Hit-or-Wait: Coordinating Opportunistic Low-effort Contributions to Achieve Global Outcomes in On-the-go Crowdsourcing

Yongsung Kim
Northwestern University
Evanston, IL
yk@u.northwestern.edu

Darren Gergle
Northwestern University
Evanston, IL
dgergle@northwestern.edu

Haoqi Zhang
Northwestern University
Evanston, IL
hq@northwestern.edu

ABSTRACT

We consider the challenge of motivating and coordinating large numbers of people to contribute to solving local, communal problems through their existing routines. In order to design such "on-the-go crowdsourcing" systems, there is a need for mechanisms that can effectively coordinate contributions to address problem solving needs in the physical world while leveraging people's existing mobility with minimal disruption. We thus introduce *Hit-or-Wait*, a general decision-theoretic mechanism that intelligently controls decisions over when to notify a person of a task, in ways that reason both about system needs across tasks and about a helper's changing patterns of mobility. Through simulations and a field study in the context of community-based lost-and-found, we demonstrate that using Hit-or-Wait enables a system to make efficient use of people's contributions with minimal disruptions to their routines without the need for explicit coordination. Interviews with field study participants further suggest that highlighting an individual's contribution to the global goal may help people value their contributions more.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords

Physical Crowdsourcing; Mobile Crowdsourcing; Decision Theory

INTRODUCTION

The growth of mobile devices in recent years has helped to bring about mobile [17, 15, 3] and physical [1, 2, 5] crowdsourcing systems that help connect people to solve local, communal problems. In these mobile and physical crowdsourcing systems, people make small contributions toward a larger collective problem, such as tracking animal species or air quality for citizen science projects or providing rides or delivering packages in commercial applications. In these systems, opportunistically relying on people to do convenient parts of the problem often leads to incomplete solutions [18, 35]. For example, time banking systems may complete only a fraction of the tasks requested, even days after the requests [18]. Yet, directing people to do inconvenient tasks decreases their willingness to complete them and therefore requires higher incentives. For example, tasks can require significant travel that strongly decreases people's willingness to complete tasks [35].

To overcome such shortcomings, recent work has proposed *on-the-go crowdsourcing* as an alternative model for connecting helpers to physical tasks by notifying people of tasks in situations where they can effectively contribute during their existing routines and routes [33, 14, 27]. Doing this should bring attention to tasks when people are likely to be able to help, thereby increasing people's willingness to participate and reducing the need to incentivize participation. Unlike existing commercial services where workers are mostly available on-demand, on-the-go crowdsourcing systems attempt to opportunistically leverage help resources as they become available, at low cost and with minimal disruption.

While previous work in on-the-go crowdsourcing has focused on studying the situations and contexts [14, 27] in which contributors may be willing and able to help, less attention has been paid to efficiently leveraging helper efforts where they are most needed. Fully realizing the benefits of on-the-go crowdsourcing requires resolving the general challenge of efficiently eliciting convenient helper contribution during their routine, while keeping the cost of disruption low by not inundating a helper with notifications of task opportunities. In practice, among the many tasks users may encounter during their routine, deciding which ones to notify them about directly affects which tasks are completed and what outcomes are prioritized. Given uncertainty in participation and potential helpers' future trajectories, it is challenging to set effective task notification policies that engage helpers where they are most needed. For example, we may wait for the best match of a helper to a task (e.g., where the task is highly valued and the helper faces low costs of diversion), only to find that the person never comes near such tasks in their realized routes. We can also be overly opportunistic, sending a person the first task they come across, only to realize that adopting this strategy while aiming to avoid over-disruption may require us to give up on better matches that become available later.

CHI '18, April 21–26, 2018, Montreal, QC, Canada
ACM 978-1-4503-5620-6/18/04. . . $15.00
DOI: https://doi.org/10.1145/3173574.3173670

To resolve this challenge, we introduce *Hit-or-Wait*, a general decision-theoretic mechanism that intelligently controls decisions over when to notify a person of a task among many tasks that they can contribute to along their existing routes, in ways that reason both about system needs across tasks and about a helper's changing patterns of mobility. Hit-or-Wait determines whether to send a task that is near a helper now—which we call *hitting*—or to *wait* for a better opportunity. Instead of simply assigning people to tasks—which is infeasible in on-the-go settings in which helpers may never go out of their way and thus may never reach assigned tasks—Hit-or-Wait continually reasons about the tradeoffs between current and future *situations* an individual may come across to make decisions about whether to notify of a task by modeling people's movements and the system's needs across tasks. Hit-or-Wait thus attempts to make effective use of every potential helper toward a global goal, while using only people's existing mobility and notifying them of tasks on the way that they can best help with. As such, it achieves much of the benefits of explicit coordination but without requiring helpers to go out of their way, to actively seek out tasks, or to reason about which task they should contribute to.

We evaluate Hit-or-Wait both in simulations and in a field deployment in the context of community-based lost-and-found, where we use Hit-or-Wait to indirectly coordinate people to look for a lost item on their way by recruiting individuals to collectively search for the item across small subregions (i.e. tasks) they may encounter. In simulations, we found that Hit-or-Wait significantly outperforms our baseline task notification strategy and approaches the performance of the myopic optimal solution which has full knowledge about future trajectories. In a field study with 25 participants, we found that Hit-or-Wait coordinated small, opportunistic contributions to achieve globally effective solutions by minimizing disruptions and maximizing the value of individual contributions. Interviews with field study participants further suggest that highlighting an individual's contribution to the global goal may help people value their contributions more.

BACKGROUND

A general challenge facing all crowdsourcing systems is the dual need to recruit contributors and to make effective use of contributions to best address task needs. In online crowdsourcing, system designers can reason about such needs separately; once a person is recruited to an effort, they land on a website where the most valued, compatible task (i.e. user can contribute) can be delivered. This separation of concerns allows one to design mechanisms for motivating and recruiting users (e.g., [6, 8, 38, 39]) independently of mechanisms for coordinating contributions (e.g., [7, 31, 41, 12]), regardless of whether tasks are assigned to workers by a system (as is typical) or dynamically determined by workers (as in [41, 12]).

In contrast, in mobile and physical crowdsourcing systems the tasks that a person can readily contribute to depend largely on the person's physical location relative to the location of tasks [21]. In other words, the tasks that are most in need of completion or that best match a worker's abilities cannot be readily presented unless the person can be motivated to arrive at the task's location. As a consequence, efforts to motivate and coordinate physical crowds cannot consider these two problems in isolation, and instead require the design of system-level mechanisms like Hit-or-Wait that are capable of reasoning jointly about the needs of the system and the changing availability of contributors.

The different models of mobile and physical crowdsourcing lead to particular challenges and tradeoffs for recruiting and coordinating workers. In the *opportunistic*, or *pull-based approach*, it is up to the workers to choose which tasks to contribute to. Even though a system can display, upon request, nearby tasks a worker can best contribute to, this approach can lead to many missed opportunities as it is only effective if workers actively look for tasks as they move about so as to happen upon high-valued tasks nearby [18].

In the *directed*, or *push-based approach*, workers are assigned tasks that best address system needs given worker locations and characteristics with the assumption that users' future routes can be determined by the system (e.g., in commercial services like Uber and PostMates) or the routes are known a priori [22, 40, 10]. This admits the use of standard optimization techniques to maximize the efficiency of the system through effective task assignments [9], but requires strong incentives to recruit on-demand workers who may have to go out of their way to complete tasks [35, 34].

In the *on-the-go approach*, workers are sent proactive task notifications in *situations where they are likely able to contribute to tasks that are valued by the system*. Given many tasks with differing values and priorities that need to be completed, effectively connecting users to tasks requires mechanisms to manage a delicate balance between recruiting users in situations where they are able to help and making efficient use of their efforts. While existing task assignment mechanisms are effective for directed approaches [11, 25, 26], they are not effective in the on-the-go setting where the user determines their own future routes and thus may never reach the tasks they are assigned. Even if these mechanisms took into account the uncertainty in future routes [9], pre-assigning tasks to users is still ineffective as it unnecessarily pre-determines who should do what tasks, which in the on-the-go setting will depend on the tasks that users actually encounter in their routes. Instead of assigning tasks, Hit-or-Wait offers a more flexible approach that reasons about whether to surface a task need at the current location or to wait to surface a different task need at a later time. To do this, Hit-or-Wait uses decision-theory over predictive models of people's routes and models of system needs to determine, on-the-fly, when to engage users for opportunistic contributions that are convenient to them, valued by the system, and that ultimately lead to globally effective solutions.

While prior work in *opportunistic planning* [19, 24, 20] had considered the problem of choosing which task to present given uncertainty over a user's route, this choice was static and assumed that a system had to make a decision at a fixed moment in time [20]. When unsure of which tasks a user may encounter, such a system may resort to asking a user directly for information about their route, which reduces uncertainty but adds extra effort on the part of the user. In order to reason

flexibly about changing conditions without user intervention, Hit-or-Wait moves away from optimizing among a set of tasks towards optimizing over immediate situations and possible future situations. This allows Hit-or-Wait to coordinate contributions dynamically, by controlling when and whether to engage a helper as they move from place to place.

In considering a dynamic sequence of decisions over whether to hit or wait, our approach bears resemblance to the use of decision-theoretic methods in online crowdsourcing that optimally control what tasks to allocate and when to stop allocating tasks [13]. Whereas efficiency is the primary reason for using decision-theory in earlier work, in our setting, the use of decision theory is further motivated by its ability to empower a seamless and lightweight form of interaction that requires no attention of potential helpers until a task request is made. Following arguments made by Kim et al. [27], we hypothesize that increasing the ability for people to conveniently contribute to local, communal problems may help to engage and sustain contributions over time, providing important benefits beyond any efficiency gain in a single scenario. As previous work has shown that highlighting the uniqueness and benefits of user contributions can elicit more contributions [4, 32], we study how helpers perceive the value of their contributions and explore ways to better communicate how Hit-or-Wait decisions make effective use of helpers' efforts.

COORDINATE ON-THE-GO CROWDS WITH HIT-OR-WAIT

In this section, we briefly review the core challenges of coordinating on-the-go contributions, introduce opportunistic *Hit-or-Wait* for coordinating on-the-go contributions in a way that achieves desired global outcomes, and describe our technical architecture that supports integrating Hit-or-Wait into on-the-go crowdsourcing systems.

As a reminder, there are several core challenges when attempting to coordinate on-the-go contributions: First, there is greater uncertainty around worker participation because on-the-go crowds consist of mobile community members and not dedicated workers. Second, task notifications need to be sensitive to the opportunistic nature of participation and cannot be overly burdensome or disruptive to potential helpers. Third, the system needs to be able to predict future routes based on current movement patterns, and make decisions while reasoning about the uncertainty of the predicted routes. Finally, the overall uncertainty that surrounds participation and future routes makes successful pre-defined task assignment implausible, and requires solutions to make decisions about when to engage potential helpers in an online manner.

To address these challenges, we present opportunistic Hit-or-Wait as a general decision-theoretic mechanism for coordinating on-the-go contributions. Hit-or-Wait aims to dynamically coordinate contributions in a way that achieves effective global outcomes by considering both current and future situations, and to notify potential helpers of tasks that they can conveniently contribute to and that most need their help.

Opportunistic Hit-or-Wait

We consider an on-the-go crowdsourcing setting with a set of tasks $\mathbf{T} = \{T_1, T_2, T_3, \ldots\}$ that are distributed across a physical space. Tasks may be of varying values that denote their priority, importance, or fit for a helper; task values are assumed known or can be estimated by the system. For any potential helper who may be able to contribute, we consider the problem of deciding, on-the-fly, which task to notify the helper of among the possibly many tasks the helper passes by. To make these decisions, the system can make use of an available *movement model*, which predicts a potential helper's future trajectories given historical data and the helper's current contexts. In order to not overly burden potential helpers, we assume that each potential helper may be notified of at most one task within a given time horizon. The goal is to notify helpers of tasks that they reach that are most valued, but with the caveat that given uncertainty in future routes it is possible to notify too early and miss a higher valued task that is reached later, or to pass on a valued task now, when in fact there are no better future opportunities on the horizon (or none at all).

To approach this problem, we model a sequence of *Hit-or-Wait* decisions with a Markov Decision Process (MDP) over a finite time horizon. A MDP consists of a set of states $s \in S$, available actions a in each state s, a transition function $P(s'|s)$ representing the likelihood of reaching state s' from state s, and a reward function $R(s,a)$ that defines the value of taking action a at state s. In Hit-or-Wait, states in the MDP represent possible situations the helper may reach. Each state s encodes the location of the helper, the task that is at that location (if any), and additionally, other contextual information about the helper's particular situational context (e.g., just left work). A helper transitions from state to state probabilistically, based on the movement model which provides the transition function $P(s'|s)$. Upon reaching a state that contains a task, the system has two possible actions: *hit* or *wait*. Hitting in state s with a task T notifies the helper of the task, and results in a reward that denotes the expected value of the helper completing the task. Waiting results in no reward and triggers a transition to the next state, while hitting triggers a transition to a terminal state to model only notifying a helper of at most one task.

In order to determine whether to notify the helper of a task in a given situation, we compute using the MDP an optimal policy π such that $\pi^t(s)$ denotes the decision to hit or to wait when the helper is at state s at time t. Computing this policy compares the expected value of hitting now with the expected value from making a decision later if we wait. Formally, we can represent the value of the optimal policy as:

$$V^t(s) = max(R(s,hit), \sum_{s'} P(s'|s)V^{t-1}(s'))$$

Which states that the expected value of the best decision $V^t(s)$ is the the maximum of the expected value of hitting now and the expected value of the best future decisions. Using this recurrence relation, we can solve for the optimal Hit-or-Wait decisions using dynamic programming.

Example Scenario: Lost and Found. To better illustrate how *Hit-or-Wait* can be used in an on-the-go crowdsourcing setting, we describe the algorithm in the context of a community-based lost-and-found scenario. Given a person

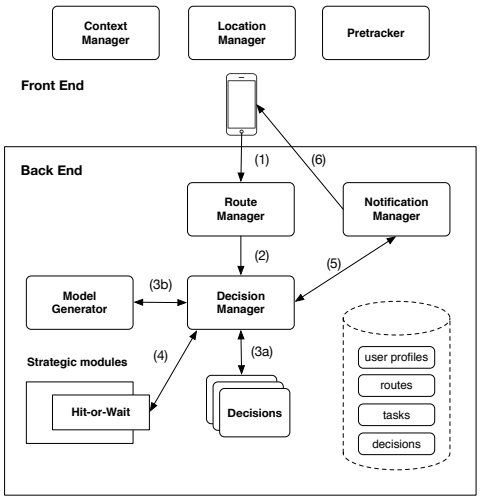


Figure 1. On-the-go crowdsourcing architecture.

who lost an item somewhere in a large region, the goal is to coordinate helpers' existing on-the-go mobility to effectively search for the item. Helpers contribute to small tasks that each request a search in a smaller subregion where the item may have been lost. The system must decide for each potential helper, whether to notify them to search in a subregion they are in, or to wait for another opportunity. The goal is to maximize the value of notifying by notifying a user in a less-searched region and wait if they are in a well-searched region.

To model and solve this problem, we can construct a Hit-or-Wait MDP for the lost-and-found scenario as follows: states represent subregions in the large region where the person might have lost an item that contain the search tasks, and the reward models the likelihood that the item is in each subregion. For instance, we may model the reward for searching in a subregion as the likelihood that the item is there following n (unsuccessful) searches, or $P(item|n)$. Assuming the likelihood of finding the item is conditionally independent given the item is in the subregion, we can compute $P(item|n)$ using Bayes' rule: $P(item|n) = \frac{P(item)P(n|item)}{P(item)P(n|item)+P(\overline{item})P(n|\overline{item})} = \frac{P(item)P(1|item)^n}{P(item)P(1|item)^n+P(\overline{item})}$, where $P(item)$ denotes the prior probability that the item is in the subregion, and $P(n|item)$ denotes the likelihood of n unsuccessful searches given that the item is there. Given that an item is more likely to be there after fewer searches, as a potential helper walks around the neighborhood where the item may have been lost, Hit-or-Wait will tend to notify them to search in a less-searched subregion than in other subregions they might encounter.

On-the-go Crowdsourcing Architecture

Building on-the-go crowdsourcing applications powered by Hit-or-Wait requires an architecture that can track location data, sense user's context, and make decisions of when and which tasks to notify based on user location and context. The architecture is described below and shown in Figure 1.

Architecture

The *Location Manager* and *Context Manager* collect user's location data and other contextual information and communicate them to the back-end.

The *Pretracker* helps deliver precise, fine-grained notifications by managing device's location accuracy depending on user's current location. For example, if a user is far from a task region, it decreases the location accuracy and increases it once the user is nearby a task region. Since it dynamically manages location accuracy, it can both save device battery and deliver fine-grained notifications at or near a task location.

The *Route Manager* processes incoming user location data in the back-end. It maps latitude and longitude pairs to states, stores new trips, or updates the existing ones in the database.

The *Model Generator* computes models that are required for strategic modules. For example, it produces the movement model for state transitions using previous route histories.

The *Decision Manager* takes as input user profiles (including routes, contexts), models, and tasks, and generates as output decisions based on a strategic module (e.g. Hit-or-Wait).

The *Notification Manager* delivers notifications to the the front-end based on the decisions made by the Decision Manager when a user meets the notification criteria, which includes but is not limited to conditions over the user location and the frequency of notifications (e.g., to model disruption and to avoid over-notifying users of tasks).

Flows

Figure 1 demonstrates how the various components interact with each other. The Route Manager receives raw GPS coordinates from the user device (1), preprocesses and stores the data, sends them to the Decision Manager (2). The Decision Manager first checks whether or not there exists decisions computed for the given location (3a), if there exists decisions, it sends the decisions to the Notification Manager (5). Otherwise, the Decision Manager requests models from Model Generator (3b). Together with the user location, the generated models, user profile, and task needs, the Decision Manager chooses a strategic module to compute decisions (4), and finally sends the decisions to the Notification Manager. The Notification Manager considers notification criteria such as the distance to a task location, user profiles, and delivers a task notification if all the criteria are met (6).

STUDY 1: SIMULATION

We conducted a simulation study in a community-based lost-and-found setting to understand (a) the performance of Hit-or-Wait mechanism for indirectly coordinating contributions towards global goals, and (b) the effect of movement model accuracy on the performance of Hit-or-Wait.

Dataset and Modeling

To train a movement model and simulate the routes of on-the-go helpers, we scraped running routes from publicly available RunKeeper data in Chicago and its northern suburban area. The dataset contains 5,983 running routes from 2,419 users. It contains a total of 590,860 latitude and longitude pairs for an average of 98.76 points per user.

We model each subregion where an item may have been lost by representing individual road segments as states. We gather

road segment data from OpenStreetMap, which treats each segment as a connection between street intersections, represented as sequence of latitude and longitude pairs that construct the segment. We preprocessed our data following the steps from [28] but adopted the following heuristic for converting GPS traces into a sequence of adjacent road segments. For each latitude and longitude pair in a runner's GPS trace, we sought a road segment within 40 meters in the OpenStreetMap dataset. If we couldn't find the nearest road segments, we marked the road as Unnamed Road. We eliminated repeated road segments to finish constructing the sequence of segments.

We used the processed data to generate a population-based movement model where the transition probability from one road segment to the next is trained using the frequencies observed in the data. We consider a first-order Markov model where predictions of next locations are conditioned only on current locations. We used population-based model instead of individual-based model because there were not enough individual route histories to train an accurate individual-based model. For routes on which no training data exists, we used a simple model trained across our dataset that assigns a probability distribution over going straight, turning left or right.

Simulation I: The Efficiency of Hit-or-Wait

Study Procedure

We compare the performance of Hit-or-Wait with a simple *node counting* algorithm and with a *myopically optimal* solution given full knowledge of people's routes. The node counting algorithm notifies a person of a task in a subregion if and only if the search count in that subregion is the lowest among all subregions. This algorithm makes efficient use of each helper's effort, but in waiting for such opportunities may fail to recruit helpers who do not approach areas with low search count. The myopically optimal solution is omniscient of a helper's routes and notifies a helper in the subregion they come across that has the lowest search count. This solution serves as an upper bound on the performance of Hit-or-Wait given perfect predictions of people's future routes.[1]

To set up lost-and-found scenarios, we chose a road segment in our dataset with the highest foot traffic and included 41 nearby road segments to form the area for our study. Within this area, we randomly selected 10 road segments to represent the search subregions where the item may have been lost. We set the reward for searching in each subregion to the likelihood that the item is there after n people have (unsuccessfully) searched in that subregion (i.e., $P(item|n)$). For each trial of our simulation, we randomly sampled 100 routes from 428 running routes from 269 unique runners.

Measures and Analysis

We measure the performance of our algorithm against other algorithms by considering the *overall search quality* and *the number of missed opportunities*. Overall search quality provides a measure of how likely a search effort (i.e., the number of searches in each subregion) is to result in finding a lost item. For simplicity, we assume that the item is equally likely to be in each state,[2] and that searches are independent conditional on the item being in the search region. We set the likelihood of finding the item after the first search given that the item is in the subregion as 0.67. We let $V(s,n)$ denote the likelihood of having found an item after n searches when the item is in state s, and compute the quality of search as: $QoS = \sum_s V(s,n)/|S|$.

For the number of missed opportunities, we measure occurrences where a person, given their actual route, could have been notified to search along their route but were not notified (regardless of the value of contribution).

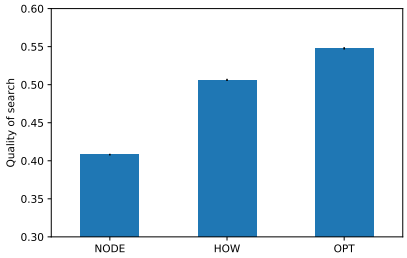


Figure 2. Simulation results comparing the overall quality of search for node counting, Hit-or-Wait, and myopic optimal. The results show that Hit-or-Wait outperformed the node counting algorithm and approached the performance of the myopic optimal solution.

Results of Simulation I

Figure 2 shows that Hit-or-Wait outperformed the node counting algorithm and approached the performance of the myopic optimal solution. It shows that Hit-or-Wait achieved 92.43% of the value of the myopic optimal solution, whereas node counting only achieved 74.5% of the value of the myopic optimal. Compared to node counting, Hit-or-Wait makes use of more of potential helpers' efforts by drastically lowering the percentage of missed opportunities compared to node counting algorithm; see Figure 3. On average, Hit-or-Wait algorithm missed 46.07% of opportunities (SD: 14.49) while node counting missed 78.68% of opportunities (SD: 10.53) by waiting for people to enter the least searched regions. While node counting notified users of the highest valued tasks exclusively, many users never reached such tasks; this led to a high percentage of missed opportunities that ultimately resulted in a lower overall quality of search than Hit-or-Wait. The myopic optimal solution has full knowledge of people's future routes, and thus misses no opportunities.

Simulation II: The Effect of Model Accuracy on Hit-or-Wait

Study Procedure

To understand the effect of movement model accuracy on the performance of Hit-or-Wait, we compare the performance of Hit-or-Wait using more and less accurate movement models

[1]To make the baseline comparison informative and compelling, we refrained from (a) comparing to approaches that notify users of tasks at non-nearby locations, which differs from our setting; and (b) comparing to directed approaches that pre-assign users tasks that may never be on their actual routes, as their performance would be similar to or worse than our chosen baseline.

[2]For the simplicity of the measure we treat the likelihood that an item is in a state as a constant, when in practice search counts contribute information about where the lost item is.

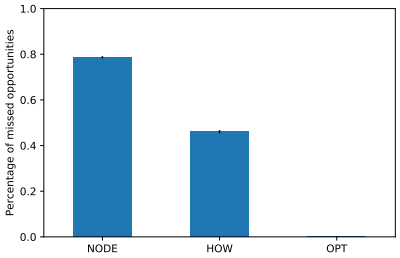


Figure 3. Percentage of missed opportunities for node counting, Hit-or-Wait, and myopic optimal solution. Hit-or-Wait algorithm missed 46.07% of opportunities (SD: 14.49) while node counting missed 78.68% of opportunities (SD: 10.53).

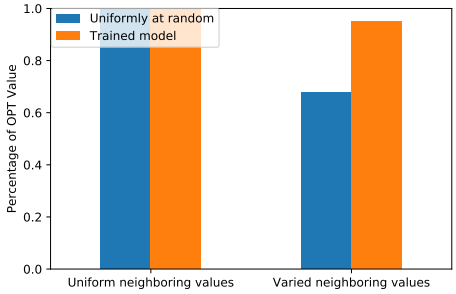


Figure 4. Effect of model accuracy on Hit-or-Wait performance in situations of uniform neighboring values and varied neighboring values.

across two types of situations: *uniform neighboring values* and *varied neighboring values*. In situations of uniform neighboring values, tasks in neighboring states are uniform in value; in such situations, we hypothesize that movement model accuracy has less impact on the performance of Hit-or-Wait because the value of future decisions is largely invariant. In situations of varied neighboring values, neighboring tasks differ in value; incorrect predictions of future routes are thus more likely to affect the quality of Hit-or-Wait decisions.

To set up an illustrative scenario, we chose a search region that consisted of three road segments that are in the area we had chosen in Simulation I for which the movement model is strongly discriminative. This allows us to observe different decisions when using Hit-or-Wait with our trained model and a less accurate model that transitions to neighboring states uniformly at random. We considered all 44 running routes that passed by this region, and considered each route as an instance of a potential search. We set the current road segment with value 0.6, and set the mean value of the neighboring road segments to 0.5. For uniform neighboring values, this should result in hit decisions at the current road segment regardless of the movement model. For varied neighboring values, we uniformly sampled a value in a range of 0.8 to 1 and set it as the value of the neighboring road segment more likely to be reached (and 1 minus that value for the other neighbor to preserve the mean of 0.5). This should allow a more accurate movement model to make wait decisions when it has strong predictions of reaching more valued states, whereas a less accurate movement model may still decide to hit.

Measures and Analysis

To study how movement model accuracy affects the performance of Hit-or-Wait with the trained model and a uniformly at random model, we measure the percentage of value captured with respect to the myopic optimal solution in situations of uniform neighboring values and varied neighboring values. We chose this measure instead of *overall search quality* because we are only looking at specific moments where we vary the task values, which means that there are no accumulated searches across the regions to compute overall search quality.

Results of Simulation II

Figure 4 shows the performance of Hit-or-Wait algorithm with a uniformly at random movement model and our trained model in the situations of uniform neighboring values and varied neighboring values. As we hypothesized, Hit-or-Wait using the uniformly at random movement model still achieves good performance in situations of uniform neighboring values, but not in the case of varied neighboring values. In the varied neighboring values case, Hit-or-Wait with our trained model captured 95.08% of the values of what myopic optimal was able to achieve, while Hit-or-Wait with the uniformly at random movement model only captured 67.66% of the values of myopic optimal. In this particular example, Hit-or-Wait made the same decisions as OPT in the case of uniformly neighboring values, as (a) incorrect predictions did not lead to any missed opportunities (e.g., the user ends up in a region without a task); and (b) the value gained for hitting in subsequent states is identical regardless of next states.

STUDY 2: FIELD DEPLOYMENT

Following our simulation study, we conducted a 10-day long field deployment of Hit-or-Wait in the lost-and-found domain to understand (a) the performance of Hit-or-Wait in comparison to a myopic optimal solution—our upper bound—in the real world, and (b) users' perceived disruptions. In addition to the simulations, this study allows us to explore the balance between hitting and waiting, the consequences of wait decisions, and when and why wait decisions may fail.

Trouve: Lost-and-Found Application

We developed a prototype, *Trouve*, a lost-and-found mobile application where users can request searches for lost items and it notifies people who pass by possible lost item regions to request that they look for the items. A user who lost an item can post a request by providing a lost item description and a possible region where they might have lost the item. When a potential helper passes by a subregion in the potential search region, they receive a notification (Figure 5a) asking if they can help look for the lost item there and then. Once they click the notification, the relevant information is shown to the user (Figure 5b). If the user decides to help, they can click "I am helping now!" to indicate their search attempt. After 30 seconds, a survey question about the perceived disruption of the 30-second search is shown to the user (Figure 5c).

Study Procedure

We recruited 25 people who had an iPhone 5S or above with iOS 10+ via flyers and local university mailing lists. 13 participants were male and 12 were female; the average age was

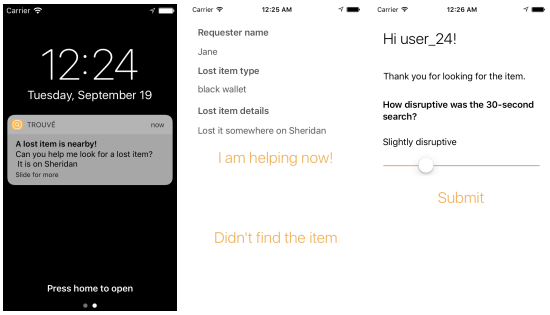


Figure 5. Trouve, a prototype lost-and-found app for the study. (a) A user receives a notification indicating a lost item is nearby; (b) the user sees the details of the lost item; (c) after 30 seconds of clicking I am helping now, the user is prompted with a survey question about perceived disruptiveness of the 30-second search.

21.7 (SD: 2.79). Participants consented to enrollment and then received the study instructions that asked participants to look for lost items for approximately 30 seconds when notified while traveling along their existing routes, and noted that searching was not mandatory for their participation. One of the authors acted as the requester, using Trouve to post lost items for participants to find.

We generated a movement model using the same procedure as in our simulation study, but with the training data coming from 51 routes from 11 recruited participants who used our location tracking app for a week prior to the study. Throughout the study we collected additional 1,490 routes and continually updated our model each time new route data was collected.

We chose two search regions near the university campus: one near the south side and another near the north side of the campus. Each search region included 5 road segments as its subregions; around 70-100 meters for each road segment. We sought to have a mixture of both high traffic and low traffic pedestrian streets within each search region, so we interviewed students who frequently traverse the regions and used the pre-study location data to help guide our choice of roads.

For each lost item request for a search region, one road segment was randomly selected as the lost item location. Based on common requests on the university's lost & found group, a lost item region was described as "somewhere on *street name #1* and *street name # 2*." Requested lost items included a wallet, a coin purse, and trinkets. The interval between task notifications (per user) was 4 hours to help minimize the overall disruption caused to the participants. Following each unsuccessful search in a subregion, we updated the search count and reduced the reward for subsequent searches in that subregion.[3] The search requests expired either when someone found the item or if no one found the item after 3 days.

[3]Due to an error in Hit-or-Wait implementation, we encoded the value of searching after n searches as $(1-P(1|item))^{n+1}$ in Study 2. While this value is decreasing in the number of searches as we would want, a more accurate estimate of the value of search should be based on $P(item|n)$ as shown earlier. Compared to using $P(item|n)$ as the reward function, our implementation overvalues states with more (unsuccessful) searches. This can lead Hit-or-Wait to make more hit decisions in well-searched regions when in actuality waiting for a less searched region would have been more valuable. As a result, the performance of Hit-or-Wait in our deployment may have been lower than if we had implemented the more accurate reward function. The error did not otherwise affect our measures, analyses, or findings.

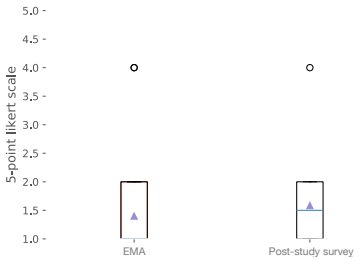


Figure 6. Responses from EMA and post-study survey about perceived disruption for 30-second searches on a 5-point Likert scale (1: not disruptive at all, 5: very disruptive).

Since the primary focus of the study was coordinating searches across road segments within a search region and not handing off found items or delivering them to a lost-and-found center, we asked the participants to simply take a picture of the found item and send it via SMS or email to the researchers. This way, we were able to verify whether or not the participants actually found the item. In the discussion, we will discuss more complex scenarios of Hit-or-Wait in which the helpers have to travel with and then hand off the found items. The participants received a $25 gift card as compensation.

Measures and Analysis

To understand the performance of Hit-or-Wait in the real-world, we considered two new measures: *the perceived cost of disruption* and *the value of waiting*. To measure the perceived cost of disruption, we used both ecological momentary assessments (EMA) and post-study survey, where the participants were asked to rate their perceived disruption on a 5-point Likert scale (ranging from "1: not disruptive at all," to "5: very disruptive"). The EMA was delivered to the participant via their smartphone (see Figure 5c) 30 seconds after they clicked "I am helping now!" We used both EMA and post-study survey to complement each other's strengths and weaknesses. On one hand, while EMA allows us to collect user responses while their memory is fresh, the responses were collected only when the participants decided to help, and therefore we miss responses when they found the tasks disruptive and did not help. This measure more effectively captures reflection on the disruption of the 30-second search task itself. On the other hand, the post-study survey allows us to capture participants' reflection on the amount of disruption they experienced throughout the study both including times when they decided to help and when they declined. One downside of this measure is that the participants' memory may not be as accurate after 10 days of study and their reporting may exhibit recall or recency bias.

In addition to *overall search quality* measure from Study 1, we added a measure that captures the value of waiting. We considered the wait decisions Hit-or-Wait made and compared the value of the eventual outcome (e.g., based on whether and where a person eventually searched) to the value if we just sent them the task then and there. To measure the real-world performance, we made this comparison by using the actual

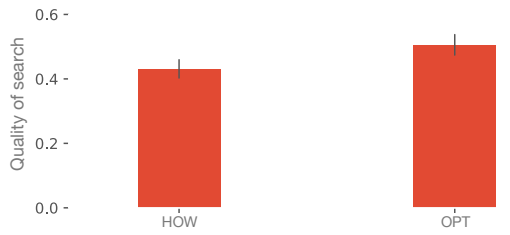


Figure 7. Overall quality of search between Hit-or-Wait and myopic optimal solution. Hit-or-Wait was able to make near optimal decisions in actual use, and make efficient use of people's search efforts.

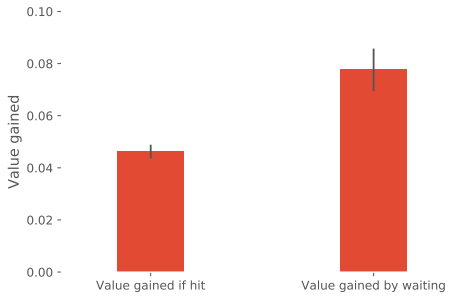


Figure 8. Comparison between the value gained if hit at the wait decision state vs. value gained from waiting and notifying at the later state.

number of searches performed in the subregion thus far to compute the likelihood that the item is still in that subregion (i.e., $P(item|n)$), instead of using the expected value of waiting as computed by Hit-or-Wait. As the system might make multiple wait decisions until it makes a hit decision, we considered only the first wait decisions for comparison.

Study Results

Searching with Low Disruption

Over the course of 10 days, the participants received 248 notifications and conducted 60 searches along their routes (24.19% acceptance rate). We found that the 30-second on-the-go searches were not disruptive to the users when they decided to help. Figure 6 shows that both the EMA (N=60 from 23 out of 25 participants) and post-study survey responses (N=24) about the perceived cost of disruption were low; the average rating from EMA is 1.39 (SD: 0.58) and the average rating from post-study survey is 1.58 (SD: 0.72), values that fall between "not disruptive at all" and "slightly disruptive."[4] Our interview findings also mirror the survey responses, as P2 said: *"So everyday I walk passed [street name] and [street name], and usually I am on my phone when I am walking and I see the alert. I usually just keep walking and look around my path and look for the item...It's not bad at all and really easy."*

Among the searches, 4 different participants found 4 items out of the 9 search requests that were made. While finding items was not a primary focus of our study, this finding demonstrates how effective coordination can make use of smaller contributions from many users to find lost items in large regions.

Maximizing User Contributions

The results show that Hit-or-Wait was effective in maximizing the user contributions by notifying the tasks where they were most needed. For 57 searches that took place in the study, the average quality of search was 0.43 for Hit-or-Wait (SD: 0.21) and 0.51 for myopic optimal (SD: 0.24), indicating that Hit-or-Wait captured 84.31% of the value of what myopic optimal is able to achieve (Figure 7).[5] Closer analysis shows that Hit-or-Wait made hit decisions in subregions with a higher search count than myopic optimal only 9.68% of the time (24 out of 248), and in 77.42% of the times (192 out of 248) it made decisions identical to the myopic optimal. These results suggest that even without explicit coordination or full knowledge of people's future routes, Hit-or-Wait is able to make near optimal decisions in actual use, and make efficient use of people's search efforts.

We found that Hit-or-Wait also made effective wait decisions. Figure 8 illustrates the value of waiting and shows that deciding to wait led to future decisions with a 67.6% increase in value compared to immediately notifying users. A paired t-test shows that there is a significant difference in the value of waiting versus not waiting ($t = 3.98$, $df = 120$, $p < 0.0001$).

When and Why Missed Opportunities Happen

Waiting for a better opportunity poses a risk of completely missing the opportunity to notify. Our results show that the variance for value gained from a wait decision is quite large (M: 0.0776, SD: 0.0893), and it is mainly due to the fact that there is zero value gained when missed opportunities occur. Our results show that 45.16% (56 out of 124) of the wait decisions resulted in missed opportunities.

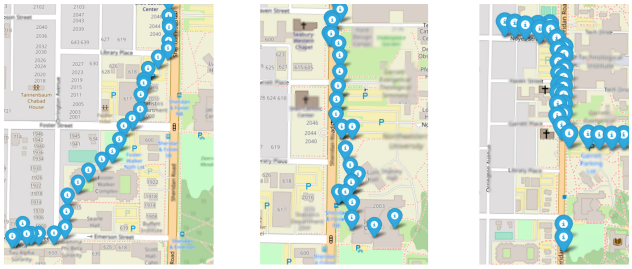
From 52 instances, outside of situations where uncertainty in future routes naturally led to users going outside of the task region, we identified three other reasons that resulted in missed opportunities (Figure 9). First, contrary to our model, people sometimes took unexpected routes or did not move to adjacent states (Figure 9a). For example, some people took shortcuts or trespassed in ways that were unexpected by our model and this led to missed opportunities where the system may have notified them in other subregions. In the future we could have models with more fine-grained state spaces, where the state is more granular than a road segment.

Second, people sometimes stopped moving and stayed at the location where the system made a wait decision (Figure 9b). These instances occurred when the participants were on their way to classes or home, and they only passed one of the subregions since their trip was cut short. Our system did not have the notion of terminal state, but in the future it could predict whether or not the current state will be the terminal state so that we can prevent such missed opportunities.

Third, inaccuracy or inconsistency in location tracking also caused some missed opportunities (Figure 9c). There are many reasons such technical failures can happen (e.g., turning

[4]One of the participants did not fill out the post-study survey and never responded to the emails.

[5]We excluded 3 searches from this analysis since they were missing the GPS location data needed to compute the measure.



(a) (b) (c)

Figure 9. Examples of missed opportunities after wait decisions: (a) taking shortcuts and did not go to adjacent states; (b) staying at the wait state (e.g., taking classes); (c) missing GPS location data.

off Wi-Fi and thus lowering the location tracking accuracy; switching between LTE and Wi-Fi while walking around the campus; turning on low-power or airplane mode). For the rest of 4 instances it did not notify due to technical failure.

FOLLOW-UP INTERVIEWS

In the follow-up interviews after the field deployment, we sought to understand how helpers perceive the value of their contributions toward the larger goal of finding the item in a large search region. We also explored ways to represent and visualize the value of contributions and use it as a tool to better communicate seemingly opaque Hit-or-Wait decisions.

Interview Setup

We invited participants for an optional 30-min interview after the field deployment and interviewed 7 participants who helped at least once during the study. Each interview lasted around 30 minutes. We chose 4 different scenarios to highlight a high-level idea of how *Hit-or-Wait* works: 1) A Hit decision is made because a user is at a road with no searches; 2) a user is at a road with some searches but the user is likely to go to another road with no searches, so it makes a Wait decision at the current road and makes a Hit decision if the user reaches the subregion with no searches; 3) a user is at a road with a few searches and the user is likely to go to a road with a fewer searches, so it makes a Wait decision at the current road, and a Hit decision if the user reaches the road with the fewer searches; 4) a user is at a road with a few searches and the user is likely to go to a road with more searches, so it makes a Hit decision at the current road.

During the interviews, we first asked users to recall their searches and tell us about the perceived value of them. We chose different contribution scenarios from a user's actual contributions (when the user's searches did not cover all four scenarios, we showed other users' searches instead) and showed the visualizations for those searches (some examples of the visualizations are shown in Figure 10). We then asked the participants to walk us through how they thought the system worked based on the visualizations. After showing the participants all of the visualizations, we again asked them about the perceived value of their searches, and we elicited their suggestions about how the system could more clearly communicate its goals and highlight the value of their contributions. The interview participants received a $5 gift card as compensation.

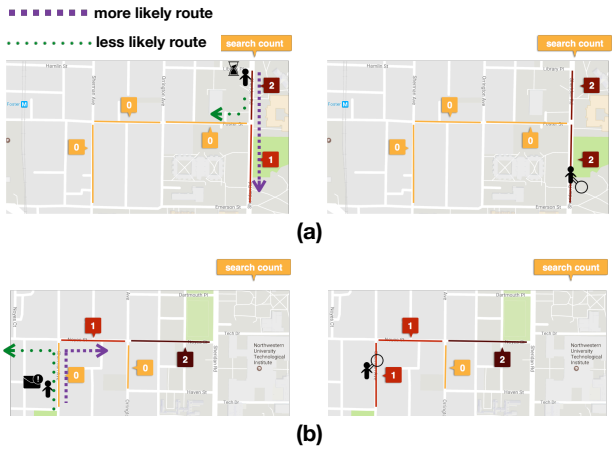


Figure 10. Wait (top) and Hit (bottom) example visualizations.

Interview Findings

Some participants perceived the value of their contributions solely on the basis of whether they found the item, and as a result they did not regard their contributions as valuable if they were unable to find the item. For example, P6 described how she thought that her contribution was not valuable: *"Well, clearly wasn't that valuable, because I never found anything."*

Some participants also assumed that the system did not take into consideration other people's searches, and perceived their contributions either as redundant or too miniscule to be valuable. One participant (P6) explained how she thought that the system was notifying everyone who passed by and as a result many people would have searched in a same region: *"there was nothing to stop someone else from doing the exact same search that I did even if I already searched that area, right?"* On the contrary, another participant assumed that she was the only one searching in a large search region and did not feel her search was ever going to be useful (P2).

However, when the participants understood the high-level idea of the mechanism of Hit-or-Wait—predicting likely routes and considering other people's searches—either through the visualizations or verbal description from the interviewer, they stated that their contributions were more valuable. P7 said: *"Oh, definitely valuable because it carefully calculates who has already [searched], so I don't feel like I am just another person who's like useless."* Another participant P2 said: *"I guess it is a lot more valuable. Because I guess I've never thought the computer was taking in how other people are doing it."*

The participants also mentioned that highlighting an individual's contribution as part of the global goal may help them value their contributions more. As P7 stated: *"Maybe also having information like if someone does find the item, then I would know I was just being helpful...So I was helping part of that even if I wasn't the exact person to find it."* The participant also said that emphasizing the uniqueness of her contributions could have helped her feel the contributions to be more valuable: *"It's nice to know that I am the first person to search like there...If I saw this while I was searching, that would've made sense and I may have felt like it's valuable."*

To summarize, our interview results show that it's important to communicate the global goal of the system and highlight the parts of the goal that the users are contributing to so as to help them to be cognizant of the value of their contributions.

DISCUSSION AND FUTURE WORK

In this paper, we introduced Hit-or-Wait, a general decision-theoretic mechanism for coordinating opportunistic contributions to achieve effective global outcomes with on-the-go crowdsourcing. We demonstrated the effectiveness of Hit-or-Wait through simulations and a field deployment, which highlighted Hit-or-Wait's ability to minimize user disruptions and maximize the value of user contributions via implicit coordination by deciding on-the-fly whether to notify or wait for better opportunities. In the rest of the section, we discuss the applicability of Hit-or-Wait to more complex scenarios and other domains; tractability and scalability; limitations of myopic Hit-or-Wait; and lastly, the general need for system-level coordination in on-the-go crowdsourcing systems.

Complex Scenarios and Applicability to Other Domains

As the goal of this paper is to explore ways to implicitly coordinate user contributions towards global outcomes, we excluded subsequent scenarios where helpers have to hand off found items or factors that could affect the willingness and convenience besides user's current location in the field deployment. To make applications like Trouve a full-fledged system with Hit-or-Wait, we could take into consideration the cost of diversion [20, 33] from a user's existing route or predicted destination [29, 42] to a hand-off location when computing the value of notifying the user. We could also include parameters that capture busyness, schedules, existence of companions, and other situational factors that are known to affect task acceptance rate [21, 27].

While we studied Hit-or-Wait in the context of a community-based lost-and-found, the general mechanism can be applied to other domains such as community sensing or for other community-based peer-to-peer services. In community sensing, Hit-or-Wait can be used to support the global goal of ensuring data coverage and fidelity, even when using low-effort contributions [37, 36]. Depending on where and how much data has been collected at different locations, Hit-or-Wait can decide when to ask for additional pieces of information, for example by making the decision to wait should the user be likely to reach other locations where data coverage is low. In community-based peer-to-peer services such as timebanking, Hit-or-Wait can be used to achieve the community goal of effectively providing help for each other by accounting for different skills, abilities, and preferences [14, 23]. Hit-or-Wait can encode the value of contributions based on required skills, priority, as well as helper preferences. For example, depending on a task's urgency, Hit-or-Wait can effectively coordinate opportunistic contributions to prioritize high-valued, urgent tasks that a user may encounter on their route.

Tractability and Scalability

Our MDP model for Hit-or-Wait scales well for reasonable state spaces; there are no immediate tractability concerns at the community- or neighborhood-scale that on-the-go crowdsourcing systems are intended to be deployed in. In cases where the state space becomes large, either in larger scale systems or by including other contextual information, we can scale further by employing standard techniques such as using coarser or factored state representations [16].

Limitations of Myopic Hit-or-Wait

One of the limitations of the current Hit-or-Wait implementation is that it makes decisions myopically without regard to the possible future routes and decisions of other helpers who may arrive. By taking into consideration others' future routes and decisions, *futuristic Hit-or-Wait* can potentially coordinate contributions more effectively, especially in cases where the contributions are contingent on differentiating factors among helpers. For instance, if only certain people have access to locations (e.g., returning a book to university library), then by predicting who will come across which locations we can more effectively coordinate these scarce resources where they are most needed. Realizing the benefits of futuristic Hit-or-Wait will require overcoming the computational challenges imposed by (a) reasoning about the potential routes and decisions of future helpers; and (b) considering the interdependencies of how current decisions can affect future decisions. Resolving these challenges to provide globally optimal solutions through opportunistic coordination will require applying and advancing existing decision-theoretic methods.

Towards System-level Coordination

Hit-or-Wait's ability to minimize disruptions and eliminate coordination costs [30] increases the ability for people to conveniently and effectively contribute to local, communal problems. In this way, Hit-or-Wait can potentially help encourage and sustain more contributions over time. We hope that such new ways of contributing to local, communal problems can provide social benefits, and create new ways of interacting with, supporting, and becoming a part of a community.

Future work on developing system-level mechanisms for coordinating on-the-go contributions may look beyond making effective use of individual contributions to considering how to engage the community of helpers as a whole. For example, a supply management framework may be able to balance the demands from requesters with disruption to potential helpers, considering both system goals such as the quality of service but also community values such as not overburdening helpers, so as to maintain a healthy pool of future helpers [27]. As some of our participants from the interviews indicated that they may be willing to deviate from their routes, there may also be opportunities to coordinate mixed models of contributing that engage both on-the-go helpers and more dedicated helpers to collectively respond to local, communal needs.

ACKNOWLEDGEMENT

We thank members of the Design, Technology, and Research program and the Delta Lab for their valuable feedback and helpful discussions. We thank Kotaro Hara for helping with collecting road segment data from OpenStreetMap dataset. This work was funded by the National Science Foundation under Grant No. 1618096.

REFERENCES

1. TaskRabbit. http://www.taskrabbit.com.
2. Uber. http://www.uber.com.
3. Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz. 2010. Location-based crowdsourcing: extending crowdsourcing to the real world. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. ACM, 13–22.
4. Gerard Beenen, Kimberly Ling, Xiaoqing Wang, Klarissa Chang, Dan Frankowski, Paul Resnick, and Robert E. Kraut. 2004. Using Social Psychology to Motivate Contributions to Online Communities. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04)*. ACM, New York, NY, USA, 212–221.
5. Victoria ME Bellotti, Sara Cambridge, Karen Hoy, Patrick C Shih, Lisa Renery Handalian, Kyungsik Han, and John M Carroll. 2014. Towards community-centered support for peer-to-peer service exchange: rethinking the timebanking metaphor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2975–2984.
6. Michael S Bernstein, Joel Brandt, Robert C Miller, and David R Karger. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 33–42.
7. Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: A Word Processor with a Crowd Inside. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 313–322.
8. Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and others. 2010. VizWiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 333–342.
9. Cen Chen, Shih-Fen Cheng, Aldy Gunawan, Archan Misra, Koustuv Dasgupta, and Deepthi Chander. 2014. TRACCS: A Framework for Trajectory-Aware Coordinated Urban Crowd-Sourcing. In *HCOMP*.
10. Yueyue Chen, Pin Lv, Deke Guo, Tongqing Zhou, and Ming Xu. 2017. Trajectory segment selection with limited budget in mobile crowd sensing. *Pervasive and Mobile Computing* 40 (2017), 123–138.
11. Shih-Fen CHENG, CHEN CEN, Thivya KANDAPPU, Hoong Chuin LAU, Archan MISRA, Nikita JAIMAN, Randy Tandriansyah DARATAN, Ming Hui KOH, and others. 2017. Scalable urban mobile crowdsourcing: Handling uncertainty in worker movement. *ACM Transactions on Intelligent Systems and Technology* 9, 3 (2017), 1.
12. Lydia B Chilton, Juho Kim, Paul André, Felicia Cordeiro, James A Landay, Daniel S Weld, Steven P Dow, Robert C Miller, and Haoqi Zhang. 2014. Frenzy: collaborative data organization for creating conference sessions. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 1255–1264.
13. Peng Dai, Mausam, and Daniel S. Weld. 2010. Decision-theoretic Control of Crowd-sourced Workflows. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI'10)*. AAAI Press, 1168–1174.
14. Afsaneh Doryab, Victoria Bellotti, Alaaeddine Yousfi, Shuobi Wu, John M. Carroll, and Anind K. Dey. 2017. If It's Convenient: Leveraging Context in Peer-to-Peer Variable Service Transaction Recommendations. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 48 (Sept. 2017), 28 pages.
15. Nathan Eagle. 2009. txteagle: Mobile crowdsourcing. *Internationalization, design and global development* (2009), 447–456.
16. Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. 2003. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research* 19 (2003), 399–468.
17. Aakar Gupta, William Thies, Edward Cutrell, and Ravin Balakrishnan. 2012. mClerk: enabling mobile crowdsourcing in developing regions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1843–1852.
18. Kyungsik Han, Patrick C. Shih, Victoria Bellotti, and John M. Carroll. 2015. It's Time There Was an App for That Too:: A Usability Study of Mobile Timebanking. *International Journal of Mobile Human Computer Interaction* 7, 2 (2015), 1–22.
19. Eric Horvitz, Paul Koch, and Muru Subramani. 2007. Mobile opportunistic planning: methods and models. In *User Modeling*, Vol. 4511. Springer, 228–237.
20. Eric Horvitz and John Krumm. 2012. Some help on the way: Opportunistic routing under uncertainty. In *Proceedings of the 2012 ACM conference on Ubiquitous Computing*. ACM, 371–380.
21. Kazushi Ikeda and Keiichiro Hoashi. 2017. Crowdsourcing GO: Effect of Worker Situation on Mobile Crowdsourcing Performance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 1142–1153.
22. Shenggong Ji, Yu Zheng, and Tianrui Li. 2016. Urban sensing based on human mobility. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1040–1051.

23. Hyunggu Jung, Victoria Bellotti, Afsaneh Doryab, Dean Leitersdorf, Jiawei Chen, Benjamin V Hanrahan, Sooyeon Lee, Dan Turner, Anind K Dey, and John M Carroll. 2016. 'MASTerful' Matchmaking in Service Transactions: Inferred Abilities, Needs and Interests versus Activity Histories. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1644–1655.

24. Ece Kamar, Eric Horvitz, and Chris Meek. 2008. Mobile opportunistic commerce: mechanisms, architecture, and application. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 1087–1094.

25. Thivya Kandappu, Nikita Jaiman, Randy Tandriansyah, Archan Misra, Shih-Fen Cheng, Cen Chen, Hoong Chuin Lau, Deepthi Chander, and Koustuv Dasgupta. 2016a. Tasker: Behavioral insights via campus-based experimental mobile crowd-sourcing. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 392–402.

26. Thivya Kandappu, Archan Misra, Shih-Fen Cheng, Nikita Jaiman, Randy Tandriansyah, Cen Chen, Hoong Chuin Lau, Deepthi Chander, and Koustuv Dasgupta. 2016b. Campus-Scale Mobile Crowd-Tasking: Deployment & Behavioral Insights. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 800–812.

27. Yongsung Kim, Emily Harburg, Shana Azria, Aaron Shaw, Elizabeth Gerber, Darren Gergle, and Haoqi Zhang. 2016. Studying the Effects of Task Notification Policies on Participation and Outcomes in On-the-go Crowdsourcing. In *HCOMP*. AAAI.

28. John Krumm. 2008. A Markov Model for Driver Turn Prediction. In *SAE World Congress & Exhibition*. SAE International.

29. John Krumm and Eric Horvitz. 2006. Predestination: Inferring destinations from partial trajectories. *Ubicomp* (2006), 243–260.

30. Thomas W Malone and Kevin Crowston. 1994. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)* 26, 1 (1994), 87–119.

31. Jon Noronha, Eric Hysen, Haoqi Zhang, and Krzysztof Z Gajos. 2011. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 1–12.

32. Al M. Rashid, Kimberly Ling, Regina D. Tassone, Paul Resnick, Robert Kraut, and John Riedl. 2006. Motivating Participation by Displaying the Value of Contribution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 955–958.

33. Adam Sadilek, John Krumm, and Eric Horvitz. 2013. Crowdphysics: Planned and Opportunistic Crowdsourcing for Physical Tasks. In *Seventh International AAAI Conference on Weblogs and Social Media*.

34. Rannie Teodoro, Pinar Ozturk, Mor Naaman, Winter Mason, and Janne Lindqvist. 2014. The motivations and experiences of the on-demand mobile workforce. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 236–247.

35. Jacob Thebault-Spieker, Loren G. Terveen, and Brent Hecht. 2015. Avoiding the South Side and the Suburbs: The Geography of Mobile Crowdsourcing Markets. In *CSCW*. ACM, New York, NY, USA, 265–275.

36. Khai N Truong, Thariq Shihipar, and Daniel J Wigdor. 2014. Slide to X: unlocking the potential of smartphone unlocking. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 3635–3644.

37. Rajan Vaish, Keith Wyngarden, Jingshu Chen, Brandon Cheung, and Michael S Bernstein. 2014. Twitch crowdsourcing: crowd contributions in short bursts of time. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 3645–3654.

38. Luis Von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 319–326.

39. Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. 2008. recaptcha: Human-based character recognition via web security measures. *Science* 321, 5895 (2008), 1465–1468.

40. Jiangtao Wang, Yasha Wang, Daqing Zhang, Feng Wang, Yuanduo He, and Liantao Ma. 2017. PSAllocator: multi-task allocation for participatory sensing with sensing capability constraints. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, 1139–1151.

41. Haoqi Zhang, Edith Law, Rob Miller, Krzysztof Gajos, David Parkes, and Eric Horvitz. 2012. Human computation tasks with global constraints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 217–226.

42. Brian D Ziebart, Andrew L Maas, Anind K Dey, and J Andrew Bagnell. 2008. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 322–331.