Lim [60] and Lin and Vincent [83], and the recent state-of-the-art algorithm by Gunawan et al. [56, 55], which hybridizes SA with ILS.

2.2 Lagrangian Relaxation for Solving ILP

Lagrangian relaxation (LR) is a well-known technique for solving difficult combinatorial problems. The central idea of the LR approach, as pointed out by Fisher [43], is that "*many difficult integer problems can be viewed as easy problems complicated by a relatively small set of side constraints.*". By moving these constraints to the objective function (i.e., dualizing these side constraints), we can provide a better lower bound (for minimization problems) more efficiently.

Note that there are also other decomposition techniques that can be explored to handle combinatorial optimization problems efficiently, such as Bender decomposition which can efficiently solve large linear programming problems that have a special block structure. Bender decomposition focuses on dualizing variables and iteratively adding new constraints, while LR focuses on dualizing constraints and penalizing constraint violations. LR is particularly powerful if the optimization problem allows further decomposition after the dualization of difficult constraints, where sub-problems can be solved efficiently in parallel.

2.2.1 Mathematical Formulation

Suppose we have the following ILP:

$$\begin{aligned} &\min c^T x \\ &s.t.\ Ax \leq b \qquad\qquad (P) \end{aligned}$$

We refer this ILP as the primal problem P.

Lagrangian relaxation relax the explicit linear constraints by bringing them into the objective function with associated vector $\boldsymbol{\lambda} = \{\ldots, \lambda_m, \ldots\}$ called the *Lagrangian multiplier*. We denote the resulting dual problem as $L(\boldsymbol{\lambda})$.

$$\min c^T x + \lambda(Ax - b) \qquad\qquad (L(\boldsymbol{\lambda}))$$

Lemma 2.2.1. *Bounding Principle: For any value of $\boldsymbol{\lambda} \geq 0$, the objective of $(L(\boldsymbol{\lambda}))$ is a lower bound on the optimal objective function value of the original optimization problem (P).*

For detailed descriptions of the important concepts and proofs for Lagrangian relaxation, please see Bertsekas [11].

According to the above Lemma, if we maximize the minimum value we obtain from $(L(\boldsymbol{\lambda}))$, we obtain a tighter bound on the objective value of (P). Thus we can address the original problem by instead solving the dual problem. In the following section, we describe the subgradient descent algorithm to approximate the Lagrangian dual.

2.2.2 Solving the Lagrangian Dual

The Lagrangian dual problem can be solved by using the following *subgradient descent algorithm* [11]:

Algorithm 2.2.2. *Subgradient Descent Algorithm*

- ***Initialization:*** *Set iteration* $t = 1$. *Set* $\boldsymbol{\lambda}_t$ *to be all zeros.*
- ***Iteration*** $t \geq 1$***:***

1. ***Solving Duals:*** *Solve the* $(L(\boldsymbol{\lambda}))$ *given* $\boldsymbol{\lambda}_t$.
2. ***Primal Extraction:*** *Obtain the primal objective function value corresponding to the current dual solution.*
3. ***Updating Lagrangian Multipliers:*** *Lagrangian multipliers are adjusted according to the degree of constraint violation; for violated (satisfied) constraints, the values of corresponding multipliers should be increased (decreased) accordingly. We adopt a well-known adaptive multiplier adjustment formula by Held et al. [58]:*

$$\lambda_{t+1} := \max\{0, \lambda_t + \alpha_t(Ax_t - b)\},$$

where α_t *represents the update step size, and is defined to be adaptive to both the duality gap and the solution quality (the magnitude of constraint violation):*

$$\alpha_t = \frac{\mu_t(F^* - L(\boldsymbol{\lambda}_t))}{||Ax_t - b||^2} \ ,$$

where F^* *is the best primal value seen so far, and* $L(\boldsymbol{\lambda}_t)$ *represents dual value obtained in iteration* t. μ_t *is in the range of* $(0, 2]$, *and is defined as:*

$$\mu_t = \begin{cases} 2, & t = 0, \\ 0.5\ \mu_{t-1}, & \text{if } L(\boldsymbol{\lambda}_t) \text{ is non-improving for } \kappa \text{ iterations.} \end{cases}$$

4. ***Termination:*** *The search process terminates either when the allocated time is up or the duality gap (i.e.,* $F^* - L(\boldsymbol{\lambda}_t)$*) is below certain*

threshold. In our implementation, we observe both termination conditions.

Chapter 3

Crowd Control

3.1 Overview

We begin by first considering the crowd control problem for a collection of interconnected service providers in leisure environments (real-world examples include the MICE[1] industry, amusement parks, and museums). Individual agents (visitors) in this environment aim to visit a sequence of selected service providers with the objective of maximizing their utilities obtained by receiving services, while observing their individual time budget limitations and service provider's time-dependent capacity constraints.

In this chapter, we investigate the problem of crowd control in leisure environments. We introduce a multi-agent version of the OP to address the crowd control issue, which we believe to be the first of its kind. We call this OP variant as Multi-agent Orienteering Problem with Time-dependent Capacity Constraints (MOPTCC). In this problem, nodes are subject to time-dependent capacity constraints and time-dependent rewards. The rewards allow us to model individual agents' preferences, while the capacity constraints enable the operator to manage and control crowds.

We depart from the classical setting of TOP and TOPTW, which is con-

[1]Meetings, incentives, conferences, and exhibitions.

cerned with the route planning for a team of agents in a centralized fashion. Instead, we treat the problem as a multi-agent planning problem where individual agents are self-interested and will scrutinize their given plans carefully. Instead of seeking for a *globally optimal plan*, we focus on identifying Nash equilibrium where individual agents cannot improve their current utilities by deviation.

3.2 Literature Review

Formally speaking, this multi-agent TOP is modeled as a game, where players are agents, player's strategy space is the set containing all possible routes, and the payoff function is the mapping from a joint strategy (routes from all players) to a vector of *payoff values* for all players. If a particular joint strategy is infeasible (e.g., if queue lengths at some service providers violate the capacity constraints), all players will receive the value of $-\infty$.

For any normal-form game, the existence of mixed strategy Nash equilibrium is guaranteed, but pure strategy Nash equilibrium does not always exist (see for instance [97]). Under certain cases, pure strategy Nash equilibrium exists. According to Debreu-Glicksberg-Fan theorem [45], if we have an infinite normal-form game where each strategy set is a compact convex subset of Euclidean space and the payoff functions are continuous and quasi-concave, then pure strategy Nash equilibrium exists. Cheng et al. [28] show that a symmetric 2-strategy game[2] must have a symmetric pure strategy Nash equilibrium. While the complexity of finding a mixed Nash equilibrium in an n-player game is still unknown, computing a mixed Nash equilibrium in a 2-player game is PPAD-complete [27]. For the case of pure strategy Nash equilibrium, determining its existence in a graphical game (a special case of normal-form game) is NP-complete [53]. From the computational perspective, it has been shown

[2]In a symmetric game, every player is identical with respect to the game rules.

that finding pure strategy Nash equilibrium is only possible in fairly small games (e.g., even for 5-player, 5-strategy games, it may take hours and sometimes days to solve). The classical approach for finding Nash equilibrium in a 2-player game is the pivot-based Lemke-Howson algorithm [79]. More recently, a mixed integer programming formulation is also proposed for solving 2-player normal-form games [110]. In cases where payoff matrix is large and complete characterization is computationally intractable (e.g., each payoff value can only be estimated by running multiple time-consuming simulations), the focus has been on computationally tractable approaches in approximately finding equilibria (e.g., see Wellman et al. [140] and Jordan et al. [62]) without complete payoff matrix.

Finally, most previous works on OP are static in that the network parameters (such as travel times and node delays) remain constant over time. This is another major feature that distinguishes our work from the literature. In the problem we are about to describe, we allow queueing times at service providers to be dependent on the number of visitors showing up in the same time period. As such, the time required to receive service from a particular provider would depend on not just this agent's strategy, but also other agents' strategies.

3.3 Problem Formulation

To understand why a game-theoretic framework would be necessary for MOPTCC, let's start with a simple example to illustrate the inadequacy of global optimum in a multi-agent environment.

3.3.1 A Motivating Example

Consider the following two-agent, two-provider problem: Let n_0 be the designated starting and ending nodes, and let n_1 and n_2 represent two providers. We assume that both agents start their trips from n_0 at time 1 and they have

to return to n_0 again on or before time 5. The travel time between any two nodes is 1. For each provider, it can only serve one agent at a time, and its service time is 1 time unit. If multiple agents request service from the same provider simultaneously, we assume that agents are to be served one after another according to their ID numbers. We further assume that the queueing policy is set to allow provider n_2 to handle at most one agent at a time (i.e., no queueing allowed) and no limit for provider n_1. Finally, we assume that agents collect their utilities when they finish their services at the provider.

Agents' time-dependent utilities for receiving services from the two providers are listed in Table 3.1:

	Agent 1		Agent 2	
t	n_1	n_2	n_1	n_2
1	1	2	3	1
2	1	2	3	1
3	2	3	3	1
4	2	3	5	2
5	2	3	5	2

Table 3.1: Agents' time-dependent utilities at different providers.

Based on the above setup, we can see that due to the time limit constraint, each agent can choose at most one provider before returning to n_0. The outcomes resulting from agents' joint decisions are summarized as the payoff matrix in Table 3.2. Note that for the joint decision (n_1, n_1), both agents would

		Agent 2	
		n_1	n_2
Agent 1	n_1	2 , 5	2 , 1
	n_2	3 , 3	$-\infty$, $-\infty$

Table 3.2: Payoff matrix for all joint decisions.

arrive at n_1 in time 2, with Agent 1 receiving service first, followed by Agent 2. Agent 1 would leave n_1 in time 3 and receives the value of 2 (according to Table 3.1); for Agent 2, he begins his service in time 3, and leaves n_1 in time 4, receiving the value of 5. For (n_1, n_2) and (n_2, n_1), since there are no

conflict, the corresponding payoff values can be directly found in row ($t = 3$) of Table 3.1. (n_2, n_2) is infeasible since provider n_2 can handle at most 1 agent (i.e., no queueing is allowed for n_2).

From Table 3.2, we can see that the global optimum is (n_1, n_1), with combined value 7. However, this solution is not stable, as Agent 1 would be better off by deviating from n_1 to n_2. In fact, the joint strategy (n_2, n_1), with combined value 6, is a Nash equilibrium.

This is a classical demonstration where selfish agents would deviate from the globally optimal solution and opt for Nash equilibria with lower combined payoff. In this instance, there are two major factors contributing to such phenomenon: 1) agents have their respective time-dependent payoffs, and 2) providers handle agents sequentially, and individual providers might be given different limits on queue lengths.

3.3.2 Centralized Formulation

Although global optimum is not very meaningful for MOPTCC, as argued earlier, we should still present the centralized formulation first. This centralized formula can serve as the comparison baseline, and it is also an important subproblem to be solved repetitively when we introduce the game-theoretic formulation.

The MOPTCC is derived from the classical single-agent OP, where n providers (nodes) are assumed to be fully connected and can be represented as a complete graph with t_{ij} denoting travel time from i to j. We assume that there are m independent agents, and let s_{ik}^t be the utility agent k receives when visiting node i in time t. The service time at provider d is a constant v_d, and the number of agents allowed to simultaneously visit provider d is capped at Q_d^{max}. The horizon of the problem is set to be T time periods. Without loss of generality, we assume that each agent k starts his trip at node 1 in time T_1^k

and should end his trip at node n before time T_n^k (nodes 1 and n can either be real or dummy nodes). Lastly, the decision variables are summarized as follows:

Variables	Descriptions
$x_{ijk}^t \in \{0,1\}$	set to 1 if agent k leaves node i at time t and goes to node j, and 0 otherwise.
$Q_d^t \in \{0,...,K\}$	denote the number of agents visiting node d at time t.

With these notations and assumptions, we can then formulate a centralized optimization problem as an integer linear program, whose objective is to maximize the combined utility received by all agents:

$$\max \sum_{t=1}^{T} \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} s_{ik}^t \, x_{ijk}^t \, . \tag{3.1}$$

The first set of constraints (3.2) ensure that for each agent k, he starts at node 1 and ends at node n:

$$\sum_{t=1}^{T} \sum_{j=1}^{n} x_{1jk}^t = \sum_{t=1}^{T} \sum_{i=1}^{n} x_{ink}^t = 1 \, , \quad \forall k. \tag{3.2}$$

Constraints (3.3) guarantee that flows are conserved at all nodes except the origin (node 1) and the destination (node n):

$$\sum_{t=1}^{T} \sum_{i=1}^{n} x_{idk}^t = \sum_{t=1}^{T} \sum_{j=1}^{n} x_{djk}^t \, , \quad \forall k, d \neq 1 \text{ or } n. \tag{3.3}$$

As in all classical OP, we assume that for each agent k, each node d is visited at most once:

$$\sum_{t=1}^{T} \sum_{j=1}^{n} x_{djk}^t \leq 1 \, . \tag{3.4}$$

Constraints (3.5) define the queue length for each node d at time t. For simplicity, we assume that service rate is 1 at all nodes. Thus Q_d^t equals the queue length from time $t-1$ plus the inflow and minus the outflow of current time t

for this node. Constraint (3.6) ensures that the queue length Q_d^t at any node d should not exceed its corresponding threshold Q_d^{max} at all times.

$$Q_d^t = Q_d^{t-1} + \sum_{k=1}^{m} \left(\sum_{i=1}^{n} x_{idk}^{t-t_{id}} - \sum_{j=1}^{n} x_{djk}^{t} \right), \qquad \forall k, d, \tag{3.5}$$

$$Q_d^t \leq Q_d^{max}, \qquad \forall k, d. \tag{3.6}$$

In constraints (3.7), the arrival and departure times for agent k at node d are constrained by taking into account all potential delays such as service time, queue length at arrival, and travel time.

$$\sum_{t=1}^{T} \sum_{i=1}^{n} \left(t + t_{id} + Q_d^{t+t_{id}} + v_d \right) x_{idk}^t = \sum_{t=1}^{T} t \left(\sum_{j=1}^{n} x_{djk}^t \right), \quad \forall k, d. \tag{3.7}$$

Finally, constraints (3.8) and (3.9) ensure that for each agent k, the schedule starts at T_1^k and ends before T_n^k.

$$\textstyle\sum_{t=1}^{T} \sum_{j=1}^{n} t \cdot x_{1jk}^t = T_1^k, \qquad \forall k, \tag{3.8}$$

$$\textstyle\sum_{t=1}^{T} \sum_{j=1}^{n} t \cdot x_{njk}^t \leq T_n^k, \qquad \forall k. \tag{3.9}$$

By expanding constraints (3.5) recursively, it can be rewritten as constraints (3.10).

$$Q_d^t = Q_d^1 + \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{s=2-t_{id}}^{t-t_{id}} x_{idk}^s - \sum_{k=1}^{m} \sum_{j=1}^{n} \sum_{s=2}^{t} x_{djk}^s \,, \quad \forall d, t. \tag{3.10}$$

When substituting Q_d^t in constraints (3.7) with (3.10), there are non-linear terms. To linearize these non-linear constraints, we introduce α_{ijdlk}^{st} to represent $x_{jdl}^s \cdot x_{idk}^t$ and β_{ijdlk}^{st} to replace $x_{djl}^s \cdot x_{idk}^t$. After the transformation,

constraints (3.7) are replaced by constraints (3.11) – (3.13).

$$\begin{aligned}\sum_{t=1}^{T}\sum_{j=1}^{n} t \cdot x_{djk}^{t} &= \sum_{t=1}^{T}\sum_{i=1}^{n} (t + v_d + t_{id})\, x_{idk}^{t} + \sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{l=1}^{m}\sum_{j=1}^{n}\sum_{s=2-t_{id}}^{t} \alpha_{ijdlk}^{st} \\ &- \sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{l=1}^{m}\sum_{j=1}^{n}\sum_{s=2}^{t+t_{id}} \beta_{ijdlk}^{st}\,, \qquad \forall d, k, \end{aligned} \tag{3.11}$$

$$\begin{cases} \alpha_{ijdlk}^{st} \le x_{jdl}^{s}\,, \\ \alpha_{ijdlk}^{st} \le x_{idk}^{t}\,, \\ \alpha_{ijdlk}^{st} \ge x_{jdl}^{s} + x_{idk}^{t} - 1\,, \end{cases} \qquad \forall i, d, j, k, l, s, t, \tag{3.12}$$

$$\begin{cases} \beta_{ijdlk}^{st} \le x_{djl}^{s}\,, \\ \beta_{ijdlk}^{st} \le x_{idk}^{t}\,, \\ \beta_{ijdlk}^{st} \ge x_{djl}^{s} + x_{idk}^{t} - 1\,, \end{cases} \qquad \forall i, d, j, k, l, s, t. \tag{3.13}$$

After linearization, the above mathematical programming model can then be solved by using standard integer linear programming solver such as CPLEX. However, such formulation does not scale well and only very small instance can be solved [25]. In this chapter, our focus is to solve MOPTCC as a game, and the above formulation can be revised to solve a single-agent version of the problem. Before introducing the equilibrium-seeking algorithm, we will first model the problem using game-theoretic framework.

3.3.3 A Game-theoretic Formulation for MOPTCC

The MOPTCC game is defined as the tuple $\Gamma = \langle \mathcal{N}, \mathcal{S}, u \rangle$, where $\mathcal{N} = \{1, \ldots, n\}$ is the set of all players (agents), $\mathcal{S} = \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ is the joint strategy space, and $u : \mathcal{S} \to R^n$ is the payoff function. When not considering

$\mathcal{S}_{-k}$, player k's strategy space is defined as:

$$\begin{aligned}\mathcal{S}_k = \quad & \{(s_k^1, \ldots, s_k^n) | s_k^i \in \{1, \ldots, n\}, \forall i; \\ & s_k^1 = 1; \exists\, d, s_k^d = n, \\ & \text{for } 1 < i < d, s_k^i \notin \{s_k^1, \ldots, s_k^{i-1}\}, \\ & \text{for } d < i \leq n, s_k^i = 0\}. \end{aligned} \tag{3.14}$$

In other words, a player's strategy must always begin with node 1, end with node n, never repeat, and if the visit sequence is shorter than n, all visits after node n must be no-op, which is denoted as 0.

Given any joint strategy profile s, we can straightforwardly compute the corresponding Q_d^t for all pairs of (t, d). We say that a joint strategy $s \in \mathcal{S}$ produces *feasible* joint orienteering plan if the resulting Q_d^t does not exceed Q_d^{max} for all pairs of (t, d). The utility function u is only defined for strategies that produce feasible joint orienteering plans. If a joint strategy s produces infeasible plan, we defined $u_k(s)$ to be $-\infty$ for all players.

As the MOPTCC game is defined as a normal-form game, all joint strategies can be played. However, due to the feasibility condition defined above, only a small fraction of strategies should ever be considered. As such, the next challenge we have to address would be to devise an algorithm that can effectively and efficiently identify *feasible* equilibrium of the MOPTCC game.

3.4 Solution Approaches

As reviewed in Section 3.2, even for a very simple game that contains only two players, it can be very computationally challenging to compute equilibrium solutions. As the number of players and the size of strategy space increase, the complexity of the equilibrium-seeking would increase quickly. In the MOPTCC game, the critical challenge is the size of the strategy space.

In fact, as in the usual OP, the size of the strategy space grows exponentially as the number of destinations increases (e.g., for problem with n destinations, the size of the strategy space is in the order of $n!$). Because of this, most traditional enumeration-based equilibrium seeking techniques (e.g., the well-known Lemke-Howson [79] algorithm) will not be effective, and we have to find alternatives.

One computational approach that shows promise in dealing with the strategy space explosion is the fictitious play algorithm, which is originally proposed by Brown [16] and later adopted by researchers in operations research and computer science in dealing with either centralized or decentralized planning problems. Without going into technical details, we can view fictitious play algorithm as a way for players to learn about how to anticipate other players' responses, so that proper strategy can be selected. The strength of the fictitious play is its simplicity, and it's known that if potential function can be defined for the game in interest, the fictitious play algorithm will converge [89]. Important classes of games that possess such property include games with identical interests (the team game) and a wide variety of congestion games.

Unfortunately, the original fictitious play algorithm has a number of undesirable properties, both theoretically and computationally. First, the equilibrium that the fictitious play algorithm could converge to (if the convergence is possible) is in mixed form, since the convergence results are all established on the belief distribution (which is probabilistic in nature). Second, in each iteration of the fictitious play algorithm, all players need to compute their *best responses* against the current belief distribution, which potentially may contain a big chunk of the original joint strategy space. This implies that the evaluation of best responses (which is based on expected) might be exponential as well.

To address the second issue, Lambert III et al. [76] have introduced the idea of sampling to the evaluation of best responses: instead of evaluating

against all possible combinations from the history in the belief distribution, a small number (in most cases, only *one* sample is needed) of joint strategies will be sampled, and the best response will be computed against these samples. Lambert III et al. [76] proved that this sampled fictitious play will converge in belief to equilibrium for games of identical interests. They then use this result to solve large-scale unconstrained discrete optimization problems as games using sampled fictitious play.

We will adopt the similar sampling idea in our first attempt to solve the MOPTCC game. However, as we are looking to generate recommendations for agents with heterogeneous preferences (represented in the form of payoff function), we will focus on finding *pure strategy equilibria* instead. As the existence of pure strategy equilibrium is not guaranteed in general, and we cannot prove it analytically due to the complexity of the formulation, we propose to use *sampled fictitious play* (SFP) algorithm to identify pure strategy equilibrium if one exists for the MOPTCC game.

3.4.1 Sampled fictitious play algorithm

In Algorithm 1, we define a variant of the SFP algorithm used in solving MOPTCC game. The major new features we implement are: 1) the handling of *infeasible* samples, which based on our earlier definition refer to joint strategies that would result in over-capacitated destination; and 2) focus on identifying pure strategy equilibrium when executing the SFP algorithm.

Algorithm 1 is a simplified skeleton that hides most implementation complexity. To start the algorithm, we first randomly generate joint strategy that is feasible by calling InitialSolutions() in line 3. This initial solution is then used to initiate the history (i.e., the belief distribution). The iteration then begins, in which a feasible joint strategy is to be sampled at the beginning of the iteration in line 7. The tighter the capacity constraint, the more difficult

Algorithm 1: Sampled fictitious play algorithm for MOPTCC games.

```
1  Input: (Γ, k_max)
2  Output: B_NE
3  B ← InitialSolutions()
4  H ← UpdateHistory({}, B)
5  k ← 1
6  while k <= k_max do
7  |   D ← Sample(H, k)
8  |   for each agent i do
9  |   |   Q_{-i} ← AggregateQueues(D_{-i})
10 |   |   (B_i, δ_i) ← BestResponse(Γ, Q_{-i})
11 |   end
12 |   H ← UpdateHistory(H, B)
13 |   if max_i δ_i = 0 then
14 |   |   B_NE ← Append(B_NE, B)
15 |   |   k ← k + 1
16 |   end
17 end
18 Return: B_NE
```

it is in sampling a feasible joint strategy. However, as the initial joint strategy is feasible, we can always find such sample. With a feasible sample joint strategy, we then solve each agent's best response problem as a mathematical program, which is to be defined later. Note that when the best response is computed in line 10, congestions at all destinations (Q_{-i}) are determined by other players' joint strategy (D_{-i}) in line 9. When computing the best response, another information we get is the individual deviation δ_i, which refers to the improvement made by choosing the best response. If δ_i is 0, it implies that player i cannot benefit from unilaterally deviating from the sampled strategy. If $\max_i \delta_i$ is 0, no player can benefit from their deviations, and D is a pure strategy equilibrium. Whenever we find such solution, we will store it in the output set (note that in practice we will store all relevant information such as utility and congestion besides just the equilibrium strategy).

In the next two subsections, we will explain how we generate random initial solutions, how do we compute best responses using mathematical programming approach.

3.4.2 Generating feasible initial solution

The initial solutions are generated using a simple greedy approach in INITIAL-SOLUTIONS(). We detail the used greedy approach as follows:

1. Initialize the congestion $\{Q_d^t\}$ to be 0 for all (t, d) pairs.
2. Choose any player k who doesn't have an itinerary yet.
3. For this agent, randomly choose one destination at a time, assuming that the current congestion is $\{Q_d^t\}$. We use rejection-base sampling: choosing all unvisited destinations with equal chance, and if the chosen destination d is at its capacity at the estimated arrival time t, another destination will be drawn.
4. For each agent, we would artificially reduce its time budget by 50%. This is to approximately factor in the potential impact of this agent's decision on other agents' increased wait time. Based on our computational experience, this damping factor can significantly improve the likelihood of us getting feasible decisions. Depending on the number of players and the problem parameters, different damping factors might be appropriate. Our computational study on 5-agent instances is summarized in Figure 3.5 in the Appendix.
5. After we have exhausted player k's time budget, we fix player k's itinerary and update $\{Q_d^t\}$.
6. If the set of free players is not empty, go to Step 2 and repeat.

We first generate initial solutions without artificially reducing player's time budget, but we soon find out that for problems with tight capacity constraints, initial solutions generated with all players spending most of their time budgets will result in joint solutions that are almost impossible to improve upon. In these cases, time budget reduction in Step 4 is shown to be very effective in

improving the efficiency of the algorithm (intuitively speaking, Step 4 allows us to reserve buffer time in the schedule generated).

3.4.3 Computing best responses

An agent's best response in the MOPTCC game can be computed using an ILP model very similar to the one introduced in Section 3.3.2, constraints (5.1)–(3.9). There are two major differences:

- The k index which represents different agent identities can be dropped. E.g., the decision variable will become only x_{ij}^t.
- All other agents' chosen strategies, which are taken from the sampled joint strategy, will collectively decide the background queue length. We define Q_{dt}^{input} to be the background queue length built up by other agents at node d in time t.

Most constraints will stay the same except constraints (3.5) and (3.7). These two sets of constraints will be rewritten as:

$$Q_d^t = Q_{dt}^{\text{input}} + \sum_{i=1}^{n} x_{id}^{t-t_{id}} \,, \qquad \forall d, t, \tag{3.15}$$

$$\sum_{t=1}^{T} \sum_{i=1}^{n} \left(t + t_{id} + Q_{d,t+t_{id}}^{\text{input}} + v_d \right) x_{id}^t = \sum_{t=1}^{T} t \sum_{j=1}^{n} x_{dj}^t \,, \qquad \forall d. \tag{3.16}$$

Since Q_{dt}^{input} is provided as problem data, the constraint is already linear and requires no linearization. Together with the fact that index k is dropped, the problem becomes much more tractable, and can be solved reasonably fast in our computational study.

3.5 Computational Experiments

In this section, we evaluate how effective our SFP variant is in finding pure strategy equilibrium. All the instances used in this section are generated randomly using our MOPTCC instance generator. The run times reported below are measured on machines running 3.16GHz Intel Xeon CPU X5460 with 16GB RAM.

3.5.1 Instance Generation

Numerical instances in our computational study are characterized by the tuple: (m, n, T, *type*), where m, n, T denote the number of agents, the number of nodes, and the number of time periods respectively. The final parameter, *type*, refers to the tightness of the instance, which can be either *loose* or *tight*. The tightness of the instance affects how node capacities (Q_d^{max}) are drawn. For *loose* instances, capacities are drawn from discrete uniform distribution between $(p \cdot m)$ and m; for *tight* instances, capacities are drawn from discrete uniform distribution between 1 and $(p \cdot m)$. In both instances, p is set to a constant between 0 and 1. In all our experiments, p is set to 0.5.

The utility value for agent k to visit node i in time t, s_{ik}^t, is assumed to be uniformly distributed between 1 to 5. The only exceptions are the start and end nodes, s_{1k}^t and s_{nk}^t, whose values are both set to be 5. Finally, all travel times (t_{ij} between nodes i and j) and service times (v_d for node d) are set to 1 for simplicity.

For our computational experiments presented in this section, we generate six categories of instances, with parameters $m \in \{2, 5, 8\}$, $n = 10$, $T = 10$, and $type \in \{loose, tight\}$. 25 random instances are generated for each category. To avoid being trapped in unpromising joint strategy subspace and increase the likelihood of finding pure strategy equilibrium, each instance is independently solved for 20 times, each time with a randomly generated initial solution

(following steps introduced in Section 3.4.2).

3.5.2 Numerical Results

(m, n, T)	$(2, 10, 10)$	$(2, 10, 10)$	$(5, 10, 10)$	$(5, 10, 10)$	$(8, 10, 10)$	$(8, 10, 10)$
Instance Type	tight	loose	tight	loose	tight	loose
Total instances	25	25	25	25	25	25
Success rate of finding equilibria	100%	100%	100%	100%	100%	100%
Avg. # of equilibria found	267.88	256.08	79.12	66.92	44.96	20.35
Avg. # of distinct equilibria found	17.88	17.76	61.4	52.64	31.68	14.00
Best equilibrium / Best feasible	100%	100%	99.94%	99.97%	98.71%	98.98%
Avg. computational time(s)	114.24	172.43	337.08	439.14	3466.80	9913.22

Table 3.3: Results for equilibria found on per-instance bases. For smaller instances (2-agent and 5-agent cases), each instance is solved by executing SFP algorithm for 20 iterations. For larger instances (8-agent), we execute SFP algorithm for 50 iterations. Note that all mentions of equilibria refer to pure strategy equilibria.

For smaller instances (2-agent and 5-agent cases), each instance is solved by executing SFP algorithm for 20 iterations. For larger instances (8-agent), we execute SFP algorithm for 50 iterations so that better solution might be found. The performance of our SFP variant in finding pure strategy equilibria is summarized in Table 3.3. As we can see from Table 3.3, SFP can identify large amount of high-quality pure strategy equilibria for smaller instances (2-agent and 5-agent cases) very quickly. When the problem expands to have 8 agents, we can see that it's drastically more difficult to find pure strategy equilibria (measured by both the computational time and the number of pure strategy equilibria found). Another interesting finding is that *loose* instances are much more difficult to solve than *tight* instances for all numbers of agents, probably because of greater degree of freedom we have (number of feasible joint strategies would be much larger when the capacity constraints are loose, and agents would have a more difficult time in producing coordinated actions as a result). We also compute the ratio between the best pure strategy equilibrium and the best feasible solution found for each instance. We can see that the ratio is reasonably high for all instances. This is an encouraging computational

result, as it's well known that the adhering to Nash equilibria might cause significant drop in social welfare (e.g., see Roughgarden and Tardos [107]'s work on quantifying the *price of anarchy* in the routing domain, i.e., the sacrifice one needs to endure for implementing Nash equilibrium solution).

Another interesting way to visualize the growing of computational complexity in finding pure strategy equilibrium is to plot the histogram of agents' maximal deviations in all iterations. Intuitively speaking, if we have lots of cases with 0 deviation, it implies that it's easy to identify pure strategy equilibrium (when the maximal deviation is 0 among all players in any iteration, it implies that a pure strategy equilibrium has been found, since no agent can benefit from deviating unilaterally). Not surprisingly, with the number of agents increasing, the performance of the algorithm takes a hit and the frequency of zero deviation should decrease. The plots in Figure 3.1 confirm our speculation. Besides steady decrease in zero-deviation cases, the distribution of deviations also gradually shifts to the right hand side, indicating greater difficulty in reaching coordinated outcomes. The instances with *loose* capacity constraints also consistently have *fatter* tails to the right, indicating that it's more difficult to identify equilibrium in general for *loose* instances.

In terms of utility value improvement, the SFP algorithm progresses very quickly. In Figure 3.2 we can observe the average progress of the SFP algorithm over the iteration, with error bars at +1 and -1 standard deviation. As illustrated in Figure 3.2, we can see that most of the progress is made at the early iterations, after which the algorithm settles down quickly, with stable values and very low standard deviations. This execution pattern is also consistent with prior research in using SFP algorithm for large-scale discrete optimization (e.g., see [49]). Do note that Figure 3.2 is relative; when we have larger number of agents, it's expected that more iterations would be necessary, however, the pattern improvement would resemble what is illustrated here.

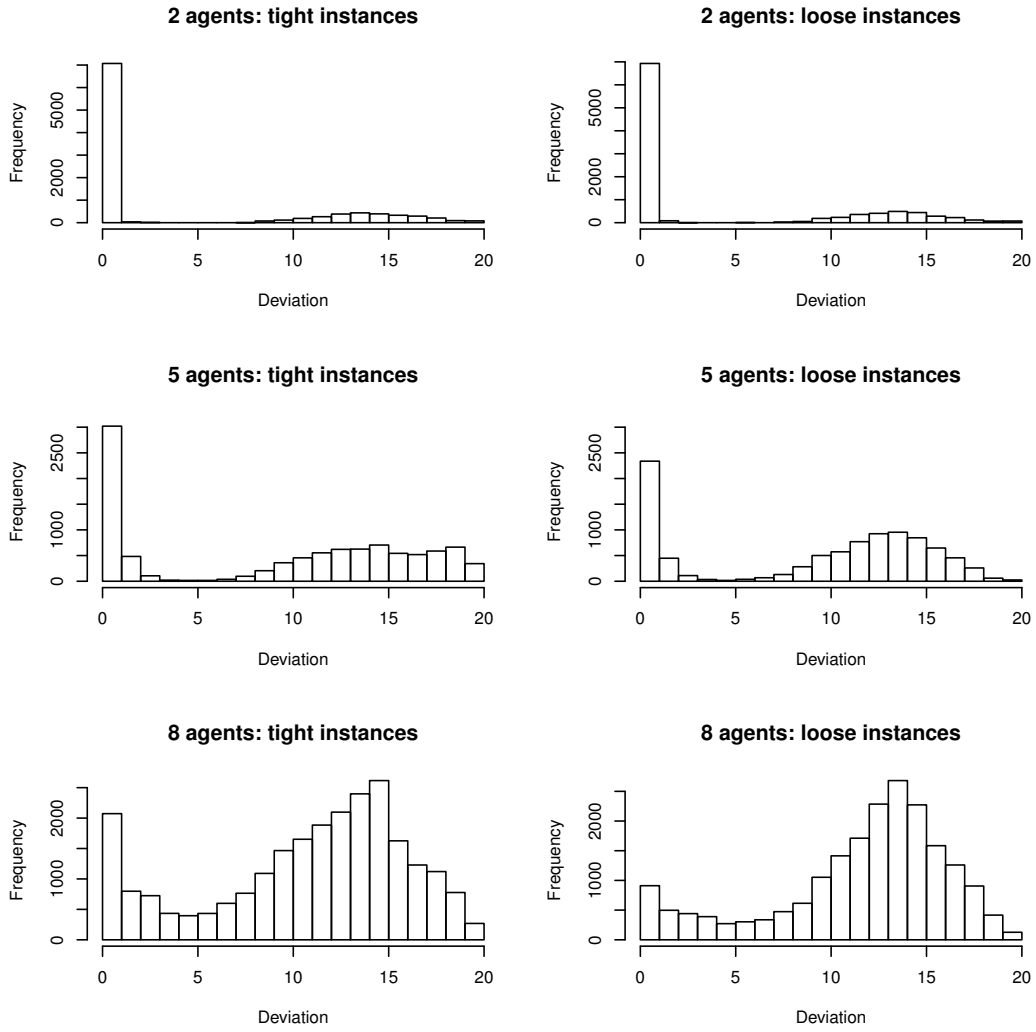


Figure 3.1: Max deviations of 2-agent, 5-agent, and 8-agent instances.

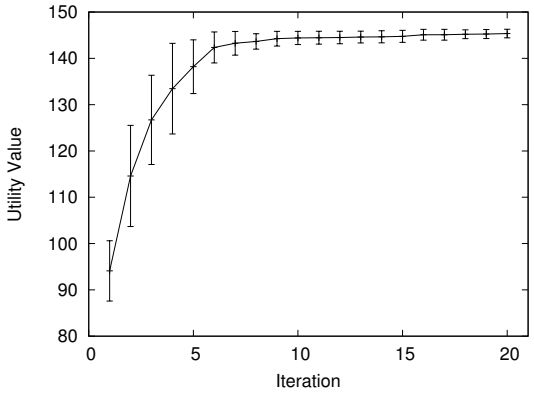


Figure 3.2: The average progress of SFP algorithm over iterations for a sample instance with random restarts (the error bar is one standard deviation over all random restarts of this instance).

3.5.3 Comparison Against Baseline

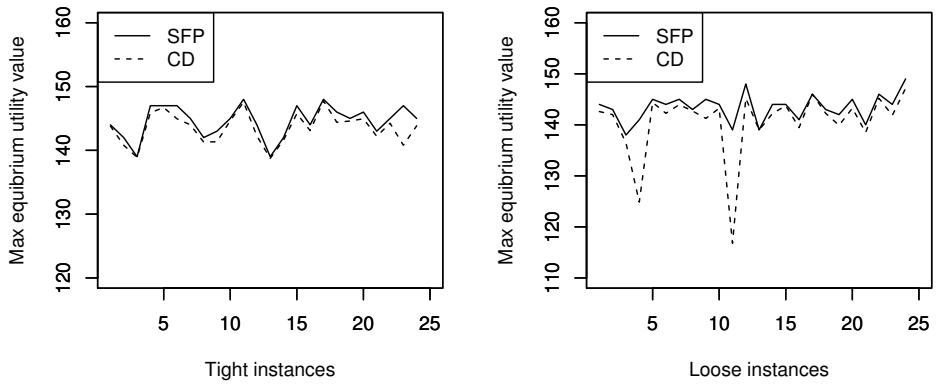


Figure 3.3: Maximal equilibrium utility value for 5-agent instances: 25 tight instances (left) and 25 loose instances (right). X-axis denotes individual instances. Y-axis represents the maximum utility value of the equilibrium solutions discovered for that instance.

To understand whether the SFP algorithm we use for MOPTCC indeed has its merits, we compare it against a popular baseline algorithm called the Coordinate Descent (CD) algorithm (similar CD algorithm has also been compared to the SFP algorithm in other domains, such as the coordinated traffic signal control [29]). The CD algorithm is a simple yet effective algorithm for solving large-scale discrete optimization (despite its simplicity, it's shown to work well in practice and in theory [127]). In the MOPTCC domain, it executes as follows: the CD algorithm would start from any player, for whom a best response against the current solution is computed and this best response will then be used to replace this player's current strategy. The CD algorithm then move on to the next player and repeat the above procedures. The CD algorithm would terminate either after no further improvement can be made after we have iterated through all agents or the computational time is up.

The comparison between the CD algorithm and the SFP algorithm is conducted for our 5-agent instances, with results plotted in Figure 3.3. From the figure we can see that the SFP algorithm is at least as good as the CD algorithm, and for some instances the SFP algorithm managed to greatly out-

perform the CD algorithm.

Finally, we extend our experiments to 8-agent and 10-agent instances, running 50 iterations of SFP algorithm. As shown in Figure 3.4, the number of iterations needed to find a equilibrium increases with the number of agents. In the 2-agent and 5-agent cases, SFP is able to find equilibrium within 20 iterations. For the 8-agent and 10 agent cases, equilibrium is seldom reached within the first 20 iterations and most equilibria are found by the second half of 50 iterations.

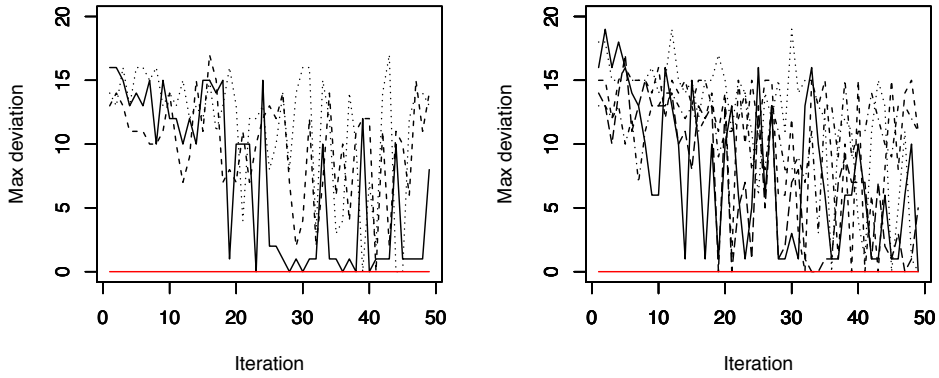


Figure 3.4: Maximal deviations for selected 8-agent (left) and 10-agent (right) instances with random restarts. Each line represents a different random restart. X-axis denotes iterations, Y-axis denotes the maximal deviation $\max_i \delta_i$. A deviation δ_i for player i can be viewed as the utility improvement made by choosing the best response. If $\max_i \delta_i$ is 0, a pure strategy equilibrium is found, i.e., no player can benefit from unilateral deviation. Setting: (m,n,T) = (m,10,10).

3.6 Summary

In this chapter, we introduce a new variant of the OP, referred as Multi-agent Orienteering Problem with Time-dependent Capacity Constraints (MOPTCC). It can be used as the starting point for modeling many combinatorial optimization problems which involve time-dependent capacity constraints; e.g., it can be applied to crowd management in leisure settings, where controlling queue lengths for various attractions is of vital concern to the operator.

To this end, we first propose a centralized model using integer linear programming formulation. Due to the distributed nature of the problem, we reformulate it in the game-theoretic framework, and we propose to use the sampled fictitious play algorithm (SFP) which is shown to be computational efficient in identifying pure strategy equilibrium. By introducing *rejection-base sampling* in the fictitious play iteration, we are able to deal with computational intractability and also the *feasibility* requirement we impose on the MOPTCC game, which is to require all capacity constraints be observed at all times. Our initial computational experiments show great promises, as we are able to find pure strategy equilibrium in all the randomly generated instances for 2-agent, 5-agent, and 8-agent MOPTCC games.

3.7 Appendix

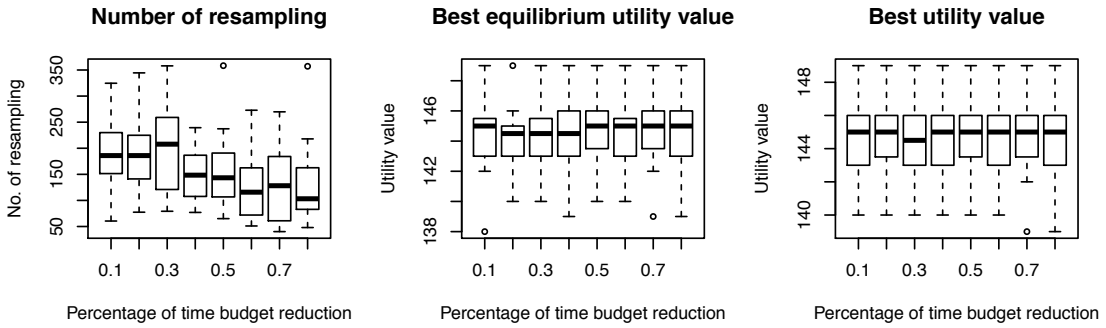


Figure 3.5: Comparison for different time budget discount ratios for 5-agent instances: (m,n,T) = (5,10,10).

The impact of using different time budget discount ratios is illustrated in Figure 3.5. We can clearly see that more aggressive time budget reduction directly leads to fewer number of re-samplings. The best utility value from discovered equilibrium solutions also improves with higher time budget discount ratios.

Chapter 4

Mobile Crowdsourcing

4.1 Overview

Mobile crowdsourcing, where citizen volunteers are incentivized to perform *location-specific* tasks, has recently attracted strong commercial interest. Most of the existing academic research and practical deployments of mobile crowdsourcing presently employ a *pull-based* model, that individual workers independently search through, and select from, the corpus of available tasks (often with built-in proximity filters that enable them to identify tasks that are close to their current location). As pointed out by [91], such pull-based embodiments of mobile crowdsourcing, suffer from the phenomenon of *super agents*, i.e., a small percentage of crowd workers who perform the majority of tasks[1]. Domination at such high level is undesirable, as many ordinary crowd workers might drop out as a result of not having enough tasks, reducing the worker pool and thus the peak capacity of the mobile crowdsourcing platform. By examining empirical data, Musthag and Ganesan conclude that the major difference between ordinary worker agents and super agents is the latter's ability in planning better routes and choosing tasks that fit their routes best.

Absent from the pull-based approaches are the fundamental concepts of (i)

[1]Based on their study, 10% of most active users complete 80% of all completed tasks.

task coordination, where the crowdsourcing platform proactively recommends tasks to each crowd-worker in a globally coordinated way, with the objective of improving the acceptance and completion rates of such tasks, and (ii) *predictive crowd-tasking*, where the recommendation or selection of tasks is not performed myopically (based on just the current location context of the workers), but instead exploits an individual's *movement trajectory* over a longer time horizon (e.g., an entire day).

In this chapter, we investigate a push based mobile crowdsourcing framework, where tasks are proactively recommended by the platform taking into account the uncertain movement patterns (as the worker may travel on different routes on different days). Such uncertainty presents a technical challenge to effective task recommendation under the push-based model, as the recommendation strategy must still seek to recommend individual workers tasks that are likely to lie along their routine commuting routes, while accounting for and trying to mitigate the likelihood that their actual routes (on a specific day) might make it infeasible for them to perform one or more of the recommended tasks.

4.2 Literature Review

Mobile Crowdsourcing: Recent approaches such as [2], [68], [67], and [61] have focused on a particular class of mobile crowdsourcing, called participatory sensing, with a goal of maximizing the number of assigned tasks based on agents' current locations. For example, Kazemi and Shahabi [68] developed a centralized allocation algorithm focused on maximizing an agent's set of allocated tasks, while satisfying a proximity constraint (which implied that some tasks could be potentially performed by multiple agents). Sadilek et al. [109] studied a general class of problems called crowdphysics where tasks requires people to collaborate in a synchronized space and time. They reduced the

problem into a graph planning problem and proposed two methods: global coordination using shortest-path algorithm and opportunistic routing based on the ranking of the time-stamped locations.

Researchers have also investigated the impact of incentives on task assignment/selection behavior in mobile crowdsourcing. [108] investigated the relative impacts of two different incentive mechanisms—micro-payments vs. weighted lotteries, on the task acceptance and completion rates of mobile agents (crowd workers). The MSensing approach by [142] focuses on incentive design for mobile crowdsourcing tasks, employing either a Stackelberg game model (for scenarios where the reward is specified by the platform) or an auction-based agent selection mechanism (where the reward is determined through competitive bidding) to model the dynamics between the pricing of individual tasks and the willingness of each agent (task worker) to perform that task. However, this approach either considers a single task in isolation or employs a cost function (for each agent) that fails to account for the regular movement trajectory of each agent. Other work on task assignment in mobile crowdsourcing addresses questions of reliability – for example, recently Boutsis and Kalogeraki [14] seek to maximize the reliability of crowdsourcing tasks by selecting agents, subject to a task completion latency constraint.

In all of these approaches, the feasibility of task assignment is defined by the task's proximity to the agent's current location. In contrast, our focus is on a coordinated mechanism for *task recommendation* that operates over a longer time horizon (e.g., a day). We also explicitly model the inherent stochasticity (uncertainty) in individual agent trajectories and seek to minimize their task-related detours.

Multi-agent Task Allocation: The planning of agent's route to perform tasks that maximizes utility can be seen as OP. The closest variant of OP to our problem is TOP. However, to the best of our knowledge, none of the previous research on TOP actually considers the incorporation of agent-specific lo-

cation information, namely the probability distribution of routine routes taken by agents. One other variant of OP is the Multi-Period Orienteering Problem with Multiple Time Windows (MuPOPTW) [125]. In MuPOPTW, sales representatives need to visit a list of mandatory customers on a regular basis, while non-mandatory customers located nearby should also be considered and integrated into the current customer tours. While one may view the set of mandatory customers as the nodes on the routine routes and non-mandatory customers as the tasks in our problem, there is no predefined visiting sequence for the mandatory customers and the stochasticity of agent routine routes is not captured in that model.

There is also a large body of work focusing on solving pickup and delivery problems [6] or dial-a-ride problems [31]; both are similar to the mobile task assignment problem we intend to study in this chapter. However, all existing models in these areas assume that a team of full-time staffs are employed and can carry out any specific order. Such assumptions are in direct conflict with the requirement that individual mobile workers should follow their respective predicted trajectories, and they could only spend limited amount of time deviating from their routes to perform assigned tasks. As such, none of existing model can be utilized straightforwardly.

Real-world Implementations: There are already several notable systems that have demonstrated values of mobile crowdsourcing. Systems such as mCrowd [141], Twitch crowdsourcing [129] and Slide-to-X [126] provide platforms to enable various crowdsourcing tasks. Commercially, the most visible examples are Uber-like services that provide a platform for willing "part-time drivers" to offer rides to passengers. Several other applications enable smart citizen monitoring by using crowdsourcing for, e.g., detecting potholes [86], mapping noise [102] and pollution [117] in urban areas, and monitoring road traffic [121, 88]. Several instances of paid mobile crowdsourcing start-ups have

also emerged commercially including FieldAgent[2], GigWalk[3], and NeighborFavor[4]. They pay workers a few dollars for micro-tasks such as price checks, product placement checks in stores, and location-aware surveys. Our real-world deployed system focuses on two core problems of all these systems: (1) assigning or recommending location-specific tasks to workers, and (2) exploring the relationship between worker behavior (e.g., accept/reject a task, time taken to choose the right task) and attributes of the tasks (e.g., rewards and locations).

Task Acceptance & Completion Behavior: Limited studies have explored the relationship between task attributes and worker behavior, especially for the crowdsourced execution of location-dependent tasks. Wang et al. [136] studied the task completion times of online tasks (posted on Amazon Mechanical Turk), and established a power-law relationship between task completion times and task-related features, such as the type of the task, the task price and the day the task was posted. Alt et al. [2] used an independently developed mobile crowdsourcing platform to discover a variety of worker preferences, including preference for performing tasks before and after business hours or involving relatively simple chores (e.g., taking pictures). More recently, Thebault-Spieker et al. [120] conducted studies on the relationship between task pricing and location, at city-scale, and showed that workers preferred to perform tasks with lower detours and that were outside economically-disadvantaged areas.

In contrast to this body of academic work on task recommendation and experimental studies on city-scale worker behavior, our focus is to empirically investigate the human dynamics of mobile crowdsourcing in an urban campus-like setting, and to uncover key behavioral differences arising from the use of push vs. pull models for task recommendation and selection.

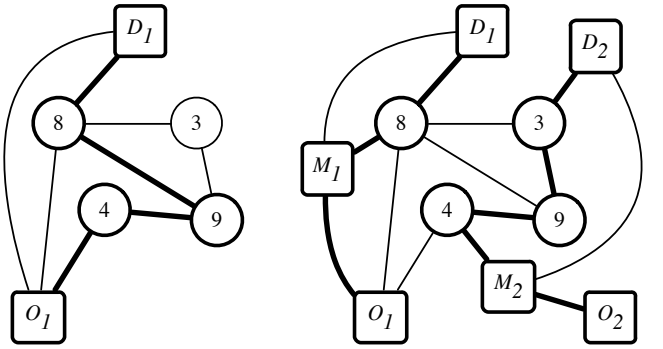
[2]http://www.fieldagent.net
[3]http://www.gigwalk.com
[4]http://www.favordelivery.com

4.3 Problem Formulation

As mentioned in Section 4.1, we are interested in creating models for recommending tasks to mobile crowd workers with known routine route distributions, so as to maximize the expected total rewards collected by all agents. The major constraints in task recommendation for each individual agent are the maximal amount of detour time allowed, the routine route that specifies the list of nodes each agent needs to traverse in order, and the probability distribution over a collection of routine routes (if this agent has multiple routine routes).

Mathematically speaking, this can be viewed as a specialized routing problem with time budget constraint and routine route requirement, and can be modeled as a variant of the OP. The classical OP can be seen in Figure 4.1a. In the classical OP, usually the requirement is for an agent to begin his trip from a given origin O_1, and to end at a given destination D_1. An agent can visit any node in-between O_1 and D_1 and incur corresponding link travel costs; yet he cannot exhaust his time budget before reaching D_1. The objective is for an individual agent to maximize value collected at visited nodes (each node comes with different value). The OP variant for the mobile crowdsourcing problem with deterministic routine route (for easy explanation) can be seen in Figure 4.1b. There are two major differences when compared to the classical OP in Figure 4.1a: 1) there are multiple agents involved, and the planner aims to optimize the sum of all agent's values; and 2) for each agent, a routine route is specified, and its partial order needs to be followed (e.g., in Figure 4.1b, agent 1 needs to follow $O_1 - M_1 - D_1$, while agent 2 needs to follow $O_2 - M_2 - D_2$). In both cases, all nodes can be visited at most once.

To model uncertainty on agent's routine route, we assume that each agent may take one of multiple routes with known probability distribution. The objective is to optimize the sum of each agent's *expected* collected values. For the rest of the section, we will introduce an integer linear programming (ILP)



(a) Classical OP. (b) OP variant for mobile crowdsourcing.

Figure 4.1: Illustrations of OP variants.

formulation for the problem of mobile crowdsourcing with routine route uncertainty. Note that our solution approach generates task recommendations for agents prior to the realization of actual routes taken. Such approach requires no agent inputs, and thus can reduce agent's cognitive burden in picking tasks; such approach is thus superior to the alternative where an agent has to first specify a deterministic route, before receiving task recommendation (this was the model in [26]).

4.3.1 Mathematical Model

Let N be the set containing both the routine nodes and the task nodes (denoted as N_t), and for all pairs (i, j), where $i, j \in N$ and $i \neq j$, let t_{ij} be the corresponding travel time to move from i to j. Let K be the set of agents, and let M_k be the set of agent k's routine routes. For each route $m \in M_k$, let β_k^m be the probability that agent k would use route m, R_k^m be the collection of all nodes in route m, o_k^m be the origin, d_k^m be the destination, and p_{ik}^m be the visit order for node $i \in R_k^m$. For each task $i \in N_t$, let s_i be its reward, and e_i be its required execution time. The upper bound of the total detour time along route m for agent k is b_k^m.

We have the following four sets of decision variables:

Note that the first set of decision variables, y_{ik}, is planner's recommendation

Variables	Descriptions
$y_{ik} \in \{0,1\}$	set to 1 when task i is recommended to agent k.
$x^m_{ijk} \in \{0,1\}$	set to 1 when agent k moves from nodes i to j when the realized routine route is m.
$v_i \in \{0,1\}$	set to 1 if request i is not assigned to any provider.
$u^m_{ik} \in \{0,\ldots,N\}$	indicate the visit order of node i for agent k, when the realized routine route is m.
$z^m_{ik} \in \{0,1\}$	set to 1 when agent k fails to complete recommended task i, when the realized routine route is m; if task i is not recommended to agent k in the first place, z^m_{ik} is set to 0 for all $m \in M_k$.

decision; while the rest of decision variables are used for evaluating the outcome of the recommendation under different routine route realizations for each agent.

The objective of the ILP formulation is to maximize expected total rewards earned by all agents, considering uncertainties over their routine routes. The quantity $y_{ik}(1 - \sum_{m \in M_k} \beta^m_k \cdot z^m_{ik})$ refers to the probability that a task i can be finished, if it is recommended to agent k (i.e., if y_{ik} is set to 1). Hence, the objective is to:

$$\max \sum_{i \in N_t} s_i \sum_{k \in K} y_{ik} \left(1 - \sum_{m \in M_k} \beta^m_k \cdot z^m_{ik}\right). \tag{4.1}$$

The above objective function is not linear, and in the next subsection, we will propose a linearized model.

Constraint (4.2) ensures that each task is recommended to at most one agent.

$$\sum_{k \in K} y_{ik} \leqslant 1, \qquad \forall i \in N_t. \tag{4.2}$$

All other constraints are at agent-route level (k, m), i.e., for each agent, $k \in K$, and for each of his routine route realization, $m \in M_k$, the same set of constraints applies. These constraints are explained as follows.

The first group of constraints ensures that flows are consistent at all nodes.

In particular, (4.3) specifies that inflow and outflow at any node d must be balanced (except for the origin and destination nodes). (4.4) specifies that all routine nodes must be visited exactly once, while all other nodes can be visited by at most once.

$$\sum_{i \in N} x_{idk}^m = \sum_{j \in N} x_{djk}^m, \qquad \forall d \in N \backslash \{o_k^m, d_k^m\}, \tag{4.3}$$

$$\begin{cases} \sum_{j \in N} x_{djk}^m \leqslant 1, & \forall d \in N \backslash R_k^m, \\ \sum_{j \in N} x_{djk}^m = 1, & \forall d \in R_k^m \backslash \{d_k^m\}, \\ \sum_{i \in N} x_{idk}^m = 1, & d = d_k^m. \end{cases} \tag{4.4}$$

The time budget constraint for each routine route is enforced in (4.5).

$$\sum_{i \in N} \sum_{j \in N} x_{ijk}^m \left(t_{ij} + e_i\right) \leqslant b_k^m. \tag{4.5}$$

The following group of constraints produces visit orders (u_{ik}^m) from flows (x_{ijk}^m), and ensures that all nodes in the routine route m are visited according to the given sequence p_{nk}^m. In particular, (4.6) states that the visit order of the origin node must be 1; (4.7) states that if j is visited immediately after i (i.e., $x_{ijk}^m = 1$), the visit order of j should be *at least* 1 more than the visit order of i (i.e., $u_{jk}^m \geq (u_{ik}^m + 1)$).

$$u_{ik}^m = 1, \qquad i = o_k^m, \tag{4.6}$$

$$(u_{ik}^m + 1) - u_{jk}^m \leqslant N\,(1 - x_{ijk}^m), \qquad \forall i, j \in N. \tag{4.7}$$

(4.8) states that the partial order between any pair of nodes in the routine route must be preserved.

$$u_{ik}^m - u_{jk}^m \geqslant p_{ik}^m - p_{jk}^m, \qquad \forall i, j \in R_k^m \;\&\; p_{ik}^m > p_{jk}^m. \tag{4.8}$$

Finally, (4.9) extracts whether a task node is bypassed ($z_{ik}^m = 1$) from the flow decision (x_{ijk}^m).

$$z_{ik}^m \geqslant 1 - \sum_{j \in N} x_{ijk}^m, \qquad \forall i \in N_t. \tag{4.9}$$

Linearization: As noted earlier, the objective function (4.1) is nonlinear, as it includes multiplicative terms composed of y_{ik} and z_{ik}^m, both of which are decision variables. We would linearize (4.1) by introducing δ_{ik}^m to replace z_{ik}^m:

- δ_{ik}^m is set to 1 if task i is recommended to agent k, yet cannot be completed when the realized routine route is m, and 0 otherwise. In other words, $\delta_{ik}^m = y_{ik} \cdot z_{ik}^m$.

With δ_{ik}^m, we can rewrite (4.1) as:

$$\max \sum_{i \in N_t} s_i \sum_{k \in K} \left(y_{ik} - \sum_{m \in M_k} \beta_k^m \cdot \delta_{ik}^m \right). \tag{4.10}$$

To characterize δ_{ik}^m, we would rewrite (4.9) for all (k, m) as:

$$\delta_{ik}^m \geqslant y_{ik} - \sum_{j \in N} x_{ijk}^m, \qquad \forall i \in N_t. \tag{4.11}$$

With the above modifications, the formulation is now linear, and can be solved as an ILP using standard commercial solvers such as CPLEX.

In this ILP formulation, we assume that each task is recommended to at most one agent. As agents might choose to ignore the task recommendation for various reasons, such single-agent recommendation scheme might not be robust in terms of task completion. In the following subsection, a straightforward extension is introduced to allow us to request for more than one agents to be considered for each task.

4.3.2 The Multi-Coverage Extension

As noted in section 4.1, an agent is more likely to accept a task if it lies along his routine route(s). In other words, the agent may choose to skip a recommended task if his realized routine route forbids him from completing the task (the computed recommendation plan maximizes expected reward, and some task recommendation might not be feasible for a particular route). To increase the robustness of the recommendation plan without making the model much more complicated, we require that each chosen task be recommended to at least η agents, where η is a given parameter. In such robust planning scheme, we will either recommend η agents to a task, or not assign the task if this is not possible. To achieve this, we would introduce two additional classes of decision variables:

- $v_i \in \{0, 1\}$: set to 1 if this task is recommended to η agents, 0 otherwise.
- $w_{ik}^m \in \{0, 1\}$: variable for linearization, representing $v_i \cdot \delta_{ik}^m$.

To realize this extension, the objective function (4.10) is rewritten as (4.12), and the assignment constraint (4.2) is rewritten as (4.13). (4.11) also needs to be included. To linearize $(v_i \cdot \delta_{ik}^m)$ with w_{ik}^m, which appears in the modified objective function (4.12), we need the set of constraints (4.14). Finally, to ensure that we only assign task i to agent k when task i can at least be completed by one of agent k's route, we also need to include constraint (4.15).

In other words, for every tuple (i, k), if $y_{ik} = 1$, $\delta_{ik}^m = 0$ for at least one $m \in M_k$.

$$\max \sum_{i \in N_t} s_i \left(v_i - \sum_{k \in K} \sum_{m \in M_k} \frac{\beta_k^m \cdot w_{ik}^m}{\eta} \right), \tag{4.12}$$

$$\begin{cases} \sum_{k \in K} y_{ik} = \eta \cdot v_i, & \forall i \in N_t, \\ y_{ik} \leqslant v_i, & \forall i \in N_t,\ k \in K, \end{cases} \tag{4.13}$$

$$\begin{cases} w_{ik}^m \leqslant v_i, \\ w_{ik}^m \leqslant \delta_{ik}^m, & \forall i \in N_t,\ k \in K,\ m \in M_k, \\ w_{ik}^m \geqslant v_i + \delta_{ik}^m - 1, \end{cases} \tag{4.14}$$

$$\sum_{m \in M_k} \delta_{ik}^m \leq y_{ik}(|M_k| - 1), \forall i \in N_t,\ k \in K. \tag{4.15}$$

4.3.3 Scalability of the Model

Although the above ILP model can be solved exactly using CPLEX, it will not scale to even moderate sizes. Yet we introduce this exact model for two purposes: 1) to provide the problem structure for use by our proposed Lagrangian relaxation heuristic (to be described in Section 4); and 2) to provide an experimental benchmark for the heuristic.

To make the ILP model more scalable, we propose the following performance-boosting preprocessing procedures on the data without affecting the optimality of the model.

- For each subproblem pair (k, m), we remove all tasks that cannot be reached within the given detour time. Therefore, instead of using global node sets N and N_t in (k, m)-level constraints, we would use (k, m)-specific node sets N_k^m and N_{tk}^m.

- To avoid having to solve shortest path routing problems explicitly in the ILP model, we pre-compute shortest path distances for all origins and destinations and store them in a N-by-N matrix. With this matrix, we

can further eliminate all non-essential nodes from N_k^m. More specifically, only nodes along the routine route m and feasible task nodes N_{tk}^m need to be included in N_k^m.

With these preprocessing steps, we are now ready to formally introduce the Lagrangian relaxation heuristic for the ILP.

4.4 Solution Approaches

In this section, we present a LR-based heuristic for our problem, and show that it is computationally efficient in producing high quality solutions.

4.4.1 Lagrangian Relaxation

In our ILP formulation, (4.11) is the set of complicating constraints that couples all subproblems together; therefore, we choose to dualize this set of constraints. Following the convention in the standard LR literature, we have Lagrangian multipliers $\boldsymbol{\lambda} = \{\ldots, \lambda_{ik}^m, \ldots\}$ associated with each of the constraints in (4.11) (indexed as (k, m, i)), and convert the objective function to be a *minimization* function. Thus, we have the following dual problem $L(\boldsymbol{\lambda})$:

$$\min \sum_{i \in N_t} s_i \left(\sum_{k \in K} \sum_{m \in M_k} \frac{\beta_k^m \cdot w_{ik}^m}{\eta} - v_i \right) \tag{4.16}$$

$$+ \sum_{i \in N_t} \sum_{k \in K} \sum_{m \in M_k} \lambda_{ik}^m \left(y_{ik} - \delta_{ik}^m - \sum_{j \in N} x_{ijk}^m \right). \tag{4.17}$$

All constraints except (4.11) remain the same. However, by observing the problem structure, we can further decompose $L(\boldsymbol{\lambda})$ into two different classes of subproblems. The first subproblem class is the *assignment subproblem*, which decides how tasks should be recommended to individual agents (involves only y_{ik}); the second subproblem class is the *routing subproblem*, which finds the

exact node visit sequence for each (k, m) tuple (involves only x^m_{ijk}). As subproblems can be solved independently, this decomposition can further improve the efficiency of our LR algorithm.

There is only one assignment subproblem in our formulation, we denote it as $g(\boldsymbol{\lambda})$ and define it as:

$$\min \sum_{i \in N_t} s_i \left(\sum_{k \in K} \sum_{m \in M_k} \frac{\beta^m_k \cdot w^m_{ik}}{\eta} - v_i \right) + \sum_{i \in N_t} \sum_{k \in K} \sum_{m \in M_k} \lambda^m_{ik} \left(y_{ik} - \delta^m_{ik} \right), \quad (4.18)$$

$$\delta^m_{ik} \leqslant y_{ik}, \ \ \forall i \in N_t, m \in M, k \in K, \quad (4.19)$$

together with constraints (4.13) and (4.14). (4.19) is included to further tighten this subproblem such that incompletion penalty will only be imposed if the task is recommended.

The routing subproblem is defined for each (k, m) tuple; therefore, the total number of routing subproblems is $\sum_{k \in K} |M_k|$. For each (k, m) tuple, let $f^m_k(\boldsymbol{\lambda}^m_k)$ be the corresponding routing subproblem, where

$$\boldsymbol{\lambda}^m_k = \{\ldots, \lambda^m_{i-1,k}, \lambda^m_{i,k}, \lambda^m_{i+1,k}, \ldots\},$$

and $f^m_k(\boldsymbol{\lambda}^m_k)$ is defined to optimize:

$$\min - \sum_{i \in N_t} \lambda^m_{ik} \sum_{j \in N} x^m_{ijk}, \quad (4.20)$$

subject to constraints (4.3) – (4.8).

The Lagrangian dual problem is solved by using a standard subgradient descent algorithm, detailed in section 2.2.2. Given a $\boldsymbol{\lambda}$ vector, we solve all dual subproblems; obtain the dual objective function value, and use a primal extraction procedure to obtain primal objective function value.

The dual objective function value $L(\boldsymbol{\lambda}_t)$ of our problem can be computed by simply summing up all objective values from the subproblems, i.e., (4.18)

and (4.20):

$$L(\boldsymbol{\lambda}_t) = g(\boldsymbol{\lambda}_t) + \sum_{k \in K} \sum_{m \in M_k} f_k^m(\boldsymbol{\lambda}_{k,t}^m). \tag{4.21}$$

The corresponding primal solution can be obtained by inserting the current routing dual solutions $\{x_{ijk}^m\}$ back into the original problem, i.e., optimizing (4.12) subject to the same set of primal constraints (4.11) and (4.13) – (4.15).

The Lagrangian multipliers$\lambda_{ik,t+1}^m$ and stepsize α_t are updated iteratively according to the functions below:

$$\lambda_{ik,t+1}^m := \lambda_{ik,t}^m + \alpha_t(y_{ik} - \delta_{ik}^m - \sum_{j \in N} x_{ijk}^m), \tag{4.22}$$

$$\alpha_t = \frac{\mu_t \left(F^* - L(\boldsymbol{\lambda}_t)\right)}{\sum_{i \in N_t, k \in K, m \in M_k} \left(y_{ik} - \delta_{ik}^m - \sum_{j \in N} x_{ijk}^m\right)^2}\ , \tag{4.23}$$

4.4.2 Speeding Up LR Implementation

The LR subproblems are independent of each other, as such, they can be solved in parallel. The performance of our LR heuristic thus depends on how subproblems are solved (slowest subproblem dictates overall solution time). The baseline approach for solving both the assignment and the routing subproblems is to optimize the mathematical programming models described in the earlier part of this section. In implementation, we utilize heuristics for both subproblems to boost performance.

The assignment subproblem can be solved exactly and efficiently using a greedy algorithm. This greedy algorithm is designed to exploit the fact that each and every task is considered independently, and all routing-related considerations (visit orders and detour times) are handled separately.

Algorithm 4.4.1. *A Greedy Algorithm for the Assignment Subproblem*
For each task i, compare the case where $v_i = 0$ (not performing the task) and $v_i = 1$ (performing the task). For $v_i = 0$, the objective value is trivially zero.

For $v_i = 1$, compute the best achievable objective value as:

1. *For $k \in K$, we can quantify agent k's contribution to the objective value by setting $y_{ik} = 1$ for components involving k in* (4.18)*:*

$$\sum_{m \in M_k} \left(\lambda_{ik}^m + \delta_{ik}^m \left(\frac{s_i \cdot \beta_k^m}{\eta} - \lambda_{ik}^m \right) \right),$$

   where we set $\delta_{ik}^m = 1$ if $\left(\frac{s_i \cdot \beta_k^m}{\eta} - \lambda_{ik}^m \right) < 0$ (because (4.18) *is a minimization problem).*

2. *Sort all agents according to their contributions in ascending order. Choose the first η agents and set $y_{ik} = 1$ for these η agents. The objective value for task i can be computed by summing contributions for these η.*

If the objective value for $v_i = 1$ is negative, set $v_i = 1$, otherwise set $v_i = 0, y_{ik} = 0, \delta_{ik}^m = 0$ for all $k \in K, m \in M_k$.

Proposition 4.4.2. *Algorithm 4.4.1 always finds optimal solution.*

Proof. In Step (2) of Algorithm 4.4.1, if we replace any of the η agents, the objective value will only increase, leading to a suboptimal solution. Similarly, in Step (1) of Algorithm 4.4.1, the objective value will only increase if δ_{ik}^m is set differently.
Therefore, Algorithm 4.4.1 will always find the optimal assignment. □

The routing subproblem, on the other hand, cannot be solved so efficiently, and are the major performance bottleneck. To investigate the trade-off between the optimality and the time performance, we define the following two LR variants based on how the routing subproblems are solved:

- **LR-Exact**: Routing subproblems are solved exactly by pure enumeration. Pure enumeration is the preferred approach in most instances since

with reasonable detour limit (say up to 30%), the number of feasible tasks will be small enough such that pure enumeration will outperform regular routing algorithm; a threshold can be set for the solver to switch to regular router if the number of feasible tasks is too large.

- **LR-Greedy**: Routing subproblems are solved using a simple greedy heuristic: an agent starts with the routine route m, and repeatedly try to evaluate the gain of inserting one of the remaining tasks into all potential slots; of all possible (task, slot) combinations, the best is chosen (thus greedy). There is no optimality guarantee, but it can solve routing subproblems very efficiently.

The LR-Greedy heuristic is very efficient, and can finish within $O(|R_k^m| \cdot |N_t|)$ even in the worst case (when there is no limit on worker's detour time). The LR-Exact algorithm, on the other hand, can perform very poorly as it simply enumerates all feasible task combinations. For smaller detour limits (e.g., the 30% as mentioned above), this might not be an issue. But for higher detour limits, the LR-Exact approach will experience exponential execution time growth in the number of tasks, and will not be scalable.

In the next section, we will empirically evaluate the effectiveness and efficiency of both approach, and quantify whether the quality and time trade-off of the LR-Greedy heuristic is worthwhile.

4.5 Computational Experiments

In section 4.4, we have addressed the task recommendation under route uncertainty using the ILP model and coming up with efficient heuristics. In this section, we investigate the computational properties of these approaches using synthetic datasets. These investigations will help us understand the performance trade-offs we would face when deploying our recommendation engine

for real-world field trials (to be reported in section 4.6.2).

More specifically, we investigate the performance of our LR heuristics from the following two perspectives:

- **LR heuristics versus the exact approach**: To understand the trade-offs between computational efficiency and solution quality, we solve the same problem instances using both our LR heuristics and the ILP model (which returns the true optimum). This evaluation can only be conducted for small instances due to the complexity of solving the ILP model.

- **LR heuristics versus deterministic heuristics**: To understand the benefits of modeling route stochasticity explicitly, we compare the performance of the LR heuristics against two deterministic heuristic approaches. One is a *push-based* deterministic model proposed in [26]; the other is a *pull-based* proximity approach that emulates current best practices. We use a city-scale network topology to perform this comparison.

4.5.1 LR Heuristics versus the Exact Approach

The purpose of this evaluation is to compare LR-Exact and LR-Greedy to the exact ILP model, both in terms of solution quality and computational time. Test instances are generated randomly with parameters (K, N_t, N), where K refers to the number of agents, N_t refers to the number of task nodes, and N refers to the total number of nodes in the network. Each agent is assumed to have two routine route candidates, where all routes have 5 nodes and are selected with equal probability. The coordinates of all nodes are generated uniformly randomly on a grid network. The distance between all pairs of nodes are Euclidean distance.

Our evaluation is summarized in Table 4.1. The gap is percentage from the optimum obtained via solving ILP model exactly. From these small testing

(K, N_t, N)	ILP	LR-Exact		LR-Greedy	
	time	gap	time	gap	time
(2, 4, 40)	0.8s	0%	0.09s	0%	0.05s
(4, 8, 80)	22.9s	0%	0.2s	4.12%	0.06s
(8, 16, 80)	6558s*	0.06%	14.8s	0.06%	0.14s

Table 4.1: LR heuristics vs. ILP: on both quality and time.
(*: We cut off CPLEX solver as the optimality gap is only 0.06%)

instances, we can see that both LR heuristics can produce close-to-optimum solutions very quickly. But examining the results closer, we can see that the efficiency of LR-Exact and LR-Greedy can potentially be different by one to two orders of magnitude.

4.5.2 LR Heuristics versus Deterministic Heuristics

To empirically quantify the benefits of generating recommendations considering routine route uncertainties, we introduce the following two deterministic heuristic baselines to compare with:

- **Deterministic-ILS**: Following the deterministic model and the iterated local search (ILS) heuristic proposed by Chen et al. [26], we designate each agent's routine route to be the route with highest probability. We denoted this baseline as **DILS**.

- **Proximity-Based Approach**: This heuristic emulates how most pull-based mobile crowdsourcing platforms work nowadays. At each decision epoch, agents who are available will be given the opportunity to pick desired tasks based on proximity. We denoted this baseline as **Proximity**.

The network used in this evaluation is based on the actual public transit network in Singapore (4,296 nodes and 10,129 edges), which contains all stops from the metro and bus services. All-pair-shortest distance matrix is computed a priori. To reflect the heterogeneous travel patterns of agents, we include two

types of agents: 80% of *normal agents* who compute back and forth between fixed locations (e.g., home and office), and 20% of *freelancers* whose routine routes have randomly chosen origin and destination nodes. For both agent types, the origin nodes are randomly picked from the non-central zones, while the end nodes are from the central zones (reflecting a commuting pattern from "home" to "office"). Two routes are constructed for each agent as follows: (i) the shortest path between the chosen origin and destination, and (ii) the path with the least number of stops. The probability distribution over agent k's two routes follows Bernoulli distribution with parameter α_k, where α_k is sampled uniformly from $(0, 1)$. Locations of tasks are generated according to the distribution (p_r, p_h), where p_r and p_h refer to the ratio of tasks in the non-central and the central zones. For the synthetic instances, half are generated with (p_r, p_h) equals: (60%, 40%), while the rest with (40%, 60%). Each task is associated with a fixed utility value of 100.

Detour	Estimate Bound	LR-E Gap	LR-G Gap	DILS Gap	Proximity Gap
10%	54.5%	-0.6%*	0.6%	13.2%	15.3%
20%	77.4%	-0.9%*	0.1%	10.4%	12.2%
30%	91.9%	0.1%	1.1%	7.1%	8.6%

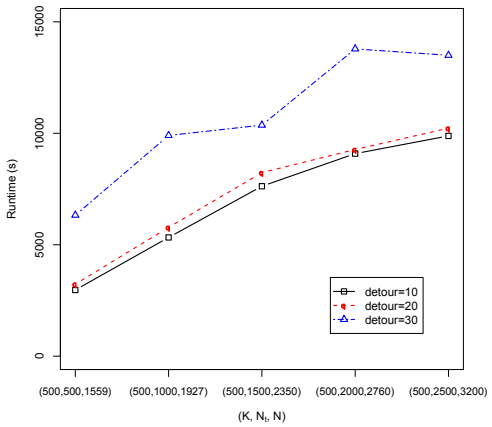
Table 4.2: LR heuristics vs. deterministic baselines.

The results for this section are summarized in Table 4.2, categorized using different detour limits for the tuple $(K, N_t, N) = (20, 30, 160)$, where 20 synthetic instances are randomly generated for every detour limit category using the above scheme. For each synthetic instance, all approaches are engaged to produce their respective task recommendation plans, which are then evaluated by 1000 routine route realizations. For each sampled route realization, we compute the percentages of tasks that can be accomplished for recommendation plans generated by all approaches (subject to the specified detour limit). For each approach, we then compute the average task completion percentage over

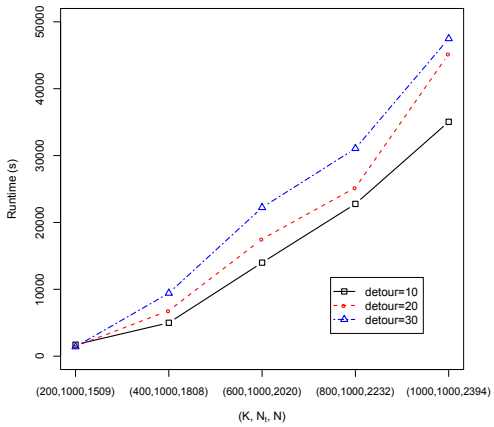
1000 route realizations and treat it as the performance of the approach.

To provide an estimated upper bound on the task completion percentage for each synthetic instance, we assume that the routine route realization is known during the evaluation phase and we solve the recommendation problem using DILS. This value is then used as the comparison baseline. The reported gaps in Table 4.2 are all relative to this estimated bound (in other words, the smaller the better). Although the estimated bounds are obtained with perfect information on route realization, due to the heuristic nature of DILS, there are cases (denoted with *) where the LR-Exact approach outperforms the estimated bounds.

From Table 4.2 we can clearly see the advantage of considering route uncertainties: both LR-Exact and LR-Greedy are able to obtain significantly smaller gap compared to deterministic alternatives. The advantage of push-based approach (DILS) over pull-based approach (Proximity) is also significant, which is consistent with previously reported results. We repeat similar experiments with a wide range of tuples (K, N_t, N), and in all instances, LR-based approaches outperform deterministic alternatives, and the advantages of our LR heuristics increase further as we tighten detour limits.



(a) Runtime(s) when N_t is increased.



(b) Runtime(s) when K is increased.

In terms of scalability, LR-Exact does not scale well. Without paralleliza-

tion, LR-Exact takes more than 9 hours to solve $(100, 200, 600)$ instances, which is of moderate size. LR-Greedy, on the other hand, is much more scalable. In Figure 4.2a, we fix K at 500 and increase N_t from 500 to 2500; in Figure 4.2b, we fix N_t at 1000 and increase K from 200 to 1000. From both figures, we empirically observe that runtime scales linearly in K and N_t.

4.6 TA$Ker: a Real-world Mobile Crowdsourcing Platform

The applicability of our push-based task recommendation is put to test in a campus-scale mobile crowdsourcing platform called TA$Ker. TA$Ker is designed to allow students in our school to register as crowd workers, and a wide variety of tasks are designed and distributed via the platform.

4.6.1 TA$Ker Architecture

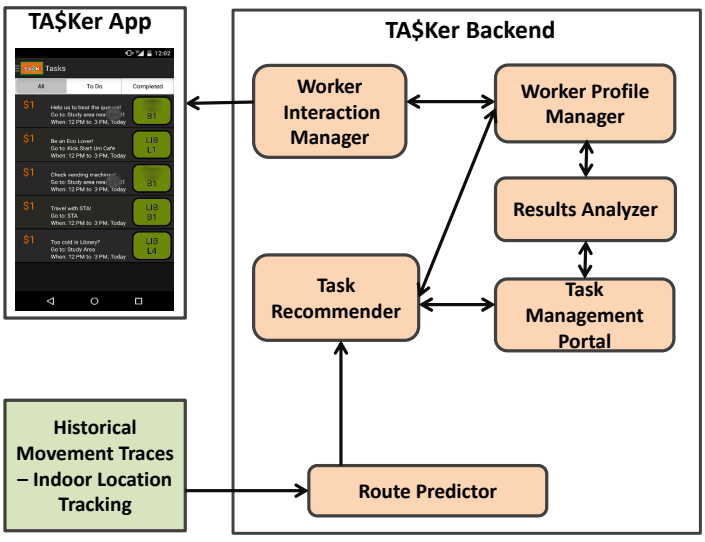


Figure 4.2: Overall architecture of the TA$Ker.

TA$Ker comprises a set of front-end and back-end components. Figure 4.2 shows the overall system architecture and illustrates the various individual

components and their interactions. The *Worker Interaction Manager* is responsible for handling interactions with individual workers via the mobile App (such as providing a list of suggested tasks, capturing user acceptance and completion of assigned tasks). The *Worker Profile Manager* is responsible for managing worker profiles (including tasks such as handling worker registration and indicating expertise for specific task categories). The *Task Management Portal* handles the interactions with task owners, allowing task owners to specify various attributes for tasks (such as the completion deadline, the amount of payment and the task location). The *Route Predictor* utilizes historical traces of individual user movement to develop a predictive trajectory profile. Finally, the *Task Recommender* is responsible for taking as inputs the list of location-dependent tasks, and the predicted movement profile of workers, and recommending an allocation of tasks (which individual workers may or may not accept) to each individual worker.

More specifically, the Task Recommender is built based on our proposed push-based LR heuristics presented in section 4.4 with uncertain move patterns. All participating students' preferences are collected through the TA$Ker mobile App. On-campus movement traces of all participating students are collected using Wi-Fi-based indoor localization techniques. Based on the collected movement traces, Route Predictor generates the probability distribution of routine routes for all participating students (details of the TA$Ker platform can be found in [63]).

4.6.2 User Study Details

Results presented here are obtained from a user study conducted with student participants on our university campus. All experimental studies are conducted with approval from our Institutional Review Board on campus. As part of the study, we recruited 160 students, who were briefed on the functionalities of the

App (but not told about the push vs. pull modes of task recommendation). To protect privacy, we did not extract any personally identifiable information such as name, date of birth, age and contact details. We then asked the students to perform tasks using our App at various locations on the campus. The trials were conducted over a two-week period from September 23 – October 2, 2015. During the trial, each student was free to use the TA$Ker App to perform any task that was in her list of *Available Tasks*.

The tasks can be categorized into the following four categories:

1. Discrete valued multiple choice: workers have to select from a predefined set of values. Example: *Is the male toilet at the second level of the Science building clean? Yes / No.*

2. Counting-based: workers have to select from a predefined set of numerical values. Example: *How many people are queuing at the coffee shop? 0–10.*

3. Picture-based: workers are required to upload task-specific images using their smartphones. Example: *Snap a picture of the promotion sign currently hanging in front of the Men's Grooming section at the Watsons store on campus.*

4. Free text-based: workers are to type in their answers as text. Example: *Tell us the price of AXE body spray sold at the Watsons store on campus.*

To study the relative efficacy of push vs. pull models, 160 students were randomly and equally divided into the "Push" class and the "Pull" class. For students belonging to the "Push" class, they were provided task recommendations tailored to their predicted routes. In contrast, for students in the "Pull" class, they were able to see the entire set of available tasks, and have to make their own task selection. The user study was governed by several important parameters:

- *Task Time Windows:* To accommodate student's typical daily schedule, each day was divided into three 3-hour time windows: (a) 9am – 12pm, (b) 12pm – 3pm, and (c) 3pm – 6pm. New tasks were loaded into the App at the beginning of every new time window (i.e., at 9am, 12pm, and 3pm). Any task not completed by the end of its time window was considered expired and was removed from the list of available tasks.

- *Reward per Task:* As our study did not focus on incentive design, we adopted a flat reward structure: every successfully completed task resulted in an earning of $1.

- *Maximal Tasks per Time Window:* Each student was allowed to perform at most 3 tasks per time window.

- *Varying η:* To investigate the effectiveness of the coverage ratio per task, we experiment with different η values ranging from 1 to 4.

4.6.3 Performance of the Recommendation Engine

It is natural to ask: *given the inherent location errors and movement uncertainties of workers on the campus, can our recommendation strategy generate useful task recommendations?* To establish this efficacy, we compute the accuracy of two distinct recommendation strategies: (a) our proposed strategy where the recommendations take into account each worker's individual predicted trajectories, versus (b) a *trajectory-oblivious* strategy, where each worker is first assigned one of five "representative trajectories" (each of which traverses all the floors of one of five campus buildings), and the centralized recommendation algorithm is then executed on these synthetic trajectories. The recommendation error is then computed in terms of the "detour distance" to a task's location, with this detour being defined as the *minimum distance to the task location, from all reference locations that the worker actually visits*

during the task's validity period. For example, assume that the recommended task T_1 (with location l_1) is valid during (10:00am, 10:30am) and the observed series of stay locations during this period is $\{X, Y, Z\}$. The minimum needed detour is then computed as $\min\{d(X, l_1), d(Y, l_1), d(Z, l_1)\}$.

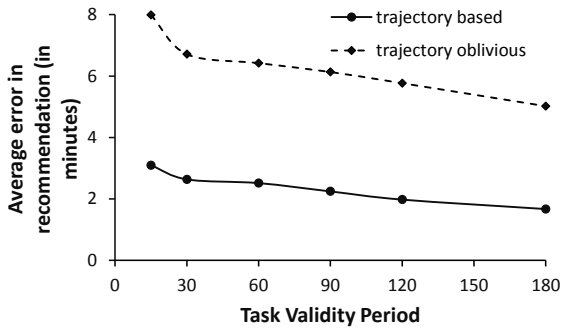


Figure 4.3: Error in recommendations.

Figure 4.3 plots the average error (detour overheads across all users) versus task validity period for these two strategies. We can see that the average error of our recommender is around 2.5 times lower compared to the trajectory-oblivious approach. More specifically, for a 15-minute validity window, our recommendation error is less than 3 minutes (equivalent to travel times between floors of the same building), while the trajectory-oblivious approach has an error of over 8 minutes (equivalent to the time needed to visit a location three buildings away). We also can see that in general, as the task validity period increases, the error in recommendation decreases, as the additional slackness in time enables us to better predict a worker's precise trajectory.

To study the efficacy of η, the multi-coverage parameter, we considered data from another pilot (March-April 2015) for a 4-week period where 80 students signed up and completed 800 tasks. During this trial period, we incremented η each week from 1 to 4. We compute the task completion rate each week by calculating the ratio between the number of completed tasks in that week and the total number of tasks accepted (normalized over number of users participated in each week). We find that increasing η indeed helps to improve the

task completion rate. With $\eta = 3$, the task completion rate is improved by 20%; when $\eta = 4$, the task completion rate is improved by 26%.

4.6.3.1 Super-Agent Phenomenon

While examining TA$Ker user's behaviors, we observe an interesting trend – a relatively small number of users generate a disproportionally large fraction of task responses. Figure 4.4 shows that 30% of active agents are responsible for 70% of total tasks completed on the TA$Ker platform. The existence of such heavily skewed behavior makes it important to focus on this critical group of users since they play an important role in the overall dynamics of the system and contribute more value to the task owners. We refer to this top 30% of active agents as *super agents*.

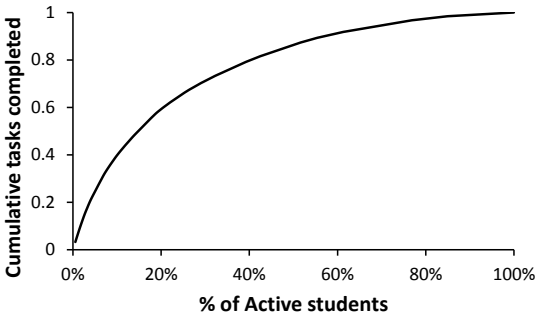


Figure 4.4: Super-agent phenomenon.

4.6.3.2 Efficiency of Users

In this section, we illustrate how we measure a user's efficiency in performing mobile crowdsourcing tasks. In particular, we provide measurements on a user's *planning efforts* and *task performance efforts* and demonstrate how being in the push/pull classes and being super/normal agents would affect these performance measures. All results are summarized in Table 4.3. The definitions of all used metrics are explained below.

Detour: To compute detour efficiency, we need to estimate detours that

Metrics	Push vs Pull		Super vs Normal		Push Class		Pull Class	
	Push	Pull	Super	Normal	Super	Normal	Super	Normal
Total detour (min)	5.2 (per task)	7.6 (per task)	45	15	40	12	62	18
Detour efficiency (cents/ min)	168	160	169	170	175	165	140	172
Task selection efficiency (min)	9	16	13	13	8	11	15	18
Performance interval (min)	8	3	6	6	8	9	3	3
Performance efficiency (distance)	one building	adjacent section	3 levels away	2 levels away	one building	3-4 levels away	2-3 sections away	adjacent section

Table 4.3: Summary of the metrics across classes and agent categories.

are related to the performance of tasks. However, measuring the total time traveled by a user is not straightforward since we need to identify the neighboring stay locations (both prior to and after the task performance) in which the user stays for a significant amount of time (in our case, more than 4 minutes) to calculate additional time elapsed for him to reach his next location after deviating from his usual route to perform the chosen task.

Let the task location be denoted as Z, we analyze the location traces, and identify locations this user stayed at, for considerably longer time before and after going to Z. We denote the location this user stayed before Z as X, and the location after Z as Y. The detour time is then $(t_{X,Z} + t_{Z,Y}) - t_{X,Y}$, where $t_{X,Z}$ denotes the travel time to reach location Z from location X.

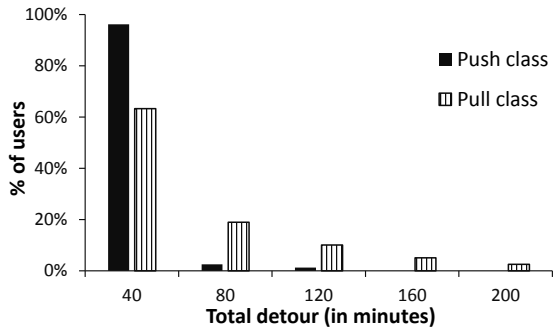


Figure 4.5: Total detour incurred during the trial.

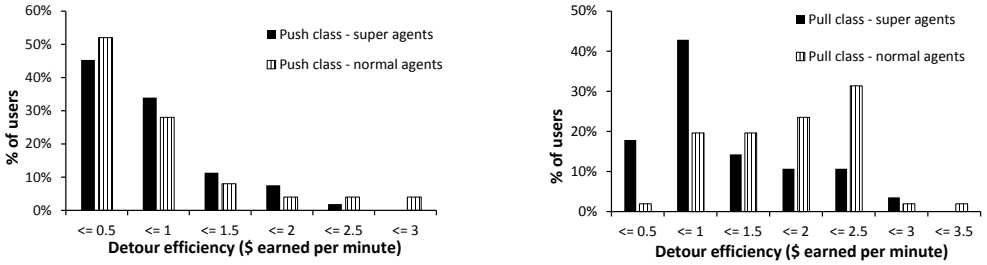
Figure 4.5 shows the histogram of the total detour made per user throughout our trial period. After averaging over the detour time over the number of completed tasks, we find that users from the push class incurred a detour

of 5.2 minutes per task, which is 2.4 minutes shorter on average than users in the pull class. A t-test confirms that the push class indeed incurs statistically lesser detour time than the pull class with a p-value of 0.0016. In terms of labor supply, super agents contributed on average 45 minutes while normal agents contributed only 15 minutes on average.

To further study how the push and pull modes affect behaviors of super agents, we separately analyze behaviors of super agents and normal agents in both push and pull classes. In the push class, super agents incurred 40 minutes of detour while normal agents incurred only 12 minutes. The same trend is observed in the pull class – super agents made significantly more detour compared to normal users (62 and 18 minutes, respectively).

Detour Efficiency: We have seen that super agents are willing to contribute more detour time, but does the extended travel lead to more earning opportunities? We measure this by the *detour efficiency*, which is defined as cents earned per detour minute. The histogram of detour efficiency for all active users is given in Figure 4.6. When broken down by the agent categories, super agents and normal agents have similar detour efficiencies at 169 and 170 cents per detour minute respectively. When broken down by the push/pull classes, the push class is earning 168 cents per detour minute, which is slightly more efficient than the pull class, which is earning 160 cents per detour minute. For normal agents in both push and pull classes, their performances are similar as well. However, the efficiency of super agents differs greatly depending on whether they are in the push or the pull class. While super agents in the push class earn 175 cents per detour minute, super agents in the pull class earn only 140 cents per detour minute, a 20% drop. This demonstrates that for dedicated mobile crowdworkers, push technology is particularly valuable as it allows workers to *outsource* their planning efforts to our recommendation engine.

Task Selection Efficiency: By analyzing how users interact with our



(a) Push-class users. (b) Pull-class users.

Figure 4.6: Detour efficiency of (a) push-class and (b) pull-class users.

App, we can quantify the "search efforts" spent by users in identifying the set of tasks to commit to. More specifically, for each time window, we estimate the task selection efficiency within this time window as the time difference between the moment a user opens the App for the first time or browses through the list of tasks (whichever happens first) and the moment the user accepts a task. Similar to earlier metrics, we provide comparison across two dimensions: between push and pull classes, and between super and normal agents.

While users from the push class spent 9 minutes browsing through the tasks list and accept the tasks, users from the pull class spent nearly twice as long as their counterparts. This is consistent with our intuition that users from the push class have the advantage (over the pull class) of browsing only the tasks in the recommended list. Interestingly, despite the fact that super agents perform more tasks than normal agents, both agent classes spend 13 minutes (on average) in making initial task commitment. When analyzing both agent categories in each class separately, we notice that both super and normal agents spend much less time in the push class than in the pull class. For super/normal agents, it is 8/11 minutes for being in the push class and 15/18 minutes for being in the pull class.

Performance Interval: To understand how far in advance does a user commit to his tasks, we also measure the time in between task selection and task performance. We find that a user from push class submits a task after

8 minutes since the acceptance time; however, pull class users submit after a mere 3 minutes. However, agent category does not seem to play a role in this metric, as the performance intervals for both super and normal agents are around 6 minutes.

Performance Efficiency (distance): We also notice that users from the pull class accept tasks mostly in the vicinity of their current locations – when they are (on average) 50 seconds away from task locations (translates to several sections away, usually on the same building level). On the other hand, push class users are more likely to accept a task further away, even when they are several buildings away. Instead of reporting the actual travel time (or distance), we report whether the distance is such that it is still on the same level (but in sections), on different levels (but still in the same building), or in different buildings.

4.7 Summary

In this chapter, we study a "push-based" paradigm for large-scale mobile crowdsourcing, where a centralized engine provides trajectory-aware task recommendations to a large pool of workers, while taking into account individual's uncertain daily movement patterns. We present a stochastic integer linear program, to find assignments that maximize the cumulative expected rewards by all workers, considering the uncertainties over their routine routes. Subsequently, to develop a solution that can scale to city-scale scenarios, we exploit the separable problem structure and apply Lagrangian relaxation and dual decomposition. Experiments using realistic movement traces over Singapore's public transport network show that our Lagrangian relaxation based heuristics can generate recommendations faster than an exact ILP formulation by more than two orders of magnitude, while suffering a less-than-1% degradation in the percentage of tasks recommended (compared to an idealized alternative

where the realized routes are known a priori).

The applicability of our push-based task recommendation is put to test in a campus-scale mobile crowdsourcing platform called TA$Ker. Real-world crowdsourcing studies show that this push-based approach can be effective even with highly uncertain routine routes, achieving more than 25% of improvement in efficiency for top workers, compared to a traditional pull-based alternative.

Chapter 5

Home Health Care

5.1 Overview

The last application domain that draws our interest is a mobile version of the workforce scheduling, where service providers are scheduled to service the customers at their specified locations. More specifically, we study the workforce scheduling for home health care services[1]. In the literature, a considerable amount of problem specific systems have been developed for home health care settings across the world. For example, Begur et al. [9] first developed a spatial decision support system for Visiting Nursing Association in the United States to minimize total travel time, taking into consideration of routing, provider availabilities, and fixed visitation frequencies. Eveborn et al. [38] presented a single day scheduling system, called LAPS CARE, for Swedish health care system that maximizes the number of served requests and considers time windows, skill requirements, and breaks. We believe the existing systems/practices fell short in the following two key aspects.

- **Failure to address the uncertainties**: Due to the varying health conditions of the patients as well as the experience of healthcare providers,

[1]These medical services range from cleaning, personal hygiene to administering some medical treatments, such as blood pressure tests, prescriptions, injections and so on

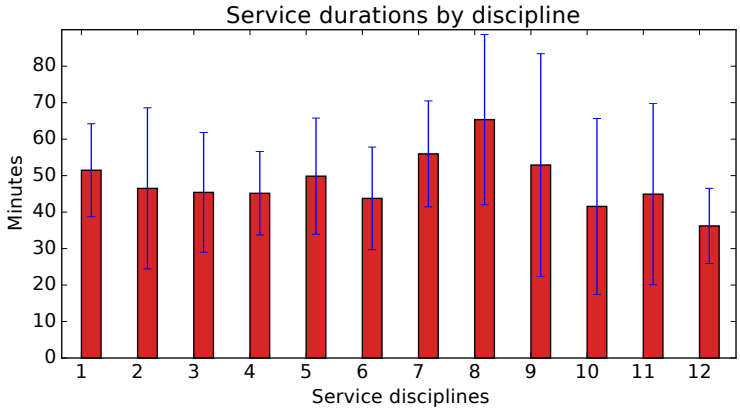


Figure 5.1: Means and standard deviations of service durations over one-month visits grouped by different service disciplines. Note, the statistics are summarized using the *actual visit* records of Sept. 2015. Discipline specifies the type of service required, e.g., speech therapy, skilled nursing, physical therapy, to name a few.

the service durations can have quite substantial variance. Figure 5.1 plots the means and standard deviations of actual service durations over one month of *actual visit records* grouped by different service disciplines. We can see that service duration uncertainty is clearly exhibited, where standard deviations under all disciplines are larger than 10 minutes up to 30 minutes. In addition, unforeseen traffic conditions, such as congestion, accidents, and breakdowns, often result in varying travel durations, thus further complicates the prediction of service start times. Faced with both types of duration uncertainties, providers responsible for scheduling are confronted with the dilemma of either over-estimating duration times so as to guarantee services and travel and, hence, under-utilizing their resources, or under-estimating those times and, thus, short-changing patients by making them wait for services. In any case, duration uncertainties can potentially deteriorate patient satisfaction and result in additional costs for the company.

- **Insufficient business considerations**: Existing automated scheduling systems differ significantly, as problems originate from different regions with various requirements and regulations. Fikar and Hirsch [41] recently

presented a comprehensive survey on the home health care problem. To the best of our knowledge, none of the existing works completely addresses all the business requirements raised by our problem.

In this chapter, we propose to develop a decision-support system for the efficient delivery of services to patients at home, while tightly integrating both patients' and providers' preferences. We focus on generating the start-of-the-day schedules, considering realistic scenarios driven by our real-world needs, with a comprehensive set of considerations for home health care settings, e.g., time windows, continuity of care, workloads, inter-visit temporal dependencies, and especially the *duration uncertainties*.

5.2 Literature Review

The Home Health Care Scheduling Problem (HHCSP) is the problem of scheduling and routing service providers to visit patients at home. It was first mentioned by Fernandez et al. [40], where community nurses are scheduled to visits patients. Essentially, HHCSP instantiates some form of *workforce scheduling* and *routing* problems. Castillo-Salazar et al. [22] review the workforce scheduling and routing problem, not limited to the health care industry and view the home health care as one of the application domains. Other similar applications can also be found in the technician scheduling, emergency services, transportation systems, or call centers [36, 22]. Problems from different domains share some commonality, and also differs substantially in problem settings. Given a large amount of literature, we focus on the research more related to our problem and heath care domain.

On the workforce scheduling aspect, literature on workforce scheduling in health care domain mainly deals with staff rostering requirements, such as skills, shifts, time-related constraints, work regulations and ect [17, 99]. On the routing aspect, Vehicle Routing Problem (VRP) [123] and OP [57] have

long been extensively studied to model various problems from transportation and logistics. In literature, HHCSP is often modeled as the VRP [41], with assumption that the workforce is sufficient and the goal is to cover all the requests with the minimal travel costs or manpower. In our problem, however, the company faces an oversubscribed situation, where the number of requests received often exceeds its service capacity, especially under the case when we care about patient preferences and continuity of care. The current practice for the company is to first schedule requests for full-time providers. Any uncovered requests are then made up by a combination of either extending the visits for full-time providers or outsourcing to part-time providers. We are concerned with scheduling full-time workforce with the objective to maximize patients' satisfaction, measured in terms of rewards. Hence, the underlying problem at stake is an OP rather than a VRP. We model the problem as a variant of the TOPTW.

There is also a thread of research dedicated to home health care. Fikar and Hirsch [41] recently present a comprehensive survey on HHCSP. The majority of the related works focuses on single-period problems, i.e., a single day as scheduling horizon, and only few papers consider multi-period ones, i.e., over multiple days. Our problem can be categorized as a *multi-period HHCSP*. The problem becomes much more challenging and complicated when going into a longer scheduling horizon, as the scheduling process involve more complex assignments, regulations, and continuity of care. Problems differ substantially due to different business considerations. Time windows, qualifications, and provider availabilities are commonly addressed in multi-period problems, while other aspects, such as workload fairness, continuity of care, inter-visit temporal dependencies are less incorporated, especially uncertainties [41]. Next, we focus on the multi-period HHCSP literature.

In multi-period HHCSP literature, there have been works addressing the workload fairness as the objective function, to minimize the workload differ-

ence among providers or optimize utilization factors [59, 8, 20, 37]. Yuan and Fügenschuh [144] handle the workload in the objective to minimize daily working hours. In our problem, the company desires a provider to service a minimum workload if possible and overworking is not preferred. We model the minimum workload as soft constraints as the penalty in the objective function and maximum workload as hard constraints.

Considering the continuity of care, works by [5, 21] model it as hard constraints that require patients to be visited strictly by the same providers over the scheduling horizon, which is not flexible to a certain extent. Nickel et al. [95] try to minimize the sum of, over all the providers, the different providers serving the same the patient, while Rodriguez et al. [106] allow a patient to be visited by a maximum number of different service providers. However, these providers are treated equally without priorities. Lin et al. [82] use five cases of hard weight allocation criteria to enforce care continuity and incorporate priority. Instead, we take a softer and flexible approach. Our objective is to maximize patients' satisfaction, measured in terms of rewards collected for serving the patients. The provider-request dependent rewards give us the flexibility to reflect the continuity of care with prioritized provider candidates and to further capture requests' information, such as type, emergency or request priority.

Inter-visit dependency is another aspect to consider. Rasmussen et al. [103] study a single-day problem with five types of temporal precedence. Existing research on the multi-period problem focuses on day-level temporal dependencies. Some impose fixed visiting days [9, 124], while some assume service frequency, often handled by using predefined service combinations as the input and let the model decide the visiting days [10, 113, 95, 144]. In our problem, we handle both fixed visiting days, as well as visiting service frequencies for patients with flexible availabilities.

With regards to handling uncertainties, we notice that most of the existing

works with uncertainties consider the uncertain demands [73, 21, 106, 15], whereas stochastic service or travel times are rarely studied in HHCSP. Yuan et al. [143] present an exact branch-and-price algorithm to tackle a single-day problem with uncertain service times, which solves small instances up to 50 patients. It focuses on the penalty for late arrivals in the objective, instead of enforcing the time windows. Errarhout et al. [37] propose a two-stage model in a multi-period setting to cater uncertain service times and solve the model by CPLEX with instances up to 11 nurses and 75 patients. However, time-windows and inter-visit temporal dependencies are not respected. The introduction of uncertain durations makes the problems with time windows even harder to solve. In this problem, we incorporate duration uncertainties by enforcing a set of time window chance constraints.

In summary, to the best our knowledge, none of the existing works can be readily extended to handle our problem.

5.3 Problem Formulation

In this section, we provide the formal definition for our multi-period home health care scheduling problem, motivated by the requirements from a leading home health care and hospice company in Pittsburgh. The problem is defined as the following tuple:

$$\langle D,\ N,\ T,\ R,\ K \rangle$$

D denotes the set of days d for the scheduling horizon, i.e., $d \in D$. Each day is discretized into minutes, indexed from 1 to 1440. N represents the set of all nodes, $N = N_t \cup N_k$, where N_t and N_k are the set of patient' requested visit locations and health care providers' start and end locations, respectively. T is the pairwise travel time matrix and $t_{ij} \in T$ denotes the travel time between

node i and j.

R represents the set of patients' requests, for the given scheduling horizon, e.g., the next 7 days. Each request represents a service task to be specifically performed at a patient's home. A request belongs to only one patient, and a patient may specify several requests over the scheduling horizon. Each request i is characterized by a tuple $\langle n_i, a_i, w_i, q_i^{req}, \{[o_{id}, c_{id}]\}\rangle$ where:

- $n_i \in N_t$, a_i and w_i are the service location, service duration, units of work required by the request, respectively.
- q_i^{req} is the service discipline specifying the type of service that the request needs, e.g., speech therapy, skilled nursing, physical therapy, just to name a few.
- $\{[o_{id}, c_{id}]\}$ refers to the set of available **time windows**, during which a patient wish servicing of a request i to be started. It is possible that a patient indicates several available time windows for the same request i over different days d – e.g., Alice is free on Monday morning and Thursday 2pm-4pm for a physical therapy treatment, from which health care providers have to decide the best time slot to allocate. More specifically, o_{id} and c_{id} are the earliest and latest start-times for a time window $[o_{id}, c_{id}]$. If a provider arrives *earlier* than o_{id}, he will *wait* until the time window opens. While arriving *later* than c_{id} will lead to the violation of the time window constraint.

K represents the set of health care providers. Each health care provider $k \in K$ has a set of qualifications that he holds. Typically, qualifications are flat and distinct, e.g., nurse, nutritionist, and therapist. Each provider is constrained by an availability time window $[T_{kd}^1, T_{kd}^2]$ on each day d, i.e., he will leave his start node N_k^1 at time T_{kd}^1, and return to the end node N_k^2 before time T_{kd}^2.

A request is considered as *completed* if and only if a **qualified and available** provider starts the service within the patient's available time window and stays with the patient for the whole service duration.

The goal is to schedule and route service providers for home health care visits on a weekly basis that considers the requirements from both patients and service providers. The problem is further subject to the following considerations:

- **Inter-visit Temporal Dependency**: A patient may subscribe several visits/requests over the same week. These requests can be temporally dependent such that request j has to be fulfilled at least D_{ij}^{-} days after and no more than D_{ij}^{+} days after servicing request i. Such inter-visits dependencies are often seen in practice.

- **Workload Fairness**: A provider is paid a fixed salary as long as he is working on a day. The company desires a provider to service at least W^{-} units of work on a daily basis, if possible. At the same time, a maximum working unit W^{+} is imposed, as overworking is not preferred.

- **Continuity of Care**: Each patient has a set of prioritized provider candidates and is preferred to be visited by his primary provider. This is important for patient satisfaction, especially for patients with chronic conditions.

- **Uncertain Durations**: In real-world scenarios, travel and service durations are usually uncertain. We assume travel times t_{ij} and service times a_i are random variables following certain distributions.

5.3.1 Mathematical Model

In this section, we first propose an ILP model for the deterministic problem, followed by incorporating the duration uncertainties. Decision variables are

summarized as follows:

Variables	Descriptions
$x^d_{ijk} \in \{0,1\}$	set to 1 if provider k serves request j right after i on day d.
$y^d_{ik} \in \{0,1\}$	set to 1 if request i is assigned to provider k on day d.
$v_i \in \{0,1\}$	set to 1 if request i is not assigned to any provider.
$e^d_{ik} \in \{1,\ldots,T\}$	service start time of request i for provider k on day d.
$f_{kd} \in \{0,1\}$	set to 1 if provider k is assigned with requests on day d.
$p_{kd} \in \{0,1\}$	set to 1 if the route for provider k on day d is penalized.

Intuitively, reward will be collected if a request is completed by a provider. Let r_{ik} be the provider-dependent rewards, defined based on the units of work (w_i) the request needs and whether this provider is the patient's primary provider under this discipline(l_{ik}). Thus, we have:

$$r_{ik} = r \cdot w_i + r^+ \cdot l_{ik},$$

where r is a constant base reward and r^+ is the additional reward assigned for the primary provider. The reward structure helps the *continuity of care*, where there is an incentive to assign primary providers to patients, and promotes the productivity, where higher reward will be collected for request requiring more units of work. The bigger the r^+, the stronger the enforcement of care continuity.

The objective of this model is to generate a sequence of requests to visit for each provider on each day that maximizes the expected total rewards collected for the whole team considering the route penalties. γ is the amount of penalty

incurred if a route does not meet the minimum workload.

$$\max \sum_{i \in N_t} \sum_{d \in D} \sum_{k \in K} r_{ik} \cdot y_{ik}^d - \gamma \cdot \sum_{d \in D} \sum_{k \in K} p_{kd}, \tag{5.1}$$

$$v_i + \sum_{d \in D} \sum_{k \in K} y_{ik}^d = 1, \qquad \forall i \in N_t, \tag{5.2}$$

$$y_{ik}^d = 0 \qquad \forall i \in N_t;\ d \in D;\ k \in \{K \setminus K_{id}\}, \tag{5.3}$$

$$\begin{cases} D_{ij}^- - M(v_i + v_j) \leqslant \sum_{k \in K} \sum_{d \in D} d \cdot (y_{jk}^d - y_{ik}^d) & \forall i, j \in N_t, \\ \sum_{k \in K} \sum_{d \in D} d \cdot (y_{jk}^d - y_{ik}^d) \leqslant D_{ij}^+ + M(v_i + v_j), & \forall i, j \in N_t, \end{cases} \tag{5.4}$$

$$y_{ik}^d \leqslant f_{kd}, \qquad \forall i \in N_t; k \in K; d \in D, \tag{5.5}$$

$$\begin{cases} W^- \cdot f_{kd} - \sum_{i \in N_t} y_{ik}^d \cdot w_i \leqslant M \cdot p_{kd}, & \forall k \in K; d \in D, \\ \sum_{i \in N_t} y_{ik}^d \cdot w_i \leqslant W^+ & \forall k \in K; d \in D, \end{cases} \tag{5.6}$$

$$y_{ik}^d \leqslant \sum_{j \in N} x_{ijk}^d \qquad \forall i \in N_t; k \in K; d \in D, \tag{5.7}$$

Constraints (5.2) ensure each request is assigned at most one provider over the entire scheduling horizon. As K_{id} denotes the set of providers who are qualified and available to serve request i on day d, constraints (5.3) make sure that requests will not be assigned to any unavailable or unqualified providers. Constraints (5.4) reflect requests' inter-visit temporal dependencies. Constraints (5.5) specify that a route exists when it contains any request. Constraints (5.6) enforce the minimum and maximum workload requirements on every route. M is a large positive number. If a provider works less than W^- units on a day, the route for that provider on that day will be penalized (i.e., $p_{kd} = 1$). Note, if a provider is not assigned any requests on a day, it will not be penalized, as there is no cost incurred by the health care company. Constraints (5.7) bind the decision variables y_{ik}^d with decision variables x_{ijk}^d, which ensure that requests are assigned only when they can be visited.

The rest of the constraints (5.8) - (5.12) are provider-day (k, d) level routing

constraints. For each provider $k \in K$ on each day $d \in D$, the same set of constraints applies.

$$\begin{cases} \sum_{j \in N} x^d_{ijk} = \sum_{j \in N} x^d_{jik}, & \forall i \in N \setminus \{N^1_k, N^2_k\}, \\ \sum_{j \in N} x^d_{ijk} - \sum_{j \in N} x^d_{jik} = 1, & i = N^1_k, \\ \sum_{j \in N} x^d_{jik} - \sum_{j \in N} x^d_{ijk} = 1, & i = N^2_k, \end{cases} \tag{5.8}$$

$$e^d_{ik} = T^1_{kd}, \qquad i = N^1_k, \tag{5.9}$$

$$e^d_{ik} + a_i + t_{ij} - e^d_{jk} \leqslant M(1 - x^d_{ijk}), \ \ \forall i, j \in N, \tag{5.10}$$

$$o_{id} \leqslant e^d_{ik} \leqslant c_{id}, \qquad \forall i \in N_t, \tag{5.11}$$

$$e^d_{ik} \leqslant T^2_{kd}, \qquad i = N^2_k. \tag{5.12}$$

Constraints (5.8) ensure that every provider k starts and ends at his specified nodes N^1_k and N^2_k and the connectivity of the nodes. Note that, e^d_{ik} refers to the start service time of node i. Early service or late service will cause the violation of the request time window constraints. Constraints (5.9) initialize the start service time of the start node for provider k. Timing consistency constraints are enforced in constraints (5.10). For every provider k, if node j is visited immediately after node i (i.e., $x^d_{ijk} = 1$), the start service time e^d_{jk} of node j should be at least $(a_i + t_{ij})$ time units later than the start service time e^d_{ik} of node i (also eliminates the subtours). Constraints (5.11) make sure that requests' time windows are respected. Finally, constraints (5.12) limit the time budget.

5.3.2 Modeling Duration Uncertainty

To incorporate the uncertain durations, we then extend the deterministic model. We assume travel times t_{ij} and service times a_i are random variables following certain distributions. We assume these distributions are given as problem input, which can be derived from historical records obtained from the

home health care domain. Note that in the deterministic model, these durations only affect the time window and time budget constraints. Thus, to model duration uncertainty, we replace the time window constraints (5.11) and time budget constraints (5.12) by a set of chance constraints (5.13) and (5.14), while the rest of the constraints remain the same as the deterministic formulation. Thus, we have:

$$\max \quad \textit{Objective} \ (5.1)$$

$$s.t. \ \textit{Constraints} \ (5.2) - (5.10)$$

$$P(o_{id} \leqslant e^d_{ik} \leqslant c_{id}) \geqslant 1 - \alpha, \qquad \forall i \in N_t; k \in K; d \in D \tag{5.13}$$

$$P(e^d_{ik} \leqslant T^2_{kd}) \geqslant 1 - \alpha, \qquad \forall i = N^2_k; k \in K; d \in D. \tag{5.14}$$

The chance constraints (5.13) and (5.14) enforce the probability of satisfying the respective constraints are at least $1 - \alpha$, $\alpha \in [0, 1]$. α is the *risk level*, indicating the decision maker's level of conservativeness.

5.4 Solution Approaches

The deterministic model can be solved by commercial solvers such as CPLEX. However, it is not scalable with increasing number of providers and requests with longer scheduling horizon. In this section, we use Lagrangian relaxation and dual decomposition to improve its scalability.

5.4.1 Lagrangian Relaxation

We relax constraints (5.7), which couple all the assignment and routing decision variables together, into the objective function.

$$\begin{aligned}\min\ L(\lambda) = &-\sum_{i\in N_t}\sum_{d\in D}\sum_{k\in K} y_{ik}^{d}\cdot r_{ik} + \gamma\cdot\sum_{d\in D}\sum_{k\in K} p_{kd} \\ &+\sum_{i\in N_t}\sum_{d\in D}\sum_{k\in K}\lambda_{ik}^{d}(y_{ik}^{d}-\sum_{j\in N} x_{ijk}^{d}).\end{aligned} \tag{5.15}$$

We then *decompose* the relaxed dual problem $L(\ \lambda\ )$ into one assignment sub-problem, which assigns the requests to the providers, and provider-day-level sub-problems, which find the routes for the providers.

Assignment Sub-problem: The assignment sub-problem is defined as:

$$\min \sum_{i\in N_t}\sum_{d\in D}\sum_{k\in K} y_{ik}^{d}\cdot(\lambda_{ik}^{d}-r_{ik}) + \gamma\cdot\sum_{d\in D}\sum_{k\in K} p_{kd}, \tag{5.16}$$

$$s.t.\ \ Constraints\ (5.2)-(5.6)$$

The assignment ILP model can be solved exactly by CPLEX. To further scale up the sub-problem, we can apply linear relaxation. The idea is to drop integer constraints for decision variables, which transforms hard ILP problem into an easier polynomial solvable linear problem (LP). In minimization problem, the objective value achieved by the LP is always smaller or equal to that of the original ILP, thus it can serve as a lower bound for the assignment sub-problem.

Routing Sub-problems: There are K · D independent provider-day-level

routing sub-problems for each provider k and on each day d:

$$\min - \sum_{i \in N_t} \sum_{j \in N} \lambda_{ik}^{d} \cdot x_{ij}, \tag{5.17}$$

$$s.t.\ Constraints\ (5.8) - (5.12)$$

$$\sum_{j \in N} x_{ij} \leqslant 1, \qquad \forall i \in N_t, \tag{5.18}$$

$$\sum_{i \in N_t} \sum_{j \in N} x_{ij} \cdot w_i \leqslant W^{+}. \tag{5.19}$$

Constraints (5.18) and (5.19) are included to further tighten the sub-problems, such that one node is visited at most once in a route and the maximum workload is enforced (i.e., no more than W^+ units of work in a route). This routing sub-problem can be viewed as an OP, which is NP-hard. Instead of solving the routing ILP, we develop a search algorithm (2), that exploits the routing problem structure. During the search phase, we systematically extend the routes and discard unpromising dominated routes, which guarantees to find the optimal routing solution. Before going into the algorithm details, we first present the following observation.

Observation - *Route Comparability: two feasible routes are comparable if and only if they contain exactly the same set of nodes and end at the same node, i.e.,* $|r_1| = |r_2|$ *and* $r_1^k = r_2^k$ *(assume k is the last node in the route).*
Observation 1 holds because two feasible routes having the same set of nodes with different visiting sequences and the same ending node leads to the same objective value, and provides a fair starting point for further route extension.

For two comparable routes, route r_1 *dominates* route r_2 if they are comparable and the total time for r_1 is less than the total time for r_2. The intuition is that the route with the shorter total time will have more room for future insertion. In algorithm (2), we insert only nodes with positive Lagrangian multipliers(λ_{ik}^{d}) into the route to improve the objective value. A node can be inserted into the route if and only if all the time windows, time budget, and

maximum workload constraints are satisfied (i.e., GetFeasibleRequest()). We start with a route with only one node, i.e., the provider's start location. At each iteration, *non-dominated* routes from the last iteration are expanded by one more *feasible* node. A node is considered as feasible if the resulting extended route satisfies the time window and time budget requirements. By doing so, routes of longer length are generated. Only *non-dominated* routes will be stored and dominated routes will be pruned. Note, with the dominance, if there are several comparable routes with the same shortest total time, we will keep just the first one. The search procedure stops when no route can be further expanded.

Algorithm 2: BFS with Dominance Pruning

```
Input: (λ^d_ik, N)
R ← Initialize(), continue←TRUE
while continue do
    for r ∈ R do
        N_feasible ← GetFeasibleRequest(r, N, λ^d_ik)
        for n ∈ N_feasible do
            r' ← ExtendRoute(r, n, λ^d_ik)
            R ← UpdateNonDominatedSet(r', R)
        end
    end
    continue ← CheckContinue()
end
r* ← UpdateBest(R)
```

Since all the sub-problems are independent from each other, our decomposition based approach allows further parallelization for the sub-problems in the system implementation, which would largely speed up the solution approach. Note, in the experiment section, sub-problems are run sequentially, without parallelization.

After dual decomposition, we solve the Lagrangian dual problem through projected sub-gradient descent algorithm, detailed in section 2.2.2. At each iteration, sub-problems are solved given the Lagrangian multipliers λ_t. λ_t are

updated iteratively through the master function:

$$\lambda^d_{ik,t+1} := \lambda^d_{ik,t} + \alpha_t(y^d_{ik} - \sum_{j \in N} x^d_{ijk}) \qquad \forall i \in N_t, k \in K, d \in D. \tag{5.20}$$

In order to iteratively move towards the optimal solution and determine the convergence, we need the best primal solution with the dual solutions at each iteration. However, dual solutions may not always result in a feasible primal solution, as a request may appear in several routing solutions. Let $z^d_{ik} = \sum_{j \in N} x^d_{ijk}$ where x^d_{ijk} are the decisions from routing sub-problems. To restore a good feasible *primal solution* from the routing solutions, we solve the following ILP:

$$\min - \sum_{i \in N_t} \sum_{d \in D} \sum_{k \in K} r_{ik} \cdot y^d_{ik} + \gamma \cdot \sum_{d \in D} \sum_{k \in K} p_{kd}, \tag{5.21}$$

$$s.t.\ Constraints\ (5.2) - (5.6)$$

$$y^d_{ik} \leqslant z^d_{ik} \qquad \forall i \in N_t; k \in K; d \in D. \tag{5.22}$$

To improve the scalability of primal extraction, we also developed a *greedy local search* heuristic. Given routing solutions as the base, we try to greedily resolve the conflicts of same requests appearing in several routes and insert unassigned requests using local search heuristic (several local search operations are used, such as insert, replace and etc).

5.4.2 Handling Duration Uncertainty

To solve the chance constrained model, we apply Sample Average Approximation (SAA) [96] to reduce realization set to manageable size and convert the stochastic formulation into a deterministic one. We randomly generate a set of independent and identically distributed samples, $S = \{\xi_1, \xi_2, ..., \xi_s\}$, for all t_{ij} and a_i from the known distributions, and check whether time window and

time budget constraints are satisfied. Note, these duration distributions can be derived based on the domain knowledge and historical data. We approximate the probabilities as:

$$P\left(o_{id} \leqslant e^d_{ik} \leqslant c_{id}\right) \approx |S^+|/|S|$$
$$P\left(e^d_{ik} \leqslant T^2_{kd}\right) \approx |S^+|/|S| \tag{5.23}$$

where $|S^+|$ are the number of samples under which the corresponding constraints are satisfied. Note that, when approximating chance constraints on a discrete set of samples, it is important to identify a smaller risk threshold $\alpha^{'}$, where $\alpha^{'} < \alpha$. Since SAA replaces the original distributions with empirical distributions obtained from the samples, a smaller $\alpha^{'}$ is used to hedge against the under-representation of the limited samples.

The resulting formulation is an ILP that still maximizes the total collected rewards but also considers the constraints over all the samples. Similarly, the chance constrained criteria can be incorporated into our solution approach by modifying the routing sub-problems, i.e., specifically the GetFeasibleRequest() Function in our specialized search routine. Now, feasible requests are generated by checking the chance constraints over all the duration samples (against $\alpha^{'}$), instead of just the deterministic durations.

Sample selection heuristic: The scalability and quality of the SAA method depend on the size and representativeness of the sample set. So instead of use a fixed large sample set S', we use a small representative subset $S \in S'$, where we try to select samples that are as different as possible. Intuitively, the smaller the durations, the lesser chance of constraint violation. We first generate a large amount of samples from the duration distributions, say $|S'| = 1000$. We then sort the samples according to certain metric, i.e., $d_s = \sum_{i \in N} \sum_{j \in N} t_{ij} + \sum_{i \in N_t} a_i$. We then uniformly select $|S|$ samples from $|S'|$ based on the distances d_s.

5.5 Computational Experiments

In this section, we empirically demonstrate the efficiency and effectiveness of our approaches on instances adapted from a real world dataset.

5.5.1 Instance Generation

The problem instances considered in our experiments are adapted from a real-world dataset from a leading home health care and hospice company in Pittsburgh, USA. The data contains one month of visit-related information for September 2015. Each service provider is characterized by start geo-coordinates and his qualifications. Also, each patient is associated with a home geo-coordinates and provider preferences. A patient may have several visit records over the week. Each visit record contains information such as a visit datetime, actual service duration, type, discipline, and provider assigned. As the dataset contains the actual visits information and not the input requests for the scheduling, we need to generate the requests from the dataset.

To broaden the analysis, we also synthetically generate an additional set of problem instances with time windows, inter-visit dependencies and duration uncertainties[2]. We first retrieve the visits within the specified scheduling horizon and the subregions. From the selected visits, we retrieve the corresponding patients' and providers' information. Deterministic pair-wise Haversine distances (i.e., great-circle distances) are computed based on their actual geo-information and converted into travel times a priori (with a travel speed of 30mph). We then synthetically generate some additional patients' request-related information, such as available time windows and inter-visit dependencies. We assume visit i's start service time s_i is the midpoint for request i's time window. Two types of request time windows are then speci-

[2]Due to patient privacy concerns, it is not possible for us to make actual problem data provided by the HHC Company publicly available. However, the additional synthetically generated problems are accessible: sites.google.com/site/homehealthcarewebsite/

fied as $[s_i - tw/2, s_i + tw/2]$, with time-window width tw set to 2 hours and 6 hours, denoted as *ins-tight* and *ins-loose* respectively. 2-hour time-window is rather realistic while 6-hour time-window provides more scheduling flexibility. To reflect the inter-visit temporal dependencies, we randomly select 40% of the patients and filter out the patients with only one request. For each selected patient, we then synthetically generate $[D_{ij}^-, D_{ij}^+]$ for all his request pairs $(i, j) \in R_k$. Additionally, we allow those selected requests to have several available days, while the rest of the requests have to be visited on the same days as the actual visits. Lastly, we assume providers are available on the days of the visits, and they work from 7am to 5pm. Provider-dependent rewards are generated based on r and r^+. Here, we assume the base reward r is a fixed value of 100.

For stochastic instances, in lieu of having distributions based on historical data, we assume durations $t \in T$ (both travel and service times) are normally distributed $N(\mu_t, \sigma_t^2)$ with μ_t equal deterministic durations and σ_t as 10 minutes.

5.5.2 Algorithms Compared

The performances and runtime of our proposed methods depend on how sub-problems are solved. To investigate the trade-off between the optimality and the time performance, in the experiment section, we evaluate the following four algorithms. The routing sub-problems are solved by the search algorithm (2). These algorithms differ from each other on how the assignment sub-problem is solved and how the primal solution is extracted at each iteration. More specifically, we have:

- DLR-E: Both assignment and primal extraction are solved exactly by ILP formulation.
- DLR-H: This is the relaxed version of DLR-E that solves the assignment

sub-problem by linear relaxation and primal extraction by the greedy local search heuristics.

- SLR-E/ SLR-H: Each extends DLR-E and DLR-H respectively to handle duration uncertainties, by applying SAA in the routing search procedure.

In all the experiments, the route penalty γ is set to 80. The cut-off running time for all approaches is set to **10 minutes**, if not specified in experiments. Experiments were conducted on a machine with i7-4790 CPU@3.6GHz and 32 GB RAM.

5.5.3 Numerical Results

We first compare the scalability of exact ILP with DLR. After running CPLEX for exact ILP on small instances (e.g., (D, R, K)=(7, 359, 44)) for 2 hours, it turns out that optimality cannot be reached and running out of memory. Our approaches can return good solutions within 10 minutes even for large-scale instances up to size (D, R, K)=(7, 4203, 273).

Results on Deterministic HHCSP: We first compare the quality of the schedules produced by our LR-based approaches against the actual schedules derived from company-provided visit data. This instance is of size (D, R, K)=(7, 2062,199), where all the requests have fixed visiting days and no time windows. To compare the trade-off between generating more *valid* routes and assigning more *primary* providers, we test the instance with different $r^+ \in \{100, 50\}$. Results are summarized in Table (5.1). Metrics are measured in percentage of relative difference, normalized by those of the actual visits.

Table (5.1) shows that LR-based approaches improve the objective value by at least 10%. The number of route metric is measured in terms of route reduction. DLR-E generates *fewer routes* compared to the others, i.e., less *manpower* required from the company. In terms of *workload fairness*, we compare the number of *valid* routes generated by each approach. A route is valid

r^+	Objective		No. of Routes		No. of Valid Routes		Primary Assigned	
	DLR-E	DLR-H	DLR-E	DLR-H	DLR-E	DLR-H	DLR-E	DLR-H
100	20.32%	19.72%	-2.59%	0.00%	17.65%	0.74%	46.12%	47.25%
50	14.79%	12.27%	-13.45%	0.17%	52.21%	-3.68%	37.23%	46.65%

Table 5.1: Comparison of DLR-E and DLR-H against actual schedules on one instance of size (D, R, K)=(7, 2062, 199) with different r^+.

if it meets the minimum and maximum workloads. The more valid routes generated, the better. Again, DLR-E outperforms the rest. For *continuity of care*, we compare the total number of primary providers assigned. Both LR-based approaches substantially assign more primary providers compared to actual visits, especially DLR-H. Results also show that increasing r^+ biases the solutions towards more primary providers assigned, a smaller r^+ tends to generate solutions with fewer routes. This demonstrates the flexibility of our methods on generating solutions for different focuses.

Table (5.2) describes the key results on the performance comparison between the DLR-E and DLR-H on both *ins-tight* and *ins-loose* instances with time-windows and inter-visit dependencies. Results are averaged over 10 random instances of each instance type.

Instance	DLR-E		DLR-H		
type	gap%	tPerItr(s)	gap%	nGap%	tPerItr(s)
ins-tight	4.46%	137.01	10.52%	6.03%	53.72
ins-loose	0.27%	132.97	7.71%	3.08%	54.26

Table 5.2: Comparison on synthetic instances with time-windows, temporal dependencies and (D, R, K)=(7, 2062, 199).

The gap here refers to the gap between the best primal and the best dual solution found. Table (5.2) clearly shows that both DLR-E and DLR-H are able to get provably near-optimal solutions, with optimality gaps less than 4.46% for DLR-E and 10.52% for DLR-H. nGap represents the normalized gap, which is calculated as the percentage difference between the best primal solution found by DLR-H and the best dual solution found by DLR-E (tighter

lower bound). We can see that the actual optimality gaps for DLR-H should be smaller than nGap, i.e., less than 6% on both instances. tPerItr represents the runtime per iteration. DLR-E provides better solution quality while DLR-H provides a trade-off between solution quality and runtime, which finds good solutions within a shorter time.

Results on Stochastic Extension: Due to durational uncertainties, there is a probability that a request cannot be served within the time window or that a route may exceed its time budget. In this section, we examine the solution quality based on 1000 random duration realizations. Results are averaged over 10 random instances of each instance type and an instance is evaluated over all the realizations. In the experiments, we set our risk level α as 0.3. The smaller the α' we set, the more conservative we are against the representativeness of the selected samples. Meanwhile, increasing the sample size leads to better approximations for the real distributions, but less efficient. Based on our initial set of parameter experiments (omitted here due to space limit), we set α' as 0.15 and sample size as 60. Samples are generated using our sample selection heuristic.

We focus on two evaluation metrics: 1) Chance constraint violation ratio, the number of chance constraints that are violated, normalized by the total number; 2) Expected objective, the average objective value achieved by the solution. For these results, a chance constraint is considered as violated if more than $\alpha \cdot 1000$ times over the 1000 realizations ($\alpha = 30\%$ here), this constraint is not satisfied. The expected objective is calculated as the sum of provider-dependent rewards of all requests, whose time window chance constraints are not violated, averaged over the random instances.

We compare the stochastic extensions with both deterministic approaches, DLR-E and DLR-H. We evaluate the deterministic approaches with two set of duration input from the distributions: mean and maximum durations. We use the durations that are 2σ upper away from the means, i.e., 97.75% percentile,

	Expected Objective	TW Violations	TB Violations
DLR-E: mean	382600	76.7/1987.2	14/508.2
DLR-H: mean	387470	65.9/1991.3	11.6/531
DLR-E: max	319780	0/1597.4	0/512.2
DLR-H: max	340320	0/1708.6	0/551.9
SLR-E	388350	0.2/1938	0 /511
SLR-H	394090	0.1/1960	0 /534

Table 5.3: Results on synthetic instances of *ins-tight* with (D, R, K) = (7, 2062, 199). TW denotes time-window chance constraints, while *TB* refers to the time budget chance constraints.

to approximate the maximum values.

We can observe from Table (5.3), our deterministic approaches with mean-duration are sensitive to duration uncertainties on *ins-tight* instances, where time window chance constraints and time budget chance constraints can be easily violated. On the other hand, deterministic approaches with max-duration can guarantee services and travel. However, they suffer from worse expected objectives, where providers are underutilized. While both SLR-E and SLR-H, are robust towards the stochasticity, achieving almost no violations of the chance constraints. Stochastic approaches slightly outperform the deterministic counterparts with mean-duration on expected objective values and they substantially outperform the deterministic counterparts with max-duration on this metric. Thus, more sophisticated approaches, i.e., our stochastic approaches, to model uncertainty are warranted.

5.6 Summary

We investigate the multi-period home health care scheduling problem driven by the real-world needs. An integer linear programming model is proposed to formulate the deterministic problem. We further extend it with chance constraints to handle stochastic travel and service times, which is useful for real-world situations. Subsequently, to develop a solution that can scale to city-

scale scenarios, we apply the Lagrangian relaxation and exploit the separable problem structure to decompose the formulation into smaller sub-problems. Finally, we solve the chance constrained problem by applying sample average approximation. With this sampling based approach incorporated, as long as there is a distribution simulator driven by the historical duration data, we can provide proactive solutions, which react well to potential uncertainties.

Chapter 6

Conclusion and Future Work

In this thesis, we propose to investigate real-world decision support problems for agent management and coordination in urban environments, which inevitably involve agents with different behavior patterns and preferences and often require orienteering agents in such environments.

In Chapter 2, we give an overview of the OP, which can serve as the starting point to model a broad range of real-world problems. In Chapters 3–5, we specifically explore three application-driven problems that aim to manage agents with a personal touch. Chapter 2 handles the crowd control problem in leisure environments with non-cooperative agents by adapting the sampled fictitious play. Two problems studied in Chapters 3-4 share some commonality that cooperative agents are required to perform location-based tasks independently, and Lagrangian relaxation is applied in both cases. These two problems also differ significantly: while the mobile crowdsourcing problem focuses on task allocation, taking into consideration of the uncertain movement patterns, the workforce scheduling problem is concerned with generating the actual schedules by considering various requirements from both patients and providers.

In the following sections, we will provide brief summaries of the contributions and discuss potential future works that can be extended from this thesis.

6.1 Crowd Control

We addressed the crowd control problem in leisure environments, where individual agents aim to visit a sequence of selected attractions with the objective of maximizing their rewards according to their own preferences while observing individual's time budget limitations and attractions' capacity constraints. In this problem, we focused on identifying pure strategy Nash equilibrium solutions where individual agents cannot gain by deviation. Below, we identify several specific future research directions.

- In the current work, the initial computational experiments show great promises, as we are able to find pure strategy equilibrium in all the randomly generated instances for 2-agent, 5-agent, and 8-agent games. The immediate next step is to further improve the computational approach so that it can scale to even larger instances. In the proposed sampled fictitious play algorithm, each player's best response is computed by solving an exact ILP model using CPLEX. Exact methods from operation research literature, such as cutting plan, branch and bound and relaxation techniques, can be explored to speed up the best response computation.

- In leisure environments, visitors may exhibit similar preference patterns, which can be categories into different groups, such as thriller seekers, water lovers, and family with kids. While current work considers each individual visitor as an agent, another direction to explore is to aggregate the visitors into different flow groups with preferences and treat the aggregated groups as agents. Instead of looking for pure strategy equilibrium, we try to identify mixed strategy Nash equilibrium for agents.

- Current work assumes that we have complete information about the time-dependent payoff matrix. However, in reality, payoff matrix is often large, especially when more agents are involved in the games and complete

characterization is computationally intractable, e.g., each payoff value can only be estimated by running multiple time-consuming simulations. It is also possible that the payoffs are noisy. To handle these, one possible direction is to estimate games through simulation and sampling. We may pursue computationally tractable approaches such as empirical game-theoretic analysis framework to approximately find equilibria.

6.2 Mobile Crowdsourcing

We investigated the push-based mobile crowdsourcing where tasks are proactively recommended by the platform to workers, taking into account the uncertain movement patterns of the workers. The objective is to maximize rewards for all assigned tasks while ensuring that each individual's task-related detour from their routine routes stays within a specified time budget. The applicability of our push-based task recommendation was put to test in a campus-scale mobile crowdsourcing platform called TA$Ker. There are significant opportunities for further improvement and refinement.

- To make the task assignment algorithms applicable to a wider variety of problem settings, the algorithms have to be enhanced to deal with additional task-related constraints, such as task time windows, sequential dependencies among tasks, or finer-grained time separation requirements for recurring tasks. Worker-level considerations can also be incorporated, such as how to ensure the fairness among the mobile workers, how to synchronize mobile workers to perform some collaborative tasks, how to measure the quality of the work done by the mobile workers.

- The model can be further extended to consider the case of task bundles, where workers must pick an entire set of tasks. Such task bundling is becoming commonplace in urban delivery services as it helps workers

amortize their travel overheads. An open question is whether such bundles can be effectively formed without knowing each worker's trajectories, but based on the overall city-scale models of aggregate commuting flows.

- Most existing task assignment models focus on centrally assigning tasks from platform's perspective, which assumes that workers will perform the tasks as long as they are assigned. However, it is possible that workers may reject the tasks assigned due to various reasons. Thus, it is more efficient and effective to combine the centralized assignment model with worker selecting model so that workers may select the tasks they are good at and then compete with others to obtain the reward.

- Incentive mechanism plays an important role in maximizing the number of tasks performed for any mobile crowdsourcing platform. The most popular incentive currently for the crowd workers to perform the tasks is using the monetary reward. How should we price the tasks or task bundles such that workers are incentivized and task completion rate is maximized, is an interesting research question to be explored. Several directions may be explored, such as conventional micro-payment, group-oriented macro-rewards, elastic pricing models.

- Further empirical studies can be performed on the TA$Ker platform to systematically explore other facets of mobile crowdsourcing, including evaluating the performances of different assignment algorithms, comparing various incentive mechanisms, understanding the effects of pricing strategies.

6.3 Workforce Scheduling

We studied a weekly home health care scheduling problem that generates the start-of-the-day schedules for the service providers to visit patients at home,

with the objective to maximize rewards for the company. We considered the scenario driven by real-world needs, with a comprehensive set of considerations, e.g., time windows, continuity of care, workloads, inter-visit temporal dependencies, and especially the stochastic service and travel durations. This problem can be extended from following directions.

- While current model maximizes the total rewards collected by the health care company, other aspects may be explored for the objective function, such as minimizing distance traveled, reducing the manpower costs, or maximizing the service level. Different objectives can also be combined through various means, e.g., constructing weighted objective functions, considering different objectives in a lexicographic order, or approximating the Pareto optimal frontier.

- From the problem setting perspective, additional considerations may be incorporated, such as mandatory breaks (e.g., lunch breaks), synchronization of services that requires multiple providers serving the same requests simultaneously, and the utilization of part-time workers to supplement the full-time workforce. Aside from that, uncertain demands are frequently observed in the real life due to various reasons. Emergent requests may occur or services may be canceled at a short notice. When generating the schedules, it is important to also incorporate the demand uncertainty into the problem formulation.

- Most of the research on this problem focuses on the single modal transport, e.g., usually assuming the use of vehicles. However, the needs to combine multiple modes of transport (e.g., vehicles, bikes, public transports and walking) or sharing the resources are often observed in real life, especially at the urban centers where public transport system is convenient or vehicle resources are limited. The model can be further extended to multi-modal transport scenario to increase the efficiency of workers,

improve the utilization of vehicles and decrease the costs.

- Other heuristics can be also studied for this problem, such as iterative local search, tabu search, and large variable neighborhood search, to compare against our proposed approaches.

6.4 Challenges for Future Works

This thesis has direct relevance to the emerging concept of smart cities, which enables better coordination among interactive agents in usage of services/facilities in urban environments. To going further, we will build upon this thesis towards the following four general directions:

- **Handling more realistic urban challenges:** In the near future, we are interested in solving more realistic problems arising from the urban environments and scaling up to city scale. For example, mobile crowdsourcing is only applicable in urban cities, where worker pool is sufficient and there are increasing demands for services. A number of research questions are left unaddressed in this thesis. For instance, how to maintain worker pool so that the platform could sustain? How to handle various real-word considerations, such as congestions, to better utilize workforce in urban environments?

- **Coping with uncertainties**: We live in a world of uncertainty in many sense. It is important to deal with sources of uncertainties such as demand, supply, durations, breakdowns, cancellations of bookings, etc. How to generate robust solutions that that can be executed with high probability in uncertain urban environments remains to be studied.

- **Utilizing human data**: Mobile technology changes the way businesses engage with people. At the individual level, there is an increasing trend

in using mobile apps to generate schedule or recommendation. At the company level, people's digital traces can be aggregated and analysed to predict demand patterns, movements patterns or other preference information. How to better utilize massive human behaviour and preferences data collected through various means, such as mobile devices, to provide context-aware recommendations and incentivize the urban dwellers, is important to be further explored.

- **Moving from offline to online**: Lastly, the approaches presented in this thesis actually compute offline strategy to be executed online. The reactive online approach is a challenging direction which is not yet explored. In the future, we would like to close the loop by moving to online reactive algorithms to manage schedule throughout its execution, including incorporating unexpected events that occur throughout days.

Bibliography

[1] Orienteering, author=Wikipedia, howpublished = `https://en.wikipedia.org/wiki/orienteering`, note = Accessed: 2016-10-31.

[2] Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 13–22. ACM, 2010.

[3] Claudia Archetti, Alain Hertz, and Maria Grazia Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1): 49–76, 2007.

[4] Claudia Archetti, Martin W. P. Savelsbergh, and M. Grazia Speranza. The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, 254(2):472–480, 2016.

[5] Rym Ben Bachouch, Alain Guinet, and Sonia Hajri-Gabouj. A decision-making tool for home health care nurses planning. In *Supply Chain Forum: an International Journal*, volume 12, pages 14–20. Taylor & Francis, 2011.

[6] Roberto Baldacci, Enrico Bartolini, and Aristide Mingozzi. An Exact Algorithm for the Pickup and Delivery Problem with Time Windows. *Operations Research*, 59(2):414–426, 2011.

[7] Reuven Bar-Yehuda, Guy Even, and Shimon Moni Shahar. On approximating a geometric prize-collecting traveling salesman problem with time windows. *Journal of Algorithms*, 55(1):76–92, 2005.

[8] David Barrera, Nubia Velasco, and Ciro-Alberto Amaya. A network-based approach to the multi-activity combined timetabling and crew scheduling problem: Workforce scheduling for public health policy implementation. *Computers & Industrial Engineering*, 63(4):802–812, 2012.

[9] Sachidanand V Begur, David M Miller, and Jerry R Weaver. An integrated spatial dss for scheduling and routing home-health-care nurses. *Interfaces*, 27(4):35–48, 1997.

[10] Ashlea R Bennett and Alan L Erera. Dynamic periodic fixed appointment scheduling for home health. *IIE Transactions on Healthcare Systems Engineering*, 1(1):6–19, 2011.

[11] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[12] Hermann Bouly, Duc-Cuong Dang, and Aziz Moukrim. A memetic algorithm for the team orienteering problem. *4or*, 8(1):49–70, 2010.

[13] Sylvain Boussier, Dominique Feillet, and Michel Gendreau. An exact algorithm for team orienteering problems. *4OR: A Quarterly Journal of Operations Research*, 5(3):211–230, 2007.

[14] Ioannis Boutsis and Vana Kalogeraki. On task assignment for real-time reliable crowdsourcing. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, June 2014.

[15] John Bowers, Helen Cheyne, Gillian Mould, and Miranda Page. Continuity of care in community midwifery. *Health care management science*, 18(2):195–204, 2015.

[16] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[17] Edmund K Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *Journal of scheduling*, 7(6):441–499, 2004.

[18] Steven E Butt and Tom M Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1): 101–111, 1994.

[19] Vicente Campos, Rafael Martí, Jesús Sánchez-Oro, and Abraham Duarte. Grasp with path relinking for the orienteering problem. *Journal of the Operational Research Society*, 65(12):1800–1813, 2014.

[20] Paola Cappanera and Maria Grazia Scutellà. Joint assignment, scheduling, and routing models to home care optimization: a pattern-based approach. *Transportation Science*, 49(4).

[21] Giuliana Carello and Ettore Lanzarone. A cardinality-constrained robust model for the assignment problem in home care services. *European Journal of Operational Research*, 236(2):748–762, 2014.

[22] J Arturo Castillo-Salazar, Dario Landa-Silva, and Rong Qu. Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, 239(1):39–67, 2016.

[23] I-Ming Chao, Bruce L Golden, and Edward A Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475–489, 1996.

[24] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3): 464–474, 1996.

[25] Cen Chen, Shih-Fen Cheng, and Hoong Chuin Lau. The multi-agent orienteering problem. In *Tenth Metaheuristics International Conference*, Singapore, August 2013.

[26] Cen Chen, Shih-Fen Cheng, Aldy Gunawan, Archan Misra, Koustuv Dasgupta, and Deepthi Chander. TRACCS: Trajectory-aware coordinated urban crowd-sourcing. In *2nd AAAI Conference on Human Computation and Crowdsourcing*, pages 30–40, 2014.

[27] Xi Chen and Xiaotie Deng. Settling the complexity of two-player nash equilibrium. In *Foundations of Computer Science (FOCS)*, 2006.

[28] Shih-Fen Cheng, Daniel M Reeves, Yevgeniy Vorobeychik, and Michael P Wellman. Notes on equilibria in symmetric games. In *Proceedings of the 6th International Workshop On Game Theoretic And Decision Theoretic Agents GTDT 2004*, pages 71–78, 2004.

[29] Shih-Fen Cheng, Marina A. Epelman, and Robert L. Smith. CoSIGN: A parallel algorithm for coordinated traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):551–564, 2006.

[30] Clickz. Personalization helps amazon prevail. `https://www.clickz.com/personalization--helps--amazon--prevail/24250`, 2015. Accessed: 2016-10-22.

[31] J.-F. o. Cordeau and G. Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. pages 89–101, 2003.

[32] Duc-Cuong Dang, Rym Nesrine Guibadj, and Aziz Moukrim. A pso-based memetic algorithm for the team orienteering problem. In *European Conference on the Applications of Evolutionary Computation*, pages 471–480. Springer, 2011.

[33] Duc-Cuong Dang, Racha El-Hajj, and Aziz Moukrim. A branch-and-cut algorithm for solving the team orienteering problem. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 332–339. Springer, 2013.

[34] Duc-Cuong Dang, Rym Nesrine Guibadj, and Aziz Moukrim. An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229(2):332–344, 2013.

[35] Daniel Duque, Leonardo Lozano, and Andrés L Medaglia. Solving the orienteering problem with time windows via the pulse framework. *Computers & Operations Research*, 54:168–176, 2015.

[36] Andreas T Ernst, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, 153(1):3–27, 2004.

[37] A Errarhout, S Kharraja, and C Corbier. Two-stage stochastic assignment problem in the home health care. *IFAC-PapersOnLine*, 49(12): 1152–1157, 2016.

[38] Patrik Eveborn, Patrik Flisberg, and Mikael Rönnqvist. Laps care—an operational system for staff planning of home care. *European Journal of Operational Research*, 171(3):962–976, 2006.

[39] Dominique Feillet, Pierre Dejax, and Michel Gendreau. The profitable arc tour problem: solution with a branch-and-price algorithm. *Transportation Science*, 39(4):539–552, 2005.

[40] Aurora Fernandez, G Gregory, A Hindle, and AC Lee. A model for community nursing in a rural county. *Journal of the Operational Research Society*, 25(2):231–239, 1974.

[41] Christian Fikar and Patrick Hirsch. Home Health Care Routing and Scheduling: A Review. *Computers & Operations Research*, 77:86–95, 2017.

[42] Matteo Fischetti, Juan Jose Salazar Gonzalez, and Paolo Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998.

[43] Marshall L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, 27(1):1–18, 1981.

[44] Robert C Ford and Duncan R Dickson. Enhancing customer self-efficacy in co-producing service experiences. *Business Horizons*, 55(2):179–188, 2012.

[45] Drew Fudenberg and Jean Tirole. Game theory, 1991. *Cambridge, Massachusetts*, 393:12, 1991.

[46] Luca Maria Gambardella, Roberto Montemanni, and Dennis Weyland. Coupling ant colony systems with strong local searches. *European Journal of Operational Research*, 220(3):831–843, 2012.

[47] Michel Gendreau, Gilbert Laporte, and Frederic Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32(4):263–273, 1998.

[48] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106(2):539–545, 1998.

[49] Archis Ghate, Shih-Fen Cheng, Stephen Baumert, Daniel Reaume, Dushyant Sharma, and Robert L Smith. Sampled fictitious play for multi-action stochastic dynamic programs. *IIE Transactions*, 46(7):742–756, 2014.

[50] Bruce L. Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307—318, 1987.

[51] Bruce L Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval research logistics*, 34(3):307–318, 1987.

[52] Bruce L Golden, Qiwen Wang, and Li Liu. A multifaceted heuristic for the orienteering problem. *Naval Research Logistics (NRL)*, 35(3): 359–366, 1988.

[53] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Pure nash equilibria: hard and easy games. In *Journal of Artificial Intelligence Research*, pages 215–230, 2003.

[54] Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. An iterated local search algorithm for solving the orienteering problem with time windows. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 61–73. Springer, 2015.

[55] Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. Sails: Hybrid algorithm for the team orienteering problem with time windows. In *Proceedings of the 7th multidisciplinary international scheduling conference (MISTA 2015), Prague, Czech Republic*, pages 276–295, 2015.

[56] Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. Well-tuned ils for extended team orienteering problem with time windows. In *LARC Technical Report Series*. Singapore Management University, 2015.

[57] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 2016.

[58] Michael Held, Philip Wolfe, and Harlan P Crowder. Validation of subgradient optimization. *Mathematical programming*, 6(1):62–88, 1974.

[59] Alain Hertz and Nadia Lahrichi. A patient assignment algorithm for home care services. *Journal of the Operational Research Society*, 60(4): 481–495, 2009.

[60] Qian Hu and Andrew Lim. An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 232(2):276–286, 2014.

[61] Shenggong Ji, Yu Zheng, and Tianrui Li. Urban sensing based on human mobility. In *2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1040–1051, 2016.

[62] Patrick R. Jordan, Yevgeniy Vorobeychik, and Michael P. Wellman. Searching for approximate equilibria in empirical games. In *Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1063–1070, 2008.

[63] Thivya Kandappu, Archan Misra, Shih-Fen Cheng, Nikita Jaiman, Randy Tandriansyah, Cen Chen, Hoong Chuin Lau, Deepthi Chander, and Koustuv Dasgupta. Campus-scale mobile crowd-tasking: Deployment & behavioral insights. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, pages 800–812. ACM, 2016.

[64] Salil S Kanhere. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *12th IEEE International Conference on Mobile Data Management*, pages 3–6, 2011.

[65] Marisa G Kantor and Moshe B Rosenwein. The orienteering problem with time windows. *Journal of the Operational Research Society*, 43(6): 629–635, 1992.

[66] Marisa G. Kantor and Moshe B. Rosenwein. The orienteering problem

with time windows. *The Journal of the Operational Research Society*, 43 (6):629–635, 1992.

[67] Leyla Kazemi and Cyrus Shahabi. A privacy-aware framework for participatory sensing. *ACM SIGKDD Explorations Newsletter*, 13(1):43–51, 2011.

[68] Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *20th International Conference on Advances in Geographic Information Systems*, pages 189–198. ACM, 2012.

[69] Liangjun Ke, Claudia Archetti, and Zuren Feng. Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54(3): 648–665, 2008.

[70] Liangjun Ke, Laipeng Zhai, Jing Li, and Felix TS Chan. Pareto mimic algorithm: An approach to the team orienteering problem. *Omega*, 61: 155–166, 2016.

[71] C Peter Keller. Algorithms to solve the orienteering problem: A comparison. *European Journal of Operational Research*, 41(2):224–231, 1989.

[72] Morteza Keshtkaran, Koorush Ziarati, Andrea Bettinelli, and Daniele Vigo. Enhanced exact solution methods for the team orienteering problem. *International Journal of Production Research*, 54(2):591–601, 2016.

[73] Paulien M Koeleman, Sandjai Bhulai, and Maarten van Meersbergen. Optimal patient and personnel scheduling policies for care-at-home service facilities. *European Journal of Operational Research*, 219(3):557–563, 2012.

[74] Nacima Labadie, Jan Melechovskỳ, and Roberto Wolfler Calvo. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics*, 17(6):729–753, 2011.

[75] Nacima Labadie, Renata Mansini, Jan Melechovskỳ, and Roberto Wolfler Calvo. The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15–27, 2012.

[76] Theodore J. Lambert III, Marina A. Epelman, and Robert L. Smith. A fictitious play approach to large-scale optimization. *Operations Research*, 53(3):477–489, 2005.

[77] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete applied mathematics*, 26(2-3):193–207, 1990.

[78] Adrienne C Leifer and Moshe B Rosenwein. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research*, 73(3):517–523, 1994.

[79] Carlton E Lemke and Joseph T Howson, Jr. Equilibrium points of bimatrix games. *Journal of the Society for Industrial & Applied Mathematics*, 12(2):413–423, 1964.

[80] Yun-Chia Liang, Sadan Kulturel-Konak, and Alice E Smith. Meta heuristics for the orienteering problem. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 384–389. IEEE, 2002.

[81] Yun-Chia Liang, Sadan Kulturel-Konak, and Min-Hua Lo. A multiple-level variable neighborhood search approach to the orienteering problem. *Journal of Industrial and Production Engineering*, 30(4):238–247, 2013.

[82] Meiyan Lin, Kwai Sang Chin, Xianjia Wang, and Kwok Leung Tsui. The therapist assignment problem in home healthcare structures. *Expert Systems with Applications*, 62:44–62, 2016.

[83] Shih-Wei Lin and F Yu Vincent. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1):94–107, 2012.

[84] R Mansini, M Pelizzari, and R Wolfer. A granular variable neighbourhood search heuristic for the tour orienteering problem with time windows. Technical report, Technical Report RT 2006-02-52, University of Brescia, Italy, 2006.

[85] Yannis Marinakis, Michael Politis, Magdalene Marinaki, and Nikolaos Matsatsinis. A memetic-grasp algorithm for the solution of the orienteering problem. In *Modelling, Computation and Optimization in Information Systems and Management Sciences*, pages 105–116. Springer, 2015.

[86] Artis Mednis, Girts Strazdins, Reinholds Zviedris, Georgijs Kanonirs, and Leo Selavo. Real-time pothole detection using android smartphones with accelerometers. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops*, 2011.

[87] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[88] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: Rich monitoring of road traffic conditions using mobile smartphones. In *7th ACM Conference on Embedded Networked Sensor Systems*, 2008.

[89] Dov Monderer and Lloyd S Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996.

[90] R. Montemanni and L. Gambardella. Ant colony system for team orienteering problems with time windows. *Foundations of Computing and Decision Sciences*, 34(4):287—306, 2009.

[91] Mohamed Musthag and Deepak Ganesan. Labor dynamics in a mobile micro-task market. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 641–650, 2013.

[92] Shanthi Muthuswamy and Sarah S Lam. Discrete particle swarm optimization for the team orienteering problem. *Memetic Computing*, 3(4): 287–303, 2011.

[93] National Association for Home Care & Hospice. Basic statistics about home care. *Washington, DC: National Association for Home Care & Hospice*, pages 1–14, 2010.

[94] United Nations. World urbanization trends 2014: Key facts. `https://esa.un.org/unpd/wup/Publications/Files/WUP2014-Highlights.pdf`, 2008. Accessed: 2016-10-22.

[95] Stefan Nickel, Michael Schröder, and Jörg Steeg. Mid-term and short-term planning support for home health care services. *European Journal of Operational Research*, 219(3):574–587, 2012.

[96] BK Pagnoncelli, Shabbir Ahmed, and A Shapiro. Sample average approximation method for chance constrained programming: theory and applications. *Journal of optimization theory and applications*, 142(2): 399–416, 2009.

[97] C. Papadimitriou. Algorithms, games, and the internet. In *Symposim on Theory of Computing (STOC)*, pages 749–753, 2001.

[98] Population Reference Bureau. Fact Sheet: Aging in the United

States. http://www.prb.org/Publications/Media-Guides/2016/aging-unitedstates-fact-sheet.aspx, 2016.

[99] Abdur Rais and Ana Viana. Operations research in healthcare: a survey. *International transactions in operational research*, 18(1):1–31, 2011.

[100] R Ramesh, Yong-Seok Yoon, and Mark H Karwan. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing*, 4(2): 155–165, 1992.

[101] Ram Ramesh and Kathleen M Brown. An efficient four-phase heuristic for the generalized orienteering problem. *Computers & Operations Research*, 18(2):151–165, 1991.

[102] Rajib Kumar Rana, Chun Tung Chou, Salil S Kanhere, Nirupama Bulusu, and Wen Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 105–116, 2010.

[103] Matias Sevel Rasmussen, Tor Justesen, Anders Dohn, and Jesper Larsen. The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. *European Journal of Operational Research*, 219(3):598–610, 2012.

[104] Giovanni Righini and Matteo Salani. Dynamic programming for the orienteering problem with time windows. *Note del Polo-Ricerca*, 91, 2006.

[105] Giovanni Righini and Matteo Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4):1191–1203, 2009.

[106] Carlos Rodriguez, Thierry Garaix, Xiaolan Xie, and Vincent Augusto.

Staff dimensioning in homecare services with uncertain demands. *International Journal of Production Research*, 53(24):7396–7410, 2015.

[107] Tim Roughgarden and Eva Tardos. How bad is selfish routing? *Journal of ACM*, 49(2):236–259, 2002.

[108] John P. Rula, Vishnu Navda, Fabián E. Bustamante, Ranjita Bhagwan, and Saikat Guha. No "one-size fits all": Towards a principled approach for incentives in mobile crowdsourcing. In *15th Workshop on Mobile Computing Systems and Applications*, pages 3:1–3:5, 2014.

[109] Adam Sadilek, John Krumm, and Eric Horvitz. Crowdphysics: Planned and opportunistic crowdsourcing for physical tasks. In *7th International AAAI Conference on Weblogs and Social Media*, pages 536–545, 2013.

[110] Tuomas Sandholm, Andrew Gilpin, and Vincent Conitzer. Mixed-integer programming methods for finding nash equilibria. In *National Conf. on Artificial Intelligence (AAAI))*, 2005.

[111] Michael Schilde, Karl F Doerner, Richard F Hartl, and Guenter Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201, 2009.

[112] Zülal Sevkli and F Erdogan Sevilgen. Discrete particle swarm optimization for the orienteering problem. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[113] Yufen Shao, Jonathan F Bard, and Ahmad I Jarrah. The therapist routing and scheduling problem. *IIE Transactions*, 44(10):868–893, 2012.

[114] Wouter Souffriau, Pieter Vansteenwegen, Joris Vertommen, Greet Vanden Berghe, and Dirk Van Oudheusden. A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, 22(10):964–985, 2008.

[115] Wouter Souffriau, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. A path relinking approach for the team orienteering problem. *Computers & Operations Research*, 37(11):1853–1859, 2010.

[116] Wouter Souffriau, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 47(1):53–63, 2013.

[117] Matthias Stevens and Eliie D'Hondt. Crowdsourcing of pollution data using smartphones. In *Workshop on Ubiquitous Crowdsourcing, held at 12th ACM Conference on Ubiquitous Computing*, 2010.

[118] Hao Tang and Elise Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407, 2005.

[119] M Fatih Tasgetiren. A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic and Social Research*, 4(2):1–26, 2001.

[120] Jacob Thebault-Spieker, Loren G. Terveen, and Brent Hecht. Avoiding the south side and the suburbs: The geography of mobile crowdsourcing markets. In *18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 2015.

[121] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *7th ACM Conference on Embedded Networked Sensor Systems*, 2009.

[122] Tommy Thomadsen and Thomas K Stidsen. The quadratic selective travelling salesman problem. Technical report, 2003.

[123] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[124] Andrea Trautsamwieser and Patrick Hirsch. A branch-price-and-cut approach for solving the medium-term home health care planning problem. *Networks*, 64(3):143–159, 2014.

[125] Martin Romauch Karl F. Doerner Tricoire, Fabien and Richard F. Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers and Operations Research*, 37(2):351–367, 2010.

[126] Khai N. Troung, Thariq Shihipar, and Daniel J. Wigdor. Slide to x: Unlocking the potential of smartphone unlocking. In *32nd SIGCHI Conference on Human Factors in Computing Systems*, 2014.

[127] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[128] T. Tsiligirides. Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35(9):797–809, 1984.

[129] Rajan Vaish, Keith Wyngarden, Jingshu Chen, Brandon Cheung, and Michael S. Bernstein. Twitch crowdsourcing: Crowd contributions in short bursts of time. In *32nd SIGCHI Conference on Human Factors in Computing Systems*, 2014.

[130] Pieter Vansteenwegen. Planning in tourism and public transportation-attraction selection by means of a personalised electronic tourist guide and train transfer scheduling. 2008.

[131] Pieter Vansteenwegen and Dirk Van Oudheusden. The mobile tourist guide: an or opportunity. *OR insight*, 20(3):21–27, 2007.

[132] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1): 118–127, 2009.

[133] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290, 2009.

[134] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10, 2011.

[135] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[136] Jing Wang, Siamak Faridani, and Panagiotis Ipeirotis. Estimating the completion time of crowdsourced tasks using survival analysis models. In *Workshop on Crowdsourcing for Search and Data Mining*, 2011.

[137] Qiwen Wang, Xiaoyun Sun, Bruce L Golden, and Jiyou Jia. Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research*, 61(1):111–120, 1995.

[138] Xia Wang, Bruce L Golden, and Edward A Wasil. Using a genetic algorithm to solve the generalized orienteering problem. In *The vehicle routing problem: latest advances and new challenges*, pages 263–274. Springer, 2008.

[139] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a

path using sparse trajectories. In *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 25–34, 2014.

[140] Michael P. Wellman, Daniel M. Reeves, Kevin M. Lochner, Shih-Fen Cheng, and Rahul Suri. Approximate strategic reasoning through hierarchical reduction of large symmetric games. In *Twentieth National Conference on Artificial Intelligence*, pages 502–508, 2005.

[141] Tingxin Yan, Matt Marzilli, Ryan Holmes, Deepak Ganesan, and Mark Corner. mCrowd: A platform for mobile crowdsourcing. In *7th ACM Conference on Embedded Networked Sensor Systems*, 2009.

[142] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *18th Annual International Conference on Mobile Computing and Networking*, 2012.

[143] Biao Yuan, Ran Liu, and Zhibin Jiang. A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill requirements. *International Journal of Production Research*, 53(24):7450–7464, 2015.

[144] Zhi Yuan and Armin Fügenschuh. Home health care scheduling: a case study. In *proceedings of the 7th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2015), 25 - 28 Aug 2015, Prague, Czech Republic*, pages 555–569, 2015.

[145] Alexandros Zenonos, Sebastian Stein, and Nicholas R Jennings. Coordinating measurements for air pollution monitoring in participatory sensing settings. In *14th International Conference on Autonomous Agents and Multiagent Systems*, pages 493–501, 2015.

[146] Yu Zheng, Tong Liu, Yilun Wang, Yanmin Zhu, Yanchi Liu, and Eric

Chang. Diagnosing new york city's noises with ubiquitous data. In *2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 715–725, 2014.