國立清華大學電機資訊學院資訊工程研究所
碩士論文
Department of Computer Science
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master Thesis

研究行動群眾感知系統下的繞行規劃問題
Detour Planning Problem on Mobile Crowdsensing Systems

廖宸誌
Chen-Chih Liao

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 104 年 6 月
June, 2015

國立清華大學
資訊工程研究所

碩士論文

研究行動群眾感知系統下的繞行規劃問題

廖宸誌 撰

1046

中文摘要

群眾感知系統是近年流行的一種工作外包的平台，此系統將感測或多媒體工作外包給有行動裝置 的工人。在群眾感知系統中，工人使用智慧型手機來工作，這些智慧型手機具備了能夠執行感測 以及多媒體工作的能力，例如偵測噪音或拍照。我們為了有效的分配感測以及多媒體工作給工人 而提出了一個群眾感知系統，並且我們只專注於時空相關的工作，這種工作必須在特定的地點以 及時間來被執行。每個工人會提供他的目的地以及預計到達的時間給我們的系統，並且這些工人 為了最大化利潤不介意遵從我們的繞行規劃路線來執行工作。一旦工人上傳相關的資訊道系統上， 工人將會收到各自的繞行規劃路線，而每條繞行規劃路線都是由一群工作組成並且有特定的順序 讓工人去遵循。工人若是依循繞行規劃路線執行工作，那他將會收到他所能獲得的最大的利潤。 我們稱上述的問題為繞行規劃問題，並且又提出一個更複雜的多人繞行規劃問題。這兩個問題的 差別只在於繞行規劃問題每次運算繞行規劃路線時只考慮一個工人。在本篇論文中，我們提出了 繞行規劃演算法 (DP) 以及多人繞行規劃演算法 (MDP) 去解決我們提出的兩個問題。我們使用真 實資料去驅動模擬器，其模擬結果也顯示我們演算法的可行性以及效率。在未來我們將會實作此 系統，以及解決其中多個困難的挑戰。

Abstract

Crowdsensing is a popular paradigm that outsources sensory/multimedia tasks to mobile workers. In the crowdsensing systems, workers perform diverse tasks such as detecting sensory data and taking pictures by employing (using) their smartphones, which are equipped with sensing and multimedia functions. We provide a crowdsensing system to efficiently delegate sensory/multimedia tasks to mobile workers, and we focus on spatial-temporal tasks that must be conducted at specific locations and time. Each worker supplies his/her destination with a deadline to our system and does not mind taking detour paths to maximize profits. Once workers submit their profiles to our system, they will receive detour paths, which consist of tasks in particular orders. Workers execute tasks by following their detour paths and receive maximal profits. We formulate this problem as a detour planning problem, and the advanced problem is multi-users detour planning problem. The difference between these two problems is that the detour planning problem just considers a worker at a time. In this thesis, we develop a detour planning algorithm (DP) and a multi-users detour planning algorithm (MDP) to solve problems respectively. We simulate the extensive trace-driven scenarios and demonstrate the effectiveness and efficiency of our algorithms. Developing a working prototype on Android OS and addressing other challenging aspects of the considered systems are our future tasks.

Contents

List of Figures

List of Tables


TSING HUA UNIVERSITY

Chapter 1

Introduction

1.1 Motivation

Nowadays, smartphone users are ubiquitous, and the smartphones have equipped with a lot of sensors, such as the GPS reader, the microphone, the camera, and etc. The prospect of smartphones has huge potential, if we can make good use of these sensors. For example, if smartphone users have free time, they can use their smartphones to perform some tasks. Smartphone users can take a picture or read sensory data in their leisure time, but they may not volunteer to do these tasks. Therefore, we need some incentives [33] (e.g. scores, and rewards) to encourage smartphone users to contribute their efforts. Due to the possibility of smartphones, we propose the *mobile crowdsensing (MCS) system* . In general, *crowdsensing* [7] systems collect sensory data from crowds, which are especially network-based, and they analyze sensory data to understand the circumstance of a specific area. Thus, it utilizes mobile sensing (e.g. opportunistic sensing, and participatory sensing) and human-in-the-loop for gathering a large number of sensory data. Moreover, crowdsensing is regarded as the extension of *crowdsourcing*, because of the concept that it assigns outsourcing tasks to uncertain crowds. Crowdsourcing refers to a distributed problem solving paradigm. In the crowdsourcing system, *requesters* submit their *requests* into the platform. Requesters can be companies, organizations, or individuals, and requests include various requests (e.g., shooting photos, recording videos, or collecting sensory data). These requests are traditionally performed by employees. However, now these requests are outsourced to public crowds who are willing to complete them for incentives, also known as *rewards*. Furthermore, each request is associated with a *deadline* and an amount of reward. People who accept the requests are referred to as the *workers*. Workers earn rewards only if they complete the corresponding requests before their deadlines. In our MCS system, we hire mobile smartphone users as workers, because we focus on spatial-temporal requests, which must be performed at specific locations and time. In this

thesis, we discuss the main problem, *detour planning problem*, on the MCS system. The problem is related to the *orienteering problem*. Next, we will introduce the orienteering problem in sec. 1.2, and then we discuss the research problems in sec. 1.3.

1.2 Orienteering Problem

We introduce the orienteering problem in the following. This problem comes from a popular outdoor sport which is similar to treasure hunt. In this sport, each player is assigned to an identical map. There are notations of rewards (e.g., money, or scores) on the map. The goal of the game is to collect as many rewards as possible. However, the time is limited. The players need to achieve the goal before the game is over. When they are finding their own path, they need to take the distance and the value of the rewards into consideration. In short, the orienteering problem is to plan the path, and the workers follow their paths to collect maximal rewards during the game. The path planning is important in this game, and there are already many literatures discussing how to solve this problem [28]. We are inspired by the orienteering problem. In the following sections, we describe our research problems in detail.

1.3 Research Problems

1.3.1 Detour Planning Problem

We study a new class of crowdsensing systems for serious applications of multimedia content gathering [16]. In the considered systems, users submit geospatial- and temporal-dependent requests to collect multimedia content, such as sensor readings from sparsely-deployed nodes, recorded videos of specific events, and photos of sightseeing sites. The corresponding users could be utility companies who need to retrieve smart-meter readings, police departments who need to collect evidence of criminal scenes, people who need photos of memorable locations for pleasure, and so on. The workers are smartphone users who are heading to their final *destinations*, but have some time to spare. The workers would not mind to take some detour paths for small rewards, which can be monetary, credits, or points, as long as they can reach the destinations in time.

The considered systems are unique because the requests are associated with deadlines and geospatial locations, while it takes some time for the workers to travel from the location of one request to that of another one. Moreover, existing systems require workers to *compete* against others for requests. We argue that the approach is not effective for our crowdsensing systems because: (i) while workers have time to take detour paths, they are

not capable coming up with the best detour paths and (ii) too many competitions may lead to excessive duplicated completion and wasted efforts, which degrades the overall system performance in the end.

In this thesis, we design a *detour planning algorithm (DP)* for our crowdsensing system. The proposed algorithm runs on a server, and generates a detour path for each worker. The resulting detour path is optimal in the sense that it maximizes his/her received profits, while guaranteeing the worker can reach his/her destination in time. We implement the proposed algorithm in a simulator, and compare its performance against several baseline algorithms. We also crawl the location traces of a large number of photos from Flickr [6], and use the traces to drive the simulator. The simulation results show that our algorithm: (i) outperforms the other algorithms by up to 100% improvement, (ii) runs efficiently and always terminates in 82 ms, and (iii) is able to scale to large problems.

1.3.2 Multi-users Detour Planning Problem

In the previous section, we only consider single user at a time. However, there are usually multiple users exist simultaneously in a real system. Under the premise, we use DP to compute detour paths. We may find that some users go far away to perform the requests which are closer to other users, and a small number of users receive the most requests. The case leads to the result that most workers are idle, so they may leave the system. Though some workers can receive optimal paths from the results of DP, it degrades the performance of the overall system. Thus, we propose to make good use of all users and reduce the traveling cost of each user. The *traveling costs* include the time the user spent and the fuel he/she consumed to move to the targeted destination. We refer to this problem as *multi-users detour planning problem*, which is more complicated than the detour planning problem.

In this system, we consider how to balance the burden among multiple users. Furthermore, we consider the *battery level* of the smartphones and the *accuracy* of the smartphone sensors [11, 15]. We choose the feasible workers who are able to perform requests within the constraint of his/her battery level and satisfy the *quality* requirement of requests.

We design a *multi-users detour planning algorithm (MDP)* to solve this problem. MDP is a heuristic algorithm which can compute the solution in real-time. In MDP, we use a *utility function* to figure out which user is the best choice to a request, and then MDP return results in real-time. Because users cannot wait long in a practical system, they may leave the system due to lack of the chances to earn rewards. The simulation results show that MDP: (i) achieves at most 1.4 times the profit of DP and 2.9 times the profit of the baseline, (ii) saves up to 51% energy compared to DP, (iii) achieves almost

100% completed requests ratio if workers are sufficient, and (iv) runs in the real-time (< 21s).

1.4 Contributions of Thesis

The proposed MCS system makes the following contributions:

1. We address the detour planning problem on the MCS system. The detour planning problem is more general than the orienteering problem with time window (OPTW), because the detour planning problem further considers the *feasible spots*, *traveling cost*, *energy consumption*, and *the accuracy*.

2. We formulate the (single-user/multi-users) detour planning problem on the MCS system. MCS maximizes rewards of the system by computing a detour path for each worker, and it may attract more workers to contribute their efforts within their available time. Moreover, each worker could arrive at their destinations on time. For the single-user detour planning problem, we propose detour planning algorithm (DP) to optimally solve this problem, and DP is based on dynamic programming. As for the multi-users detour planning problem, we propose multi-users detour planning algorithm (MDP) to efficiently assign requests to workers.

3. We implement traced-driven simulators to simulate the MCS system. The evaluation results show that the MCS system is suitable to be implemented to a practical system. Our MDP algorithm can achieve near optimal solution, and the running time is in real time. Thus, MCS well utilizes each worker with the best utility value and encourages as many workers to participate in our system.

4. We found that the MCS system is also appropriate for *urban computing*, and the proposed solutions could help the city planners to solve the urban problems. The city planners can issue his/her requests (e.g. traffic congestion, air pollution, and noise detection), and workers can help to collect these data for sustainable urban futures.

Chapter 2

Related Work

2.1 Crowdsourcing

The crowdsourcing paradigm has been employed in the literature to solve various real-life problems. Many of these studies leverage smartphones for their high penetration rate. For example, Li et al. [14] design a distributed question and answer system, called SOS, for smartphone users. SOS employs crowdsourcing to find the answers for the questions that can not be addressed by search engines. It also leverages social networks to propagate the unanswered questions via friends' connections. Yan et al. [32] adopt crowdsourcing to develop an information sharing system, called CrowdPark. A driver who will need a parking spot soon may use his/her smartphone to reserve it in advance. For a driver who is vacating a parking spot, he/she can notify the CrowdPark server for selling the parking spot to incoming drivers. Crowdsourcing paradigm has also appeared in commercial systems. For example, Roamler [20] allows companies to define different tasks for iOS users. The users install an iOS application to receive and complete tasks so as to earn money or points. The Roamler server pushes tasks to users based on their preferences and locations. Different from our work, the works in [14, 32, 34, 35] and commercial systems [20] concentrate on building the crowdsourcing platforms, which facilitate interactions between the requesters and workers, but do not *guide* the workers for more efficient decisions.

Some other crowdsourcing systems guide the workers for better decisions. For example, Yuen et al. [36] study the task matching problem in crowdsourcing. Their system generates a task list for each worker based on his/her preference and historical performance, which helps workers to concentrate on completing tasks rather than searching for ideal tasks. Different from our work, the problem in [36] is not geospatial-dependent. Boutsis and Kalogeraki [2,3] propose a real-time task assigning algorithm, which efficient assigns tasks to the crowd. The crowd may achieve high quality results and return it in real-time.

Even they consider the geospatial-dependent tasks, they won't assign a detour path and the tasks without time-window. Bassem and Bestavros [1] tackle a Geo-temporal Request Satisfaction (GRS) problem from a game-theoretic perspective, and propose a heuristic algorithm that works as follows. The algorithm derives a path from the starting location to the final destination using Yen's ranking algorithm [17], and chooses the task with the highest reward on this path. The set up in [1] is quite different from ours, because they allow workers to compete for tasks. In contrast, we believe that *coordinated* detour planning for minimizing wasted worker efforts is critical to the overall system performance. Moreover, we take the properties of mobile multimedia into considerations while designing the system. Last, their heuristic algorithm does not give optimal paths.

2.2 Crowdsensing

Several studies [4, 9, 22, 25, 31, 31] discuss the crowdsensing with smartphones. Xiao et al. [31] discuss large-scale mobile crowdsensing systems, and they address the barriers and make potential solutions. Sherchan et al. [22] propose an architecture for mobile crowdsensing, and they focus on efficiently collecting and analyzing mobile sensory data. Talasila et al. [25] use image processing techniques to validate mobile sensors and increase the reliability of sensed data. Cardone et al. [4] propose a geo-social crowdsensing platform to foster mobile users by their profiles to sense the environment of cities. Hasenfratz et al. [9] use off-the-shelf sensors and smartphones to sense the air pollution. They focus on efficient collecting the data, and the difference is that they didn't discuss how to guide workers to follow detour paths for maximal profits.

There are some crowdsensing works [5, 12, 23] which focus on location dependent tasks. He et al. [23] design an approximation algorithm to pursuit maximal rewards from location dependent tasks. They also propose a pricing mechanism for attracting smartphone users. Feng et al. [5] design a mechanism to assign location dependent tasks to smartphone users, and the objective is to minimize the cost (e.g., power consumption) of smartphones. Jaimes et al. [12] design a mechanism to greedily select location-based workers with a fixed power budget, and their algorithm maximize the sensing coverage of the target area. Even they are also dealing with the location dependent tasks, but their tasks didn't with time-window. Moreover, they won't simultaneously consider the reward, the cost (e.g., energy cost, and traveling cost), and the accuracy of tasks.

Chapter 3

Detour Planning Problem

3.1 Formulate the Detour Planning Problem

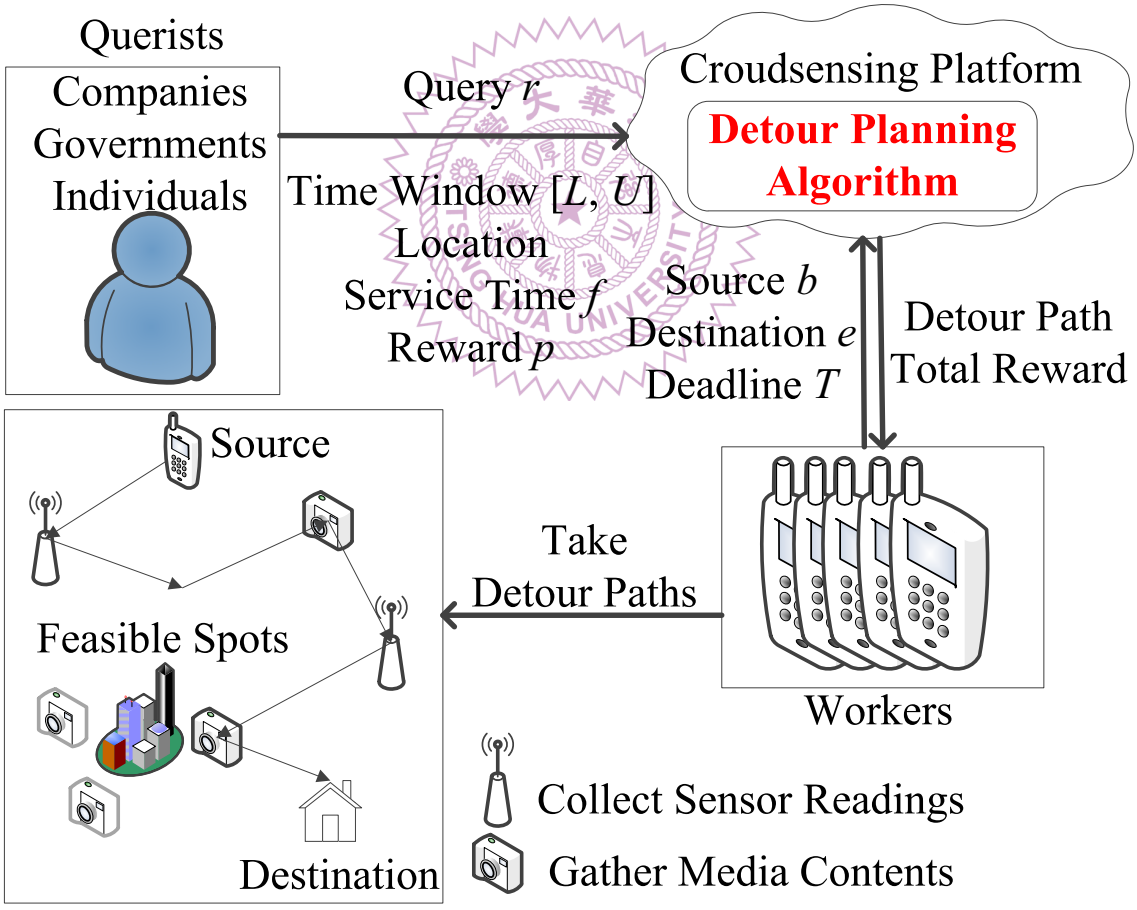


Figure 3.1: The considered detour planning problem. This thesis concentrates on multimedia content gathering.

Fig. 3.1 illustrates the considered crowdsensing system, which consists of three entities: (i) the platform, (ii) requesters, and (iii) workers. Let N be the number of to-

tal requests that haven't been assigned to any workers. A request sends a request r_i ($1 \leq i \leq N$) with a time window $[L_i, U_i]$, reward p_i, location, and service time f_i to the platform using a web site or via a smartphone application. The requesters can be companies, governments, and individuals. If r_i is completed by a worker between time L_i and U_i, that worker receives reward p_i. A worker sends his/her current, or *source*, location b, destination location e, and deadline T to the crowdsensing platform. Whenever there is a new worker, the crowdsensing platform computes the detour path $\mathbf{X}$ for the worker using the *detour planning* (DP) algorithm proposed in Sec. 3.2. Moreover, a new detour path is generated if a worker is late or lost when following the previously computed detour path. The resulting detour path allows the worker to maximize his/her total reward, and ensures the worker to arrive the destination by the specific deadline. Upon receiving the detour path, a worker follows it to complete the individual requests.

Requesters may submit several types of requests. Fig. 3.1 shows two representative types: (i) sensor data collection and (ii) media data gathering. Examples of the latter type include taking a photo of a landmark, which may be done at multiple close-by locations referred to as *feasible spots*. We let $z_i \geq 1$ ($1 \leq i \leq N$) be the number of feasible spots of request r_i. We let f_i be the *service time* of request r_i, i.e., the amount of time a worker has to spend at r_i. To plan the detour paths, we also need to know that travel time between any two feasible spots. We let m_{i_x,j_y} be the travel time from feasible spot x of request r_i to feasible spot y of request r_j, and collectively write this distance matrix as $\mathbf{M}$. Similarly, c_{i_x,j_y} and $\mathbf{C}$ represent the *cost* of traveling from a request to another, which can be attributed to gas cost and car depreciation rate. Both $\mathbf{M}$ and $\mathbf{C}$ are predetermined, for example, by using Google Map and other online maps. $\mathbf{E}$ is a large constant.

Next, we define the decision variables and some intermediate variables for the considered worker assigning problem. We use boolean variables to represent the path $\mathbf{X}$. Specifically, $x_{i,j} = 1$ if the detour path leads a worker moving from request i to j; and $x_{i,j} = 0$ otherwise. We also use s_i to represent the worker's planned arrival time at the location of request r_i. We let $u_{i,j} = 1$ ($1 \leq i \leq n$, $1 \leq j \leq z_i$) if feasible spot j of request r_i is on the detour path; and $u_{i,j} = 0$ otherwise.

Inspired by the formulations given in Vansteenwegen et al. [28], we mathematically formulate our worker assigning problem as:

$$\max \sum_{i=1}^{N-1}\sum_{j=2}^{N}[p_i - \sum_{a=1}^{z_i}\sum_{b=1}^{z_j} c_{i_a,j_b} u_{i,a} u_{j,b}]x_{i,j} \tag{3.1}$$

$$s.t. \sum_{j=2}^{N} x_{1,j} = \sum_{i=1}^{N-1} x_{i,N} = 1 \tag{3.2}$$

$$\sum_{i=1}^{N-1} x_{i,k} = \sum_{j=2}^{N} x_{k,j} \leq 1, \forall k = 2, ..., N-1 \tag{3.3}$$

$$\sum_{i=1}^{N-1}\sum_{j=2}^{N}(\sum_{a=1}^{z_i}\sum_{b=1}^{z_j} m_{i_a,j_b} u_{i,a} u_{j,b} + f_i)x_{i,j} \leq T_{max} \tag{3.4}$$

$$\sum_{j=1}^{z_i} u_{i,j} \leq 1, \forall i = 1, ..., N \tag{3.5}$$

$$s_i + f_i + \sum_{a=1}^{z_i}\sum_{b=1}^{z_j} m_{i_a,j_b} u_{i,a} u_{j,b} - s_j \leq E(1 - x_{i,j}), \forall i, j = 1, ..., N \tag{3.6}$$

$$L_i \leq s_i, \forall i = 1, ..., N \tag{3.7}$$

$$s_i + f_i \leq U_i, \forall i = 1, ..., N \tag{3.8}$$

$$x_{i,j}, u_{i,j} \in \{0, 1\}. \tag{3.9}$$

The objective function Eq. (3.1) is to maximize the total collected reward minus travel cost, which is referred to as profit. The constraints in Eq. (3.2) ensure that the path starts from request 1 to request N. The constraints in Eq. (3.3) make sure that every feasible spot is visited once. The constraints in Eq. (3.4) ensure that the total time of each path doesn't exceed the deadline specified by each worker. The constraints in Eq. (3.5) make sure that only one feasible spot of each request is visited. The constraints in Eq. (3.7) set the timeline of the path. The constraints in Eqs. (3.8) and (3.9) consider the worker's arrival time and service time.

Hardness of our problem. The worker assigning problem is a *generalized* version of the *orienteering problem* (OP), which computes a path to visit some locations in order to maximize the profit and arrive at the destination in time. Golden et al. [8] show that OP problems are NP-hard, and Vansteenwegen et al. [28] present four OP variations: (i) single worker, (ii) multiple workers, (iii) single worker with time-windowed requests, and (iv) multiple workers with time-windowed requests. The first two variations (without time-windowed requests) have been well studied in the literature, e.g., Schilde et al. [21] and Vansteenwegen et al. [27] propose algorithms to solve single- and multiple-worker OP problems without time-windowed requests. To our best knowledge, OP problems with time-windowed requests have not been throughly studied. The two most recent works are: Righini and Salani [19] and Montemanni and Gambardella [18], which solve the

Detour Planning (DP) Algorithm

Let $\alpha_{i,j} = < \mathbf{V}_{\emptyset}, 0, 0, \mathbf{u}_{\emptyset} >, \forall \alpha_{i,j} \in \chi$
Let $\alpha_{1,1}.\mathbf{V} = \alpha_{1,1}.\mathbf{V} \cup \{v_{1,1}\}$; $\alpha_{1,1}.\mathbf{u} = \alpha_{1,1}.\mathbf{u} \cup \{u_{1,1} = 1\}$
Enqueue $\alpha_{1,1}$ into $\mathbf{E}$ //E: vertices to visit
while $\mathbf{E} \neq \emptyset$ **do**
  Dequeue $\alpha_{i,x} \in \mathbf{E}$
  //Extend the $\alpha_{i,x}$ to a new $\alpha_{j,y}$
  for each neighboring $\alpha_{j,y} \in \chi \backslash \{\alpha_{i,k}\}_{1 \leq k \leq z_i}$ **do**
    Let $temp.\mathbf{V} = \alpha_{i,x}.\mathbf{V} \cup \{v_{j,y}\}$; $temp.w = \alpha_{i,x}.w + p_j - c_{i_x,j_y}()$;
    $temp.e = \alpha_{i,x}.e + m_{i_x,j_y} + f_j$; $temp.\mathbf{u} = \alpha_{i,x}.\mathbf{u} \cup \{u_{j,y} = 1\}$ //extand the path
    if $temp.e \leq U_j$ **then**
      if $temp.w \geq \alpha_{j,y}.w$ **then**
        Let $\alpha_{j,y} = temp$; **Enqueue** $\alpha_{j,y}$ into $\mathbf{E}$
Let $\mathbf{X}^* = \alpha_{n,1}$

Figure 3.2: Pseudocode of finding the optimal path $\mathbf{X}^*$.

single- and multiple-worker variations, respectively.

The considered worker assigning problem (Eqs. (3.1)–(3.9)) is close to the OP problem with single worker and time-windowed requests. However, our crowdsensing problem has the following unique features:

1. Each request r_i is associated with a non-trivial service time f_i. For example, it may take a worker 3 minutes to download the sensor readings or take a photo.
2. Each request r_i may be satisfied from z_i feasible spots. $z_i = 1$ if the request can only be performed at a specific location.
3. Relocating a worker from request r_i to r_j imposes nontrivial cost $c_{i,j}$ on that worker.

Therefore, the existing OP solutions cannot be applied to our problem. We develop a worker assigning algorithm in Sec. 3.2.

3.2 Detour Planning Algorithm: DP

We develop an optimal algorithm for our detour planning problem. Our Detour Planning (DP) algorithm is inspired by the dynamic programming algorithm presented in Righini and Salani [19], but we explicitly take the three unique features (see Sec. 3.1) into considerations. We let $\alpha_{i,j}$ be the state of a potential detour path from source location s to feasible spot j of request r_i. More specifically, we define $\alpha_{i,j} = < \mathbf{V}, w, e, \mathbf{u} >$, where $\mathbf{V}$ is the current path, w is the profit, e is the elapse time, and $\mathbf{u}$ keeps track of the visited

requests and feasible spots. We let χ=$\{\alpha_{i,j}\}_{1\leq i\leq N, 1\leq j\leq z_i}$ be the set of all the states with the maximum objective function values known so far. Fig. 3.2 presents the pseudocode of our algorithm, which consists of two steps: (i) initialization and (ii) expansion. In particular, lines 1–2 initialize all $\alpha_{i,j}$, and lines 4–11 iteratively extend $\alpha_{i,x}$ to $\alpha_{j,y}$. The optimal detour path is stored in $\alpha_{n,1}$, which is returned as $\mathbf{X}^*$ in line 12.

We analyze the efficiency of the proposed DP algorithm below.

Remark 1 *The DP algorithm given in Fig. 3.2 has a time complexity of* $O((NZ)2^{NZ})$, *where* Z *is the maximum feasible spots for each request. That is* $Z = \max_{1\leq i\leq N}\{z_i\}$. *This is because the dequeue command at line 5 may obtain* $\sum_{k=0}^{(NZ)-2}\binom{(NZ)-2}{k}$ *states, and the for-loop starts from line 7 has a complexity of* $O(NZ)$ *as it may check all the states. Hence, the time complexity of the DP algorithm is* $O((NZ)2^{NZ})$.

We next briefly explain a possible optimization that can be applied to the DP algorithm.

Remark 2 *The proposed DP algorithm can be optimized in various ways. For example, Righini and Salani [19] propose a bidirectional approach, in which they use DP algorithm from both sides, the beginning and the end. Each direction extends the path and stops right before exceeding the half of the total number of requests. Then, the two detour paths are merged into an optimal detour path.*

3.3 Approximation Algorithm: DPA

In previous sections, we present an optimal algorithm DP for detour planning problem. In this section, we propose a (pseudo-)approximation algorithm DPA, based on DP. The main difference between them is that the running time of DPA depends on a user selected parameter $\epsilon \in (0, 1]$. We define a scaling factor $F = \frac{\epsilon p_{max}}{N}$, where p_{max} is the maximal score among all locations, and N is the total number of locations.

DPA works as follows. First, we scale the score p to $p'_n = \lfloor \frac{p_n}{F} \rfloor$ for all location n. Second, we apply dynamic programming to compute the path with the highest score. In particular, we iteratively expand the path from a user's source to his/her destination. For each location l, we record multiple subpaths from the source to itself. We only keep the potential *subpaths* that may become part of the optimal paths. We define *potential* subpaths as the subpaths that either reach fewer locations, spend less time, or achieve higher accumulated score, compared to any known subpaths at l. Upon reaching the destination, we return the best known path X.

In the DPA algorithm, larger F (larger ϵ) leads to more subpaths with the same score, which in turn allows us to drop more subpaths and thus achieve lower complexity. However, doing so affects the optimality of path X as well. Let X^* be the optimal path. We denote the total scores of X and X^* as $P(X)$ and $P(X^*)$, and we can derive $P(X) \geq (1-\epsilon)P(X^*)$ (the proof is similar to the one in [13]). This means that DPA achieves an approximation gap of ϵ, while higher ϵ leads to both lower complexity and higher approximation gap.

3.4 Evaluations

In this section, we conduct trace-driven simulations to evaluate the proposed DP algorithm.

Table 3.1: Ontime Ratio (%) of Various Algorithms

City	**Taipei**			**Vancouver**		
Algorithm	**HR**	**CR**	**DP**	**HR**	**CR**	**DP**
Deadline $T = 1$	0	0	100	0	0	100
2	4.1	4.1	100	4.1	0	100
4	0	0	100	0	0	100
8	0	0	100	0	4.1	100
16	29.1	58.3	100	33.3	41.6	100
City	**Taipei**			**Vancouver**		
Algorithm	**HR**	**CR**	**DP**	**HR**	**CR**	**DP**
No. Requests $N = 5$	12.5	8.3	100	0	0	100
10	0	0	100	0	4.1	100
15	0	8.3	100	0	0	100
20	0	0	100	0	0	100
25	0	4.1	100	0	0	100

3.4.1 Setup

We have developed a simulator for the detour planning problem using C/C++, and run the simulations on a commodity PC with an AMD 2.6 GHz CPU. Within the simulator, we have implemented the proposed DP algorithm[1]. We are not aware of any algorithms solving the considered problem, thus we have also implemented four heuristic

[1]We thank the authors of [19] for sharing their datasets and algorithm implementation with us.

algorithms, Highest-Reward (HR), Highest-Reward with OnTime constraints (HROT), Closest-Request (CR) and Closest-Request with OnTime constraints (CROT), for comparisons. The HR algorithm works as follows. It gives requests with higher rewards higher priority, and iteratively extends the detour path to the request with the highest priority. It stops when adding the next request renders the worker missing his/her deadline. The CR algorithm works in a similar way, but gives the closest request higher priority. The HR and CR algorithms mimic human behaviors when no algorithm is used to generate detour paths. We then create the ontime versions (HROT and CROT) of the algorithms, which validate the ontime constraint before extending a detour path to the next request. In particular, HROT and CROT do not consider request r_i unless the worker may travel from r_i to e before time T. Hence, HROT and CROT guarantee that workers can reach their destinations in time.

To drive our simulator, we consider the actual requests of multimedia content gathering, and we collect actual geospatial traces from Flickr [6] as follows. We first collect the names of 25 attractions from travel Web sites. In particular, Taipei [24] and Vancouver [26] are considered in our evaluations. We then look up the longitude/latitude of each attraction, and search for Flickr photos that are taken at close-by longitude/latitude and are tagged with the attraction's name. By close-by, we refer to the photos taken within 1 km radius of each attraction. We end up with having up to 2000 photos for each attraction. We extract the longitudes/latitudes from individual photos, and cluster them into a few feasible spots using hierarchical clustering with a threshold of 500 meters. The mean longitude/latitude of each cluster is used as the location of a feasible spot. The number of feasible spots for each attraction is between 1 and 31, depending on the attraction's height and popularity.

Each simulation lasts for 25 hours. We assume the requests follow a Poisson arrival process and we set the average number of requests per hour to be 4. Each request happens at a random attraction, with a time window size between 1–6 hours and a reward between 1–40 U.S. dollars, both follow uniform distributions. We vary three system parameters in the simulations. $T \in \{1, 2, 4, 8, 16\}$ is the deadline of arriving at the destination, $N \in \{5, 10, 15, 20, 25\}$ is the number of attractions, and $C \in \{0, 0.06, 0.12, 0.24, 0.48\}$ dollars per km is the average gas and car depreciation cost. We let $T = 8$, $N = 10$, and $C = 0.12$ if not otherwise specified. We run each simulation 24 times, and report the simulation results with 95% confidence intervals whenever applicable. We consider four performance metrics: (i) *late time* of HR and CR, (ii) *ontime ratio*, which is the fraction of workers who reach their destinations in time, (iii) *profit* of the resulting detour path, and (iv) *running time* of each algorithm.

3.4.2 Results

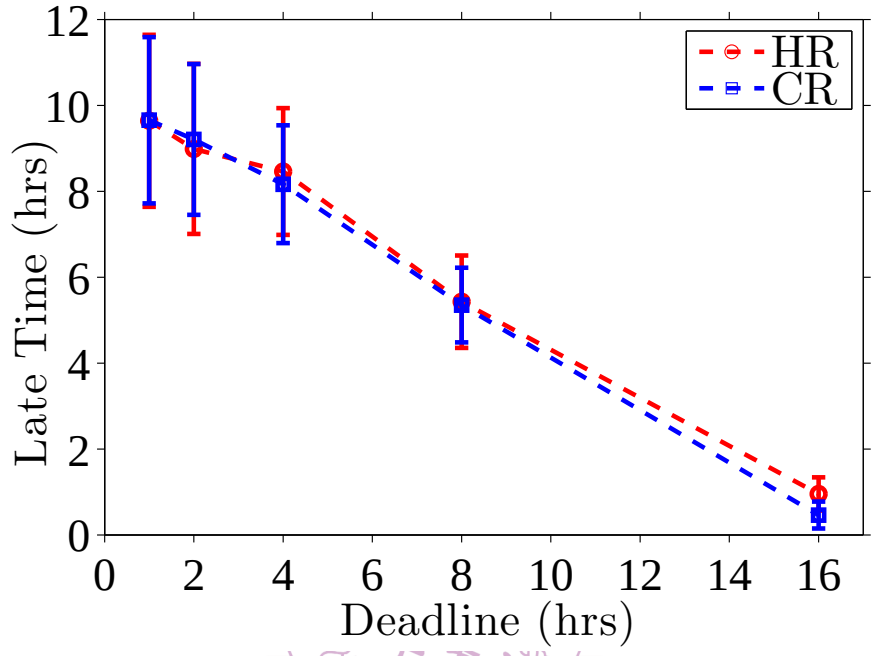
Importance of the DP problem. In Fig. 3.3, We first report sample late time from the HR and CR algorithms, which mimic human behavior. This figure reveals that the HR and CR algorithms lead to up to almost 10 hrs of average late time. This is significant considering that the deadline of arriving at the destinations is as short as 1 hr. Long late time will drive the potential workers away from the crowdsensing systems. We next present the ontime ratio of HR, CR, and DP algorithms in Table 3.1. This table clearly shows the benefit of the DP algorithm: the computed detour paths are always ontime. Fig. 3.3 and Table 3.1 depict the importance of the considered DP problem, as human computed detour paths (mimicked by HR and CR) lead to late arrivals at the destinations. Since the HR and CR algorithms do not meet the basic requirement of the DP problem, we no longer consider them in the rest of this paper.

Optimized profits. Fig. 3.4 presents the profits of different algorithms under varying deadlines. This figure reveals that the DP algorithm always outperforms the HROT and CROT algorithms. Moreover, the performance gain of the DP algorithm over the other two algorithms increases with longer deadlines. In particular, with $T = 16$, the DP algorithm almost doubles the profits, compared to the other two algorithms.

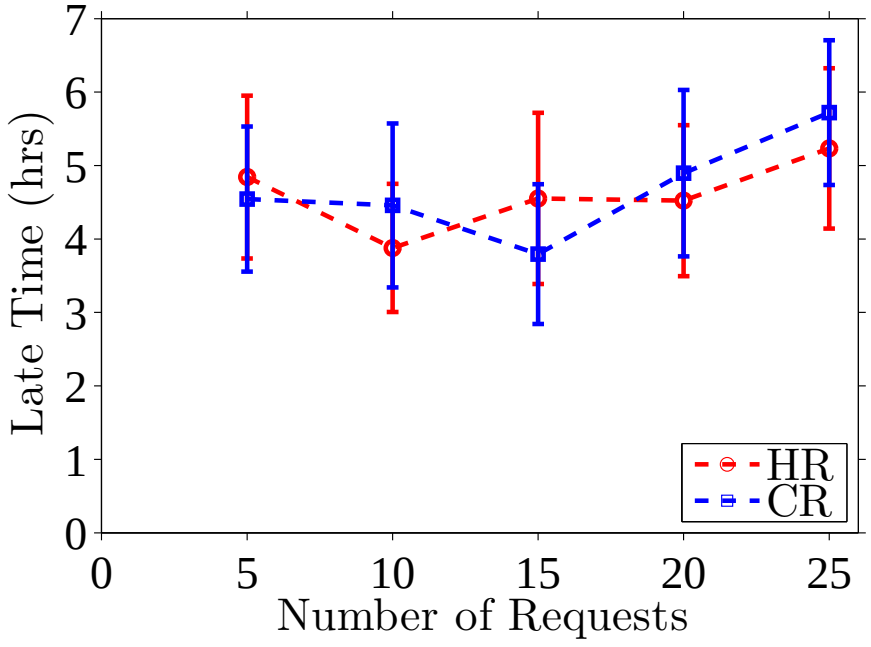
Efficiency. Fig. 3.5 plots the running time of the considered algorithms. We observe that the heuristic algorithms finish in no time, and the DP algorithm takes about 6 ms to terminate in the worst case, which shows its efficiency.

Scalability of the DP algorithm. We present the simulation results with diverse numbers of requests in Figs. 3.6 and 3.7, for resulting profits and running time, respectively. These two figures demonstrate that the DP algorithm scales to large numbers of requests and feasible spots quite well. In Taipei, the average number of feasible spots per-request is 1.96 and the maximal number of feasible spots per-request is 15. In Vancouver, the average and maximal numbers of feasible spots per-request are 6.48 and 31, respectively. Fig. 3.6 reveals that the DP algorithm outperforms all other algorithms under all considered numbers of requests. Fig. 3.7 depicts that the DP algorithm terminates much slower in Vancouver (Fig. 3.7(b)) than in Taipei (Fig. 3.7(a)): up to 82 ms running time is observed in Vancouver. This difference can be explained by the fact that the attractions in Vancouver have more feasible spots (162 in total), compared to the attractions in Taipei (49 in total), as reported above. We conclude that the DP algorithm does scale to 162 feasible spots, yet runs in < 82 ms, which essentially is in real-time.

Implication of cost. Fig. 3.8 reports the sample resulting profits from the algorithms with different travel cost. This figure reveals that, when the per-km cost is lower, the proposed DP algorithm results in higher profits. Moreover, the DP algorithm outperforms the heuristic algorithms under all considered travel costs.

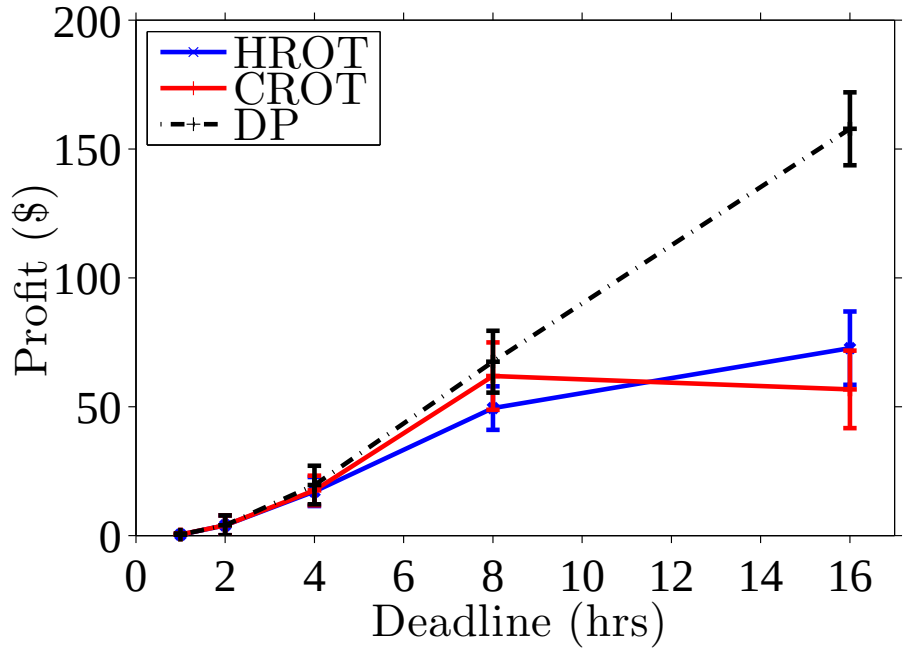


(a)

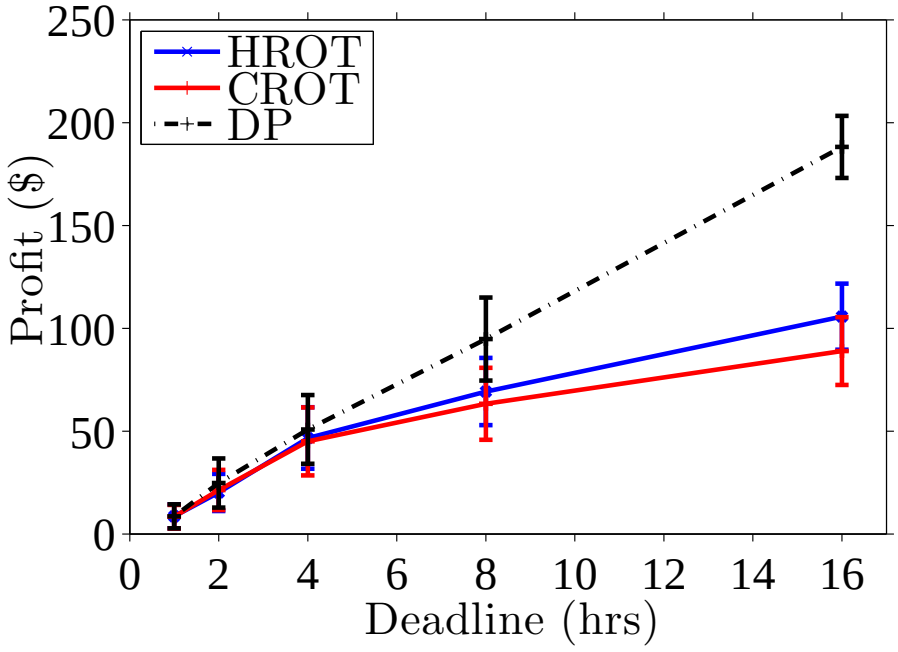


(b)

Figure 3.3: Late time of the HR and CR algorithms: (a) varying deadlines in Taipei and (b) varying numbers of requests in Vancouver.

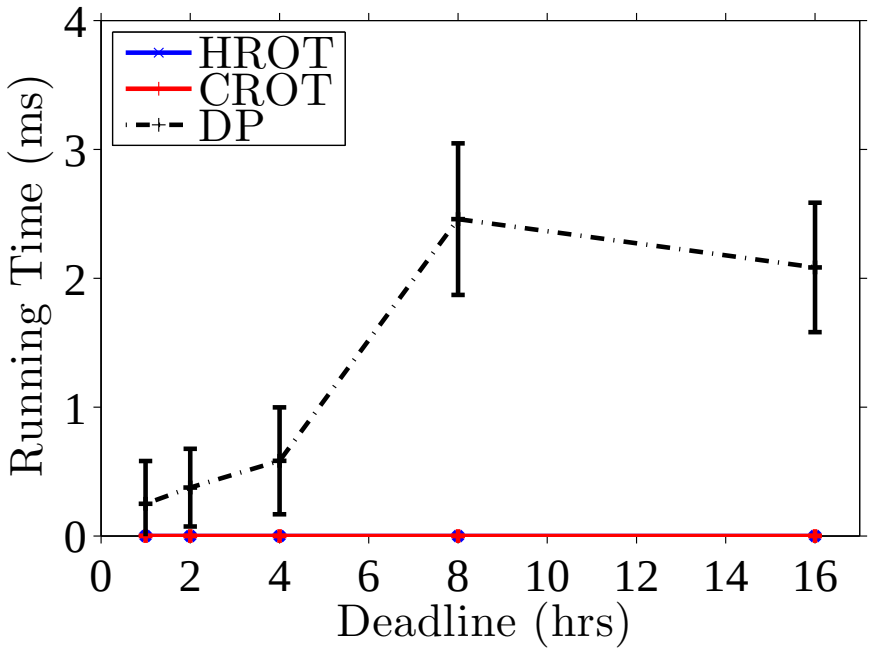


(a)

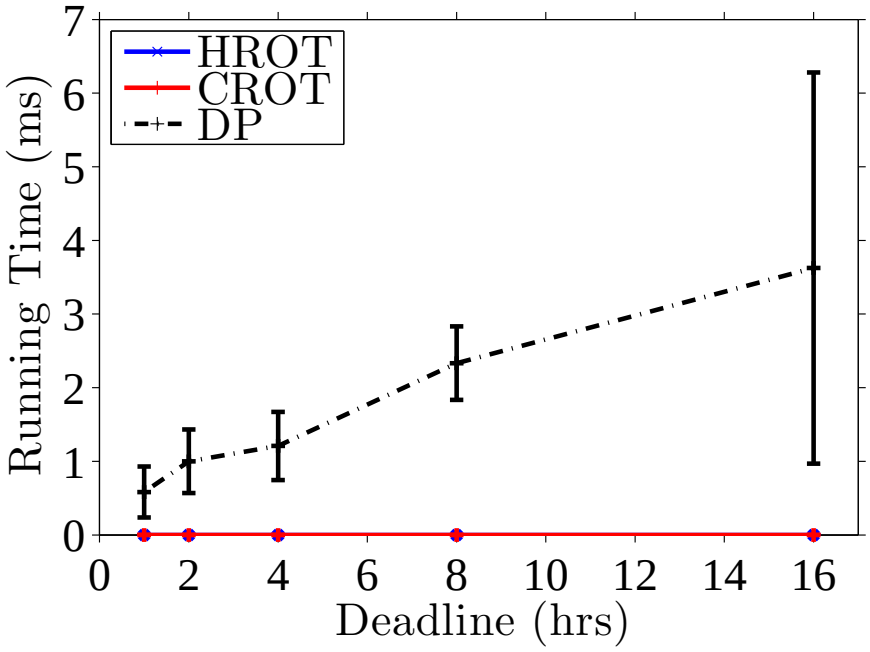


(b)

Figure 3.4: Resulting profit with varying deadlines, from: (a) Taipei and (b) Vancouver.

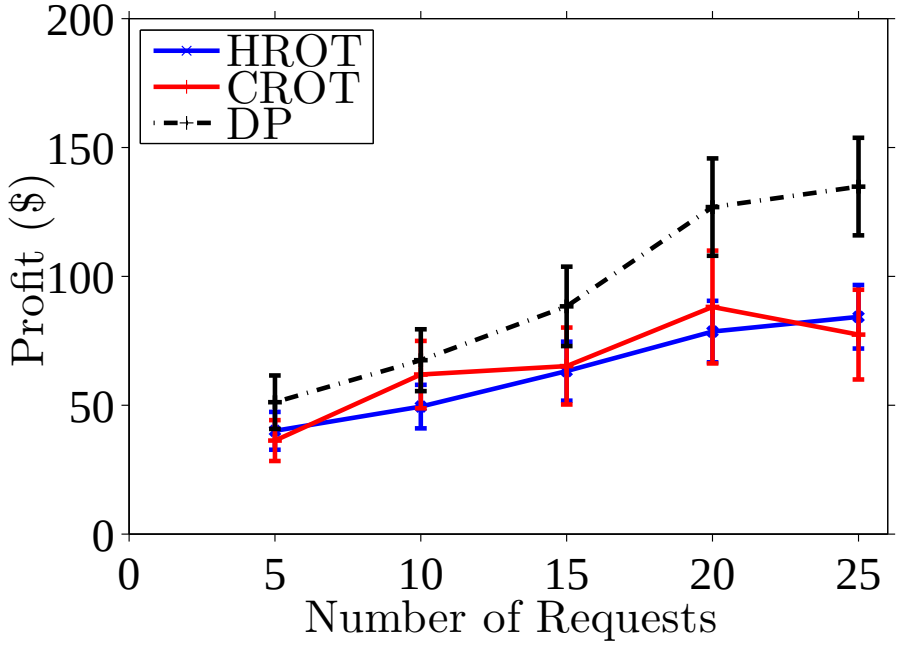


(a)

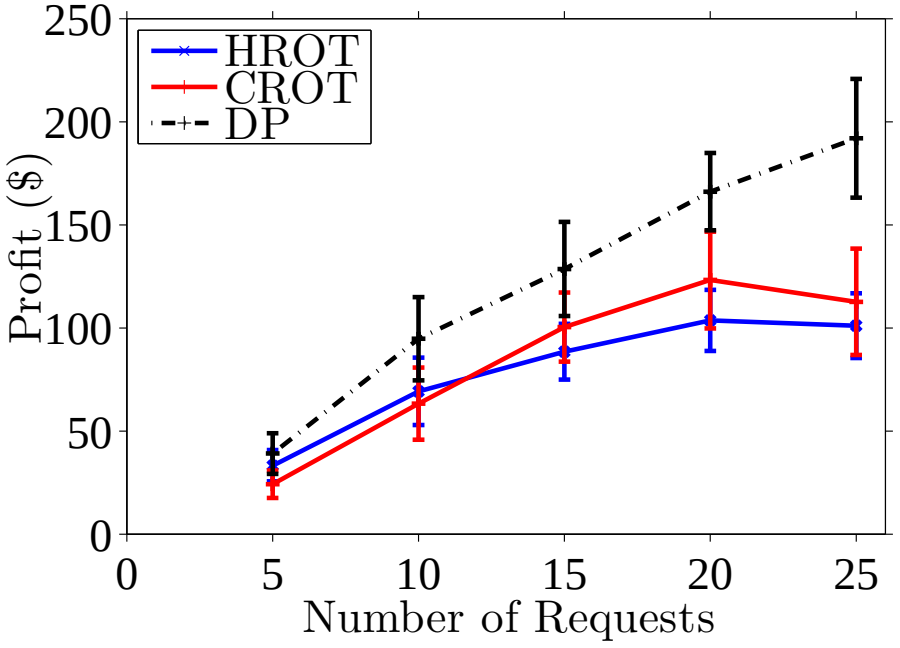


(b)

Figure 3.5: Running time with varying deadlines, from: (a) Taipei and (b) Vancouver.

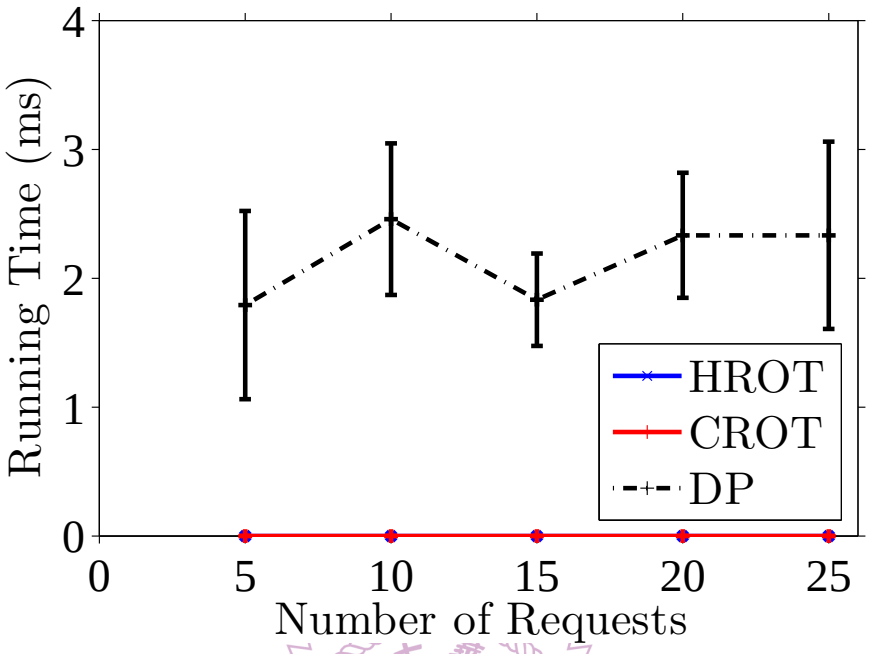


(a)

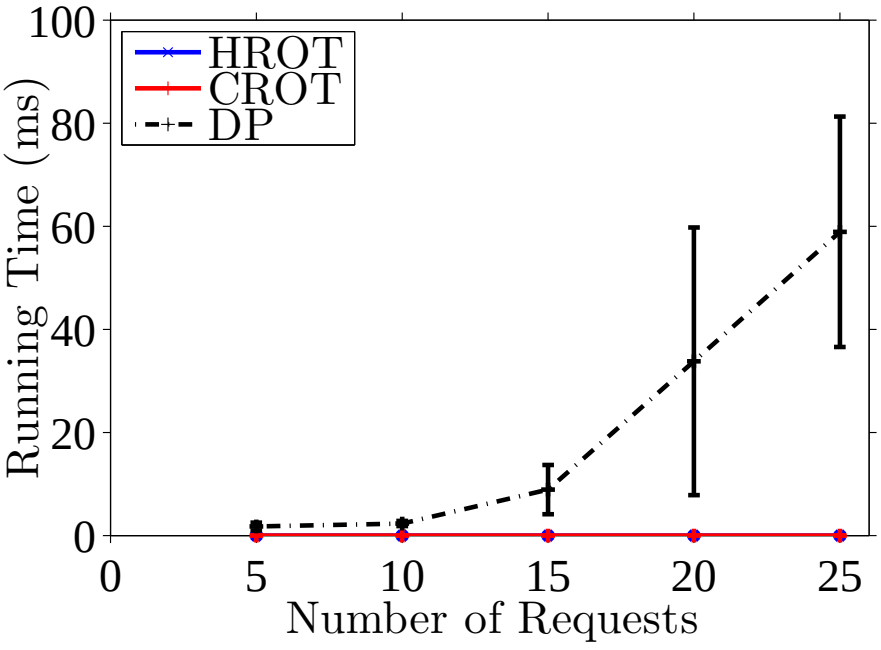


(b)

Figure 3.6: Resulting profits with varying numbers of requests, from: (a) Taipei and (b) Vancouver.

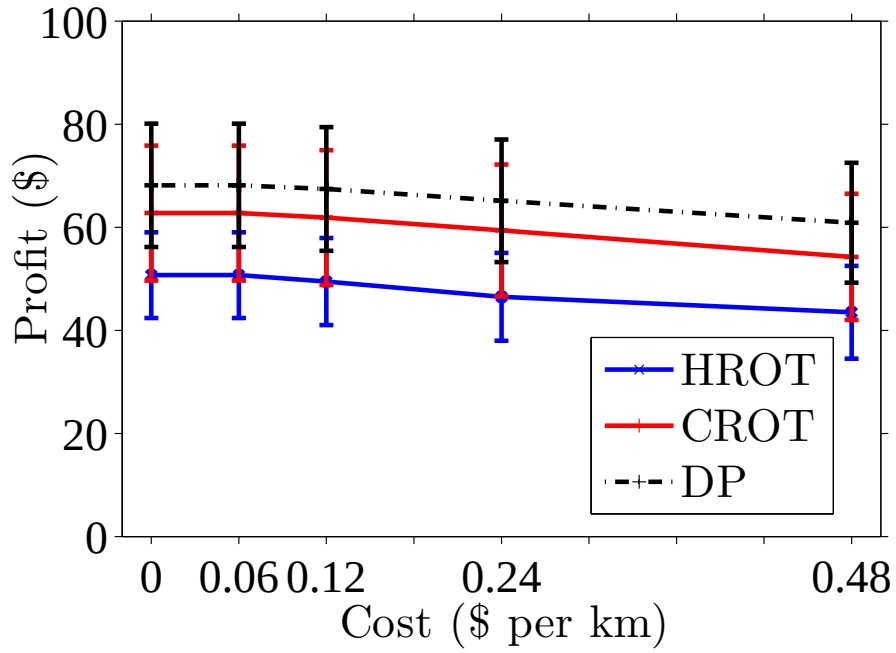


(a)

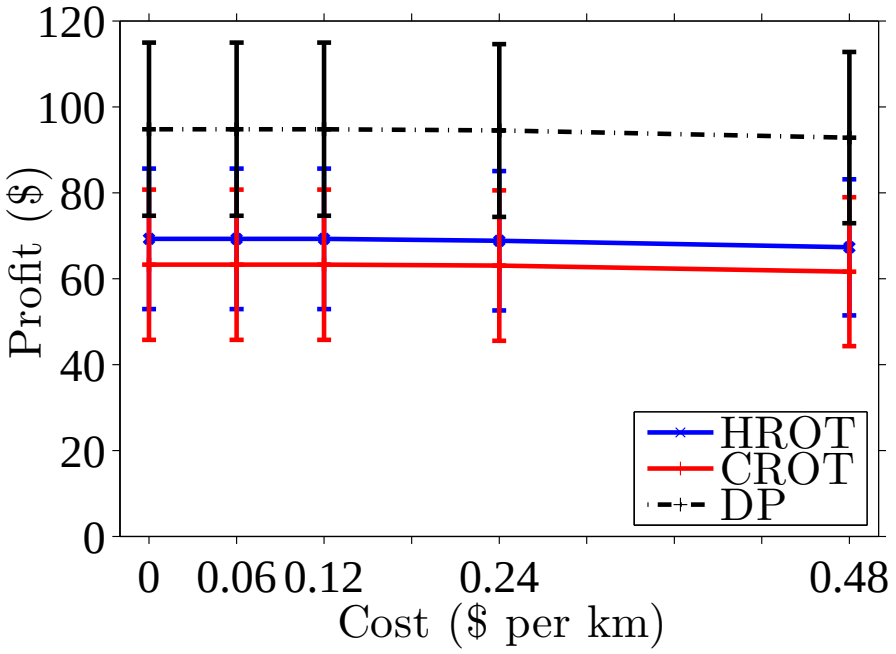


(b)

Figure 3.7: Running time with varying numbers of requests, from: (a) Taipei and (b) Vancouver.

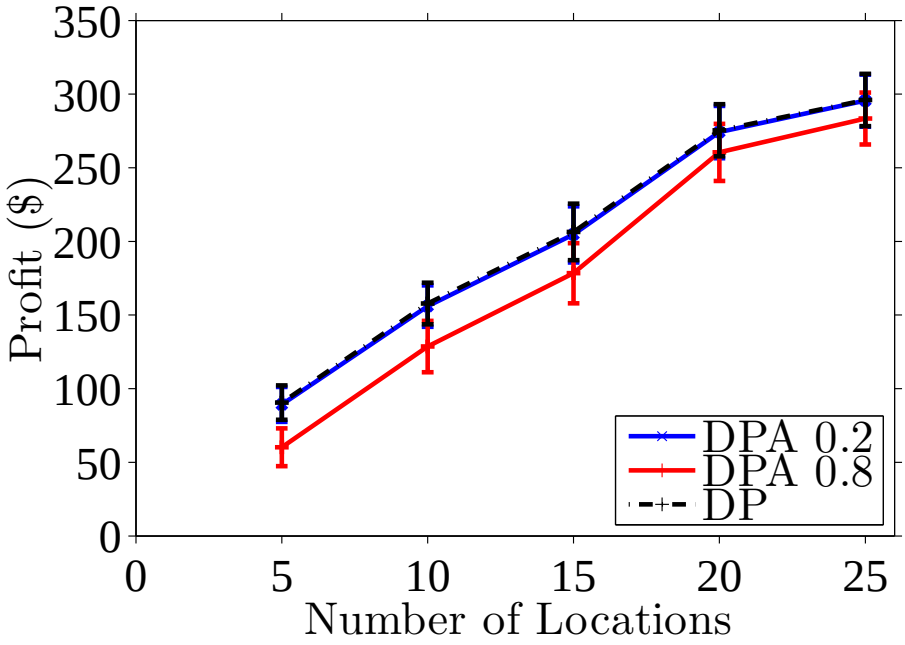


(a)

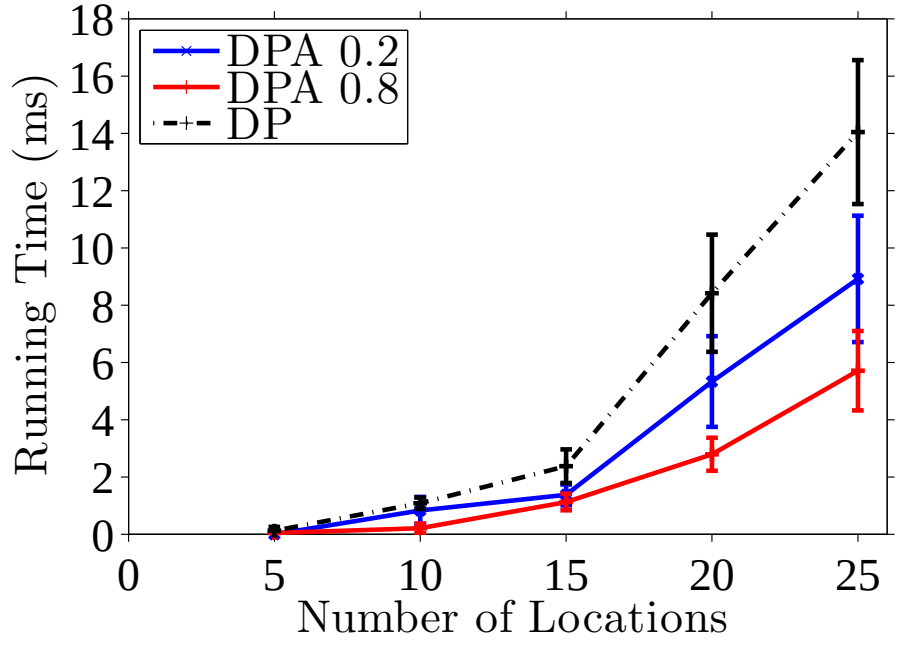


(b)

Figure 3.8: Resulting profits with different travel cost, from: (a) Taipei and (b) Vancouver.



(a)



(b)

Figure 3.9: Results from Taipei dataset: (a) profit and (b) running-time.

DPA is Efficient. Fig. 3.9(a) presents the profit with different number of locations. We observe that DPA with $\epsilon = 0.8$ achieves almost optimal reward (DP) in every number of locations, and DPA with $\epsilon = 0.2$ slightly deviates from DP and the reward is more closer in larger number of locations. Moreover, Fig. 3.9(b) shows that, compared to DP, DPA with $\epsilon = 0.2$ achieves 2X speed up, and DPA with $\epsilon = 0.8$ even achieves 3X speed up.

Tradeoff Complexity and Optimality. Fig. 3.9 demonstrates that the DPA algorithm successfully allows the users to tradeoff complexity and optimality. Because users can let ϵ be smaller to achieve the almost optimal reward (Fig. 3.9(a)), the better solution also consumes more running time for computing the result (Fig. 3.9(b)).

3.5 Discussion

We are actively developing a complete crowdsensing system for multimedia content gathering. The following challenges arise when designing such a geospatial- and temporal-dependent system.

1. The systems should produce a *feasible* detour path for each new worker. A detour path is feasible if and only if the worker can reach his/her destination in time.
2. The systems should compute the detour paths to maximize the overall productivity in the format of the total worker profit.
3. The systems should simultaneously compute multiple users, because we must make a good use of all users.
4. The systems should concerns the energy consumption and sensor accuracy when assign requests to users.

We address the first two challenges so far. The remaining two challenges, as well as other practical concerns, are we discussed them in the next section. Furthermore, we name the more complicated version of detour planning problem as multi-users detour planning problem, and it will solve the four challenges.

Chapter 4

Multi-users Detour Planning Problem

4.1 Formulate the Multi-users Detour Planning Problem

We regard the multiple detour planning problem as a more complicated version of the detour planning problem. We use almost variables which are in Sec.3.1, and we add/re-define some variables to fit the more complicated version. About the worker, we let o_w and d_w be the start location and end location of worker w, and the battery level of worker w is g_w. We reuse boolean variables X, but it represents multiple paths now. More specifically, $x_{w,i,j} = 1$ if the detour path instructs the worker w to move from query i to j; and $x_{w,i,j} = 0$ otherwise. For the requests, we must satisfy the quality q_i of each request i, and each request i consumes energy δ_i. $a()$ is the accuracy model for evaluating how many workers to achieve required quality of each request.

We formulate the multiple detour planning problem of maximizing the overall rewards, and it is derived from the formulation of the detour planning problem in Sec. 3.1.

$$\max \sum_{i=1}^{N-1} \sum_{j=2}^{N} [p_i - \sum_{a=1}^{z_i} \sum_{b=1}^{z_j} c_{i_a,j_b} u_{i,a} u_{j,b}] x_{w,i,j} \tag{4.1}$$

$$s.t. \sum_{w=1}^{W} \sum_{j=1, j!=o_w}^{N} x_{w,o_w,j} = \sum_{w=1}^{W} \sum_{i=1, i!=d_w}^{N} x_{w,i,d_w} = 1 \tag{4.2}$$

$$\sum_{i=1}^{N-1} x_{w,i,k} = \sum_{j=2}^{N} x_{w,k,j} \leq a(q_k), \forall w = 1, ..., W, \forall k = 2, ..., N-1 \tag{4.3}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} (\sum_{a=1}^{z_i} \sum_{b=1}^{z_j} m_{i_a,j_b} u_{i,a} u_{j,b} + f_i) x_{w,i,j} \leq T_w, \forall w = 1, ..., W \tag{4.4}$$

$$\sum_{j=1}^{z_i} u_{i,j} \leq 1, \forall i = 1, ..., N \tag{4.5}$$

$$s_i + f_i + \sum_{a=1}^{z_i} \sum_{b=1}^{z_j} m_{i_a,j_b} u_{i,a} u_{j,b} - s_j \leq M(1 - x_{w,i,j}), \forall i, j = 1, ..., N, \forall w = 1, ..., W \tag{4.6}$$

$$B_i \leq s_i, \forall i = 1, ..., N \tag{4.7}$$

$$s_i + f_i \leq U_i, \forall i = 1, ..., N \tag{4.8}$$

$$x_{w,i,j}, u_{i,j} \in \{0, 1\}. \tag{4.9}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} \delta_i x_{w,i,j} \leq g_w, \forall w = 1, ..., W \tag{4.10}$$

The objective function Eq. (4.1) is to maximize the benefit, which is defined as overall collected rewards minus overall travel costs. The constraints in Eq. (4.2) ensure that all paths starts from query 1 to query N for each worker. The constraints in Eq. (4.3) ensure that every feasible spot is visited once of the specific worker. The constraints in Eq. (4.4) ensure that the total time of each path doesn't exceed the deadline specified by each worker. Eqs. (4.1)–(4.4) are slightly revised from Eqs. (3.1)–(3.4), and the constraints in Eqs. (4.5)–(4.9) are same to Eqs. (3.5)–(3.9). Eq. (4.10) prevents the worker run out his/her battery.

4.2 Multi-users Detour Planning Algorithm: MDP

We apply a heuristic solution, named multi-users detour planning (MDP) algorithm, in the MCS system, because the optimal solution (DP) is not designed to solve multi-users detour planning problem. The objective of the MDP algorithm is to maximize the profits and minimize the traveling cost of each worker. Therefore, we let $h_{w,n}$ be the utility of the worker w to perform request n:

$$h_{w,n} = \frac{p_n + \sum_{\substack{k=1 \\ k \neq n}}^{N} \frac{p_k}{m_{w,n,k}}}{m_{w,a_w,n}} \tag{4.11}$$

We define input $i_w =< o, d, g, T >$ of worker w, where o is the source, d is the destination, g is the energy, and T is the deadline. Then the input $j_n =< l, [B-U], p, q, f, \delta >$ of request n, where l is the location, $[B - U]$ is the time-window, p is the reward, q is the quality, f is the service time, and δ is the energy consumption. The algorithm receives these input data and outputs detour path X. Fig. 4.1 presents the pseudocode of our algorithm. Lines 1–2 initialize the current location r_w and time y_w of each worker w, and the returned paths X. Line 4 computes all utility value $H = \{h_{w,n}\}$. Line 5–15 continuously update the returned paths X, which satisfy all constraints. Finally, it returns all detour paths in line 16.

Multiple Detour Planning (MDP) Algorithm

Input: $\mathbf{I} = \{i_w =< o, d, g, T >, \forall w = 1 \text{ to } W\}$
$\mathbf{J} = \{j_n =< l, [B - U], p, q, f, \delta >, \forall n = 1 \text{ to } N\}$
Output: $\mathbf{X}$

$r_w = i_w.o, y_w = 0, \forall w = 1$ to W
$\mathbf{X} = \emptyset$
while $\mathbf{J} \neq \emptyset$ and $\mathbf{I} \neq \emptyset$ **do**
  Compute $\mathbf{H}$, and then sort($\mathbf{H}$) order by $h_{w,n}$ decreasing
  while $\mathbf{H} \neq \emptyset$ **do**
    $h_{w,n} =$ pop($\mathbf{H}$)
    if $j_n.[B - U]$, $i_w.g$, and $i_w.T$ are satisfied **then**
      $\mathbf{X} = \mathbf{X} \cup x_{w,a_w,n}$
      Update r_w, y_w, and $i_w.g$
      if j_n satisfy its quality **then**
        Remove j_n from $\mathbf{J}$
      break
  if $\mathbf{X}$ does not update **then**
    $\mathbf{X} = \mathbf{X} \cup x_{w,a_w,i_w.d}$
    Remove i_w from $\mathbf{I}$
Return $\mathbf{X}$

Figure 4.1: Pseudocode of finding all feasible paths $\mathbf{X}$.

4.3 Evaluations

4.3.1 Setup

In the evaluation, we use the dataset, which is collected from the real-world. We collect the dataset through a PTT, which is the most popular bulletin board system in Taiwan. PTT includes at least 1.5 million registered users and more than 20000 boards. On the whole, PTT is one of the largest forums, and users post more than 20000 articles everyday. We gather the articles from the most popular board in PTT, and we totally collected 5700 articles in 10 days from April 11 to April 20, 2014. Then, we use the information in the articles to be the input of requests and workers. For simulating spatial-temporal requests, we extract IP addresses of the articles. The IP addresses are assumed to be the requested locations, and the posted time is the start time of the time window of the request. We use triangulation [29,30] to approximate geo-locations from their IP addresses. We hired three servers to be the anchors to help us to acquire the geo-locations from each IP address. In specific, the servers ping each IP address and analyze the round-trip time

(RTT) to measure the distances between the target IPs and the servers. Then we estimate the real location according to these distances. Furthermore, we divide Taiwan into grids, and we determine the real location on grids of each IP address. We compute the mean squared error between grids and estimated locations, and we set the estimated locations to appropriate grids with minimal mean squared error. Besides, we randomly assign two grids to be the source and destination of each worker. Moreover, we simulate the mobility of idle workers by random waypoint model. Because workers won't stay at specific locations too long, we let workers change their moving ways in a pre-defined period. If workers received requests, they stop following the random waypoint model. They follow the assigned detour path, and they keep following the random waypoint model after they finish whole requests.

We implement a traced-driven simulator in Java, and the simulator is driven by the PTT dataset. Each request use the real posted time be the begin time of time window. Then we randomly assume its fake posted time is uniformly early in [1, 5] hours and the end time of time window is uniformly later in [1, 5] hours. The profit is distributed in [1, 40] U.S. dollars. The required quality of requests varies form 1, 2, 3, 4, 5. The service time is uniformly distributed in [1, 3] minutes. The request energy cost is in the work [10]. Each worker randomly chooses two locations from dataset to be his/her source and end locations, and we let the larger posted time of the two locations to be the deadline of the worker. The battery levels of workers are uniformly distributed in [20%, 80%]. In our experiments, we considered 100 workers and varied the number of queries in [50, 100, 200, 400, 800], and we also fixed the number of requests to 800 and varied the number of workers in [50, 100, 200, 400, 800].

In the following, we introduce the performance metrics for evaluating our solutions: 1) *profit* as the total reward of performed requests takes out the moving cost among the locations of requests; 2) *traveling cost* as total consumed energy (e.g. gasoline) of workers in moving, and we represent the consumed energy by joule (J); 3) *completed requests ratio* as the number of requests, which are completed with satisfied qualities and performed within time-windows, over the total number of requests. In our experiments, we implement three algorithms: 1) multi-users detour planning algorithm (MDP), 2) detour planning algorithm (DP), and 3) nearest algorithm (CROT). We compare MDP against other two algorithms to evaluate the performance.

4.3.2 Results

Superior profits. Fig. 4.2 presents the average profit under varying the numbers of requests. We observed that MDP is always better than DP and CROT. But MDP and

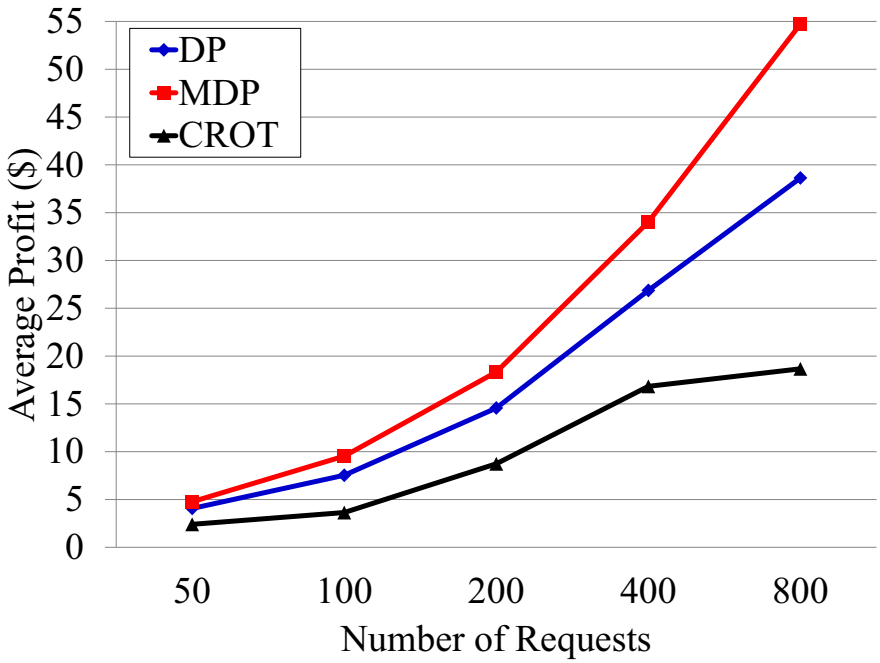


Figure 4.2: Resulting profit with varying requests.

DP are achieve almost same profit when the number of request is 50. The reason is that the number of requests is relatively small than the 100 workers, 100 workers can perform most requests in both solutions. When the number of request increases, MDP outperforms DP and CROT. MDP achieves at most 1.4 times profits of DP at the number of request is 800, and MDP earns at least 1.9 times profits of CROT.

Efficient movement. Fig. 4.3 presents the average traveling cost under varying the numbers of requests. The figure shows that CROT consumes less traveling cost than MDP and DP. Because CROT assigns requests to nearest workers, workers consume less traveling cost to arrive at locations of nearby requests. Even CROT saves more traveling cost, CROT achieves less profits than MDP and DP (Fig. 4.2). We observed Fig. 4.3 and Fig. 4.2 when the numbers of request are 50 and 800. Separately, MDP consumes 1.1 and 1.2 times traveling cost of CROT, but MDP achieves 1.9 and 2.9 times profits of CROT. When the number of requests is 100, MDP saves 51% traveling cost compared to DP , and achieves 1.26 times profits of DP. Thus, MDP assigns workers more efficient than DP and CROT while maximizing the profits.

Limited workload. Fig. 4.4 presents the completed requests ratio under varying the numbers of requests. MDP achieves 95% completed requests ratio at the number of requests is 50, and MDP finishes more requests than DP and CROT about 14% and 47%, separately. However, the differences between the algorithms are gradually decreasing when the number of requests is increasing. Because the number of workers is fixed to 100

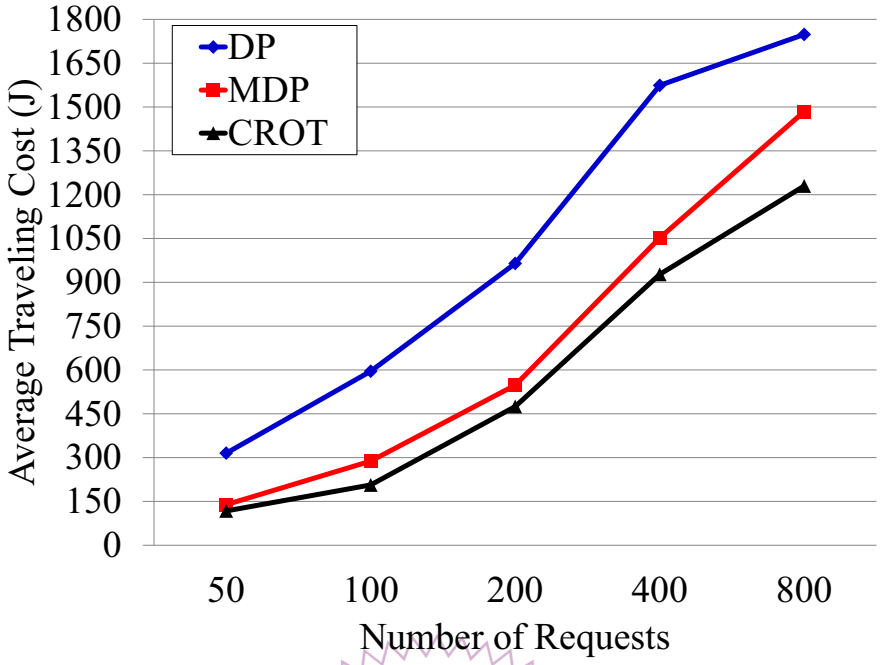


Figure 4.3: Resulting traveling cost with varying requests.

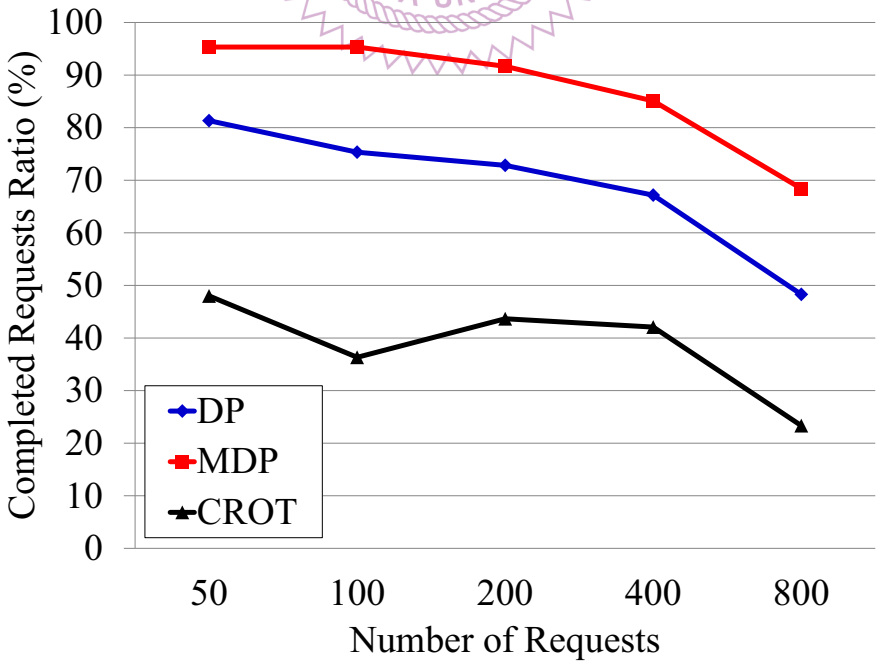


Figure 4.4: Resulting completed requests ratio with varying requests.

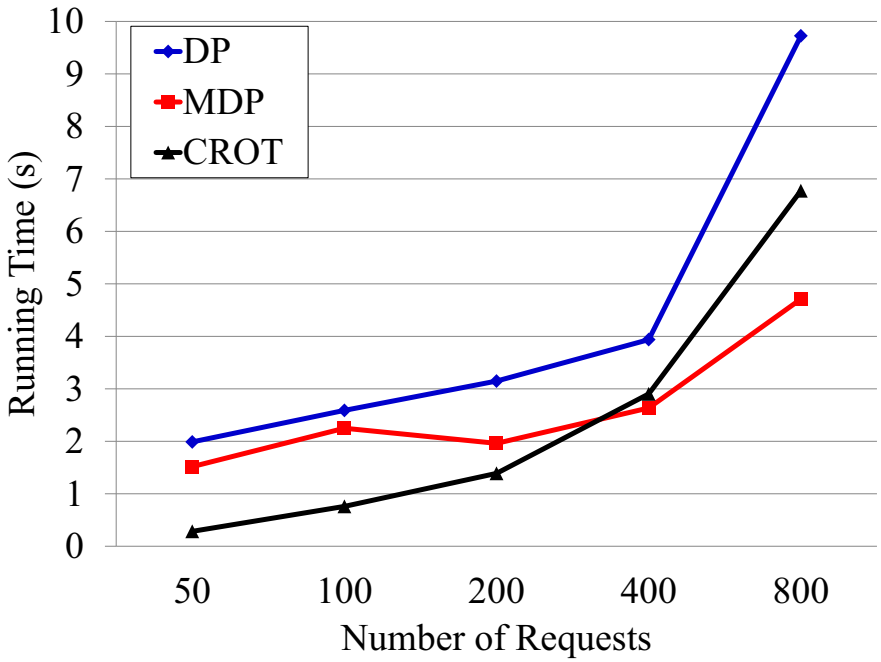


Figure 4.5: Resulting the running time with varying requests.

and many temporal constraints, they can't perform a large number of requests. When the number of requests is 800, MDP still achieves more requests than DP and CROT about 20% and 45%, respectively.

Real-time computation. Fig. 4.5 presents the running time under varying the numbers of requests. When the number of requests is smaller than 400, we observed that MDP is slower than CROT. Even MDP consumes a little bit more time than CROT, the computation of MDP is still in a real-time, which is less than 3 seconds. Nevertheless, MDP is faster than CROT when the number of requests is larger and equal than 400. MDP spends at most 4.7 seconds when the number of requests is 800, and MDP is 5 seconds and 2 seconds faster than DP and CROT, respectively.

Raising overall profits. Fig. 4.6 presents the average profit under varying the numbers of workers. We observed that three algorithms rapidly increase the average profit at the number of workers from 50 to 200. Because the number of requests is 800, the 50 to 200 workers could easily perform some requests of them. MDP stably increases until there are 800 workers, but the overall profit still increases. When the number of workers is 800, MDP achieves 2.64 times profits of CROT. When the number of workers is 50, MDP achieves 1.6 times profits of DP. Even the achieved profits are close between MDP and DP while increasing number of workers, MDP outperforms DP all the time.

Assigning workers well. Fig. 4.7 presents the average traveling cost under varying the numbers of workers. We noticed that MDP consumes less traveling costs than others

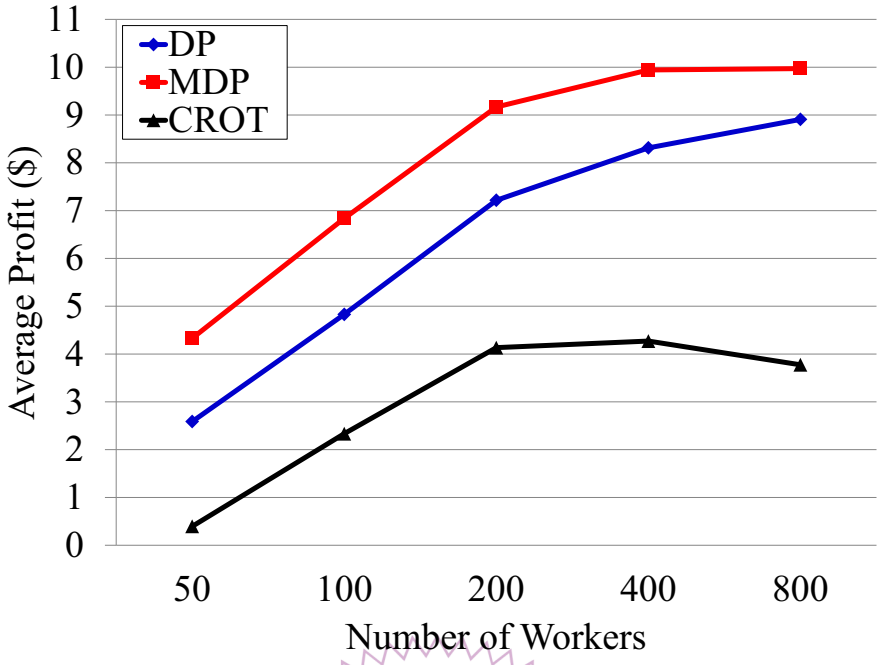


Figure 4.6: Resulting profit with varying workers.

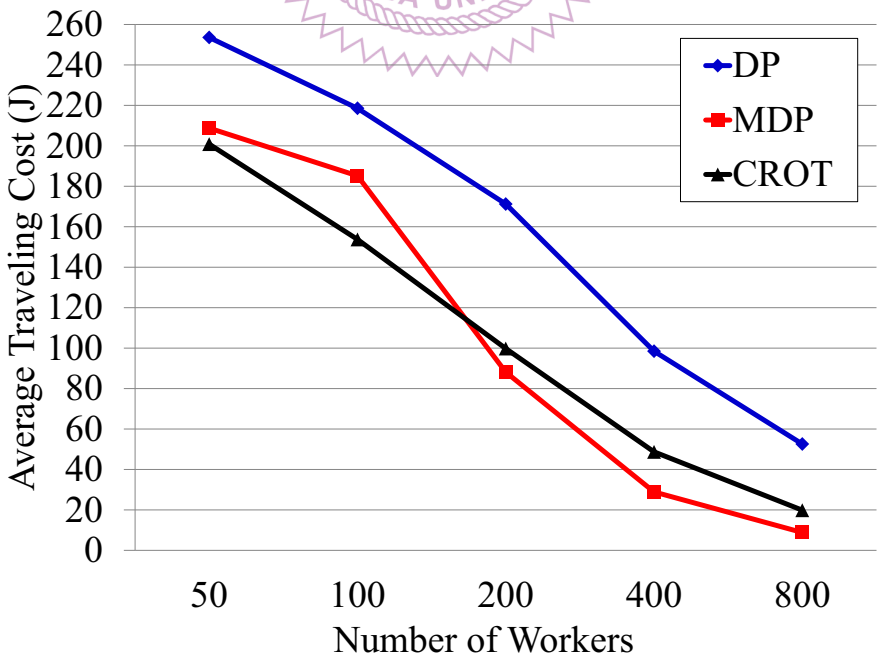


Figure 4.7: Resulting traveling cost with varying workers.

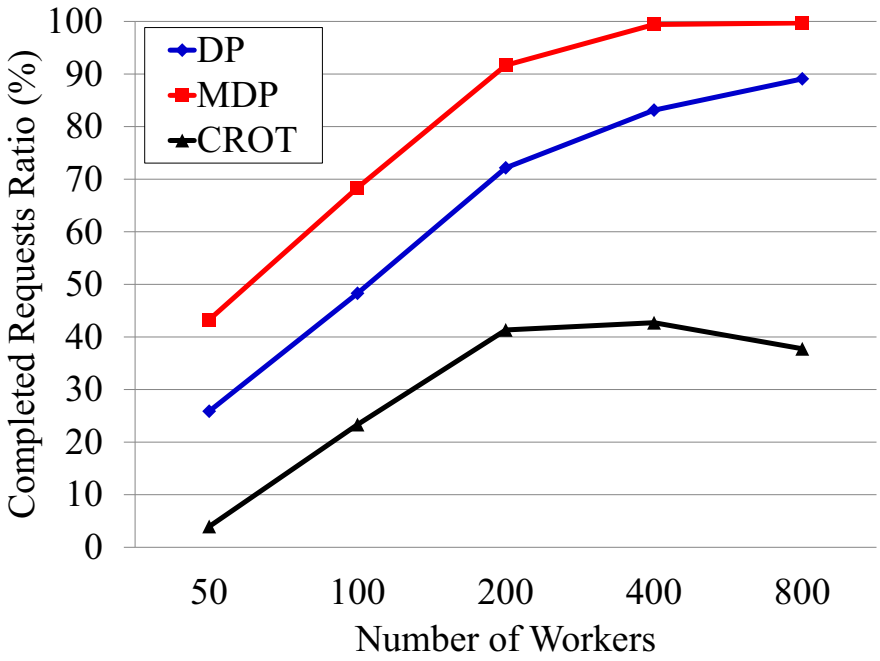


Figure 4.8: Resulting completed requests ratio with varying workers.

when the number of workers is larger and equal than 200. It reveals that MDP makes good assignments when large numbers of workers are in the system, and MDP only consumes at most 1.35 times traveling cost than CROT when smaller numbers of workers are in the system. But achieved profits from MDP are 10 and 3 times against CROT when numbers of workers are 50 and 100. Moreover, MDP saves up to 83% traveling cost compared to DP when the number of workers is 800.

Fine completed ratio. Fig. 4.8 presents the completed requests ratio under varying the numbers of workers. We found that MDP and DP achieves near nearly 100% and 83% completed requests ratio when the number of workers is 400. There are two reasons. 1) MDP assigns as many workers as possible to achieve the required quality, which is evaluated from the accuracy model. 2) MDP assigns a worker to a location of a request, which closes to some requests, and the worker may perform them to reach required quality and earn more profit. Otherwise, we think that DP assigns workers to achieve their optimal profit, but the assignment from DP may be not appropriate under some cases, such as a worker receives a request which is closer to other workers.

Scalability. Fig. 4.9 presents the running time under varying the numbers of workers. We observed that MDP is faster than DP and CROT in each case, and MDP achieves 3.4X and 2.3X speed up compared to DP and CROT when the number of workers is 800.

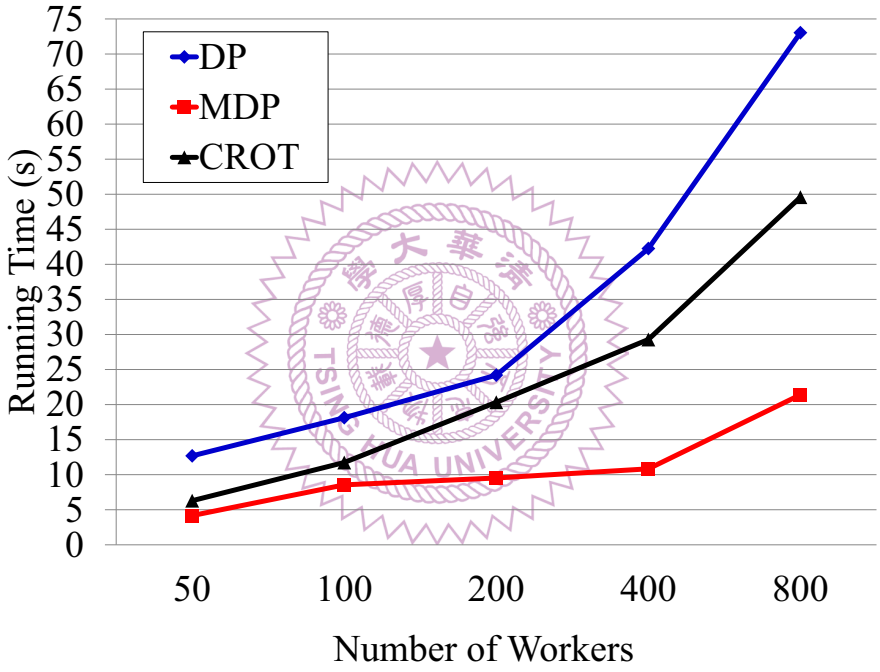


Figure 4.9: Resulting the running time with varying workers.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

- In this thesis, we propose a mobile crowdsensing (MCS) system, and we discuss the detour planning problem and the extended multi-users detour planning problem. The difference between the orienteering problem with time windows and the proposed problems is the more consideration about feasible spots, traveling cost, energy consumption, and the accuracy of sensory data.

- In MCS system, workers aim for maximal profit and perform every request within its time and energy constraints. Moreover, workers must arrive at their destination at the planned time. We mathematically formulate the single-user and multi-users detour planning problems, and we address detour planning algorithm (DP) and multi-user detour planning algorithm (MDP) to solve the two problems, respectively.

- Furthermore, we implemented traced-driven simulators to simulate the MCS system, and we use a real dataset, which is collected in 10 days from PTT, the most popular bulletin board system in Taiwan. The simulation results of DP show that our algorithm: (i) outperforms the other algorithms by up to 100% improvement, (ii) runs efficiently and always terminates within 82 ms, and (iii) is able to scale to larger problems. The simulation results of MDP show that MDP: (i) achieves at most 1.4 times the profit of DP and 2.9 times the profit of the baseline, (ii) saves up to 51% energy compared to DP, (iii) achieves almost 100% completed requests ratio if workers are sufficient, and (iv) runs in the real-time (< 21s).

5.2 Future work

- We plan to improve the mechanism of submitting the results. How to efficiently upload the results to the crowdsensing system with correct timestamps and GPS data is important. Workers have diverse network conditions and different sizes of results. We need to decide which results should be uploaded in different network conditions. The mechanism of uploading results also influences the performance of the system. Workers may perform the task in correct time, but they upload them later due to the worse network condition. We have to distinguish whether the results are feasible or not. Furthermore, we could also find malicious workers who upload results in wrong timestamps and GPS data. Since there are privacy issues, workers may not agree to provide their tracks.

- We think that our MCS system is possible to attract many workers to participate into the system, so the MCS system is appropriate for urban computing by utilizing these workers. The city planners can submit specific requests to detect the environment of the city and track the mobility of workers to analyze the traffic condition or the usage of land. Thus, workers can help the city planners to plan the future constructions.

- Due to the difficulties of deciding rewards, we plan to gamify our crowdsensing system, which workers must pay for playing games. We transfer profits/rewards to scores of games, and we use the augmented reality technique to trigger games. The challenges are to unify games to the system and players play games smoothly.

- We plan to implement a working prototype on Android smartphones. Thus, we can design the experiments for real workers, and we may deploy a practical system.

Bibliography

[1] C. Bassem and A. Bestavros. Mechanism design for spatio-temporal request satisfaction in mobile networks. Technical report, Boston University, February 2012.

[2] I. Boutsis and V. Kalogeraki. Crowdsourcing under real-time constraints. In *Proc. of IEEE International Symposium on Parallel Distributed Processing (IPDPS'13)*, pages 753–764, Boston, May 2013.

[3] I. Boutsis and V. Kalogeraki. On task assignment for real-time reliable crowdsourcing. In *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS'14)*, pages 1–10, Madrid, Spain, June 2014.

[4] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. *IEEE Communications Magazine*, 51(6):112–119, June 2013.

[5] Z. Feng, Y. Zhu, Q. Zhang, L. Ni, and A. Vasilakos. Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing. In *Proc. of IEEE International Conference on Computer Communications (INFOCOM'14)*, pages 1231–1239, April 2014.

[6] flickr, October 2012. `http://www.flickr.com/`.

[7] R. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Communication Magazine*, 49(11):32–39, November 2011.

[8] B. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307–318, 1987.

[9] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. Participatory air pollution monitoring using smartphones. In *Proc. of International Workshop on Mobile Sensing (in conjunction with ACM/IEEE IPSN)*, Beijing, China, April 2012.

[10] T. Hou, J. Lin, C. Hsu, Y. Chung, and C. King. OCI: A middleware framework for optimal context inference on smartphones. Technical report, 2013. `http://nmsl.cs.nthu.edu.tw/dropbox/oci.pdf`.

[11] T.-F. Hou. Optimizing mobile middleware for coordinated sensor activations. Master's thesis, National Tsing Hua University, Taiwan, 2014.

[12] L. Jaimes, I. Laurens, and M. Labrador. A location-based incentive mechanism for participatory sensing systems with budget constraints. In *Proc. of IEEE International Conference on Pervasive Computing and Communications (PerCom'12)*, pages 103–108, March 2012.

[13] K. Lai and M. Goemans. The knapsack problem and fully polynomial time approximation schemes (FPTAS). *`http://math.mit.edu/~goemans/18434S06/knapsack-katherine.pdf`*, 2006.

[14] Z. Li, H. Shen, G. Liu, and J. Li. SOS: A Distributed Mobile Q&A System Based on Social Networks. In *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS'12)*, pages 627–636, Macau, China, June 2012.

[15] C. Liao, T. Hou, T. Lin, Y. Cheng, A. Erbad, C. Hsu, and N. Venkatasubramanian. SAIS: Smartphone Augmented Infrastructure Sensing for Public Safety and Sustainability in Smart Cities. In *Proc. of ACM International Workshop on Emerging Multimedia Applications and Services for Smart Cities (EMASC'14)*, pages 3–8, Orlando, FL, November 2014.

[16] C. Liao and C. Hsu. A detour planning algorithm in crowdsourcing systems for multimedia content gathering. In *Proc. of ACM International Workshop on Mobile Video (MoVid'13)*, pages 55–60, Oslo, Norway, February 2013.

[17] E. Martins and M. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. *4OR: A Quarterly Journal of Operations Research*, 1(2):121–133, June 2003.

[18] R. Montemanni., D. Weyland, and L. Gambardella. An enhanced ant colony system for the team orienteering problem with time windows. In *Proc. of IEEE International Symposium on Science and Society (ISCCS'11)*, pages 381–384, Kota Kinabalu, Malaysia, July 2011.

[19] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with

dynamic programming. *Computers and Operations Research*, 36(4):1191–1203, April 2009.

[20] Roamler, 2011. http://www.roamler.com/services.aspx.

[21] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201, September 2009.

[22] W. Sherchan, P. Jayaraman, S. Krishnaswamy, A. Zaslavsky, S. Loke, and A. Sinha. Using on-the-move mining for mobile crowdsensing. In *Proc. of IEEE International Conference on Mobile Data Management (MDM'12)*, pages 115–124, Bengaluru, Karnataka, India, July 2013.

[23] H. Shibo, S. Hoon, Z. Junshan, and C. Jiming. Toward optimal allocation of location dependent tasks in crowdsensing. In *Proc. of IEEE International Conference on Computer Communications (INFOCOM'14)*, pages 745–753, Toronto, Canada, April 2014.

[24] Taipei travel net, October 2012. `http://www.taipeitravel.net/en/`.

[25] M. Talasila, R. Curtmola, and C. Borcea. Improving location reliability in crowd sensed data with minimal efforts. In *Proc. of Joint IFIP/IEEE Wireless and Mobile Networking Conference (WMNC'13)*, pages 1–8, Dubai, United Arab Emirates, April 2013.

[26] Vancouver landmarks, October 2012. `http://www.hotels.com/de169712-la/all-landmarks-in-vancouver-canada/`.

[27] P. Vansteenwegena, W. Souffriaua, G. Berghe, and D. Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1):118–127, July 2009.

[28] P. Vansteenwegena, W. Souffriaua, and D. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, February 2011.

[29] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang. Towards street-level client-independent ip geolocation. In *Proc. of USENIX Conference on Networked Systems Design and Implementation (NSDI'11)*, pages 365–379, 2011.

[30] B. Wong, I. Stoyanov, and E. Sirer. Octant: A comprhensive framework for the geolocalization of internet hosts. In *Proc. of USENIX Conference on Networked Systems Design and Implementation (NSDI'07)*, pages 313–326, 2007.

[31] Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan. Lowering the barriers to large-scale mobile crowdsensing. In *Proc. of ACM International Workshop on Mobile Computing Systems and Applications (HotMobile'13)*, pages 9:1–9:6, Jekyll Island, Georgia, 2013.

[32] T. Yan, B. Hoh, D. Ganesan, K. Tracton, T. Iwuchukwu, and J.-S. Lee. Crowdpark: A crowdsourcing-based parking reservation system for mobile phones. Technical report, University of Massachusetts Amherst, 2011.

[33] D. Yang, G. Xue, X. Fang, and J. Tang. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *Proc. of ACM International Conference on Mobile Computing and Networking (Mobicom'12)*, pages 173–184, New York, 2012.

[34] M. Yuen, L. Chen, and I. King. A survey of human computations systems. In *Proc. of IEEE Symposium on Social Computing Applications (SCA'09)*, pages 723–728, Vancouver, Canada, October 2009.

[35] M. Yuen, I. King, and K. Leung. A survey of crowdsourcing systems. In *Proc. of IEEE International Conference on Social Computing (SocialCom'11)*, pages 766–773, Boston, MA, October 2011.

[36] M. Yuen, I. King, and K. Leung. Task matching in crowdsourcing. In *Proc. of IEEE International Conference on Internet of things, and Cyber, Physical and Social Computing(iThings/CPSCom'11)*, pages 409–412, Dalian, China, October 2011.