A Server-Assigned Spatial Crowdsourcing Framework

Hien To, University of Southern California
Cyrus Shahabi, University of Southern California
Leyla Kazemi, Microsoft[1]

With the popularity of mobile devices, *spatial crowdsourcing* is rising as a new framework that enables human workers to solve tasks in the physical world. With spatial crowdsourcing, the goal is to crowdsource a set of spatiotemporal tasks (i.e., tasks related to time and location) to a set of workers, which requires the workers to physically travel to those locations in order to perform the tasks. In this paper, we focus on one class of spatial crowdsourcing, in which the workers send their locations to the server and thereafter the server assigns to every worker tasks in proximity to his location with the aim of maximizing the overall number of assigned tasks. We formally define this *maximum task assignment (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, including the spatial distribution and the travel cost of the workers. MTA is based on the assumptions that all tasks are of the same type and all workers are equally qualified in performing the tasks. Meanwhile, different types of tasks may require workers with various skill-sets or expertise. Subsequently, we extend MTA by taking the expertise of the workers into consideration. We refer to this problem as the *maximum score assignment (MSA)* problem and show its practicality and generality. Extensive experiments with various synthetic and two real-world data sets show the applicability of our proposed framework.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications Spatial databases and GIS

General Terms: Algorithms

Additional Key Words and Phrases: Crowdsourcing, Spatial Crowdsourcing, Participatory Sensing

1. INTRODUCTION

With the ubiquity of smart phones and wireless network bandwidth improvements, every person with a mobile phone can now act as a multi-model sensor collecting and sharing various types of high-fidelity spatiotemporal data instantaneously (e.g., picture, video, audio, location, time, speed, direction, acceleration). Exploiting this phenomena, a new framework for efficient and scalable data collection has emerged, namely *spatial crowdsourcing* [Kazemi and Shahabi 2012]. With spatial crowdsourcing, the goal is to crowdsource a set of *spatial tasks* (i.e., tasks related to a location) to a set of *workers*, which requires the workers to physically go to those locations in order to perform the tasks. Spatial crowdsourcing has application in numerous domains such as journalism, tourism, intelligence, disaster response and urban planning. To illustrate, consider a disaster-response scenario, where Red Cross (i.e., *requester*) is interested in collecting pictures and videos of disaster areas from various locations of a city. With spatial crowdsourcing, the requester issues a query to a spatial crowdsourcing server (*SC-server*). Consequently, the SC-server crowdsources the query among the available workers in the vicinity of the events. Once the workers document their events with their mobile phones, the results are sent back to the requester.

While crowdsourcing has largely been used by both research communities (e.g., image processing [Chen et al. 2009], [Sorokin and Forsyth 2008] and [Whitehill et al. 2009], databases [Franklin et al. 2011], [Marcus et al. 2011], [Parameswaran et al. 2012] and [Demartini et al. 2013]) and industry (e.g., oDesk [oDesk 2005], MTurk [mechanical turk 2005] and CrowdFlower [Crowdflower 2009]), spatial crowdsourcing only recently received some attention [To et al. 2014], [Musthag and Ganesan 2013],

[1]The work been completed when the author was a PhD student at USC's Infolab.

[Kazemi et al. 2013], [Kazemi and Shahabi 2012] and [Alt et al. 2010]. 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 campaigns include [UCB 2008], [Hull et al. 2006] and [Mohan et al. 2008]. For example, the Mobile Millennium project [UCB 2008] 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 CarTel [Hull et al. 2006] and Nericell [Mohan et al. 2008], which used mobile sensors/smart phones mounted on vehicles to collect information about traffic, WiFi access points on the route and road information. All these previous studies on participatory sensing focus on a single campaign and try to address challenges specific to that campaign. However, our focus is on devising a generic crowdsourcing framework, similar to MTurk where multiple campaigns can be handled simultaneously, but spatial. 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. To move from single-campaign and customized participatory-sensing to multi-purpose and generic spatial crowdsourcing, the system needs to scale. That is, it should be able to efficiently assign tasks to workers.

In this paper, we first introduce our problem focus, spatial crowdsourcing, in the context of crowdsourcing. Next, we focus on one class of spatial crowdsourcing, known as *server assigned*, 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 server, who receives the location of the workers, assigns to every worker his nearby tasks. The optimization goal is to maximize the number of assigned tasks while conforming to the constraints of the workers, referred to as *maximum task assignment (MTA)*. The solution to the MTA could be straightforward if the SC-server had a global knowledge of both the spatial tasks and the workers. However, the server is continuously receiving spatial tasks from requesters and task inquiries from the workers. Therefore, the server can only maximize the task assignment at every time instance (i.e., local optimization) with no knowledge of the future. We proposed three alternative solutions to the MTA problem. The first approach, namely *Basic*, follows the local optimization strategy by maximizing the task assignment at every time instance. Our second approach, called *Least Location Entropy Priority (LLEP)*, improves the overall task assignment by assigning higher priority to spatial tasks located in worker-sparse areas (i.e., places with lower location entropy). The intuition is that spatial tasks are more likely to be performed in future if they are located in worker-dense areas (i.e., areas with high population of workers or high location entropy). With spatial crowdsourcing, the travel cost of the workers becomes critical. Thus, in our third approach, namely *Close Distance Priority (CDP)*, we incorporated the travel cost of the workers into the task assignment by assigning higher priority to the tasks with lower travel cost.

So far we assume that all tasks are of the same type and all workers are equally qualified in performing the tasks. We relax these assumptions by allowing each worker to have a set of skills and every spatial task to have a type. Subsequently, we define *expertise match* as an assignment of a task to a worker, in which the worker has the required qualification to perform the task (e.g., the task of taking a high quality picture is assigned to a photographer). Thus, we assign higher *scores* to expertise matches

than to non-expertise matches (i.e., matches that satisfy only worker's constraints). Consequently, we formalize a new problem named *Maximum Score Assignment (MSA)*, whose optimization goal is to maximize the total score assignment while conforming to the constraints of the workers. To solve MSA, we extend the same heuristics to MTA, including Basic, LLEP and CDP. Assuming there is only one task type and all workers are the same, MSA becomes MTA, i.e., MSA is a generalization of MTA. Therefore, in the experiments we only evaluate the solutions to MSA.

Our extensive experiments on both real and synthetic data show that in comparison with Basic, our LLEP improves the overall task assignment (i.e., total score) by up to 35%, while the CDP approach can improve the travel cost of the workers by up to 90%. Furthermore, with our real data sets, LLEP shows its superiority in maximizing the number of expertise matches by up to 30% when compared to the other approaches. Moreover, an unexpected positive side effect of minimizing the travel cost in CDP is maximizing the number of expertise matches. As a result, the CDP approach is better than LLEP by up to 15% in maximizing the number of expertise matches with our synthetic data. Consequently, based on the objective of the crowdsourcing application (i.e., maximizing the assignment or minimizing travel cost), either of the LLEP or CDP approaches can be selected. Finally, by varying the expertise match score, we can flexibly change the optimization goal (e.g., from maximum assigned tasks to maximum expertise matches), showing the practicality and generality of MSA when compared to the MTA problem.

A related problem namely online matching [Karp et al. 1990] and [Kalyanasundaram and Pruhs 2000] can be considered as a special case of our task assignment (i.e., MTA and MSA) where the worker set (or task set) is given in advance while items in the other set arrive one at a time or in batch. In Section 2, we distinguish MTA and MSA from the online bipartite matching problems, which do not use any spatial knowledge. We also experimentally compared our approaches, Basic and LLEP with a typical online bipartite matching algorithm namely Ranking [Karp et al. 1990]. Experimental results on both real and synthetic data show that our approaches outperforms the Ranking algorithm in terms of maximizing the number of assigned tasks.

In a preliminary version of this work [Kazemi and Shahabi 2012], we introduced MTA and the three approaches, Basic, LLEP, CDP. This article subsumes [Kazemi and Shahabi 2012] by relaxing the assumptions of identical workers and tasks as well as solving the MSA problem, which is a generalization of MTA. We also related our problem to the online bipartite matching problem and added a new set of experiments, in order to compare our heuristics with a solution to the online matching problem [Karp et al. 1990], called the Ranking algorithm. For this comparison, we needed to modify our experimental setup so that the worker-set as well as the cost of matches are known in advance. Moreover, in this paper, to evaluate the approaches, we generated synthetic data sets in a more systematic manner. We also used a new real-world data from Yelp, which is suitable for evaluating MSA.

The remainder of this paper is organized as follows. In Section 2, we review the related work. Section 3 presents the taxonomy for crowdsourcing and introduces our problem focus, spatial crowdsourcing. In Section 4, we discuss a set of preliminaries in the context of spatial crowdsourcing. Thereafter, in Sections 5 and 6 we formally define the MTA problem and its extension, MSA, then explain our assignment solutions. Section 7 reports on our experimental results. Finally, Section 8 concludes the paper.

2. RELATED WORK

Crowdsourcing: Crowdsourcing has recently been attracting extensive attention in the research community. A recent survey in this area can be found in [Kittur et al. 2013]. With a growing recognition of crowdsourcing, many crowdsourcing services in-

cluding oDesk [oDesk 2005], MTurk [mechanical turk 2005] and CrowdFlower [Crowdflower 2009] have emerged, which allow requesters to issue tasks that workers can perform for a certain reward. Crowdsourcing has been largely adopted in a wide range of applications. Examples of such applications include but not limited to image search [Yan et al. 2010], natural language annotations [Snow et al. 2008], video and image annotations [Chen et al. 2009], [Sorokin and Forsyth 2008] and [Whitehill et al. 2009], search relevance [Alonso et al. 2008] and [Bozzon et al. 2012], social games [Von Ahn and Dabbish 2008] and [Guy et al. 2011] and graph search [Parameswaran et al. 2011]. Moreover, the database community has utilized crowdsourcing in database design, query processing [Franklin et al. 2011], [Marcus et al. 2011], [Parameswaran et al. 2012], [Demartini et al. 2013] and [Zhao et al. 2013] and data analytics [Liu et al. 2012] and [Wang et al. 2012]. In [Franklin et al. 2011], a relational query processing system is proposed, which uses crowdsourcing to answer queries that cannot otherwise be answered. As part of the crowdsourced database systems, human-powered versions of the fundamental operators, such as sort and join [Marcus et al. 2011] and filter [Parameswaran et al. 2012] were developed. Recently in [Liu et al. 2012], a system was developed to improve the accuracy of data analytics jobs by exploiting crowdsourcing techniques.

Spatial Crowdsourcing: Despite all the studies on crowdsourcing, spatial crowdsourcing only recently received some attention [To et al. 2014], [Musthag and Ganesan 2013], [Kazemi et al. 2013], [Dang et al. 2013], [Kazemi and Shahabi 2012], [Bulut et al. 2011] and [Alt et al. 2010]. In [Alt et al. 2010], a crowdsourcing platform is proposed, which utilizes location as a parameter to distribute tasks among workers. In [Kazemi and Shahabi 2012], a spatial crowdsourcing platform whose goal is to maximize the number of assigned tasks is proposed. Since the workers cannot always be trusted, another work [Kazemi et al. 2013] aims to tackle the issue of trust by having tasks performed redundantly by multiple workers. In [Dang et al. 2013], the problem of complex spatial tasks (i.e., each task comprises of a set of spatial sub-tasks) is introduced, in which the assignment of the complex task requires performing all of its sub-tasks. Recently in [To et al. 2014], the authors introduced the problem of protecting worker location privacy in spatial crowdsourcing. This study proposes a framework that achieves differentially-private protection guarantees.

Moreover, 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 a given query [Bulut et al. 2011]. 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. In [Musthag and Ganesan 2013], spatiotemporal dynamics in mobile task markets (e.g., [Agent 2010; Gigwalk 2010]) were studied.

The well-known concept of participatory sensing could be deemed as one class of spatial crowdsourcing, in which workers form a campaign to perform sensing tasks. Examples of participatory sensing campaigns include [UCB 2008], [Kazemi and Shahabi 2011], [Cornelius et al. 2008], [Hull et al. 2006] and [Mohan et al. 2008]. 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 studies on participatory sensing (e.g., [Kazemi and Shahabi 2011]) 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 can be used for any type of spatial crowdsourcing system.

Another class of spatial crowdsourcing is known as volunteered geographic information (or VGI), whose goal is to create geographic information provided voluntarily by individuals. Examples for this class include Google Map Maker [map maker 2008], OpenStreetMap [Openstreetmap 2004] and WikiMapia [Wikimapia 2006]. 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 voluntarily 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 ([map maker 2008] and [Wikimapia 2006]), 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.

Matching Problems: One can consider the task assignment problem in spatial crowdsourcing as the online bipartite matching problem ([Karp et al. 1990] [Kalyanadundaram and Pruhs 1993] [Khuller et al. 1994], [Kalyanasundaram and Pruhs 2000] and [Mehta et al. 2007]). Online bipartite matching problem is the most relevant variation to spatial crowdsourcing as it captures the dynamism of tasks arriving at different times. However, in online bipartite matching one of the item sets is given in advance and items from the other set arrive (usually one at a time), while in spatial crowdsourcing both sets, i.e., workers and tasks, can come and go without our knowledge. Thus, to some extent, online matching can be considered as a special case of our task assignment where the worker set (or task set) is fixed. In addition, with online matching, the cost/weight of any match is known in advance. However, with spatial crowdsourcing, the cost for a worker to perform a task mainly corresponds to the time it takes for him to travel to the locations of the tasks. As the result, the cost of a task is not a fixed value but is dependent on the worker's prior location. Hence, the sequence in which the tasks are performed impacts the cost. That is, with spatial crowdsourcing, the cost of the execution of a set of tasks is the distance of the shortest path that starts from the workers current location and goes through the locations of all the assigned tasks. On the other hand, with online matching [Kalyanasundaram and Pruhs 2000] the overall cost for one worker would be the sum of the distances between the worker and each assigned task. Finally, the performance of an online algorithm is often evaluated based on competitive ratio - the ratio between its performance and the offline algorithm's performance. The online algorithm is competitive if its competitive ratio is bounded under any circumstance; this is not the goal of MSA which focuses on the average performance.

Some recent studies in spatial matching [Wong et al. 2007] and [Yiu et al. 2008] do focus on efficiency and use the spatial features of the objects for more efficient assignment. Spatial matching is a one to one (or in some cases one to many) assignment between objects of two sets where the goal is to optimize over some aggregate (etc., sum, max) function of the distance between matched objects. For example, the objective in [Varadarajan 1998] is to pair up 2N points in the plane into N pairs such that sum of the Euclidean distances between the paired points is minimized. In [Wong et al. 2007], given a set of customers and a set of service providers with limited capacities, the goal is to assign the maximum number of customers to their nearest providers, among all the providers whose capacities have not been exhausted in serving other closer customers. These studies assume a global knowledge about the locations of all objects exists a priori and the challenge comes from the complexity of spatial matching. However, spatial crowdsourcing differs due to the dynamism of tasks and workers (i.e., tasks and workers come and go without our knowledge), thus the challenge is to perform the task assignment at a given instance of time with the goal of global

optimization across all times. Moreover, the fact that workers need to travel to task locations causes the landscape of the problem to change constantly. This will add another layer of dynamism to spatial crowdsourcing that renders it a unique problem.

Expertise matching is the concept of assigning queries to experts that has gained extensive interest from various research fields for its wide range of applications such as product-reviewer alignment, product-endorser matching, paper-reviewer assignment [Mimno and McCallum 2007], [Sun et al. 2008] and [Hettich and Pazzani 2006]. In [Tang et al. 2012], a framework for expertise matching with various constraints was introduced, which is capable of rendering the optimal solution. However, none of these studies address the problem of expertise in spatial matching. Our objective is different than that of these studies as we address the problem of maximum expertise matches, while they try to find the best individual match based on some models/methods.

Vehicle Routing Problems: Modeling the assignment cost as the shortest path visiting the location of multiple tasks brings another class of problems to attention. In this context the assignment problem in spatial crowdsourcing becomes similar to the Traveling Salesman Problem (TSP) [Lawler et al. 1985] and the Vehicle Routing Problem (VRP) [Toth and Vigo 2001]. The goal of VRP is to minimize the cost of delivering goods located at a central depot to customers who have placed orders for such goods with a fleet of vehicles. The online versions of both TSP and VRP have been studied to some extent where new locations to visit are revealed incrementally. Since there is only one salesman in the standard version of TSP, here we focus on VRP. Different variations of VRP have been studied, yet there are differences between task assignment in spatial crowdsourcing and these variations. With VRP, a server can pay a penalty and deny visiting a location; however, in spatial crowdsourcing the goal is to maximize the number of assigned tasks so the worker does not have the option of denying a task. Furthermore, with VRP, all servers start from the same depot whereas in spatial crowdsourcing each worker can have a different starting location. Moreover, with VRP we have a fixed number of servers whereas in spatial crowdsourcing the same type of dynamism for tasks can apply to the workers. That is, workers can be added (removed) to (from) the system at any time.

3. SPATIAL CROWDSOURCING

Spatial crowdsourcing opens up a new mechanism for spatial tasks (i.e., tasks related to a location) to be performed by humans. Consequently, we formally define spatial crowdsourcing as the process of crowdsourcing a set of spatial tasks to a set of human workers where performing an outsourced task is only possible if the workers are physically at the location of the task, termed spatial task. In this section, we present a taxonomy for crowdsourcing (Figure 1). First, we classify spatial crowdsourcing based on worker's motivation. Next, we define two modes of task publishing in spatial crowdsourcing. Finally, we classify the workers into two groups based on whether or not they have constraints.

3.1. Worker's Motivation

A major challenge in any crowdsourcing system is how to motivate people to participate. Four levels of worker motivation can be found in [Quinn and Bederson 2011], including pay, altruism, fun and implicit. To simplify, crowdsourcing can be classified based on the motivation of the workers into two classes: *reward-based* and *self-incentivised* (Figure 1). With reward-based spatial crowdsourcing, every spatial task has a price (assigned by a requester) and workers will receive a certain reward for every spatial task they perform correctly. Examples of this class include [Agent 2010; Gigwalk 2010]. With self-incentivised spatial crowdsourcing, workers volunteer to perform the tasks or usually have other incentives rather than receiving a reward such

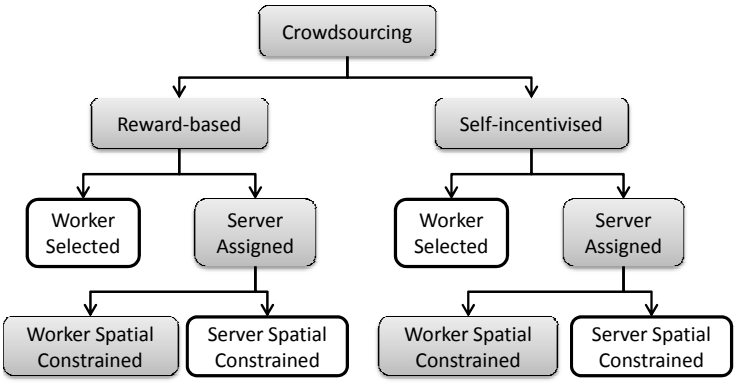


Fig. 1: The focus of this paper, spatial crowdsourcing, is shown in grey.

as documenting an event or promoting their cultural, political or religious views. An example of this class is [UCB 2008], in which more than 5000 users voluntarily install traffic software onto their phones and report traffic information. Our work focuses on this class of spatial crowdsourcing.

3.2. Task Publishing Modes

Next, we define two task publishing modes in spatial crowdsourcing, *Worker Selected Tasks (WST)* and *Server Assigned Tasks (SAT)*. With the WST mode, the SC-server publishes the spatial tasks and online workers can choose any spatial task in their vicinity without the need to coordinate with the server. One advantage of the WST mode is that the workers do not need to reveal their locations to SC-server since they can choose any arbitrary task in their vicinity autonomously. However, one drawback of this mode is that the server does not have any control over the allocation of spatial tasks. This may result in some spatial tasks never be assigned, while others are 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 may not result in a globally optimal assignment. An example of the WST mode is [Alt et al. 2010], where the workers browse for available spatial tasks, and pick the ones in their neighborhood.

With the SAT 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 nearby tasks while maximizing the overall task assignment. Examples of this mode of spatial crowdsourcing include [Kazemi and Shahabi 2012] and [Kazemi and Shahabi 2011]. In [Kazemi and Shahabi 2011], a framework for small campaigns is proposed, where workers are assigned to their closeby sensing tasks. However, the drawback is that the workers should report their locations to the server for every assignment, which can pose a privacy threat. Recently in [To et al. 2014], a framework was proposed to sanitize workers' locations according to differential privacy while still using SC-server as a broker to assign tasks to workers. A real-world example of the SAT mode is Uber[2], a mobile app that connects passengers with drivers of vehicles for ridesharing. Our focus in this paper is on this mode of spatial crowdsourcing.

[2]https://www.uber.com/

3.3. Worker's Constraints

Finally, in the case of SAT, we divide the workers into two groups based on whether or not they have constraints. With workers without constraints, the server has full flexibility on how tasks should be assigned to the workers. This means that workers only send their locations to the server, and the server assigns every spatial task to its nearby worker [Kazemi and Shahabi 2011]. With workers with constraints, the server needs to satisfy the constraints while assigning the tasks. An example of spatial constraint is that every worker only accepts spatial tasks in a spatial region (i.e., his working region).

4. PRELIMINARIES

In this section, we define a set of preliminaries in the context of self-incentivized SAT spatial crowdsourcing with constraints. First, we formally define a spatial task.

Definition 4.1 (*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 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 by traveling to the location. An example of a spatial task is taking a picture of a particular dish in a restaurant. This means that the worker needs to physically go to the exact location of the restaurant in order to take the picture. Each task has a deadline and it can appear at several time instances until reaching its deadline [3]. For example, if one time instance is 2 minutes long and the task deadline is within 40 minutes, the task can last for 20 time instances.

Definition 4.2 (*Spatial Crowdsourced Query*). A spatial crowdsourced query (or SC-Query) of form $<t_1, t_2, ...>$ asked by a requester is a set of spatial tasks t_i that needs to be crowdsourced.

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. We assume that all workers perform every task correctly. Thus, the SC-server needs to assign every spatial task to one and only one worker. Figure 2 shows our spatial crowdsourcing framework. In the following we formally define a worker.

Definition 4.3 (*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.

We assume that all tasks are of the same type and all workers have the same set of skills. We will relax these assumptions in Section 6. Once a worker goes online, he sends a task inquiry to the SC-server (Figure 2). We now formally define the task inquiry.

Definition 4.4 (*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. The other constraint, $maxT$, is the maximum number of tasks

[3]We included task deadline to maintain the generality of our definition and the full consideration of the temporal aspects of tasks is beyond the scope of this paper. For now, we simply use the tasks' deadlines to exclude them for assignment in the time instances passed their deadlines.

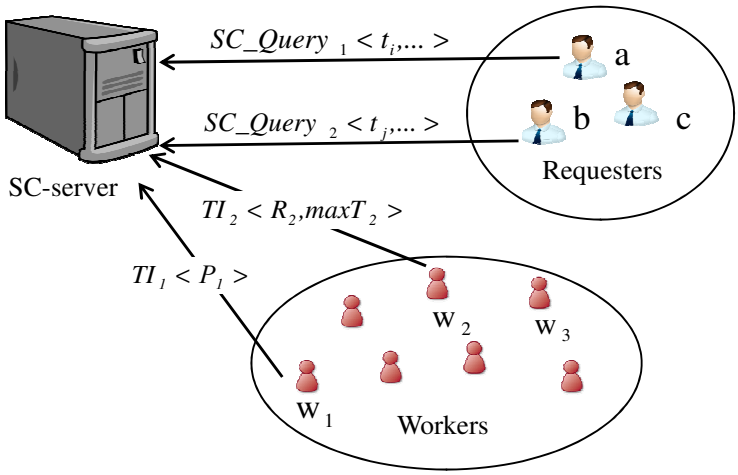


Fig. 2: The spatial crowdsourcing framework.

that the worker is willing to perform in *one* time instance. The worker can have either or both of the constraints[4]. Figure 2 shows an example of task inquiry, in which worker w_1 has only one constraint (i.e., R_1) and worker w_2 has both constraints (i.e., R_2 and $maxT_2$). Note that the task inquiry is defined for the SAT mode with constraints scenario (see Taxonomy in Figure 1), where workers should send both their locations and constraints to the SC-server for proper task assignment.

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 4.5 (*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 matches of form $<w, t>$, where a spatial task t is assigned to a worker w, while satisfying the workers' constraints, Rs and $maxTs$. Also, $|I_i|$ denotes the number of tasks that are assigned at time instance s_i.

Consequently, the task assignment instance set must conform to the constraints of the workers. Given a worker w with both constraints (R and $maxT$), this means for every match $<w, t>$ in I_i, the spatial task t must be located inside the spatial region R of the worker. Moreover, every worker w can be assigned to at most $maxT$ number of tasks (i.e., the number of matches in I_i including w is at most $maxT$).

5. MAXIMUM TASK ASSIGNMENT

5.1. Problem Definition

Based on the preliminaries in Section 4, we now define the *Maximum Task Assignment (MTA)* problem.

Definition 5.1 (*Maximum Task Assignment (MTA)*). Given $\phi = \{s_1, s_2, ..., s_n\}$, each s_i is a time instance, let I_i be the assigned tasks at time instance s_i. The maximum

[4]We assume that workers' constraints (i.e., R and $maxT$) will not change over time instances.

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., $\sum_{i=1}^{n} |I_i|$) is maximized.

In order to solve the MTA problem, the SC-server should have a global knowledge of all the spatial tasks and the workers. This would allow the server to optimally assign the tasks to the workers. However, the server does not have such knowledge. At every instance of time, the server receives a set of new tasks from the requesters, and a set of new task inquiries from the workers. Therefore, the 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 server tries to optimize the task assignment locally at every instance of time.

In the following, we propose three solutions to this problem. All the solutions follow the local optimal assignment strategy. The first approach tries to solve MTA by maximizing the task assignment at every instance of time (Greedy). The second approach tries to improve the optimization by applying a heuristic that utilizes the location entropy of an area, to maximize the overall assignment (Least Location Entropy Priority). Finally, the third approach tries to maximize the task assignment while taking into account the travel cost of the workers (Close Distance Priority).

5.2. Assignment Protocol

The main challenges of spatial crowdsourcing are due to the large-scale, ad-hoc and dynamic nature of the workers and tasks. First, to continuously match thousands of spatial crowdsourcing campaigns, where each campaign consists of many spatiotemporal tasks, with millions of workers, an SC-Server must be able to run efficient task-assignment strategies that can scale. Second, the task-assignment must be performed frequently and in real-time as new tasks and workers become available or as tasks are completed (or expired) and workers leave the system.

5.2.1. Basic Strategy. As discussed earlier, with this approach the idea is to do the maximum assignment at every instance of time. This strategy only performs local optimization; therefore, it may not result in a globally optimal answer. Given a set of online workers $W_i = \{w_1, w_2, ...\}$, and a set of available tasks $Ti = \{t_1, t_2, ...\}$ 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 (MTA) instance problem. Thus, our goal in this approach is to maximize the overall assignment by solving the MTA instance problem for every instance of time.

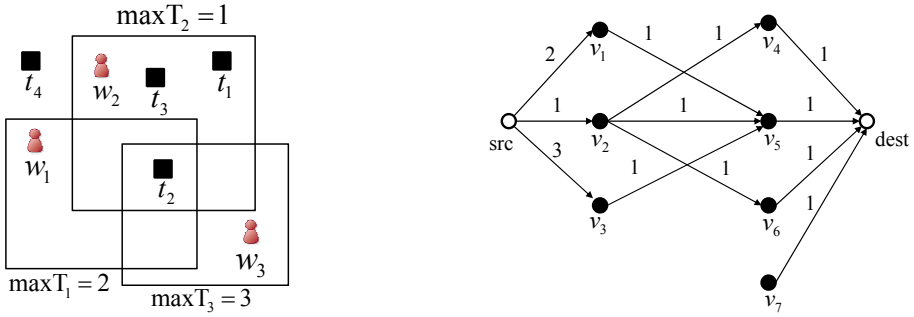
The idea of solving the MTA instance problem is to utilize the spatial constraints of workers to ensure that tasks are properly assigned. Note that without the constraints, a worker might be assigned to a spatial task in a far distance from his location. However, with spatial crowdsourcing, since workers 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 MTA 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. $|W_i|$ is the number of edges connecting the new src vertex to the vertices mapped from W_i, denoted by (src, v_j). We set the capacity of each of these edges, $c(src, v_j) = maxT_j$, 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 only one worker. Finally, 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. We have m edges of this kind and set their 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). Therefore, the vertex mapped from w_1 can transfer flow to only one vertex mapped from the task (i.e., v_5). 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_5, v_6, v_7) 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 $G_i(V, E)$

Fig. 3: An example of the reduction of the MTA 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 MTA instance problem. One of the well-known techniques in computing the maximum flow is the Ford-Fulkerson algorithm [Kleinberg and Tardos 2006]. 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.

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.

5.2.2. Least Location Entropy Priority (LLEP). The problem with the Basic strategy is that at every instance of time, it only tries to maximize the current assignment locally. 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. Our second approach LLEP improve the Basic approach by exploiting the spatial characteristics of the environment during the assignment, one of which is the distribution of the workers in that area. The observation is that a task is more likely to be performed in the future if it is located in an area with higher population of workers. Therefore, the idea is to assign higher priority to tasks that are located in worker-sparse areas, so that those tasks can be assigned before the ones in worker-dense areas. Consequently, there are more tasks to be assigned.

We used *location entropy*, which was first introduced in [Cranshaw et al. 2010] to measure the diversity of unique visitors of a location. Location entropy takes into account the total number of workers in the task's location as well as the relative proportion of their future visits to that location. A location has a high entropy if many workers visit that location with equal proportions[5]. In contrast, a location will have a low entropy if there are only a few workers visit. Thus, our heuristic is to give higher priority to tasks that 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, and O_{wl} be the set of visits that worker w has made to the location l. The probability that a random draw from O_l belongs to O_{wl} is $P_l(w) = \frac{|O_{wl}|}{|O_l|}$, which is the fraction of total visits to l that belongs to worker w. The location entropy for l is computed as follows:

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

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 MTA instance problem by reducing it to the minimum-cost maximum flow problem [Kleinberg and Tardos 2006]. 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 MTA instance problem is reducible to the minimum-cost maximum flow problem.*

PROOF. We prove this theorem 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., $a(u, v_{|W_i|+j}) =$

[5]The concept of visiting the same location is not of primary concern in this paper but for the sake of completeness we assume some sort of a discretization of space (e.g., a grid) and thus a location is represented by a grid cell, which can be visited by several workers.

$Entropy(l_j)$). 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 $Entropy(l_1)$ be the location entropy of the spatial task t_1. Since t_1 is located in the spatial regions of the workers w_2, we set the cost of the edge (v_2, v_4) to $Entropy(l_1)$.

According to the above theorem, our problem becomes 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 [Kleinberg and Tardos 2006] is to first find the maximum flow of the network using Ford-Fulkerson or any other algorithm that 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:

$$min \sum_{(u,v)\in E} f(u,v) \times a(u,v)$$
$$s.t : f(u,v) \leq c(u,v); f(u,v) = -f(v,u)$$
$$\sum_{w\in V} f(u,w) = 0, \forall u \neq src, dst$$
$$\sum_{w\in V} f(src,w) = f_{max}; \sum_{w\in V} f(w,dst) = f_{max}$$

Since all constraints are linear, and our goal is to optimize a linear function, we can solve this in polynomial time by using CPLEX solver, IloCplex [CPLEX 2007]. To summary, 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.

5.2.3. Close Distance Priority (CDP). Both Basic and LLEP approaches try to maximize overall task assignment. However, they did not consider the travel cost (e.g., in time or distance) of the workers during the assignment process. This human travel cost becomes a critical issue since workers need to physically travel to the location of the spatial task in order to perform the task. Thus, we incorporate the travel cost of the workers in the assignment process with an intuitive assumption that tasks that are closer to a worker have smaller travel costs. Our goal is to maximize the task assignment at every time instance while minimizing the travel cost of the workers whenever possible. This means that we still try to maximize the overall task assignment. However, we assign higher priorities to tasks that are closer in spatial distance to the worker.

In our specific problem, we define the travel cost between a worker w and a spatial task t in terms of the Euclidean distance[6] between the two, denoted by $d(w, t)$. Consequently, by computing the distance between every worker and the spatial tasks inside his spatial region, we can associate higher priorities to the closer tasks. Thus, our problem is to assign the maximum number of tasks during every time instance, while

[6]Other metrics such as network distance are also applicable.

the total travel cost of the assignment is the lowest. Consequently, the problem turns into the minimum-cost MTA instance problem. Therefore, a similar solution to that of Section 5.2.2 but with a different cost function (i.e., the travel cost) can be applied to solve this problem.

6. MAXIMUM SCORE ASSIGNMENT

Thus far, we assumed that all tasks are of the same type and all workers have identical qualifications. However, workers may have different expertise. Thus, we aim to explore the task assignment problem considering expertise of the workers. The intuition is that workers are more likely to provide higher quality results when performing a task in their expertise. For example, a chef can be a good candidate to perform the task of rating a particular dish in a restaurant.

6.1. Formal Definitions

We focus on an expertise model, in which each worker has a set of skills. Also, every spatial task has a type with the same value domain as that of the worker expertise. We now redefine spatial task and worker as follows:

Definition 6.1 (*Spatial Task*). A spatial task t of form $<l, q, e, s, \delta>$ is the same as defined in Definition 4.1, except that every spatial task has a task type e.

Definition 6.2 (*Worker*). A worker, denoted by w, is the same as defined in Definition 4.3, except that every worker has a set of skills, denoted by E. Worker w with skill $e_i \in E$ means that the worker has the expertise to perform the task with type e_i.

The Task Inquiry is redefined as follows.

Definition 6.3 (*Task Inquiry*). Task inquiry of a worker is the same as defined in Definition 4.4, except the inquiry also includes his expertise $<e_1, e_2, ...>$.

We now define an expertise match.

Definition 6.4 (*Expertise Match*). An expertise match is an assignment of a worker w to a spatial task t in his spatial region R (i.e., $t \in R$), denoted by $<w, t>$, where the worker w with the skills set E has the expertise to perform the task t with type e (i.e., $e \in E$).

An example of an expertise match can be assigning a photographer to take a picture. The expertise of the photographer is photography and the type of the given task is to take a high-quality picture. Note that every worker can have multiple skills, but every task has one and only one type.

Consider task types, maximizing the number of assigned tasks is no longer our optimization goal. Instead, we introduce the concept of *score* and formally define the *Maximum Score Assignment (MSA)* problem as follows:

Definition 6.5 (*Score*). For every match of form $<w, t>$, we define a score value, $score(w, t)$, which indicates how well the worker w performs the task t based on the worker expertise.

Definition 6.6 (*Maximum Score Assignment (MSA)*). Given a time interval $\phi = \{s_1, s_2, ..., s_n\}$, let S_i be the total score assignment at time instance s_i, i.e., $S_i = \sum_{j=1}^{|I_i|} score_j$. The maximum score assignment problem is the process of assigning tasks to the workers during the time interval ϕ such that the total score $\sum_{i=1}^{n} S_i$ is maximized.

Figure 4a shows a set of workers, represented by circles, along with their constraints and a set of tasks, represented by squares at time instance s_i. There are two types of tasks and two corresponding workers' skills (i.e., each worker has one skill), which are represented by the colors of black and white. Every worker w_j is associated with a spatial region R_j, where $j = 1, 2, 3$. As shown in the figure, worker w_2 can only accept tasks inside his spatial region (i.e., t_1,t_2 and t_3). Moreover, while w_2 is only willing to accept one task (i.e., $maxT_2 = 1$), w_1 and w_3 do not limit the number of acceptable tasks (i.e., $maxT_1 = maxT_3 = \infty$). An expertise match between a task and a worker happens when the task is inside the spatial region of the worker and they have the same color (i.e., $<w_2, t_2>$). With MTA, domain experts define a score for every match. In this particular example, the expertise match score is 3 while the scores of all non-expertise matches are 1. Our optimization goal is to maximize the total score assignment.

6.2. Assignment Protocol

Similar to MTA, a global solution to MSA is not feasible unless we are clairvoyant. Instead, the SC-server tries to optimize the task assignment locally at every instance of time. In the following, we extend three local optimal strategies, Basic, LLEP and CDP to the MSA problem.

6.2.1. Basic Strategy. The basic approach uses local optimization at every time instance rather than global optimization. That is, for each time instance s_i, we have a set of online workers $W_i = \{w_1, w_2, ...\}$ and a set of available tasks $T_i = \{t_1, t_2, ...\}$; the goal is to assign the tasks to the workers such that the total score is maximized. The total score is $S_i = \sum_{j=1}^{|I_i|} score(w_j, t_j)$, where $|I_i|$ is the number of tasks that are assigned at time instance s_i. We refer to this as maximum score assignment (MSA) instance problem. Thus, our goal in this approach is to maximize the overall score assignment by solving the MSA instance problem.

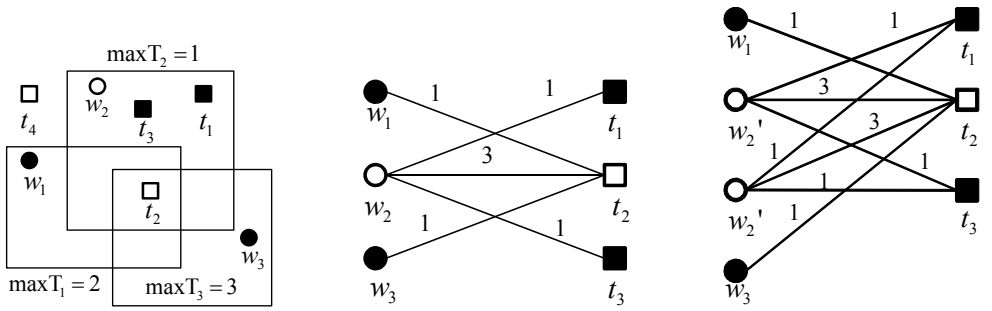
During the task inquiry, every online worker forms two constraints: the spatial region R, and the maximum number of acceptable tasks $maxT$. Using the following theorems, we first solve the MSA instance problem with the assumption of $maxT = 1$ for all workers by reducing it to the *maximum weighted bipartite matching (MWBM)* problem. Thereafter, we solve the problem by relaxing this assumption.

LEMMA 1. *The maximum score assignment instance problem, with $maxT = 1$ for all workers, is reducible to the maximum weighted bipartite matching problem.*

PROOF. We prove the lemma 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 an undirected graph whose vertices can be partitioned as $V = W_i \cup T_i$, where each worker w_j maps to a vertex in W_i and each spatial task t_k maps to a vertex in T_i. There is an edge $e_{j,k} \in E$ connecting a vertex w_j in W_i and another vertex t_k in T_i if t_k is in the spatial region of w_j. G_i is a bipartite graph since every edge $e_{j,k} \in E$ has one end in W_i and the other end in T_i. In addition, $<w_j, t_k>$ is a valid match only if both w_j and t_k appear in at most one edge in E (i.e., $maxT = 1$ for all workers, and every task is to be assigned to at most one worker). We set the weight of every edge $e_{j,k}$ to $score(w_j, t_k)$. Consequently, the MSA instance problem becomes finding the maximum matching in the weighted bipartite graph G_i. □

Figure 4b depicts the bipartite graph G_i of the example in Figure 4a. As shown earlier, worker w_2 can only accept tasks in his spatial region (i.e., $t_1, t_2,$ and t_3). Therefore, there are three edges connecting w_2 to these tasks. Note that the black and white col-

ors represent two types of tasks and the corresponding skills of workers. The following theorem follows from Lemma 1:



(a) An example of worker set W_i and task set T_i

(b) Bipartite graph with $maxT_i = 1\ \forall i$

(c) Transformed bipartite graph, $maxT_2 = 2$ and $maxT_{1,3} = 1$

Fig. 4: An example of the reduction of the MSA instance problem to the maximum weighted bipartite matching problem at instance s_i

THEOREM 3. *The maximum score assignment instance problem is reducible to the maximum weighted bipartite matching problem.*

PROOF. This theorem extends Lemma 1 to any value of $maxT$. The idea is to reduce the problem to the case where $maxT = 1$, then use Lemma 1 to solve the derived problem. For every worker w_j whose $maxT$ is larger than 1, we replace it by N logical workers. These logical workers are identical to the original one (i.e., location, spatial region, expertise), except its $maxT$ is one. Let $|T_j|$ denote the number of tasks in the spatial region of worker w_j, N is computed as follows:

$$N = \begin{cases} maxT & \text{for } 1 < maxT \leq |T_j| \\ |T_j| & \text{otherwise} \end{cases} \tag{2}$$

This equation means that if $maxT$ is infinity (i.e., $maxT$ is not specified), the number of logical workers is equal to the number of tasks in the spatial region of the replaced worker. That is, the worker is willing to perform any number of tasks in his spatial region. Otherwise, the number of logical workers is the smaller number between $maxT$ and $|T_j|$. Using Lemma 1, a corresponding bipartite graph G'_i is constructed on the new set of workers. With this transformation, the original MSA instance problem is equivalent to finding maximum weighted matching in the derived graph G'_i. □

Figure 4 better clarifies this transformation from the example in Figure 4a. Assuming $maxT_2 = 2$, Figure 4c presents the transformed bipartite graph G'_i. It shows that the worker w_2 whose $maxT_2 = 2$ was replaced by two identical workers w'_2 whose $maxT = 1$.

With above reductions, we can now use any algorithm for maximum weighted bipartite matching to solve the MSA instance problem. One of the well-known methods to compute weighted matching is the Hungarian algorithm [Papadimitriou and Steiglitz 1998]. The input of the Hungarian algorithm is a matrix of the costs. Thus, we need to present the bipartite graph G as a matrix of size $|W| \times |T|$, where the workers and the tasks are represented as rows and columns, respectively. Each cost in the matrix is negative score value of each match. Consequently, in order to solve the MSA problem we repeat the MSA instance problem for every instance of time.

With the Basic approach, maximum score assignment at every instance of time does not necessarily result in a globally optimal answer due to its local optimization. To improve the Basic strategy, we now present two spatial heuristics in turn.

6.2.2. Least Location Entropy Priority (LLEP). As mentioned in Section 6.2.2, the problem with the Greedy approach is that at every instance of time, it only tries to maximize the current assignment, without considering future optimizations. Thus, we can use some heuristics to maximize the overall assignments. One of the heuristics is to exploit the spatial characteristics of the environment during the assignment, one of which is the distribution of the workers in that area. Similar to Section 5.2.2, we use location entropy as the indicator of the worker density.

Our goal in this approach is to maximize the total score during every instance of time while the total cost associated to the assigned tasks is the lowest. With the following theorem, we can solve the minimum-cost MSA instance problem by reducing it to the minimum-cost maximum weighted bipartite matching, that is, finding the MWBM of minimum total cost.

THEOREM 4. *The minimum-cost maximum score assignment instance problem is reducible to the minimum-cost maximum weighted bipartite matching problem.*

PROOF. We prove this theorem 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 weighted bipartite graph constructed in the proof of Theorem 3. Note that the weight of every edge $e_{j,k}$ is $score(w_j, t_k)$. We associate to every edge $e_{j,k} \in E$ the cost of t_k, which is the entropy of location l_k (i.e., $Entropy(l_k)$). Thus, by finding the minimum-cost maximum score in graph G'_i, we have assigned the maximum score with the minimum cost. □

In the example of Figure 4c, let $Entropy(l_2)$ be the location entropy of the spatial task t_2. Since t_2 is located in the spatial regions of all workers, we set the cost of all the edges connecting to t_2 to $Entropy(l_2)$.

According to Theorem 4, solving our problem is equivalent to solving minimum-cost MWBM at every time instance. In order to solve this problem, our approach is to first find the maximum matching using any algorithm that computes MWBM (e.g., Hungarian algorithm). Thereafter, the cost of the matching can be minimized by applying an Integer Programming technique (e.g., branch and bound technique).

Let $G = (V, E)$ be an undirected graph constructed in the proof of Theorem 3. Every edge $e_{j,k} \in E$, which connects a worker w_j and a task t_k has a cost $c_{j,k}$. This cost is the location entropy of the task's location l_k, computed by Equation 1. Let $x_{j,k}$ be an indicator variable for each edge $e_{j,k}$ such that

$$x_{j,k} = \begin{cases} 1 & \text{if the edge } e_{j,k} \in M \\ 0 & \text{otherwise} \end{cases}$$

where M is any matching of G. Let f_{max} be the total score of M_{max}. The goal is to minimize the total cost of the matching, which can be formulated as follows:

$$min \sum_{j,k} x_{j,k} c_{j,k}$$

$$s.t : \sum_{j,k} x_{j,k} w_{j,k} = f_{max}; x_{j,k} \in \{0, 1\}$$

$$\forall w_j \in W_i, \sum_{k} x_{j,k} \leq 1; \forall t_k \in T_i, \sum_{j} x_{j,k} \leq 1$$

This is a mixed integer programming problem. Using the branch and bound technique offered by CPLEX [CPLEX 2007], in our particular experiments, we can find the optimal solution to the MSA instance problem with thousands of workers and tasks within a few seconds. However, finding the global optimal solution across many time instances is a much more complex problem as the number of variables is very large (i.e., x_{jkt}, where the subscripts j, k and t represent the dimensions of worker, task and time, respectively). The computation challenge of this problem is beyond the scope of this work.

Finally, our LLEP strategy solves the MSA problem by computing the minimum-cost maximum score problem for every time instance.

6.2.3. Close Distance Priority (CDP). As presented in Section 5.2.3, with the CDP strategy, we incorporate the travel cost of the workers in the assignment process. Our goal is to maximize the total score at every time instance while minimizing the travel cost of the workers whenever possible. We defined the travel cost between a worker w and a spatial task t as the Euclidean distance between the two, denoted by $d(w, t)$. By computing the travel cost of every match $<w, t>$, we can associate to every pair the corresponding cost. Similar to the LLEP approach, our goal in CDP is to maximize the total score during every time instance, while the total cost of the selected edges is the lowest. This means that the matches with lower costs are likely to be selected first. Similarly, the problem turns into the minimum-cost MWBM instance problem, and therefore, a similar solution to that of Section 6.2.2 but with a different cost function can be applied to solve this problem.

We now show that MSA is a generalized MTA with Theorem 5. Assuming the scores of all expertise matches are equal and the scores of all non-expertise matches are equal, the effect of expertise match score is as follows.

THEOREM 5. *Maximum score assignment instance results in maximum expertise matches if the expertise match score is higher than double of the non-expertise match score.*

PROOF. We first assume that $maxT = 1$ for all workers. Let M_{max} be the optimal solution of the maximum score assignment instance problem. If there exists an unassigned expertise match $<w, t>$ in M_{max} whose both ends w and t are not in any other assigned expertise match, to increase the total score we can always replace at most two assigned non-expertise matches by $<w, t>$. On the other hand, if there does not exist an unassigned expertise match in M_{max}, the number of expertise matches in M_{max} is already maximized. Thus, maximum score assignment instance results in maximum expertise matches in the case of $maxT = 1$ for all workers. If $maxT$ is larger than one, we can use Theorem 3 to reduce the problem to the case of $maxT = 1$. □

7. PERFORMANCE EVALUATION

We conducted several experiments on both real-world and synthetic data to evaluate the performance of our proposed approaches: Basic Greedy (GR), LLEP, and CDP. Since the maximum task assignment (MTA) problem is a special case of the maximum score assignment (MSA) problem, we will evaluate MSA that subsumes MTA [7]. Below, we first discuss our experimental methodology. Next, we present our experimental results.

7.1. Experimental Methodology

We performed four sets of experiments. In the first two sets of experiments, we evaluated the scalability of our proposed approaches by varying both the average number of

[7]For MTA experiments please refer to [Kazemi and Shahabi 2012]

W/T	SYN-UNIFORM		SYN-SKEWED	
Sparse	Avg : 1	SD : 1	Avg: 1	SD : 4
Medium	Avg : 3	SD : 1.7	Avg: 3	SD : 8
Dense	Avg : 5	SD : 2.2	Avg: 5	SD : 12

Table I: Distribution of the synthetic data for W/T

workers whose spatial regions contain a given spatial task, namely workers per task (W/T), and the average number of spatial tasks which are inside the spatial region of a given worker, denoted by tasks per worker (T/W). Similar results were observed for varying both T/W and W/T since T/W grows as W/T increases. Hereafter, we present the results for varying W/T only. In the third set of the experiments, we evaluated the impacts of the workers' constraints on the performance of our approaches. Note that every worker has two constraints: R and $maxT$. However, we only evaluated the impact of one of them (i.e., $maxT$) on our approaches, since both constraints have similar effects. Finally, to evaluate the impact of expertise match score, we used three performance measures: 1) the average total score per time instance, 2) the average number of expertise matches per time instance and 3) the average travel cost for a worker to perform a spatial task. The travel cost is measured in terms of the Euclidean distance between the worker and the location of the task. We conducted our experiments on various synthetic (SYN) and two real-world (REAL) data sets. With our synthetic experiments we used two distributions: uniform (SYN-UNIFORM) and skewed (SYN-SKEWED). In the following, we discuss our data sets in more details.

With the first set of synthetic experiments, in order to evaluate the impact of W/T, we considered three cases (Table I), sparse, medium, and dense. The average number of W/T is 1, 3, and 5, respectively (i.e., we increased the size of spatial region R to achieve these numbers). This means that we consider an area to be worker-dense, if the average number of workers who are eligible to perform a single spatial task is 5, whereas in a sparse case, the average number of W/T is 1. In our experiments on SYN-UNIFORM, the average number of W/T varies with a small standard deviation (from 1 to 2.2), whereas in our experiments on SYN-SKEWED, the average number of W/T varies with a large standard deviation (between 4 to 12). With the uniform distribution (SYN-UNIFORM), both workers and tasks are uniformly distributed. Whereas, in order to generate the SYN-SKEWED data set, the workers were formed into four Gaussian clusters (with $\sigma = 0.05$ and randomly chosen centers) while the tasks were uniformly distributed.

Since we do not have access to real-world spatial crowdsourcing data, we used real-world data from other applications, namely Gowalla[8] and Yelp[9], to emulate the spatial crowdsourcing application. Hence, we used the users in these applications as workers, the point of interests as tasks and the check-in (for Gowalla) and review (for Yelp) as completed tasks. A summary of these data sets is given in Table II.

Gowalla is 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 20 days in 2010, including 10,273 spots (e.g., restaurants), covering the state of California. Moreover, we assumed that Gowalla users are the workers of our spatial crowdsourcing system, and checking in a spot is equivalent to accepting a spatial task at that location. 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. Since users may have various check-ins during a day, for every

[8] snap.stanford.edu/data/loc-gowalla.html
[9] yelp.com/dataset_challenge

	Gowalla	Yelp
No. of tasks	20,000	11,537
No. of workers	10,273	43,873
No. of task types	5	508
Grid size	50,000×50,000	10,000×10,000
Avg. No. of skills per worker	1	8
Avg. No. of W/T	4	44

Table II: Statistics of the real data sets, Gowalla and Yelp.

user w, we set $maxT$ as the number of check-ins of the user in that day, and we set R as the minimum bounding rectangle of those checked-in locations. Moreover, the spatial tasks were randomly generated for the given spots in the area. In order to compute the location entropy, we discretized the latitude and longitude space into 50,000×50,000 grid (approximately 30x30 meters per grid cell). For every grid cell, we computed location entropy based on Equation 1. The average number of W/T of Gowalla data set was computed as 4. This also confirms our choices of parameters for the synthetic data sets.

The Yelp data set was captured in the greater Phoenix, Arizona, including locations of 11,537 businesses (e.g., restaurants), 43,873 users and 229,907 reviews. For our experiments, we assumed that the businesses are the spatial tasks, Yelp users are the workers, and reviewing a business is equivalent to accepting a spatial task at its location. To elaborate, we assumed business categories are task types, and skill-sets of each worker are the category lists of the businesses that he reviewed, referred to as a review list. For each worker, we set R as the minimum bounding rectangle of the locations of his review list. Moreover, we set $maxT$ as the size of his review list at one instance of time. With Yelp data, we generated 20 time instances by dividing the review time interval into 20 equal periods. Other parameters are shown in Table II.

Finally, in all of our experiments, we fixed the time interval to 20 days and set the duration of every spatial task to 10 days (i.e., $\phi = 20, \delta = 10$). Furthermore, without loss of generality, unless mentioned otherwise, we set the scores of all expertise matches to 3 and the scores of all non-expertise matches to 1. With the SYN experiments, we fixed the number of new workers and the number of new tasks at each instance time to 500 and 1000, respectively. For all workers, the value of $maxT$ is 20. Furthermore, with SYN and Gowalla data sets, there are 5 values of task types. The type of each task was chosen randomly within these values. Moreover, each worker has one and only one skill, selected randomly from these task types. For each of our experiments, we ran 50 cases, and reported the average of the results. Finally, experiments were run on an Intel(R) Core(TM)i7-2600 CPU @ 3.40 GHz with 8 GB of RAM.

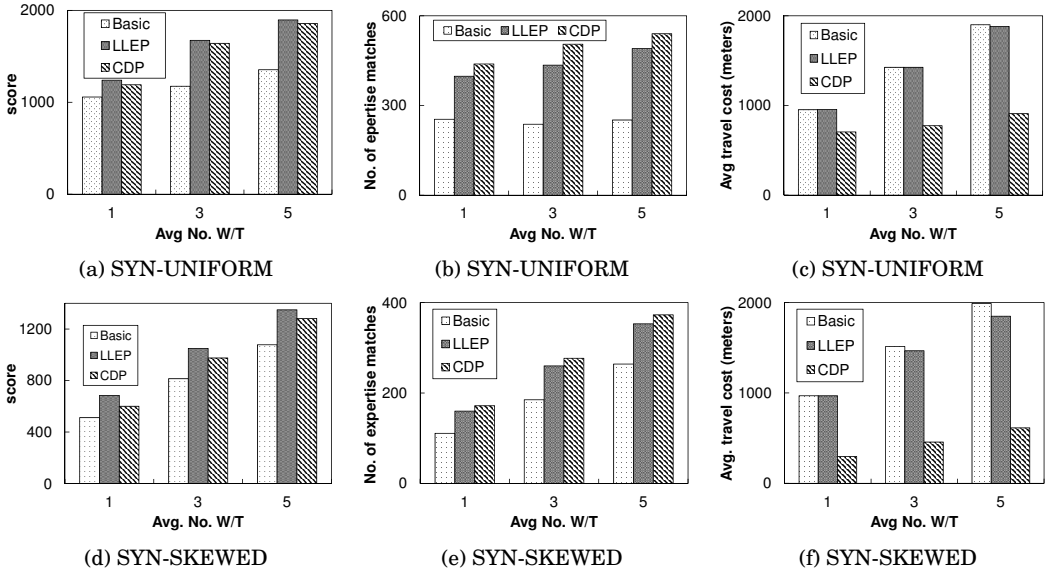
7.2. Experimental Results

7.2.1. Effect of Number of Workers per Task (W/T). In the first set of experiments, we evaluated the performance of our approaches by varying the number of workers whose spatial regions contain a given spatial task. Figures 5a, 5d, 5b and 5e depict the result of our experiments on both SYN-UNIFORM and SYN-SKEWED. The figures show that the total score and the number of expertise matches increase as the number of W/T grows. The reason is that more resources become available to perform tasks. Moreover, Figures 5a and 5d show that LLEP is better than both Basic (GR) and CDP in terms of the average total score (up to 35%) due to applying the location entropy heuristic. However, LLEP does not necessarily maximize the number of expertise matches. Figures 5b and 5e show that CDP is better than the others in terms of the number of expertise matches (up to 50%). The reason is that CDP first maximizes the total score. It then tries to minimize the total travel cost whenever possible, which results in minimizing

Dataset	#Tasks	Basic	LLEP	CDP	Avg. #Assigned tasks	Non-assigned (%)
Gowalla	1000	405	496	459	454	55.6
Yelp	577	469	500	464	478	17.1
SYN-UNIFORM	1000	750	796	753	767	23.3
SYN-SKEWED	1000	290	325	251	289	71.1

Table III: The percentiles of non-assigned tasks per time instance. We measured the average number of assigned tasks over three heuristics, Basic, LLEP and CDP.

the number of assigned tasks (i.e., the more tasks assigned, the more total travel cost). As a result, CDP increases the number of expertise matches.



(a) SYN-UNIFORM (b) SYN-UNIFORM (c) SYN-UNIFORM

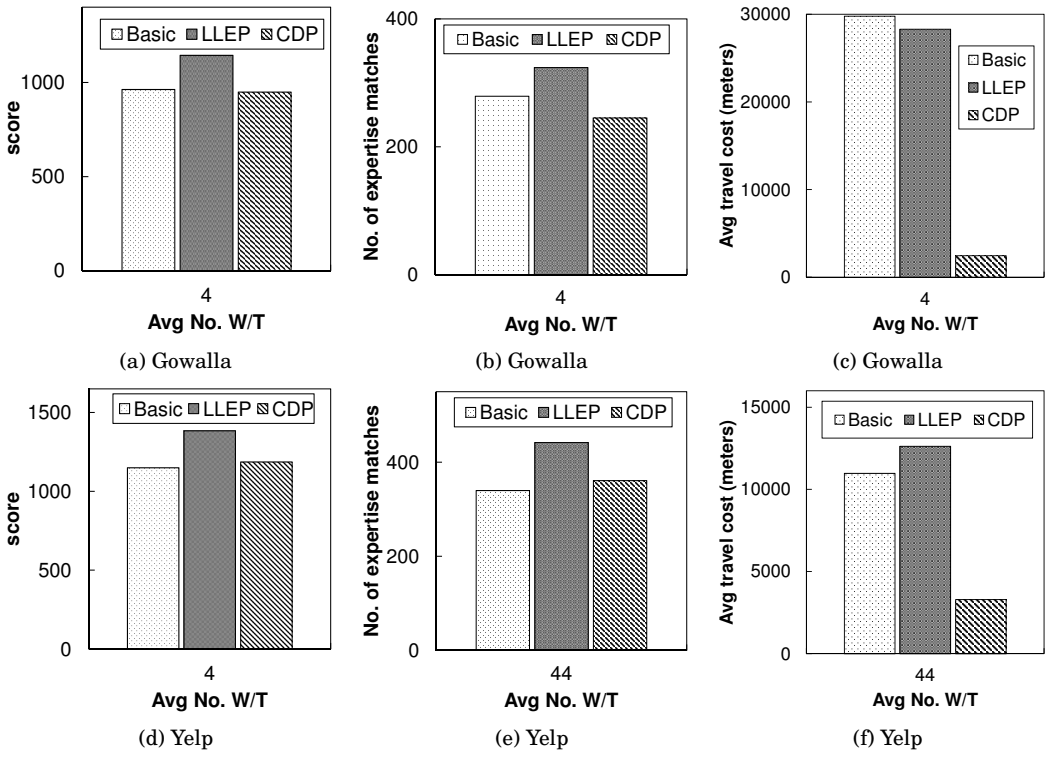
(d) SYN-SKEWED (e) SYN-SKEWED (f) SYN-SKEWED

Fig. 5: Effect of W/T on the synthetic data - per time instance

Figure 6 depicts similar experiments using two real data sets. With Gowalla data, Figures 6a and 6b show that LLEP improves Basic greedy (GR) and CDP significantly both in terms of the total score as well as the number of expertise matches (up to 25%). Similar results are shown on Yelp data. Particularly, Figures 6d and 6e show that LLEP outperforms Basic and CDP in task assignment (up to 25%). The superiority of the LLEP strategy in both cases (e.g., the total score and the number of expertise matches) can be explained that LLEP took much advantage of location entropy of the real check-in data. Table III shows that a large number of tasks (about 56% for Gowalla, 17% for Yelp, 23% for SYN-UNIFORM and 71% for SYN-SKEWED) remained unassigned. This happens due to different reasons such as the constraints of the workers (e.g., the covering area of the workers may overlap with only a small number of tasks) or the expiration of the unassigned tasks.

In the next set of experiments, we measured the impact of varying the number W/T on the average travel cost of the workers in performing a given task in each algorithm. Figures 5c, 5f, 6c and 6f depict the result of our experiments on both SYN and REAL data. As Figures 5c and 5f show, the average travel cost of the workers increases in all cases because as the number of W/T grows, more tasks are assigned, in which some farther tasks can result in high average travel distance. In addition, we observed that

CDP improves the travel cost of the workers significantly as compared with Basic (GR) and LLEP in both SYN and REAL data, which demonstrates the effectiveness of the travel cost heuristic. Particularly, Figures 5c and 5f show 50% and 70% improvements of CDP as compared with the Basic approach in SYN-UNIFORM and SYN-SKEWED, respectively. Moreover, Figures 6c and 6f demonstrate that CDP improves the travel cost of the workers in Gowalla and Yelp data by 90% and 70%, respectively.



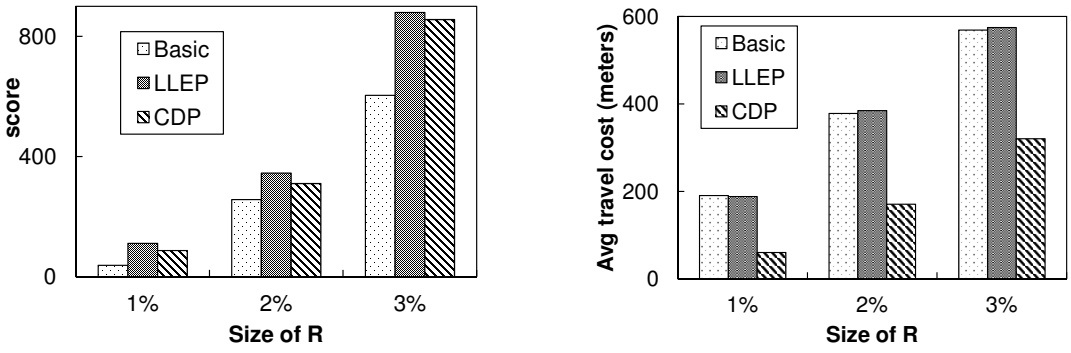
(a) Gowalla (b) Gowalla (c) Gowalla

(d) Yelp (e) Yelp (f) Yelp

Fig. 6: Effect of W/T on the real data - per time instance

7.2.2. Effect of Worker's Constraints. We evaluated the performance of our approaches with respect to expanding the spatial region of every worker R. In addition, we only reported our experiments on SYN-UNIFORM since similar trends were observed for the skewed distribution. As Figure 7a shows, as we vary the size of R from [1%$\times$1%] to [3%$\times$3%] of the entire area, the total score increases. The reason is that larger spatial regions cover larger number of spatial tasks. Consequently, the number of tasks per worker, i.e., T/W, increases, which leads to more tasks being available to be performed by workers. Next, Figure 7b shows the effect of varying R on the travel cost. The figure illustrates that the travel cost increases with the expansion of R. This is because when R grows, farther tasks could be assigned, which leads to the increase in the travel cost.

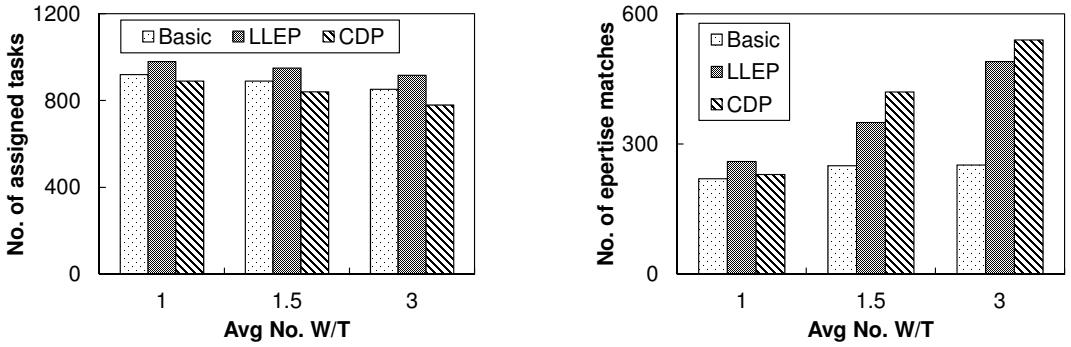
7.2.3. Effect of Expertise Match Score. Next, we evaluated the impact of varying the expertise match score on MSA with the SYN-UNIFORM data. Figures 8a and 8b show that as we vary the expertise match score from 1 to 3, the number of assigned tasks decreases while the number of expertise matches grows. The reason is that when expertise match and non-expertise match scores are equal, MSA reduces to the MTA prob-



(a) Avg. score (b) Avg. travel distance

Fig. 7: Effect of spatial region R - per time instanc e - SYN-UNIFORM

lem, which maximizes the number of assigned tasks. Moreover, as proved in Theorem 5, when the expertise match score is more than two times the score of a non-expertise match, the number of expertise matches is maximized. This means that by varying the expertise match score from 1 to any value that is greater than 2, we can flexibly change our optimization goal from maximizing the assigned tasks to maximizing the expertise matches.



(a) Avg. No. of assigned tasks (b) Avg. No. of expertise matches

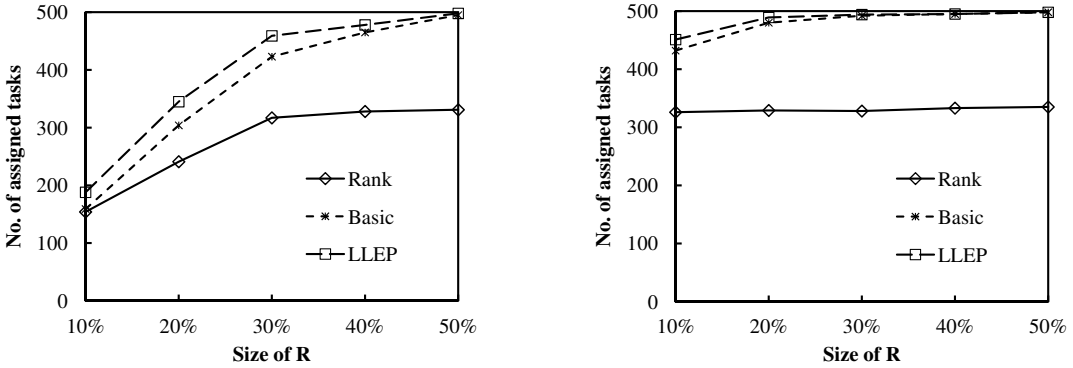
Fig. 8: Effect of score - per time instance - SYN-UNIFORM

To summarize, the main observation from this set of experiments is that LLEP outperforms both Basic and CDP in terms of the overall task assignment (i.e., the total score), while the CDP approach is superior in terms of the travel cost. The non-spatial greedy algorithm (i.e., Basic) achieves optimal assignment per time instance, given the local knowledge up to that time instance. However, a simple heuristic that predicts the spatial distribution of the workers in the future, using their past location entropy, i.e., LLEP, achieves 35% improvement over greedy. Meanwhile, another simple heuristic that gives priority to the distance traveled by the workers, i.e., CDP, achieves 90% superiority over the greedy in terms of travel cost. This shows the importance of considering the spatial aspect of our problems, i.e., MSA, when devising new solutions. Thus, based on the objective of the spatial crowdsourcing application (i.e., maximizing

the assignment or minimizing travel cost), either of the LLEP or CDP approaches can be selected.

7.2.4. Fixing the Worker Set (Online Matching). In the final set of experiments, we compared the performance of Basic and LLEP with the Ranking algorithm to online bipartite matching problem [Karp et al. 1990][10]. In order to compare our heuristics with Ranking, we needed to modify our experimental setup so that the worker set is known in advance (i.e., we fixed the worker set and varied the task set at every time instance). For this comparison, each worker can only perform one task, $maxT = 1$. Other values of $maxT$ can be reducible to the case of $maxT = 1$ as shown in Theorem 3. With the synthetic data sets, the worker set size is 500 while the the task set size is 25 per time instance. With real data sets, we merged 20 worker sets from the previous experiments and kept the task sets the same. This merging results in the worker set sizes of Gowalla and Yelp 25,000 and 10,000, respectively. Other parameters, i.e., the number of time instances and the task duration are the same as before.

Figure 9 presents the results for the synthetic data sets. Figure 9a shows that the number of assigned tasks increases for all algorithms as the sizes of the spatial regions of the workers increase. The reason is that as the spatial regions expand, a task can be performed by many workers. Particularly, LLEP and Basic approaches outperform the Ranking algorithm by up to 60%. Figure 9b shows a similar trend, except a higher number of assigned tasks is observed in SYN-UNIFORM. This difference is because most of the tasks are covered by many workers in SYN-UNIFORM while in SYN-SKEWED, a few tasks are covered by a large number of workers yet the others may not be covered by any worker.



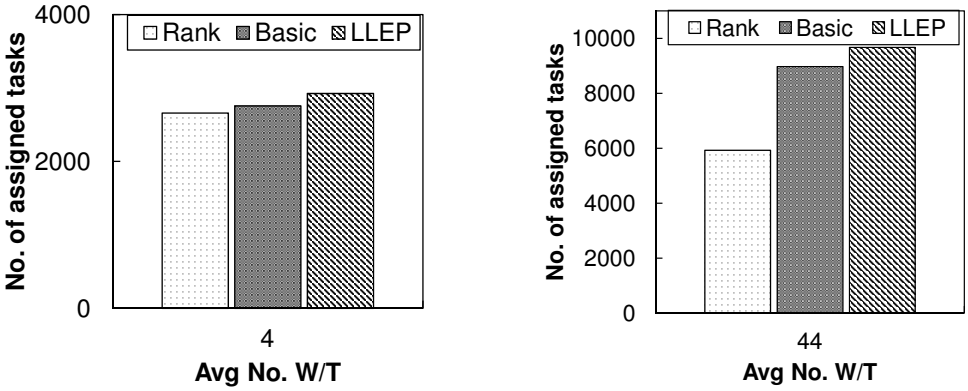
(a) SYN-SKEWED (b) SYN-UNIFORM

Fig. 9: Comparison of our approaches with the Ranking algorithm on the synthetic data sets

Figure 10 depicts the comparison for the real data sets. First, the number of assigned tasks in Gowalla (2500-3000) is smaller than that in Yelp (6000-10,000). The reason is that W/T in Yelp is much higher than that in Gowalla. As a result, one task can be assigned to many workers, which means that the derived bipartite graph in Figure 4b is denser and its search space is larger. Another observation is that the Ranking algorithm performs as good as Basic and LLEP in Gowalla data (Figure 10a) while it does not perform as well in Yelp data (figure 10b) (LLEP improves Rank by 46%).

[10]This randomized solution does not use any spatial knowledge and has a competitive ratio of 0.63.

This result suggests that our heuristics result in much better assignments than the Ranking algorithm in the large search space.



(a) Gowalla (b) Yelp

Fig. 10: Comparison of our approaches with the Ranking algorithm on the real data sets

To summarize, the solutions to online bipartite matching, such as Ranking, are simplistic (e.g., assume the arrival of one item at a time) as the goal is to find a performance bounded by computing competitive ratio, which is different than our objective to devise practical heuristics for a real-world problem.

8. CONCLUSION

In this paper, we introduced spatial crowdsourcing as the process of crowdsourcing a set of spatial tasks to a set of workers. We formally defined the MTA problem and its extension, MSA, which considers worker expertise. Subsequently, we proposed three solutions to MTA and MSA, namely Basic, LLEP, and CDP. In our experiments on both real and synthetic data, we show the superiority of our heuristics LLEP and CDP in maximizing the overall task assignment and minimizing the travel cost, respectively, when compared to the Basic approach. We also compared our approaches with the Ranking algorithm for solving a variant of online bipartite matching problem where each task has a deadline. Experimental results show that our solutions outperform Ranking in maximizing the number of assigned tasks.

The main contribution of the paper is on the formulation of the MTA and MSA problems, their formalization and the heuristics proposed to solve them. The main lesson learned is that a greedy algorithm that achieves optimal assignment per time-instance without any spatial consideration, i.e., Basic, is not the best approach to solve the spatial crowdsourcing problem. Instead, a heuristic that predicts the spatial distribution of the workers in future, using their past location entropy (i.e., LLEP), or a heuristic that gives priority to the distances traveled by the workers, i.e., CDP, outperforms the non-spatial greedy approach.

ACKNOWLEDGMENTS

This research has been funded in part by Award No. 2011-IJCX-K054 from National Institute of Justice, Office of Justice Programs, U.S. Department of Justice, as well as NSF grant IIS-1320149, the USC Integrated Media Systems Center (IMSC), and unrestricted cash gifts from Google and Northrop Grumman. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and

do not necessarily reflect the views of any of the sponsors such as the National Science Foundation or the Department of Justice.

REFERENCES

Field Agent. 2010. (2010). http://www.fieldagent.net/

Omar Alonso, Daniel E Rose, and Benjamin Stewart. 2008. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, Vol. 42. ACM, 9–15.

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.

Alessandro Bozzon, Marco Brambilla, and Stefano Ceri. 2012. Answering search queries with crowdsearcher. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 1009–1018.

Muhammed Fatih Bulut, Yavuz Selim Yilmaz, and Murat Demirbas. 2011. Crowdsourcing location-based queries. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. IEEE, 513–518.

Kuan-Ta Chen, Chen-Chi Wu, Yu-Chun Chang, and Chin-Laung Lei. 2009. A crowdsourceable QoE evaluation framework for multimedia content. In *Proceedings of the 17th ACM international conference on Multimedia*. ACM, 491–500.

Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, and Nikos Triandopoulos. 2008. Anonysense: privacy-aware people-centric sensing. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 211–224.

ILOG CPLEX. 2007. 11.0 Users Manual. *ILOG SA, Gentilly, France* (2007).

Justin Cranshaw, Eran Toch, Jason Hong, Aniket Kittur, and Norman Sadeh. 2010. Bridging the gap between physical location and online social networks. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*. ACM, 119–128.

Crowdflower. 2009. (2009). http://www.crowdflower.com/

Hung Dang, Tuan Nguyen, Hien To, and Cyrus. 2013. Maximum Complex Task Assignment: Towards Tasks Correlation in Spatial Crowdsourcing. (2013).

Gianluca Demartini, Beth Trushkowsky, Tim Kraska, and Michael J Franklin. 2013. CrowdQ: Crowdsourced Query Understanding.. In *CIDR*.

Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. CrowdDB: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 61–72.

Gigwalk. 2010. (2010). http://gigwalk.com/

Ido Guy, Adam Perer, Tal Daniel, Ohad Greenshpan, and Itai Turbahn. 2011. Guess who?: enriching the social graph through a crowdsourcing game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1373–1382.

Seth Hettich and Michael J Pazzani. 2006. Mining for proposal reviewers: lessons learned at the national science foundation. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 862–871.

Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. 2006. CarTel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, 125–138.

Bala Kalyanadundaram and Kirk R. Pruhs. 1993. Online Weighted Matching. *Journal of Algorithms* (1993), 478–488. Issue 3.

Bala Kalyanasundaram and Kirk R Pruhs. 2000. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science* 233, 1 (2000), 319–325.

Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. 1990. An Optimal Algorithm for On-line Bipartite Matching. 352–258.

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

Leyla Kazemi and Cyrus Shahabi. 2012. GeoCrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 189–198.

Leyla Kazemi, Cyrus Shahabi, and Lei Chen. 2013. GeoTruCrowd: Trustworthy Query Answering with Spatial Crowdsourcing. (2013).

Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. 1994. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science* (1994), 255–267. Issue 2.

Aniket Kittur, Jeffrey V Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. 2013. The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 1301–1318.

Jon Kleinberg and Eva Tardos. 2006. *Algorithm design*. Pearson Education India.

Eugene L Lawler, Jan Karel Lenstra, AHG Rinnooy Kan, and David B Shmoys. 1985. *The traveling salesman problem: a guided tour of combinatorial optimization*. Vol. 3. Wiley New York.

Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang. 2012. Cdas: a crowdsourcing data analytics system. *Proceedings of the VLDB Endowment* 5, 10 (2012), 1040–1051.

Google map maker. 2008. (2008). http://www.google.com/mapmaker/

Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. 2011. Human-powered sorts and joins. *Proceedings of the VLDB Endowment* 5, 1 (2011), 13–24.

Amazon mechanical turk. 2005. (2005). http://www.mturk.com/

Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. 2007. Adwords and generalized online matching. *Journal of the ACM (JACM)* 54, 5 (2007), 22.

David Mimno and Andrew McCallum. 2007. Expertise modeling for matching papers with reviewers. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 500–509.

Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. 2008. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 323–336.

Mohamed Musthag and Deepak Ganesan. 2013. Labor dynamics in a mobile micro-task market. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 641–650.

oDesk. 2005. (2005). https://www.odesk.com/

Openstreetmap. 2004. (2004). http://www.openstreetmap.org/

Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications.

Aditya Parameswaran, Anish Das Sarma, Hector Garcia-Molina, Neoklis Polyzotis, and Jennifer Widom. 2011. Human-assisted graph search: it's okay to ask questions. *Proceedings of the VLDB Endowment* 4, 5 (2011), 267–278.

Aditya G Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. 2012. Crowdscreen: Algorithms for filtering data with humans. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 361–372.

Alexander J Quinn and Benjamin B Bederson. 2011. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1403–1412.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. 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*. Association for Computational Linguistics, 254–263.

Alexander Sorokin and David Forsyth. 2008. Utility data annotation with amazon mechanical turk. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE, 1–8.

Yong-Hong Sun, Jian Ma, Zhi-Ping Fan, and Jun Wang. 2008. A hybrid knowledge and model approach for reviewer assignment. *Expert Systems with Applications* 34, 2 (2008), 817–824.

Wenbin Tang, Jie Tang, Tao Lei, Chenhao Tan, Bo Gao, and Tian Li. 2012. On optimization of expertise matching with various constraints. *Neurocomputing* 76, 1 (2012), 71–83.

Hien To, Gabriel Ghinita, and Cyrus Shahabi. 2014. A Framework for Protecting Worker Location Privacy in Spatial Crowdsourcing. *Proceedings of the VLDB Endowment* 7, 10 (2014).

Paolo Toth and Daniele Vigo. 2001. *The vehicle routing problem*. Siam.

UCB. 2008. (2008). http://traffic.berkeley.edu/

Kasturi R Varadarajan. 1998. A divide-and-conquer algorithm for min-cost perfect matching in the plane. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*. IEEE, 320–329.

Luis Von Ahn and Laura Dabbish. 2008. Designing games with a purpose. *Commun. ACM* 51, 8 (2008), 58–67.

Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1483–1494.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*. 2035–2043.

Wikimapia. 2006. (2006). http://wikimapia.org/

Raymond Chi-Wing Wong, Yufei Tao, Ada Wai-Chee Fu, and Xiaokui Xiao. 2007. On efficient spatial matching. In *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 579–590.

Tingxin Yan, Vikas Kumar, and Deepak Ganesan. 2010. 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*. ACM, 77–90.

Man Lung Yiu, Kyriakos Mouratidis, Nikos Mamoulis, and others. 2008. Capacity constrained assignment in spatial databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 15–28.

Zhou Zhao, Wilfred Ng, and Zhijun Zhang. 2013. CrowdSeed: query processing on microblogs. In *Proceedings of the 16th International Conference on Extending Database Technology*. ACM, 729–732.