GeoCrowd: Enabling Query Answering with Spatial Crowdsourcing

Leyla Kazemi
IMSC
University of Southern California
Los Angeles, CA 90089-0781
lkazemi@usc.edu

Cyrus Shahabi
IMSC
University of Southern California
Los Angeles, CA 90089-0781
shahabi@usc.edu

ABSTRACT

With the ubiquity of mobile devices, *spatial crowdsourcing* is emerging as a new platform, enabling *spatial tasks* (i.e., tasks related to a location) assigned to and performed by human workers. In this paper, for the first time we introduce a taxonomy for spatial crowdsourcing. Subsequently, we focus on one class of this taxonomy, in which workers send their locations to a centralized server and thereafter the server assigns to every worker his nearby tasks with the objective of maximizing the overall number of assigned tasks. We formally define this *maximum task assignment (or MTA)* problem in spatial crowdsourcing, and identify its challenges. We propose alternative solutions to address these challenges by exploiting the spatial properties of the problem space. Finally, our experimental evaluations on both real-world and synthetic data verify the applicability of our proposed approaches and compare them by measuring both the number of assigned tasks and the travel cost of the workers.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms

Keywords

Spatial Crowdsourcing, Crowdsourced Query, Spatial Task Assignment

1. INTRODUCTION

Due to the ubiquity of sensors, every person with a mobile phone can now act as a multi-modal sensor collecting various types of data instantaneously (e.g., picture, video, audio, location, time, speed, direction, acceleration). Many studies suggest significant future growth in the number of mobile smart phone users, the phone's hardware and software features, and the broadband bandwidth. Therefore, it is critical to fully utilize this new platform for various tasks, among which the most promising is *spatial crowdsourcing.*

In this paper, we introduce spatial crowdsourcing as the process of crowdsourcing a set of *spatial tasks* (i.e., tasks related to a location) to a set of *workers*, which requires the workers to perform the spatial tasks by physically traveling to those locations. Consider a scenario, in which a *requester* is interested in collecting pictures and videos of the anti-government demonstrations from various locations of a city. With spatial crowdsourcing, the requester, instead of traveling to the locations of each of the events, issues his query to a spatial crowdsourcing server (or SC-server). Consequently, the SC-server crowdsources the query among the available workers in the vicinity of the events. Once the workers document their nearby events, the results are sent back to the requester.

While crowdsourcing has recently attracted both research communities (e.g., database [19], image processing [14, 32], NLP [31]) and industry (e.g., Amazon's Mechanical Turk [1] and CrowdFlower [3]), only a few work [12, 10, 23] have studied spatial crowdsourcing. Moreover, most existing work on spatial crowdsourcing focus on a particular class of spatial crowdsourcing called *participatory sensing.* With participatory sensing, the goal is to exploit the mobile users, for a given campaign, by leveraging their sensor-equipped mobile devices to collect and share data. Some real-world examples of participatory sensing projects include [2, 10, 21, 26]. For example, the Mobile Millennium project [10] by UC Berkeley is a state-of-the-art system that uses GPS-enabled mobile phones to collect en route traffic information and upload it to a server in real time. The server processes the contributed traffic data, estimates future traffic flows and sends traffic suggestions and predictions back to the mobile users. Similar projects were implemented earlier by CalTel [21] and Nericell [26] which used mobile sensors/smart phones mounted on vehicles to collect information about traffic, WiFi access points on the route and road condition. In CycleSense [2], bikers report their biking routes to a server during their daily commute in the Los Angeles area, along with information about air quality, hazards, traffic conditions, accidents, etc.

All these previous studies on participatory sensing focus on

ACM SIGSPATIAL GIS '12, November 6-9, 2012. Redondo Beach, CA, USA

a single campaign and try to address challenges specific to that campaign. More examples of single campaigns include [18], which is a campaign for watching petro prices, and [29] which is a campaign for monitoring the urban air pollution. However, our focus is on devising a generic crowdsourcing framework, similar to Amazon Turk, but spatial, where multiple campaigns can be handled simultaneously. Moreover, most existing studies on participatory sensing focus on small campaigns with a limited number of workers, and are not scalable to large spatial crowdsourcing applications. Finally, spatial crowdsourcing subsumes participatory sensing by introducing a general framework, which allows any form of spatial tasks to be assigned and performed by humans.

In this paper, for the first time we introduce a taxonomy for spatial crowdsourcing. First, we classify spatial crowdsourcing based on people's motivation. Thereafter, we define two modes for spatial task publishing. Finally, we define two ways for spatial task assignment. We focus on one class of spatial crowdsourcing, in which a set of workers send their *task inquiries* to a SC-server. The task inquiry of a worker, which includes his location along with a set of constraints (e.g., a region), is a request that the worker issues to inform the SC-server of his availability to work. Consequently, the SC-server, who receives the location of the workers, assigns to every worker his nearby tasks. In this class of spatial crowdsourcing, the main optimization goal is to maximize the overall task assignment while conforming to the constraints of the workers. We refer to this problem as the *maximum task assignment (MTA)* problem. The solution to the MTA problem could be straightforward if the SC-server had a global knowledge of both the spatial tasks and the workers. However, the SC-server is continuously receiving spatial tasks from requesters and also task inquiries from the workers. Therefore, the SC-server can only maximize the task assignment at every time instance (i.e., local optimization) with no knowledge of the future.

We propose three alternative solutions to the MTA problem. Our first approach, namely *Greedy (GR)*, follows the local optimization strategy by maximizing the task assignment at every time instance. The Greedy approach utilizes the constraints of the workers to assign to every worker his nearby tasks. Our second approach, called *Least Location Entropy Priority (LLEP)*, improves the Greedy approach by utilizing the entropy of the location. The location entropy heuristic is based on the intuition that spatial tasks are more likely to be performed in future if they are located in areas with higher population of workers (i.e., higher location entropy). Therefore, the LLEP approach improves the overall task assignment by assigning higher priority to spatial tasks located in places with lower location entropy, as they are less likely to be completed in future. With spatial crowdsourcing, since workers should physically travel to a location in order to perform a task, the travel cost of the workers is also an important factor. Therefore, our third approach, referred to as *Nearest Neighbor Priority (NNP)*, incorporates the travel cost of the workers into the task assignment by assigning higher priority to the tasks with lower travel cost. Our extensive experiments on both real and synthetic data show that in comparison with GR, our LLEP approach can improve the number of assigned tasks by up to 36%, while the NNP approach can improve the travel cost of the workers by up to 41%. Consequently, based on the objective of the application, either LLEP or NNP can be applied to solve the MTA problem.

The remainder of this paper is organized as follows. Section 2 introduces our taxonomy for spatial crowdsourcing. In Section 3, we discuss a set of preliminaries in the context of spatial crowdsourcing, and formally define the MTA problem. Thereafter, in Section 4 we explain our assignment solutions. Section 5 presents the experimental results. In Section 6, we review the related work. Finally, in Section 7 we conclude and discuss the future directions of this study.

2. A TAXONOMY OF SPATIAL CROWDSOURCING

Spatial crowdsourcing opens up a new mechanism for spatial tasks (i.e., tasks related to a location) to be performed by humans. In this section, we define a taxonomy[1] for spatial crowdsourcing (Figure 1). First, we classify spatial crowdsourcing based on people's motivation. Next, we define two modes of task publishing in spatial crowdsourcing. Finally, we define two ways for spatial task assignment in spatial crowdsourcing.

2.1 Spatial Crowdsourcing Classification

Spatial crowdsourcing is the process of crowdsourcing a set of spatial tasks to a set of workers, which requires the workers to be physically located at that location in order to perform the corresponding task. Spatial crowdsourcing can be classified based on the motivation of the workers into two classes: *reward-based* and *self-incentivised* (see Figure 1).

Reward-based Spatial Crowdsourcing

With reward-based spatial crowdsourcing, every spatial task has a price and workers will receive a certain reward for every spatial task they perform correctly. An example of reward-based spatial crowdsourcing is [37], where every worker can receive a small reward for completing and sharing a sensing task.

Self-incentivised Spatial Crowdsourcing

This class of spatial-crowdsourcing is for people who are self-incentivised to perform tasks voluntarily. Here, people usually have other incentives rather than receiving a reward such as documenting an event or promoting their cultural, political, or religious views. An example of this class includes a participatory sensing campaign [2, 10], in which a group of people are willing to voluntarily report traffic events (e.g., accidents) by leveraging their sensor-equipped mobile devices. Our focus in this paper is on this class of spatial crowdsourcing.

2.2 Spatial Task Publishing Modes

With spatial crowdsourcing, tasks can be published in two different modes: *Worker Selected Tasks (WST)* and *Server Assigned Tasks (SAT)*.

[1] Note that even though the taxonomy can be generalized to any type of crowdsourcing (i.e., spatial or non-spatial), in this paper we focus on the taxonomy in the context of spatial crowdsourcing.

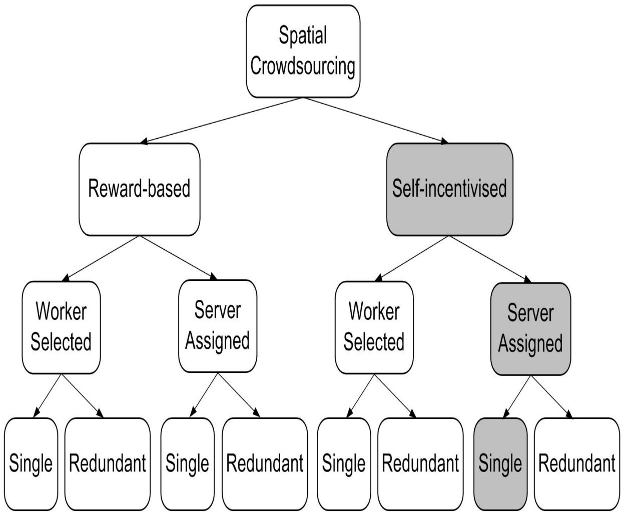


Figure 1: A taxonomy of spatial crowdsourcing. The focus of this paper is shown in grey.

Worker Selected Tasks (WST) Mode

With this mode, the SC-server publicly publishes the spatial tasks and online workers can choose any spatial task in their vicinity without the need to coordinate with the SC-server. One advantage of this mode is that since the workers can choose any arbitrary task in their vicinity autonomously, they do not need to reveal their locations to the SC-server for every assignment. However, one drawback of this mode is that the SC-server does not have any control over the allocation of spatial tasks. This may result in some spatial tasks never be assigned, while others get assigned redundantly. Another drawback of WST is that workers choose tasks based on their own objectives (e.g., choosing the k closest spatial tasks to minimize their travel cost), which is not necessarily the ultimate objective of the SC-server (i.e., maximizing the overall task assignment). An example of the WST mode is [12], where users browse for available spatial tasks, and pick the ones in their neighborhood.

Server Assigned Tasks (SAT) Mode

In this mode, the SC-server does not publish the spatial tasks to the workers. Instead, any online worker sends his location to the SC-server. The SC-server after receiving the locations of all online workers, assigns to every worker his closeby tasks. The advantage of SAT is that unlike WST, the SC-server has the big picture, and therefore, can assign to every worker his closeby tasks while maximizing the overall task assignment (i.e., global optimization). However, the drawback is that workers should report their locations to the SC-server for every assignment, which can pose a privacy threat. An example of SAT mode is [23], which proposes a framework for small campaigns, where workers are assigned to their closeby sensing tasks. Our focus in this paper is on this mode of spatial crowdsourcing.

2.3 Spatial Task Assignment Modes

So far, we discussed different types of spatial crowdsourcing, and how spatial tasks can be published. However, we did not discuss how to verify the validity of the tasks completed by workers. A malicious worker might intentionally complete a task incorrectly (e.g., being dishonest about physically going to the location of the spatial task). In this section, we define two modes for task assignment in terms of how to verify the validity of the spatial tasks: *single-based task assignment* and *redundant-based task assignment.*

Single Task Assignment

The general assumption here is that workers are trusted, and therefore, they complete the spatial tasks correctly without any malicious intentions. Consequently, every spatial task is only assigned to one worker (preferably to the closest worker). Examples of this class are [15, 23]. In this paper our assumption is that all workers are trusted, and thus, we focus on single-based task assignment.

Redundant Task Assignment

Here, the intuitive assumption, based on the idea of the wisdom of crowds [33], is that the majority of the workers can be trusted. Thus, the data with the majority vote is verified as correct. This indicates that instead of each spatial task be completed by a particular worker, it should be completed by k closeby workers, where k is defined by the requester who issued the task. Consequently, the higher the value of k, the more chance that the completed task is correct. An example of non-spatial redundant-based task assignment is Amazon's Mechanical Turk [1].

3. PRELIMINARIES

In this section, we define a set of preliminaries in the context of self-incentivised single-based spatial crowdsourcing with the SAT mode. First, we formally define a *spatial task*.

Definition 1 (Spatial Task). *A spatial task t of form $<l, q, s, \delta>$ is a query q to be answered at location l, where l is a point in the 2D space. The query is asked at time s and will be expired at time $s + \delta$.*

Note that the query q of a spatial task t can be answered by a human only if the human is physically located at location l. For example, consider a scenario, in which the spatial task is to take a picture from a particular building. This means that the worker needs to physically go to the exact location of the building in order to take the picture. For simplicity, we assume that all tasks take the same amount of time to finish. With this definition, we now define the spatial crowdsourced query.

Definition 2 (Spatial Crowdsourced Query). *A spatial crowdsourced query (or SC-Query) of form ($<t_1, t_2, ...>$ $, k$) is a set of spatial tasks and a parameter k issued by a requester, where every spatial task t_i is to be crowdsourced k number of times.*

After receiving the SC-queries from all the requesters, the *spatial crowdsourcing server (or SC-server)* assigns the spatial tasks of these SC-queries to the available workers. Note that with the single task assignment mode, the SC-server should assign every spatial task to only one worker (i.e.,

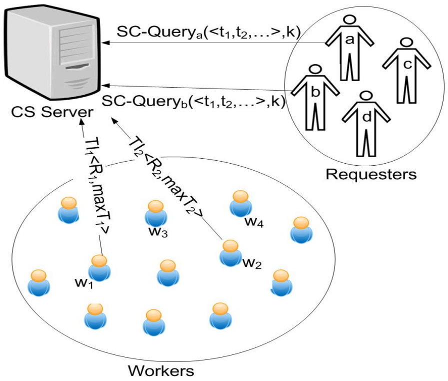


Figure 2: The spatial crowdsourcing framework

$k = 1$). Figure 2 shows our spatial crowdsourcing framework. In the following we formally define a worker.

Definition 3 (Worker). *A worker, denoted by w, is a carrier of a mobile device who volunteers to perform spatial tasks. A worker can be in an either online or offline mode. A worker is online when he is ready to accept tasks.*

Note that with self-incentivised spatial crowdsourcing, workers volunteer to perform spatial tasks without expecting any reward. Once a worker goes online, he sends a task inquiry to the SC-server (Figure 2). We now formally define the task inquiry.

Definition 4 (Task Inquiry or TI). *Task inquiry is a request that an online worker w sends to the SC-server, when ready to work. The inquiry includes location of w, l, along with two constraints: A spatial region R, and the maximum number of acceptable tasks $maxT$. The spatial region R represented by a rectangle is the area in which the worker can accept spatial tasks. In other words, any task outside the region will be rejected by the worker. Moreover, $maxT$ is the maximum number of tasks that the worker is willing to perform.*

Note that the task inquiry is defined for the SAT mode, where workers should send their locations to the SC-server for proper task assignment. The workers can also specify other constraints in their task inquiry (e.g., category of the task, amount of time they have). However, in this work we only consider two constraints for every worker (i.e., R and $maxT$).

Once the workers send their task inquiries, the SC-server assigns to every worker a set of tasks, while satisfying each worker's constraints. However, the task assignment is not a one-time process. The SC-server continuously receives SC-queries from requesters and task inquiries from workers. Therefore, we define the notion of task assignment instance set, which is the set of assigned tasks for a given instance of time.

Definition 5 (Task Assignment Instance Set). *Let W_i=$\{w_1, w_2, ...\}$ be the set of online workers at time s_i. Also, let T_i=$\{t_1, t_2, ...\}$ be the set of available tasks at time s_i. The task assignment instance set, denoted by I_i is the set of tuples of form <w,t>, where a spatial task t is assigned to a worker w, while satisfying the workers' constraints. Also, $|I_i|$ denotes the number of tasks, which are assigned at time instance s_i.*

Consequently, the task assignment instance set must conform to the constraints of the workers. This means that for every tuple $<w,t> \in I_i$, the spatial task t must be located inside the spatial region R of worker w. Moreover, every worker w can be assigned to at most $maxT$ number of tasks (i.e., the number of tuples in I_i including w is at most $maxT$).

Based on the above definition, We now define the *maximum task assignment* problem.

Definition 6 (Maximum Task Assignment (MTA)). *Given a time interval $\phi = \{s_1, s_2, ..., s_n\}$, let $|I_i|$ be the number of assigned tasks at time instance s_i. The maximum task assignment problem is the process of assigning tasks to the workers during the time interval ϕ, while the total number of assigned tasks (i.e., $\Sigma_{i=1}^{n}|I_i|$) is maximized.*

Note that in the ideal case, all tasks will be assigned to all workers. However, this might not be practical due to the constraints of the workers. Therefore, our optimization goal is to maximize the number of assigned tasks.

4. ASSIGNMENT PROTOCOL

In order to solve the MTA problem, the SC-server should have a global knowledge of all the spatial tasks and the workers ([34, 36]). This would allow the SC-server to optimally assign every task to every worker, so that the total number of assigned tasks is maximized. However, the SC-server does not have such knowledge. At every instance of time, the SC-server receives a set of new tasks from the requesters, and also a set of new task inquiries from the workers. Therefore, the SC-server only has a local view of the available tasks and workers at any instance of time. This means that a global optimal assignment is not feasible. Instead, the SC-server tries to optimize the task assignment locally at every instance of time. The SC-server does this by utilizing the spatial information that workers share during their task inquiries. In the following, we propose three solutions to this problem. All the solutions follow the local optimal assignment strategy. Our first approach tries to solve MTA in a greedy way by maximizing the task assignment at every instance of time. Our second approach tries to improve the optimization by applying a heuristic, which utilizes the location entropy of an area, to maximize the overall assignment. Finally, our third approach tries to maximize the task assignment while taking into account the travel cost of the workers.

4.1 Greedy (GR) Strategy

As discussed earlier, with this approach the idea is to do the maximum assignment at every instance of time. The reason this approach is called Greedy is that at every instance of time, it only tries to maximize the current assignment (i.e., local optimization instead of global optimization). Note that this does not necessarily result in a globally optimal answer. Given a set of online workers W_i=$\{w_1, w_2, ...\}$, and a set of available tasks T_i=$\{t_1, t_2, ...\}$ at time instance s_i, the goal is to assign maximum number of tasks in T_i to workers in W_i for every instance s_i, which is equivalent to maximizing $|I_i|$. We refer to this as maximum task assignment instance problem. Thus, our goal in this approach is to maximize the overall assignment by solving the maximum task assignment instance problem for every instance of time.

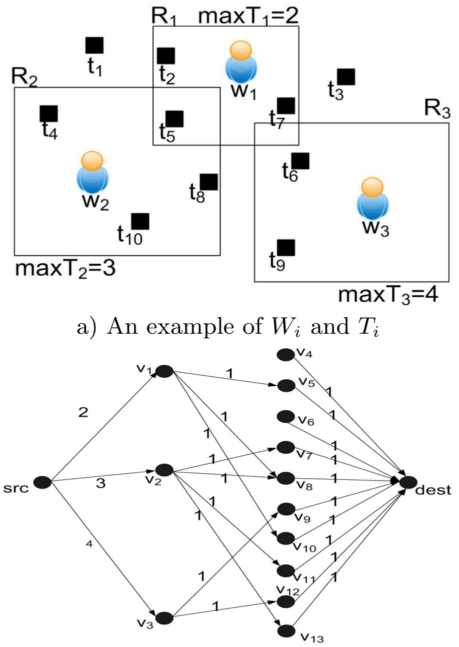
In order to solve the maximum task assignment instance problem, the idea is to utilize the constraints of the workers to guarantee that tasks are properly assigned. Note that without the constraints, a worker might be assigned to a spatial task in a far distance from his location. However, with spatial crowdsourcing, since workers need to physically go to a location to perform a spatial task, the goal is to assign only a number of tasks within a given distance to the workers. During the task inquiry, every online worker forms two constraints: the spatial region R, and the maximum number of tasks $maxT$. This means that every worker is willing to perform at most $maxT$ tasks, which should not be outside his spatial region R. With the following theorem, we can solve the maximum task assignment instance problem by reducing it to the maximum flow problem.

THEOREM 1. *The maximum task assignment instance problem is reducible to the maximum flow problem.*

PROOF. *We prove this for time instance s_i with W_i=$\{w_1, w_2, ...\}$ as the set of online workers, and T_i=$\{t_1, t_2, ...\}$ as the set of available spatial tasks. Let G_i=(V, E) be the flow network graph with V as the set of vertices, and E as the set of edges at time instance s_i. The set V contains $|W_i|$+$|T_i|$+2 vertices. Each worker w_j maps to a vertex v_j. Each spatial task t_j maps to a vertex $v_{|W_i|+j}$. We create a new source vertex src labeled as v_0, and a new destination vertex dst labeled as $v_{|W_i|+|T_i|+1}$.*

The set E contains $|W_i|$+$|T_i|$+m edges. There are $|W_i|$ edges connecting the new src vertex to the vertices mapped from W_i. For a given edge connecting the src vertex to vertex v_j (mapped from w_j) denoted by (src, v_j), we set the capacity to $maxT_j$ (i.e., $c(src, v_j)$=$maxT_j$), since every worker is only capable of performing maxT number of tasks. There are also $|T_i|$ edges connecting the vertices mapped from T_i to the new dst vertex. We set the capacity of each of these edges to 1, since every task is to be assigned to one worker (i.e., single task assignment). Every worker w_j has a spatial region constraint R_j, and can only perform tasks inside its spatial region. Thus, for every worker w_j we add an edge from v_j to all the vertices mapped from T_i, which are inside the spatial region R_j. For each of these m edges, we also set the capacity to one. □

Figure 3 better clarifies this reduction. Figure 3a shows an example of a set of workers W_i and a set of available tasks T_i at time instance s_i. Every worker w_j is associated with a spatial region R_j. The corresponding flow network graph G_i is depicted in Figure 3b. As shown in the figure, worker w_1 can only accept tasks inside his spatial region (i.e., t_2, t_5, and t_7). Therefore, the vertex mapped from w_1 can transfer flow to only the three vertices mapped from those tasks (i.e., v_5, v_8, and v_{10}). Moreover, w_1 is only willing to accept two tasks since $maxT_1 = 2$. Therefore, the capacity of the edge (src, v_1) is 2. Finally, the capacity of all the edges connecting the vertices mapped from spatial tasks (i.e., v_4..v_{13}) to the destination vertex dst are 1, since every spatial task is to be assigned to one worker.



a) An example of W_i and T_i

b) Flow network graph $G_i = (V, E)$

Figure 3: An example of the reduction of the maximum task assignment instance problem to the maximum flow problem at instance s_i

By reducing to the maximum flow problem, we can now use any algorithm that computes the maximum flow in the network to solve the maximum task assignment instance problem. One of the well-known techniques in computing the maximum flow is the Ford-Fulkerson algorithm [24]. The idea behind Ford-Fulkerson algorithm is that it starts sending flow from the source vertex to the destination vertex, as long as there is a path between the two with available capacity. Consequently, in order to solve the MTA problem we repeat this step for every instance of time.

The Greedy approach can be marginally improved by incorporating conventional non-spatial task scheduling approaches [8] such as FIFO, or FEFO (first expired, first out)[2]. With FEFO, the expiration time of every task can be utilized as a tiebreaker in the assignment process by prioritizing the tasks based on their expiration. However, in this paper our goal is to exploit the spatial properties of the problem space. Therefore, we introduce two spatial heuristics in our following two approaches.

[2] Our experiments showed that the impact of incorporating these conventional task scheduling approaches is marginal.

4.2 Least Location Entropy Priority (LLEP) Strategy

The problem with the Greedy strategy is that at every instance of time, it only tries to maximize the current assignment, without considering future optimizations. Even though we are clairvoyant on neither the future SC-queries from the requesters nor the future task inquiries from the workers, we can use some heuristics to maximize the overall assignments. One of the heuristics that can improve the task assignment process is to exploit the spatial characteristics of the environment during the assignment, one of which is the distribution of the workers in that area. Since every spatial task is linked to a location in the environment, a task is more likely to be completed when located in areas with higher worker densities. Therefore, the idea is to assign higher priority to tasks which are located in worker-sparse areas.

We use *entropy* of a location to measure the total number of workers in that location as well as the relative proportion of their future visits to that location. We refer to this as *location entropy*. Location entropy was first introduced in [16]. A location has a high entropy if many workers visit that location with equal proportions. Conversely, a location will have a low entropy if the distribution of the visits to that location is restricted to only a few workers. Thus, our heuristic is to give higher priority to tasks which are located in areas with smaller location entropy, because those tasks have lower chance of being completed by other workers.

We now formally define the location entropy. For a given location l, let O_l be the set of visits to location l. Thus, $|O_l|$ gives the total number of visits to l. Also, let W_l be the set of distinct workers that visited l. Moreover, let $O_{w,l}$ be the set of visits that worker w has made to the location l. The probability that a random drawn from O_l belongs to $O_{w,l}$ is $P_l(w) = \frac{|O_{w,l}|}{|O_l|}$, which is the fraction of total visits to l that belongs to worker w. The location entropy for l is computed as follows:

$$Entropy(l) = -\sum_{w \in W_l} P_l(w) \times \log P_l(w) \tag{1}$$

By computing the entropy of every location, we can associate to every task t_i of form $<l_i, q_i, s_i, \delta_i>$ a certain cost, which is the entropy of its location l_i. Accordingly, tasks with lower costs have higher priority, since they have a smaller chance of being completed. Thus, our goal in this approach is to assign the maximum number of tasks during every instance of time while the total cost associated to the assigned tasks is the lowest. We refer to this problem as the minimum-cost maximum task assignment instance problem. With the following theorem, we can solve the minimum-cost maximum task assignment instance problem by reducing it to the minimum-cost maximum flow problem [6]. A minimum cost maximum flow of a network G=(V, E) is a maximum flow with the smallest possible cost.

Theorem 2. *The minimum-cost maximum task assignment instance problem is reducible to the minimum-cost maximum flow problem.*

Proof. *We already proved in Theorem 1 that the maximum task assignment instance problem is reducible to the maximum flow problem. In the minimum-cost maximum task assignment instance problem, every task is associated with a cost. We prove this for time instance s_i with W_i=$\{w_1, w_2, ...\}$ as the set of online workers, and T_i=$\{t_1, t_2, ...\}$ as the set of available tasks. Let G_i=(V, E) be the flow network graph constructed in the proof of Theorem 1. For every task t_j, let V_j be the set of all vertices mapped from workers W_i which have edges connected to the vertex mapped from t_j (i.e., $v_{|W_i|+j}$). For every vertex $u \in V_j$, let $(u, v_{|W_i|+j})$ be the edge connected to $v_{|W_i|+j}$. We associate to $(u, v_{|W_i|+j})$ the cost of t_j (i.e., $a(u, v_{|W_i|+j}) = Entropy(l_j)$). Moreover, we set the cost of all other edges in E to 0. Thus, by finding the minimum-cost maximum flow in graph G_i, we have assigned the maximum number of tasks with the minimum cost.* □

In the example of Figure 3, let $Entropy(l_5)$ be the location entropy of the spatial task t_5. Since t_5 is located in the spatial regions of the workers w_1 and w_2, we set the cost of both edges (v_1, v_8) and (v_2, v_8) to $Entropy(l_5)$.

According to the above theorem, solving our problem is equivalent to solving the minimum-cost maximum flow problem at every time instance. In order to solve the minimum-cost maximum flow problem, one of the well-known techniques [6] is to first find the maximum flow of the network using Ford-Fulkerson or any other algorithm which computes the maximum flow. Thereafter, the cost of the flow can be minimized by applying linear programming.

Let G_i=(V, E) be the flow network graph constructed in the proof of Theorem 2 for time instance s_i. Every edge $(u, v) \in E$ has capacity $c(u, v) > 0$, flow $f(u, v) \geq 0$, and cost $a(u, v) \geq 0$, where the cost of sending the flow $f(u, v)$ is $f(u, v) \times a(u, v)$. Let f_{max} be the maximum flow sent from src to dst using the Ford-Fulkerson algorithm. The goal is to minimize the total cost of the flow, which can be defined as follows:

$$\sum_{(u,v) \in E} f(u, v) \times a(u, v) \tag{2}$$

with the constraints

$$f(u, v) \leq c(u, v) \tag{3}$$

$$f(u, v) = -f(v, u), \tag{4}$$

$$\sum_{w \in V} f(u, w) = 0 \text{ for all } u \neq src, dst \tag{5}$$

$$\sum_{w \in V} f(src, w) = f_{max} \text{ and } \sum_{w \in V} f(w, dst) = f_{max} \tag{6}$$

Since all constraints are linear, and our goal is to optimize a linear function, we can solve this by linear programming.

Therefore, our LLEP strategy solves the MTA problem by computing the minimum-cost maximum flow for every time instance, where the cost is defined in terms of the location entropy of the tasks.

4.3 Nearest Neighbor Priority (NNP) Strategy

With the both GR and LLEP approaches, our goal was to maximize the overall task assignment. However, we did not consider the travel cost (e.g., in time or distance) of the workers during the assignment process. With spatial crowdsourcing, the travel cost becomes a critical issue since workers should physically go to the location of the spatial task in order to perform the task. Even though the task assignment process satisfies the spatial constraint of every worker by assigning him only those tasks inside his spatial region, it does not necessarily assign to every worker those tasks with the smallest travel costs. With this approach, we incorporate the travel cost of the workers in the assignment process. Our goal is to maximize the task assignment at every time instance while minimizing the travel cost of the workers whenever possible. Intuitively, tasks which are closer to a worker have smaller travel costs. This means that we still try to maximize the overall task assignment. However, we assign higher priorities to tasks which are closer in spatial distance to the worker.

We define the travel cost between a worker w and a spatial task t in terms of the Euclidean distance[3] between the two, denoted by $d(w,t)$. Consequently, by computing the distance between every worker and his allowable spatial tasks (i.e., those inside his spatial region), we can associate higher priorities to the closer tasks. We do this by associating to every edge between a worker w and a spatial task t a certain cost, which is the distance between the two (i.e., $d(w,t)$). Thus, our problem is to assign the maximum number of tasks during every time instance, while the total cost of the assignment is the lowest. Consequently, the problem turns into the minimum-cost maximum task assignment instance problem. Therefore, a similar solution to that of Section 4.2 but with a different cost function can be applied to solve this problem.

5. PERFORMANCE EVALUATION

We conducted several experiments on both real-world and synthetic data to evaluate the performance of our proposed approaches: GR, LLEP, and NNP. Below, we first discuss our experimental methodology. Next, we present our experimental results.

5.1 Experimental Methodology

We performed three sets of experiments. In the first set of experiments, we evaluated the scalability of our proposed approaches by varying the number of spatial tasks. In the rest of the experiments, we evaluated the impact of the workers' constraints (i.e., R and $maxT$) on the performance of our approaches. With these experiments, we used two performance measures: 1) the total number of assigned tasks, and 2) the average travel cost for a worker to perform a spatial task, in which the travel cost is measured in terms of the Euclidean distance between the worker and the location of the task.

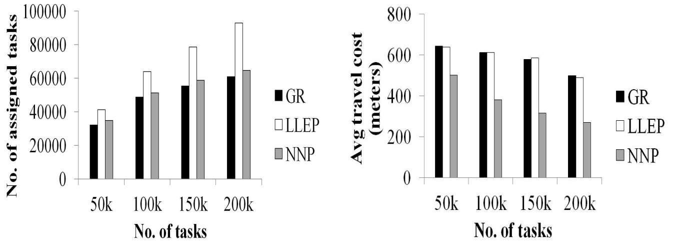
We conducted our experiments with both real-world (REAL) and synthetic (SYN) data sets. The real-world data set is obtained from Gowalla [5], a location-based social network, where users are able to check in to different *spots* in their vicinity. The check-ins include the location and the time that the users entered the spots. For our experiments, we used the check-in data over a period of 100 days in 2010, covering the state of California. Moreover, we assumed that Gowalla users are the workers of our spatial crowdsourcing system. We picked the granularity of a time instance as one day. Consequently, we assumed all the users who checked in during a day as our available workers for that day. Moreover, since users may have various check-ins during a day, for every user w, we set $maxT$ as the number of check-ins of the user in that day, and also we set R as the minimum bounding rectangle of those checked-in locations. Intuitively, checking in a spot is equivalent to accepting a spatial task at that location. Moreover, the spatial tasks were randomly generated for the given spots in the area. In order to compute the location entropy, we discretesized the latitude and longitude space into a 0.0002×0.0002 grid (approximately 30 meters × 30 meters). For every grid cell, we computed location entropy based on the definition explained in Section 4.2. With our synthetic experiments, we randomly generated data from a uniform distribution for both workers and spatial tasks. Moreover, we used a similar grid structure as in REAL for a spatial area of 50 kilometers × 50 kilometers.

In all of our experiments, we varied the number of tasks between 50k and 200k, with 100k as the default value. We also set the duration of every spatial task to 40 days (i.e., $\delta = 40$). Moreover, we fixed the time interval ϕ to 100 days. With the SYN experiments, we fixed the number of workers at 10k. Furthermore, unless mentioned otherwise, we randomly selected the value of $maxT$ between 1 to 20, and the spatial region R between 0.01 to 0.05 of the entire area. Note that since both $maxT$ and R are fixed in REAL (i.e., depend on the worker's check-ins during one day), we only conducted our first set of experiments with both REAL and SYN data sets. In the rest of the experiments, in which we need to vary either $maxT$ or R, we only used SYN. Finally, for each of our experiments, we ran 500 cases, and reported the average of the results.

5.2 Scalability

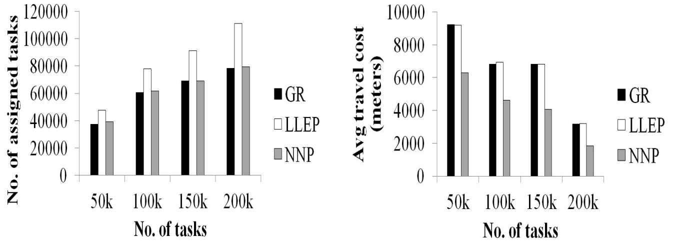
In the first set of experiments, we evaluated the scalability of our approaches by varying the number of spatial tasks from 50k to 200k. Figure 4a depicts the result of our experiments using the synthetic data. As the figure demonstrates, the assignment increases as the number of tasks grows. The figure also shows that LLEP outperforms both GR and NNP in terms of the number of assigned tasks (up to 35%), due to applying the location entropy heuristic. Furthermore, as the number of tasks grows, the impact of location entropy heuristic becomes more significant. The reason is that with a large number of tasks, more tasks appear in the spatial region of every worker, and thus, a wise selection of the tasks becomes more critical. Figure 5a depicts similar experiments using our REAL data. Similarly, the assignment increases as the number of tasks grows. Moreover, the figure shows the superiority of LLEP as compared with GR and NNP in terms of the number of assigned tasks in all cases (up to 30%). Note that our experiments on both REAL and SYN

[3] Other metrics such as network distance are also applicable



a) b)

Figure 4: Scalability - Synthetic data



a) b)

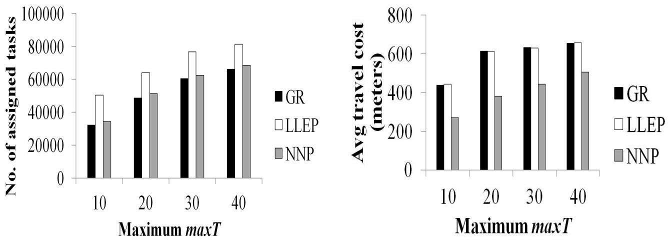
Figure 5: Scalability - Real data

data shows that a large number of tasks (more than 50%) remains unassigned. This happens due to different reasons such as the constraints of the workers (e.g., the spatial region of a worker may overlap with only a small number of tasks) or the expiration of the unassigned tasks.

Figures 4b and 5b depict the impact of varying the number of tasks on the average travel cost of the workers using SYN and REAL data, respectively. As the figures show, the average travel cost of the workers decreases in all cases because in a task-dense area, there is a higher probability that an assigned task is in a closer distance to a worker. Moreover, we observe that NNP improves the travel cost of the workers as compared with GR and LLEP by up to 45% using the SYN data and up to 42% using the REAL data, which proves the effectiveness of the travel cost heuristic.

5.3 Effect of Maximum Acceptable Tasks Constraint

In the next set of experiments, we evaluated the impact of the maximum acceptable tasks (i.e., $maxT$) constraint using the synthetic data. We increased the value of $maxT$ between [1-10] to [1-40]. Figure 6a illustrates an increase in the number of assigned tasks as $maxT$ grows. The reason is that with an increase in $maxT$, workers are willing to do more tasks, and thus, the number of assignment increases. Moreover, similar to the previous experiments, LLEP is the superior approach in terms of improving the number of task assignment (up to 36% times better than GR). However, the impact of location entropy heuristic is more significant for smaller values of $maxT$ (Figure 6a). The reason is that with smaller values of $maxT$, only a small number of tasks should be selected from those inside the spatial region of a worker. Therefore, a wise selection of tasks using the location entropy heuristic becomes more significant.



a) b)

Figure 6: Effect of $maxT$ - Synthetic data

Figure 6b depicts an increase in the travel cost as $maxT$ grows. The reason is that the higher $maxT$, the more tasks assigned to every worker, resulting in higher travel cost. Moreover, while NNP outperforms both GR and LLEP in terms of the travel cost (between 27% to 39%), its superiority is more significant with smaller values of $maxT$. The reason is that as $maxT$ grows, more tasks are selected inside the spatial region of the worker. Thus, the impact of the travel cost heuristic becomes less critical.

5.4 Effect of Spatial Region Constraint

In our final set of experiments, we measured the performance of our approaches with respect to expanding the spatial region of every worker from [0.01% 0.05%] to [0.01% 0.2%]. As Figure 7a shows, with an expansion in R, the number of assigned tasks increases. The reason is that larger spatial regions cover more number of spatial tasks. That is, more edges connect the vertices mapped from the workers to the vertices mapped from the tasks, which leads to more flow in the corresponding flow network graph. Moreover, Figure 7b shows the effect of varying R on the travel cost. The figure illustrates that the travel cost increases with an expansion in R. This is because as R grows, farther tasks will be assigned to the workers with higher probability, which increases the average travel cost.

The main observation from this set of experiments is that LLEP outperforms both GR and NNP in terms of the number of task assignment, while the NNP approach is superior in terms of the travel cost. This shows that based on the objective of the crowdsourcing application (i.e., maximizing the assignment or maximizing the assignment with the minimum possible travel cost), either of the LLEP or NNP approaches can be selected[4].

6. RELATED WORK

As discussed earlier, crowdsourcing has been gathering extensive attention in the research community. A related survey in this area can be found in [17]. With the increasing popularity of crowdsourcing, recently, a set of crowdsourcing services such as Amazon's Mechanical Turk (AMT) [1] and CrowdFlower [3] have emerged which allow requesters to issue tasks that workers can perform for a certain reward. Crowdsourcing has been largely used in a wide range of applications. Examples of such applications are image search

[4] An alternative solution is a hybrid approach which utilizes both the location entropy and the travel cost in the assignment process, which is the focus of our future work.

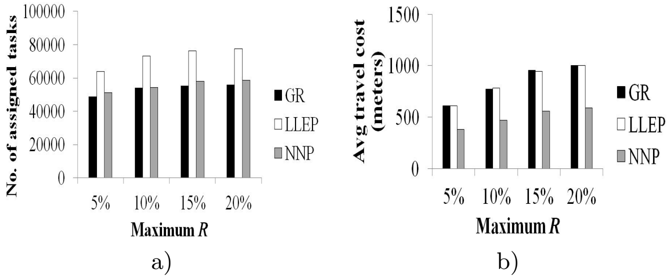


Figure 7: Effect of R - Synthetic data

[38], natural language annotations [31], video and image annotations [14, 32], social games [20, 35], graph search [28], and search relevance [11]. Also, a number of studies have focused on analyzing the crowdsourcing platforms. In [30], the demographics of the AMT workers are studied. Furthermore, [22] analyzes the marketplace for Amazon's Mechanical Turk. Moreover, the database community has utilized crowdsourcing in relational query processing [19, 25, 27]. In [19] a relational query processing system is proposed that uses crowdsourcing to answer queries that cannot otherwise be answered.

Despite all the studies on crowdsourcing, only a few studies [12, 13] have focused on spatial crowdsourcing. In [13], the problem of crowdsourcing location-based queries over Twitter has been studied, which employs a location-based service (e.g., Foursquare) to find the appropriate people to answer the given query. Even though this work focuses on location-based queries, it does not assign to users any *spatial* task, for which the user should go to that location and perform the corresponding task. Instead, it chooses users based on their historical Foursquare check-ins. Moreover, in [12], a crowdsourcing platform with WST mode is proposed, which utilizes location as a parameter to distribute tasks among worker.

One class of spatial crowdsourcing is known as participatory sensing, in which workers form a campaign to perform sensing tasks. Examples of participatory sensing campaigns include [2, 10, 21, 26], which uses GPS-enabled mobile phones to collect traffic information. Other examples are [15, 23]. In [15], a participatory sensing framework with WST mode is proposed, whereas, in [23], a participatory sensing framework with SAT mode is introduced. However, the major drawback of all the existing work on participatory sensing is that they focus on a single campaign and try to address the challenges specific to that campaign. Another drawback of most existing work on participatory sensing (e.g., [23]) is that they are designed for small campaigns, with a small number of participants, and are not scalable to large spatial crowdsourcing applications. Finally, while most existing work on participatory sensing systems focus on a particular application, our work introduces a generalized framework for any type of spatial crowdsourcing system.

Another class of spatial crowdsourcing is known as volunteered geographic information (or VGI), in which the goal is to create geographic information provided voluntarily by individuals. Some examples include WikiMapia [9], OpenStreetMap [7], and Google Map Maker [4]. These projects allow the users to generate their own geographic content, and add it to a pre-built map. For example, a user can add the features of a location, or the events occurred at that location. However, the major difference between VGI and spatial crowdsourcing is that in VGI, users unsolicitedly participate by randomly contributing data, whereas in spatial crowdsourcing, a set of spatial tasks are queried by the requesters, and workers are required to perform those tasks. Moreover, with most VGI projects ([4, 9]), users are not required to physically go to a particular location in order to generate data with respect to that location. Finally, as the name suggests, VGI falls into the class of self-incentivised spatial crowdsourcing.

7. CONCLUSION AND FUTURE WORK

In this paper, we introduced spatial crowdsourcing as the process of crowdsourcing a set of spatial tasks to a set of workers. Moreover, we defined a taxonomy for spatial crowdsourcing. We also studied one class of spatial crowdsourcing in more details and formally defined the MTA problem. Subsequently, we proposed our assignment protocol that included three various solutions to the MTA problem, namely GR, LLEP, and NNP. In our experiments on both real and synthetic data, we demonstrated that in comparison with GR, our LLEP approach can improve the number of assigned tasks by up to 36%, while the NNP approach can improve the travel cost of the workers by up to 41%.

As future work, we aim to focus on the other classes of spatial crowdsourcing. Moreover, since location privacy is one of the major impediments that may hinder workers from participation in spatial crowdsourcing, we plan to extend our work to protect the location privacy of the workers.

Acknowledgment

This research is supported in part by Award No. 2011-IJ-CX-K054 from National Institute of Justice, Office of Justice Programs, U.S. Department of Justice, as well as NSF grants CNS-0831505 (CyberTrust) and IIS-1115153, the USC Integrated Media Systems Center (IMSC), and unrestricted cash and equipment gifts from Northrop Grumman, Google, Microsoft and Qualcomm. The opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect those of the Department of Justice and the National Science Foundation.

8. REFERENCES

[1] Amazon mechanical turk. `http://www.mturk.com`.

[2] Center for embedded networked sensing (cens). `http://urban.cens.ucla.edu/projects/`.

[3] Crowdflower. `http://www.crowdflower.com`.

[4] Google map maker. `http://www.wikipedia.org/wiki/Google Map Maker`.

[5] Gowalla. `http://www.wikipedia.org/wiki/Gowalla`.

[6] Minimum-cost maximum flow problem. `http://www.wikipedia.org/wiki/Minimum-cost flow problem`.

[7] Openstreetmap. `http://www.wikipedia.org/wiki/OpenStreetMap`.

[8] Scheduling. `http://en.wikipedia.org/wiki/Scheduling(computing)`.

[9] Wikimapia. `http://www.wikipedia.org/wiki/WikiMapia`.

[10] University of california berkeley, 2008-2009. `http://traffic.berkeley.edu/`.

[11] O. Alonso, D. E. Rose, and B. Stewart. Crowdsourcing for relevance evaluation. *SIGIR Forum*, 42(2):9–15, 2008.

[12] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pages 13–22, 2010.

[13] M. Bulut, Y. Yilmaz, and M. Demirbas. Crowdsourcing location-based queries. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 513 –518, 2011.

[14] K.-T. Chen, C.-C. Wu, Y.-C. Chang, and C.-L. Lei. A crowdsourceable qoe evaluation framework for multimedia content. In *Proceedings of the 17th ACM international conference on Multimedia*, MM '09, pages 491–500.

[15] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *MobiSys '08*, pages 211–224.

[16] J. Cranshaw, E. Toch, J. Hong, A. Kittur, and N. Sadeh. Bridging the gap between physical location and online social networks. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 119–128, 2010.

[17] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, 2011.

[18] Y. F. Dong, S. Kanhere, C. T. Chou, and N. Bulusu. Automatic collection of fuel prices from a network of mobile cameras. In *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems*, DCOSS '08, pages 140–156, 2008.

[19] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 international conference on Management of data*, SIGMOD '11, pages 61–72, 2011.

[20] I. Guy, A. Perer, T. Daniel, O. Greenshpan, and I. Turbahn. Guess who?: enriching the social graph through a crowdsourcing game. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1373–1382.

[21] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *SenSys'06*, pages 125–138.

[22] P. G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS*, 17(2):16–21, 2010.

[23] L. Kazemi and C. Shahabi. A privacy-aware framework for participatory sensing. *SIGKDD Explorations*, 13(1), 2011.

[24] J. Kleinberg and E. Tardos. *Algorithm Design.* Addison-Wesley Longman Publishing Co., Inc., 2005.

[25] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.

[26] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *SenSys'08*, pages 323–336.

[27] A. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. In *Conference on Inovative Data Systems Research (CIDR 2011)*, 2011.

[28] A. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *Proc. VLDB Endow.*, 4(5):267–278, 2011.

[29] E. Paulos, R. Honicky, and E. Goodman. Sensing atmosphere. In *In Workshop on Sensing on Everyday Mobile Phones in Support of Participatory Research*, 2007.

[30] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, CHI EA '10, pages 2863–2872.

[31] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263.

[32] A. Sorokin and D. Forsyth. Utility data annotsation with amazon mechanical turk. *Computer Vision and Pattern Recognition Workshop*, 0, 2008.

[33] J. Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations.* 2004.

[34] L. H. U, M. L. Yiu, K. Mouratidis, and N. Mamoulis. Capacity constrained assignment in spatial databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 15–28. ACM, 2008.

[35] L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8):58–67, 2008.

[36] R. C.-W. Wong, Y. Tao, A. W.-C. Fu, and X. Xiao. On efficient spatial matching. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 579–590. VLDB Endowment, 2007.

[37] X. Xie, H. Chen, and H. Wu. Bargain-based stimulation mechanism for selfish mobile nodes in participatory sensing network. In *Proceedings of the 6th Annual IEEE communications society conference on Sensor, Mesh and Ad Hoc Communications and Networks*, SECON'09, pages 72–80, 2009.

[38] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 77–90.