Crowd-Based Smart Parking: A Case Study for Mobile Crowdsourcing

Xiao Chen[2,1], Elizeu Santos-Neto[1], and Matei Ripeanu[1]

[1] Department of Electrical and Computer Engineering,
University of British Columbia
[2] Department of Computer Science and Technology,
Shanghai Lixin University of Commerce
{xiaoc,elizeus,matei}@ece.ubc.ca

Abstract. An increasing number of mobile applications aim to enable "smart cities" by harnessing contributions from citizens armed with mobile devices that have sensing ability. However, there are few generally recognized guidelines for developing and deploying crowdsourcing-based solutions in mobile environments. This paper considers the design of a crowdsourcing-based smart parking system as a specific case study in an attempt to explore the basic design principles applicable to an array of similar applications. Through simulations, we show that the strategies behind crowdsourcing can heavily influence the utility of such applications. Equally importantly, we show that tolerating a certain level of freeriding increases the social benefits while maintaining quality of service level offered. Our findings provide designers with a better understanding of mobile crowdsourcing features and help guide successful designs.

Keywords: mobile crowdsourcing, smart parking, collaborative sensing.

1 Introduction

The definition of crowdsourcing has evolved to cover a variety of online activities that exploit collective contribution/intelligence to solve complex problems. Since the value of the related product or service is usually far beyond the cost of incentivizing individual participants to contribute, crowdsourcing has become an economical, effective, and justified mechanism to carry out initiatives that offer social benefits but cost too much to be deployed by any single entity. Notable examples include Wikipedia and Salt Lake City's use of crowdsourcing for transit planning [1]. A remarkable trend in crowdsourcing is the use of mobile devices: these break the time and space barriers between people and enable them to share information and knowledge. For example, mobile applications like *txteagle* are emerging alternatives to traditional platforms like AMT (Amazon Mechanical Turk) [2]. With the popularity of mobile social networking and the emergence of ideas like participatory sensing, mobile crowdsourcing has the potential to help tackle an array of new problems that involve real-time data collection from and coordination among a large

C. Borcea et al. (Eds.): MobilWare 2012, LNICST 65, pp. 16–30, 2013.

number of participants. In particular, mobile crowdsourcing can be harnessed to design smart parking solutions.

The parking problem has existed in big cities for decades. Studies show that an average of 30% of the traffic in busy areas is caused by vehicles cruising for vacant parking spots [3]. The situation is getting worse in developing countries like China, where the number of private cars has soared recently, while the investment in parking facilities has lagged. The additional traffic causes significant problems from traffic congestion, to air pollution, to energy waste. Some local governments try to mitigate these issues by deploying *smart parking* systems: systems that employ information and communication technologies to collect and distribute the real-time data about parking availability and may guide drivers so that they find parking spots quicker.

For example, the city of San Francisco installed thousands of sensors at on-street parking spaces in busy areas to make parking availability information public. Although the benefits of such a centralized approach are immediate, its huge initial investment and maintenance cost inhibits a widespread adoption in most other cities: the average maintenance cost for each sensor monitoring a single parking space is beyond $20 per month [4]. Even in San Francisco, the majority of parking spaces are not covered by the system likely due to its cost.

This paper studies the properties of crowdsourcing in the context of smart parking. More specifically, this work investigates the use of information collected through crowdsourcing for parking guidance, which is integrated into a road navigation system (as a design alternative to lower the cost to install and maintain a dedicated infrastructure). It is important to note, for example, that *Waze* [6] has already demonstrated that crowdsourcing using road navigation devices is feasible and has accumulated millions of users in over 45 countries. *Waze* collects most of the data we also employ for parking guidance; and, our system can be easily implemented as an extension to it.

This work, however, improves over existing approaches in a number of ways. First, by integrating crowdsourcing and a road navigation system, we eliminate unnecessary drivers' manual operations during the parking search process. This complies with the current safety regulation in most countries. Unlike applications such as *Open Spot* [8], which require drivers to launch them separately to search parking spots, we only ask drivers for their manual input at the beginning and the end of their trips. By simplifying operations, we are more likely to recruit a larger number of contributors, a key factor to crowdsourcing success.

Second, since drivers who contribute also benefit from the system, our approach heavily depends on a pattern of mutual assistance, which excludes the complexities caused by monetary rewards [5]. On the one hand, we demonstrated that the system is resilient to the existence of free riders (Section 5). On the other hand, as we assume a centralized control the distribution of collected date, the system can create incentives by providing users with different quality of service (e.g., better parking suggestions, request prioritization) based on their contribution records.

Finally, we guide/coordinate the crowdsourcing behavior among participants to improve data collection efficiency and system utilization. In contrast to existing approaches that only share information about parking vacancies, our system also tries

to identify occupied areas through user's sensor data (or explicit input) so as to help drivers avoid unnecessary cruising. Also, we assign parking spaces to users dynamically, according to the reported capacity of parking spots to eliminate races between participants. Furthermore, we take a proactive strategy to crowdsource when the knowledge is limited: more specifically, the system might direct drivers to unexplored areas so that it can expand its knowledge about parking availability in these areas.

Our contributions in this paper fall in two categories: On the one hand, we demonstrate, through simulations, that mobile crowdsourcing is a feasible and cost effective approach to deploy a smart parking system. On the other hand, we regard this application as a case study to demystify some rumors that have influenced the design of mobile crowdsourcing-based applications for a long time. We find that recruiting more participants may not necessarily lead to a better performance if the crowdsourcer fails to coordinate people's behavior in the context of these applications. We show that people can provide valuable data even through the simplest manual operation in a dynamic mobile environment if they are coordinated. We also discover that a proper policy to deal with free-riders will improve social benefits without sacrificing the quality of the crowdsourcing-based service. These findings can serve as a catalyst to facilitate the development of similar mobile applications and help double the number of success stories.

The rest of the paper is organized as follows: Section 2 positions this work among the related literature; Section 3 describes the parking guidance system and its different strategies to harness crowdsourcing; Sections 4 and 5 present the simulation design and the evaluation results; Section 6 concludes the paper with final remarks and discusses directions for future work.

2 Related Work

The huge demand for transportation-related services to simplify daily life is the driver for mobile crowdsourcing applications. Thanks to data crowdsourced through thousands of mobile devices, drivers are able to pick a better route to avoid a road segment that was detected as congested in the previous five minutes by Waze, to refill at a gas station with a lower price by GasBuddy [7], or find a parking place using applications like Open Spot [8]. Similarly, taxi drivers might improve their routes by knowing colleagues' trajectory [9] and commuters can get the real-time transit information from Roadify [10]. One feature shared by these mobile crowdsourcing scenarios is that they rely on data contributed by the consumers of these services. Therefore, these crowdsourcing-based services become sustainable if they can attract a sufficient amount of users.

Although the aforementioned applications have attracted great attention in the market (e.g., as estimated by their download count), they are orthogonal to the research interests of the academic community. As Kanhere discusses [11], current studies in mobile crowdsourcing or participatory sensing generally focus more on new applications (e.g., personal health monitoring [12], environmental surveillance [13],

or enhanced social media [14]) than on the impact of participating rates and crowdsourcing strategies. Issues like privacy preservation, incentive design, or evaluating the trustworthiness of data remain major concerns when deploying these applications into practice.

As far as smart parking is concerned, the majority of existing studies either assume the availability of gadgets installed at the parking lots or require *all* drivers to comply with the same protocol when reporting parking availability. Systems like [15] and SPARK [16] employ wireless sensors and, respectively, VANET (Vehicular Ad-hoc Network) devices to collect and disseminate information about parking availability to help drivers find vacant parking spaces. CrowdPark [5] assumes a seller-buyer relationship between drivers, who are going to leave or parking at the lots, to deal with the parking reservation problem. A relevant study [17] tries to realize smart parking by solving an optimal resource allocation problem according to drivers' various parking requirements. However, the reservation-based solutions might complicate drivers' operation and can collapse if only a few drivers follow their rules.

One remarkable initiative that realizes smart-parking by the infrastructure-based approach is the SFPark [18] project in San Francisco. Although the benefit is obvious, few cities worldwide can afford the high initial investment and the maintenance cost. Alternatively, some pure crowdsourcing-based solutions like Open Spot [8] are emerging but, to date, failed to solve the problem effectively. We believe there is a viable approach between these two extremes. More specifically, our approach is to introduce a central entity to coordinate participants' behavior in order to make mobile crowdsourcing not only a cheap but also an effective solution to the smart parking problem.

3 System Design

The **basic idea behind our design is to build** a system that acquires, possibly approximate or aggregate, parking availability information through crowdsourcing: each participating **driver** helps with data acquisition. **In return, the system provides** either the aggregate parking availability map and users make uncoordinated decisions or the system provides customized **recommendations of parking locations and navigation to the participants** and thus attempts to coordinate their behavior.

3.1 Assumptions

The goal of smart parking is to inform drivers of a parking vacancy as soon and as close to their destination as possible. The desired effect is to save the time and the fuel spent in cruising, reduce unnecessary walking, and reduce the traffic congestion and fuel waste. To this end, the crowd-based smart parking system collects relevant data from participating drivers, and then uses this data to navigate them to the right parking slots. For convenience, we refer to the drivers who participate in the system as *smart parkers (SP)* in contrast to those who do not participate as *ordinary drivers (OD)*.

The system consists of three components: central servers, client devices, and smart parkers. Figure 1 shows the relationships and the data flows between them. We make the following assumptions about their responsibility or functionality.

Central Servers: The servers collect data from drivers, who report their current location and destination, car speed, and parking availability on a certain street through client devices. Using the information collected in real time, the servers maintain a dynamically annotated parking availability map. When a smart parker arrives close to his destination, the servers search the dynamic map for potential parking vacancies according to the parker's current location and destination. Then they inform the client device of the search result, which might be either the specific location of the parking spot or the direction of the next turn to the parking spot.

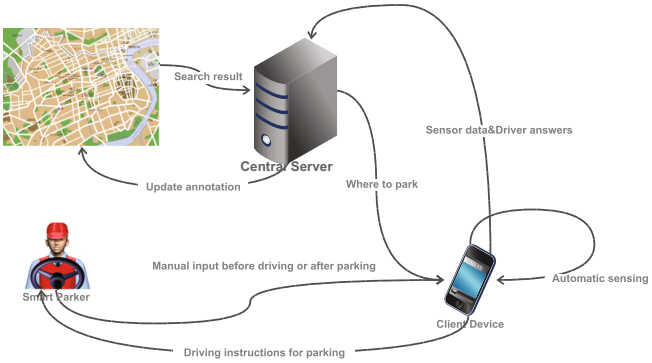


Fig. 1. Data flow among the central server, the client device, and the smart parker

In addition to this dynamic data, we also assume the servers have access to static data, which are relevant to parking guidance, such as the parking price, legal periods, and areas to park, and statistics about the arrival rate of vehicles and parking rate around a certain region during a certain period. In fact, an increasing number of cities provide these kinds of data online [20].

Client Devices: Drivers have on-board devices that can communicate with the server. They upload geo-tagged data and can download the result of queries regarding parking slots availability. It is reasonable to assume that such devices have GPS capability and Internet connection. A variety of off-the-shelf consumer electronics like smart phones, tablet PCs, and versatile GPS navigators can play this role. The client devices have a simple user interface that allows smart parkers to input relevant data manually when they are not driving. The devices can also collect geo-tagged sensor data automatically without drivers' intervention when the car is moving. We draw a self-loop on client devices in Figure 1 because the device might process the collected sensor data before sending them to the server.

Smart Parkers: Smart parkers are the drivers who have access to the service through their client devices. Like ordinary users of GPS navigators, a smart parker will input her destination before she starts driving. Then she will receive recommendations from the system about potential free parking slot when she approaches her destination. The smart parker can choose whether or not to follow such recommendations, but the client device will report her cruising trail to the server. At the beginning and the end of a trip, with the car stopped, the smart parker is expected to answer a question about parking availability in the area by manually handling the client device.

3.2 Problem, Key Questions, and Required Data

Three key questions guide the design of any crowdsourcing system: *What is the required data? How can this data be obtained through crowdsourcing? How can the acquired data be used in the specific application scenario?*

In our parking scenario, we model each road segment as a parking lot with several parking spots along it. To realize on-street smart parking, we need to navigate smart parkers to streets that are not fully occupied. In other words, we need to acquire the status of the parking availability along each road segment.

From the server's perspective, each road segment could have one of the three statuses for its parking availability: *available*, *occupied,* or *unknown*. Initially, the status of all streets is marked as *unknown*. Once information is received the status can switch to *available* or *occupied*.

Unlike smart parkers, ordinary drivers do not provide data thus when they arrive at or leave from a parking space, the change in status is not observed by the central server. Thus, for all parking spots, we automatically change the status to *unknown* when a timer expires. The timer length can be derived from statistic data or occupancy prediction [19] and can be adjusted through the observation of the crowdsourced data. In addition to the occupancy status, the system also needs to know the capacity of each on-street parking lot to determine if it can navigate two cars to the same street at the same time.

3.3 Crowdsourcing Data Acquisition

Crowdsourcing data acquisition in a mobile environment poses some challenges. An obvious problem is that a limited user interface and drivers' tight schedule require the device operation to be as simple as possible. In the case of a smart parking scenario, we might want smart parkers to observe the streets carefully and report a specific number for the parking capacity. However, most smart parkers will likely prefer to answer a much simpler Yes/No question by just pressing a button on their devices. Experiences from similar applications like Waze show that user-friendly interface and simple operation are key factors to recruit contributors.

We explore the impact of the varying accuracy of crowdsourcing based information (Table 1). *Our study (Section 5) shows that the answer to a simple Yes/No question is sufficient even with a low participation rate in the crowdsourcing system.*

Table 1. Different kinds of questions smart parkers could be asked

#	*Question*	*Answers*	*Capacity*
Q1	How many parking spots on the street?	0,1,2,3…	As the answer
Q2	Any more parking spots on the street?	Yes/No	1(Yes)/0(No)
Q3	No question	No answer	Always 1

In addition to the above requirement for a simplified operation, the limited view of the participants could restrict their ability to provide accurate data. For example, by answering, smart parkers only inform the server of the situation of the street where they parked but tell nothing about the occupied streets they cruised through. However, we can infer such information from crowdsourced sensor data. More specifically, we assume a car to be cruising if it follows the server's instructions to reach a certain road segment but still keeps moving at low speed. Then we consider the road segment where the car starts cruising as *occupied*. Furthermore, all streets the car cruises without parking can also be regarded as *occupied*. In addition, we can mark a street as *available* if a car leaves from there. Since a car's cruising speed is only 20% of its normal driving speed, we can infer the above by just observing the sensor data like speed and location. We enumerate all three kinds of inference in Table 2.

Table 2. Different types of inference through sensor data

#	Observed behavior	Inference	Capacity
I1	Reach the assigned street and continue at low speed	The assigned street is occupied	0
I2	Move at low speed after I1	The past street is occupied	0
I3	Launch the application and drive away	New vacancy in the street	+1

3.4 Parking Guidance Alternatives: Coordinated vs. Uncoordinated

Once the server annotates each street on the map with its parking availability status, the simplest way to do parking guidance is to display the locations of available parking slots on a map directly to all drivers without attempting to coordinate them. However, this uncoordinated approach (also adopted by Open Spot) can lead to several problems.

First, it is usually difficult for drivers to integrate all information on the annotated map to make a good decision when driving. They could always focus on the same parking slots reported by other drivers, which might not always be their best choice. Furthermore, when drivers cruise along occupied streets, they cannot help others to avoid such areas which in turn contribute to longer cruising time. Due to the uncoordinated nature, smart parkers are less likely to explore unknown areas, where there could be more available parking slots closer to the destination.

To mitigate the problems, we propose to coordinate the drivers (instead of letting them choose where to park by themselves). To eliminate the race between two smart parkers for the same parking spot, we keep track of the capacity of each road segment and navigate smart parkers according to the streets' current available capacity. To find out the parking status around the *unknown* areas, we assume each *unknown* street has a capacity of one. Once a street is assigned to a smart parker, its capacity is reduced by one and we only navigate cars to streets with a non-zero capacity. If the assigned street is already fully occupied when the smart parker arrives there, we navigate the car to cruise toward streets with non-zero capacity. This way, we not only help the smart parker avoid unnecessary cruising but also increase the server's knowledge about *unknown* streets. The difference between our approach and uncoordinated crowdsourcing is shown in Figure 2. In simulation experiments we explore the sensitivity of the solution to the number of smart parkers that follow the coordination suggestions of the server.

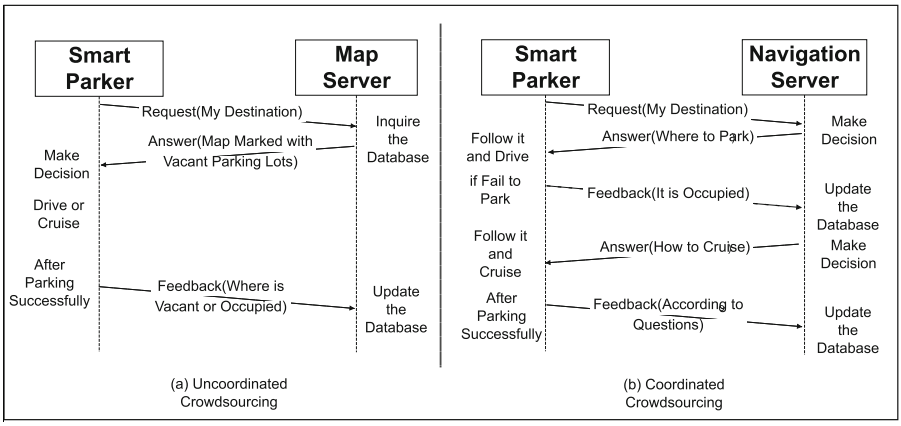


Fig. 2. Illustration between uncoordinated and coordinated parking guidance

4 Simulation Methodology

We explore the design space delimited by the design choices highlighted in the previous section through simulations. This section presents the situation settings.

4.1 Simulation Environment

To simulate the crowdsourcing-based system in the context of smart parking scenario, we need to take care of two aspects. On one hand, the simulations should reflect features in realistic road traffic environment like road layout, car following patterns, and individual driving behaviors. On the other hand, the simulation environment should be configurable to take into account the system design factors discussed in

Section 3. Since no existing simulation environments can satisfy all requirements, we modified an open source road traffic simulator, SUMO [21], to meet our needs.

SUMO is a microscopic road traffic simulator, which allows simulating thousands of vehicles moving through a road network. The simulator is capable of capturing the geospatial properties of each vehicle in motion like location and speed at any moment. This corresponds to our assumption about smart parkers: they should be able to report such information to the server. However, the existing environment heavily depends on predefined configuration files to determine the departure time and the route of each vehicle.

To simulate the dynamic scenario, in which vehicles arrive according to a Poisson process and cruise around for an open spot to park, we integrate the logic of vehicle generation and routing into the simulator. In addition, the adapted simulator adds the parking capacity as a new property of each street in the road network so that smart parkers will keep cruising in search of open parking slots until they enter a street with a non-zero parking capacity. Furthermore, we have implemented the data collection process and parking navigation inside SUMO to reflect the different crowdsourcing strategies mentioned in Section 3.

4.2 Simulated Scenario and Parameter Setting

Scenario: The simulation aims to evaluate the feasibility of the aforementioned crowdsourcing system in a simple but realistic scenario, where hundreds of vehicles are heading for the same destination during a short period of time and few cars leave the parking lots at that time. This often happens around office buildings and park-and-ride facilities [22] during rush hours or at a stadium before a game kicks off. This scenario helps us focus on the impact of different design choices for the crowdsourcing system rather than on the statistics related to parking lot usage around a certain area.

Parameter Setting: The road network in our simulations is modeled as a 1 km^2 region divided into a 9*9 grid by four-lane bidirectional streets. Each road segment has a parking capacity of 5 for either side of the street. In the simulator, the block in the center is assumed as a common destination and everyone tries to park close to it in order to reduce the walking distance. In each round of simulation, a sequence of about 1,000 vehicles enters the map according to a Poisson process. The arrival rate is set to one car every 15 seconds.

The simulator determines whether a new coming driver is a smart parker by a certain probability so that it is possible to control the approximate ratio between the two groups of drivers. If an ordinary driver cannot find an open spot on the destination street, he will have to cruise around randomly until he can find one somewhere else. The speed limit for normal driving is 50km/h while the cruising speed is below 10km/h.

When a smart parker moves close to the desired destination, the server will show her suggestions about the available parking place. If she follows the server's suggestion but reaches a fully occupied on-street lot, she also needs to cruise.

However, the parking guidance will help during the cruising if we adopt a coordinated guidance strategy. We run each simulation from 5 to 35 times and plot the average value.

5 Evaluation Results

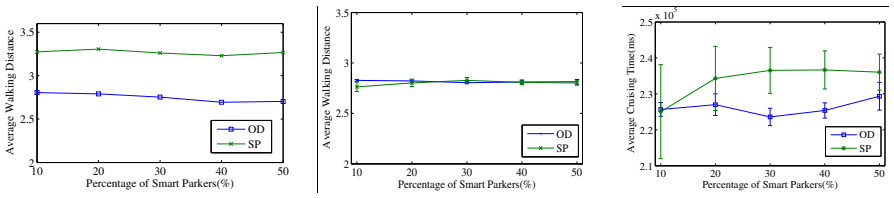
We explore the impact of three key design decisions: the impact of global coordination; the impact of collecting approximate data that leads to increased usability of the client devices; and the social impact of freeriding.

There are two success criteria for our system: the walking distance from the parking spot found to the actual destination (measured in 'blocks' – i.e., the distance between two crossroads) and the average cruising time to find a parking spot. Our results highlight that coordinated crowdsourcing is not only effective but also practical in the real world.

5.1 Uncoordinated VS Coordinated Crowdsourcing

The first question we focus on is: *Do smart parkers outperform ordinary drivers regardless of the type of crowdsroucing strategy used by the system?* We first assume that the system adopts a pure uncoordinated crowdsourcing strategy so that each smart parker just follows a predecessor who managed to park the closest to its destination and signals that parking spaces are still available in the area.

Figure 3(a) compares two groups of drivers with regard to the average walking distance. We collect the data as the participation rate (i.e., the ratio of smart parkers in the system) increases from 10% to 50% of the driver population. As Figure 3(a) shows, uncoordinated crowdsourcing leads to longer walking distance for smart parkers than for ordinary drivers. Since the system does not provide smart parkers with a global view around the region, they miss potential vacancies closer to their destination.

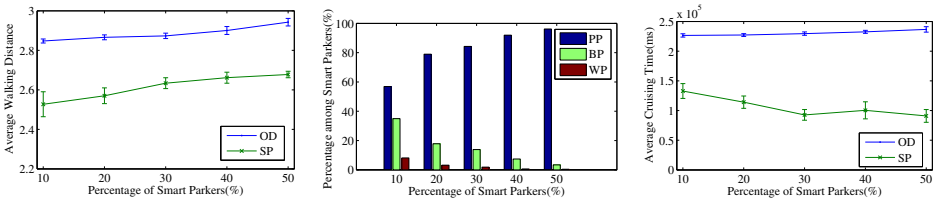


(a) (b) (c)

Fig. 3. Performance comparison between ordinary drivers and smart parkers adopting an uncoordinated crowdsourcing approach. The walking distance in (a) and (b) is measured in number of blocks away from the central destination. Error bars here indicate the 95% confidence interval.

Smart parkers might not always follow directions to the recommended spots, which are far away from the destination. Thus we next assume that they choose to cruise by themselves and ignore the system's recommendation if the recommended spots are more than three blocks away from the destination, which is the median value for the walking distance. The resulting walking distance and average cruising time are shown in figure 3(b) and 3(c) respectively. Although smart parkers don't lose to ordinary drivers in terms of average walking distance this time, about 40% of them spend more average search time than ordinary drivers.

The previous figures show that the uncoordinated crowdsourcing approach, also used by Google's Open Spot, fails to help users do a better job than ordinary drivers in the search of parking spots regardless of how many drivers participate.



(a) (b) (c)

Fig. 4. Performance of smart parkers when their behavior is coordinated in the crowdsourcing system. Error bars here indicate the 95% confidence interval.

We assume now a coordinated approach where the system collects information by applying option Q1 in table 1 and I2 in table 2. In addition, it assigns smart parkers to explore unknown areas and helps them cruise more efficiently by avoiding occupied streets. As the figure 4(a) shows, such an approach achieves a lower average walking distance for smart parkers.

To report results, we divide the smart parkers into three groups: the majority of them can find an open spot immediately according to the system's navigation and we call them perfect parkers (PP). Those who still need to cruise but spend less time than the average cruising time of ordinary drivers are referred to as better parkers (BP). The rest are called worse parkers (WP) as they spend more time cruising. Figure 4(b) shows the change of the composition of smart parkers as more drivers participate. More than 90% of the smart parkers do not need to cruise when the membership covers about 40% of all drivers. For BP and WP, we calculate their average cruising time and plot the result in figure 4(c). *All these figures show that the coordinated crowdsourcing is more effective in the smart parking scenario.*

5.2 Impact of Various Design Options Leading to Increased Usability

The second question we try to study concerns both developers and users: *Could the data collection process be simpler (user friendly) while still achieving the application objectives?* To make it clear, we let smart parkers answer simpler questions (as in

Table 1) to report relevant information to the server and then estimate their impact on system efficiency (for which we use the number of perfect parkers as a proxy). We plot the results in figure 5(a) with each line for a specific question. The figure reflects that the answers to a Yes-No parking availability question provide sufficient information for the server to implement a useful navigation service. Next, we repeat the experiments without inferring the occupancy through cruising vehicles. By comparing figure 5(a) and 5(b), we find that the information inferred through cruising cars is helpful when only a few smart parkers participate but its importance diminishes as more drivers join the system.

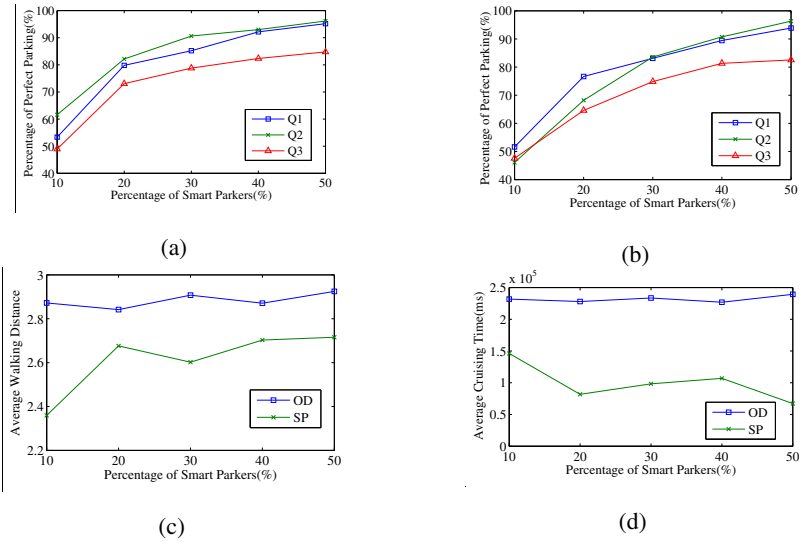


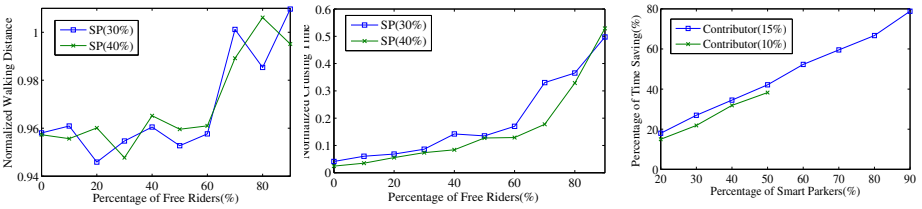
Fig. 5. Influence of various crowdsourcing options to the performance of smart parking system. In (a) and (b), we ask different questions after smart parkers find their spots. We first turn on in (a) and then turn off in (b) the option to infer occupancy through cruising cars. In (c) and (d), we only ask drivers if or not there are additional spots on the street where they parked the cars.

To increase confidence in the preliminary conclusion we can derive (i.e., answers to simpler questions can still provide sufficient information to navigate smart parkers properly) we assume that smart parkers only answer question Q2 and no information is inferred when they are cruising. In other words, the server only asks the Yes-No question this time and makes no inference about *occupied* streets when smart parkers are cruising. We compare smart parkers with ordinary drivers again with regard to the average walking distance and cruising time in figure 5(c) and 5(d) respectively. Since the majority of smart parkers (at least 50% of them) do not cruise at all, we only plot in figure 5(d) the average cruising time for those smart parkers who need to cruise, namely the better parkers(BP) and the worse parkers(WP). *The figures show that the crowdsourcing system is still effective even when participants only answer simple questions.*

5.3 The Impact of Free Riders

The last set of simulations deals with another realistic question: *How should the system handle free riders?* In the context of our system, free riders are those participants who only want to take advantage of the service but refuse to answer any question. As part of the feasibility evaluation, we need to evaluate how tolerant the crowdsourcing system is to freeriding and decide how to handle them: tolerate or attempt to exclude them.

The average walking distance among all drivers and the average cruising time of ordinary drivers do not change much across all experiments. Therefore we use them as a reference to test if smart parkers can still find the open spot quickly and park closer to their destination even in the presence of free riders. In the following simulations, we only ask Yes-No question without data inference during drivers' cruising. We assume that 30% to 40% of all drivers are smart parkers. Among the smart parkers, the percentage of free riders grows from 10% to 90%. We plot the normalized walking distance for smart parkers in figure 6(a), namely the average walking distance of the smart parkers divided by the average walking distance among all drivers. Similarly, we plot the normalized cruising time for smart parkers in figure 6(b). The figure shows that the navigation system works well until the percentage of free riders exceeds 60%. In other words, we can infer that the quality of the service is still acceptable as long as at least 12% of all drivers are willing to contribute.



(a) (b) (c)

Fig. 6. Tolerance for free riders in the crowdsourcing-based smart parking system. In (a) and (b), we assume two fixed percentages of smart parkers among all drivers and record their performance as more smart parkers become free riders. In (c), we assume two fixed percentages of contributors among all drivers and record how much cruising time can be saved as more people become free riders.

As mentioned, the average cruising time of ordinary drivers does not change much as the share of smart parkers grows. This means that the overall cruising time of all drivers will remain a constant if the smart parking system is not available. If we use this cruising time as a measure of social welfare, allowing freeriding could boost social welfare, which means the overall time and fuel consumption will be reduced, as long as the service is still usable. In the following simulations, we assume the contributors account for 10% or 15% among all drivers while the percentage of smart parkers (including both contributors and free-riders) grows from 20% to 90% of the population. Then we calculate the percentage of the time saving as long as the system

is able to keep smart parkers closer to their destination. As figure 6(c) shows, when contributors and free riders account for 15% and 35% of the population respectively, above 40% of the overall cruising time can be saved. As we mentioned before, it is difficult to maintain the quality of the service if the number of free-riders keeps growing while only 10% of all drivers contributing to the system. However, if the amount of contributors reaches 15%, the system can accommodate much more free-riders and reduce the overall cruising time significantly. *These results show that we should allow free riders to exist if there are enough contributors in the crowdsourcing system.*

6 Conclusion

Mobile crowdsourcing is an increasingly popular mechanism to realize applications that harness a large volume of real-time data to improve daily life. Crowdsourcing, however, brings several new issues that arise only in the context of participatory, peer-to-peer systems. In this paper, we take smart parking as a usecase and explore the possible design options to deal with these issues. At the same time, we summarize design guidelines to build mobile crowdsourcing applications.

In particular, our study leads to the following findings:

First, a naïve crowdsourcing implementation in a mobile environment can lead to 'herd' behavior rather than collective intelligence since each participant only has a limited view of his surroundings and a global picture of the physical world is not realizable. To deal with this issue, we propose 'coordinated crowdsourcing', in which a server integrates all information from participants and encourages them to explore unknown area. Our simulations show that the coordinated crowdsourcing is an effective approach in a mobile environment.

Second, the participation rate is more important than the volume of information each individual contributes. Our simulations show that, when the membership rate of a crowdsourcing system passes a certain threshold, the outcomes remain stable regardless of how much information each individual contributes and its accuracy. However, if the participation rate is low, a sophisticated data collection mechanism becomes necessary to compensate the lack of data sources.

Finally, the crowdsourcing-based application might continue to increase social welfare by tolerating free riders, as long as it can maintain a moderate level of contribution among participants. In the context of mobile crowdsourcing, free riders could reduce the quality of the crowdsourcing-based service as they might benefit from the system, and change the status of the physical environment without reporting new information. However, the aggregated social benefit for all participants could still rise significantly (at the cost of a slightly degraded service quality) as long as a certain percentage of the members keep contributing their data.

References

1. Brabham, D.C., et al.: Crowdsourcing Public Participation in Transit Planning: Preliminary Results from Next Stop Design Case. In: TRB 89th Annual Meeting Compendium (2010)
2. Sorokin, A., Forsyth, D.: Utility data annotation with Amazon Mechanical Turk. In: Computer Vision and Pattern Recognition Workshops (2008)
3. White, P.: No Vacancy: Park Slopes Parking Problem And How to Fix It, http://www.Transalt.org/newsroom/releases/126
4. Kessler, S.: How Smarter Parking Technology Will Reduce Traffic Congestion (2011), http://mashable.com/2011/04/13/smart-Parking-Tech/
5. Yan, T., et al.: CrowdPark: A Crowdsourcing-based Parking Reservation System for Mobile Phones. UMASS Technical Report, Tech. Rep. UM-CS-2011-001 (2011)
6. Waze, http://www.Waze.Com/
7. GasBuddy, Find Low Gas Prices in the USA and Canada, http://gasbuddy.Com
8. Kincaid, J.: Googles Open Spot Makes Parking A Breeze, Assuming Everyone Turns Into A Good Samaritan, http://techcrunch.com/2010/07/09/google-Parking-Open-Spot/
9. Li, B., et al.: Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In: Pervasive Computing and Communications Workshops (2011)
10. Lamba, N.: Social Media Trackles Traffic (2010), http://www.Wired.com/autopia/2010/12/ibm-Thoughts-on-a-Smarter-Planet-8/
11. Kanhere, S.S.: Participatory Sensing: Crowdsourcing Data from Mobile Smartphones in Urban Spaces. In: Mobile Data Management, MDM (2011)
12. Reddy, S., et al.: Image Browsing, Processing and Clustering for Participatory Sensing: Lessons from a DietSense Prototype. In: Workshop on Embedded Networked Sensors (2007)
13. Mun, M., et al.: PEIR, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research. In: MobiSys 2009 (2009)
14. Miluzzo, E., et al.: Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application. In: ACM SenSys, USA (November 2008)
15. Chinrungrueng, J., et al.: Smart Parking: An Application of Optical Wireless Sensor Network. In: Applications and the Internet Workshops (2007)
16. Lu, R., et al.: SPARK: A New VANET-Based Smart Parking Scheme for Large Parking Lots. In: INFOCOM 2009, pp. 1413–1421. IEEE (2009)
17. Geng, Y., Cassandras, C.G.: A new "smart parking" system based on optimal resource allocation and reservations. In: Intelligent Transportation Systems, ITSC (2011)
18. SFMTA, SFPark- About the Project, http://sfpark.org/about-the-Project/
19. Caliskan, M., et al.: Predicting Parking Lot Occupancy in Vehicular Ad Hoc Networks. In: IEEE 65th Vehicular Technology Conference, VTC 2007, pp. 277–281 (Spring 2007)
20. Anonymous "Parking Meter Rates and Time Limits", http://vancouver.ca/vanmap/p/parkingMeter.html
21. Behrisch, M., et al.: SUMO - Simulation of Urban MObility: An Overview. In: The Third International Conference on Advances in System Simulation, SIMUL 2011 (2011)
22. Tsang, F.W.K., et al.: Improved modeling of park-and-ride transfer time: Capturing the within-day dynamics. Journal of Advanced Transportation 39 (2005)