Reliable Diversity-Based Spatial Crowdsourcing by Moving Workers

Peng Cheng [#], Xiang Lian [*], Zhao Chen [#], Rui Fu [#], Lei Chen [#], Jinsong Han [†], Jizhong Zhao [†]

[#] *Hong Kong University of Science and Technology, Hong Kong, China*
{pchengaa, zchenah, leichen}@cse.ust.hk, rfu@ust.hk

[*] *University of Texas Rio Grande Valley, Texas, USA*
xiang.lian@utrgv.edu

[†] *Xi'an Jiaotong University, Shaanxi, China*
{hanjinsong, zjz}@mail.xjtu.edu.cn

ABSTRACT

With the rapid development of mobile devices and the crowdsourcing platforms, the spatial crowdsourcing has attracted much attention from the database community, specifically, spatial crowdsourcing refers to sending a location-based request to workers according to their positions. In this paper, we consider an important spatial crowdsourcing problem, namely *reliable diversity-based spatial crowdsourcing* (RDB-SC), in which spatial tasks (such as taking videos/photos of a landmark or firework shows, and checking whether or not parking spaces are available) are time-constrained, and workers are moving towards some directions. Our RDB-SC problem is to assign workers to spatial tasks such that the completion reliability and the spatial/temporal diversities of spatial tasks are maximized. We prove that the RDB-SC problem is NP-hard and intractable. Thus, we propose three effective approximation approaches, including greedy, sampling, and divide-and-conquer algorithms. In order to improve the efficiency, we also design an effective cost-model-based index, which can dynamically maintain moving workers and spatial tasks with low cost, and efficiently facilitate the retrieval of RDB-SC answers. Through extensive experiments, we demonstrate the efficiency and effectiveness of our proposed approaches over both real and synthetic datasets.

1. INTRODUCTION

Recently, with the ubiquity of smart mobile devices and high-speed wireless networks, people can now easily work as moving sensors to conduct sensing tasks, such as taking photos and recording audios/videos. While data submitted by mobile users often contain spatial-temporal-related information, such as real-world scenes (e.g., street view of Google Maps [1]), video clips (e.g., MediaQ [2]), local hotspots (e.g., Foursquare [3]), and traffic conditions (e.g., Waze [4]), the *spatial crowdsourcing* platform [17, 19] has nowadays drawn much attention from both academia (e.g., the database community) and industry (e.g., Amazon's AMT [5]).

Proceedings of the VLDB Endowment, Vol. 8, No. 10

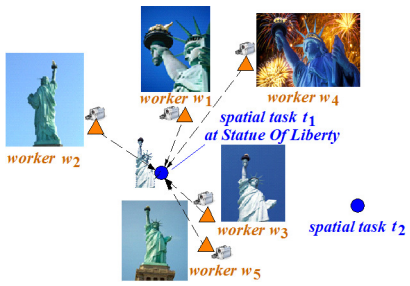


Figure 1: An Example of Taking Photos/Videos of a Landmark (Statue of Liberty) in the Spatial Crowdsourcing System.

Specifically, a spatial crowdsourcing platform [17, 19] is in charge of assigning a number of workers to nearby *spatial tasks*, such that workers need to physically move towards some specified locations to finish tasks (e.g., taking photos/videos).

Example 1 (Taking Photos/Videos of a Landmark). *Consider a scenario of the spatial crowdsourcing in Figure 1, in which there are two spatial tasks at locations t_1 and t_2, and 5 workers, $w_1 \sim w_5$. In particular, the spatial task t_1 is for workers to take 2D photos/videos of a landmark, the Statue of Liberty, while walking from ones' current locations towards it. The resulting 2D photos/videos are useful for real applications such as virtual tours and 3D landmark reconstruction. Therefore, the task requester is usually interested in obtaining a full view of the landmark from diverse directions (e.g., photos from the back of the statue that not many people saw before).*

As shown in Figure 1, workers w_1 and w_4 can take photos from left hand side of the statue, worker w_2 can take photos from the back of the statue, and workers w_3 and w_5 can take photos from the front of the statue. Since photos/videos from similar directions are not informative for the 3D reconstruction or virtual tours, the spatial crowdsourcing system needs to select those workers who can take photos of the statue with as diverse directions as possible, and then assign task t_1 to them.

Note that, when worker w_4 reaches the location of t_1, he/she can take a photo of the landmark at night with fireworks. As a result, by assigning task t_1 to worker w_4, we can obtain a quite diverse photo at night for virtual tours, compared with that taken by worker w_1 in the daytime (even though they are taken from similar angles). Thus, it is also important to consider the temporal diversity of taking the photos, in terms of the arrival times of workers at t_1. ■

Example 2 (Available Parking Space Monitoring over a Region). *In the application of monitoring free parking spaces in a spatial region, it is important to analyze the photos taken from di-*

verse directions and at different time periods of the day, and predict the trend of available parking spaces in the future. This is because some available parking spaces might be hidden by other cars for photos just from one single direction (or multiple similar directions), and moreover, photos taken at different timestamps have richer information than those taken at a single time point, for the purpose of predicting the availability of parking spaces. Therefore, in this case, the spatial crowdsourcing system needs to assign such a task to those workers with diverse walking directions to it and arrival times. ■

In this paper, we will investigate a realistic scenario of spatial crowdsourcing, where workers are dynamically moving towards some directions, and spatial tasks are constrained by valid time periods. For example, a worker may want to do spatial tasks on the way home, and thus he/she tends to accept tasks only along the direction to home, rather than an opposite direction. Similarly, a spatial task of taking photos of the statue, together with fireworks, is restricted by the period of the firework show time. Therefore, a worker can only conduct a task within the constrained time range and prefer to accepting the tasks close to his/her moving direction. In this paper, we characterize features of moving workers and time-constrained spatial tasks, which have not been studied before.

Under the aforementioned realistic scenario, we propose the problem of dynamic task-and-worker assignment, by considering the answer quality of spatial tasks, in terms of two measures: *spatial-temporal diversities* and *reliability*. In particular, inspired by the two applications above (Examples 1 and 2), for spatial tasks, photos/videos from diverse angles or timestamps can provide a comprehensive view of the landmark, which is more preferable than those taken from a single boring direction/time; similarly, photos taken by different directions and periods are more useful for the trend prediction of available parking spaces. Therefore, in this paper, we will introduce the concepts of *spatial diversity* and *temporal diversity* to spatial crowdsourcing, which capture the diversity quality of the returned answers (e.g., photos taken from different angles and at different timestamps) to spatial tasks.

Furthermore, in reality, it is possible that answers provided by workers are not always correct, for example, the uploaded photos/videos might be fake ones, or workers may deny the assigned tasks. Thus, we will model the confidence of each worker, and in turn, guarantee high *reliability* of each spatial task, which is defined as the confidence that at least one worker assigned to this task can give a high quality answer.

Note that, while existing works on spatial crowdsourcing [17, 19] focused on the assignment of workers and tasks to maximize the total number of completed tasks, they did not consider much about the constrained features of workers/tasks (workers' moving directions and tasks' time constraints). Most importantly, they did not take into account the quality of the returned answers.

By considering both quality measures of spatial-temporal diversities and reliability, in this paper, we will formalize the problem of *reliable diversity-based spatial crowdsourcing* (RDB-SC), which aims to assign moving workers to time-constrained spatial tasks such that both reliability and diversity are maximized. To the best of our knowledge, there are no previous works that study reliability and spatial/temporal diversities in the spatial crowdsourcing. However, efficient processing of the RDB-SC problem is quite challenging. In particular, we will prove that the RDB-SC problem is NP-hard, and thus intractable. Therefore, we propose three approximation approaches, that is, the greedy, sampling, and divide-and-conquer algorithms, in order to efficiently tackle the RDB-SC problem. Furthermore, to improve the time efficiency, we design a cost-model-based index to dynamically maintain moving workers and time-constrained spatial tasks, and efficiently facilitate the dynamic assignment in the spatial crowdsourcing system. Finally, through extensive experiments, we demonstrate the efficiency and effectiveness of our approaches.

To summarize, we make the following contributions.

- We formally propose the problem of reliable diversity-based spatial crowdsourcing (RDB-SC) in Section 2, by introducing the reliability and diversity to guarantee the quality of spatial tasks.
- We prove that the RDB-SC problem is NP-hard, and thus intractable in Section 3.
- We propose three approximation approaches, greedy, sampling, and divide-and-conquer algorithms, in Sections 4, 5 and 6, respectively, to tackle the RDB-SC problem.
- We conduct extensive experiments in Section 8 on both real and synthetic datasets and show the efficiency and effectiveness of our approaches.

We design a cost-model-based index structure to dynamically maintain workers and tasks in Section 7. Section 9 overviews previous works on (spatial) crowdsourcing. Finally, Section 10 concludes this paper.

2. PROBLEM DEFINITION

2.1 Time-Constrained Spatial Tasks

We first define the time-constrained spatial tasks in the crowdsourcing applications.

Definition 1. (*Time-Constrained Spatial Tasks*) *Let* $T = \{t_1, t_2, ..., t_m\}$ *be a set of* m *time-constrained spatial tasks. Each spatial task* t_i *(*$1 \leq i \leq m$*) is located at a specific location* l_i*, and associated with a valid time period* $[s_i, e_i]$. ■

In this paper, we consider spatial and time-constrained tasks, such as "taking 2D photos/videos for the Statue of Liberty together with fireworks", or "taking photos of parking places during open hours of the parking area in a region". Therefore, in such scenarios, each task can only be accomplished at a specific location, and, moreover, satisfy the time constraint. For example, photos should be taken by people in person and within the period of the firework show. Therefore, in Definition 1, we require each spatial task t_i be accomplished at a spatial location l_i (for $1 \leq i \leq m$), and within a valid period $[s_i, e_i]$.

The set of spatial tasks is dynamically changing. That is, the newly created tasks keep on arriving, and those completed (or expired) tasks are removed from the crowdsourcing system.

2.2 Dynamically Moving Workers

Next, we consider dynamically moving workers.

Definition 2. (*Dynamically Moving Workers*) *Let* $W = \{w_1, w_2, ..., w_n\}$ *be a set of* n *workers. Each worker* w_j *(*$1 \leq j \leq n$*) is currently located at position* l_j*, moving with velocity* v_j*, and towards the direction with angle* $\alpha_j \in [\alpha_j^-, \alpha_j^+]$*. Each worker* w_j *is associated with a confidence* $p_j \in [0, 1]$*, which indicates the reliability of the worker that can do the task.* ■

Intuitively, a worker w_j ($1 \leq j \leq n$) may want to do tasks on the way to some place during the trip. Thus, as mentioned in Definition 2, the worker can pre-register the angle range, $[\alpha_j^-, \alpha_j^+]$, of one's current moving direction. In other words, the worker is only willing to accomplish tasks that do not deviate from his/her moving direction significantly. For other (inconvenient) tasks (e.g., opposite to the moving direction), the worker tends to ignore/reject the task request, thus, the system would not assign such tasks to this worker. In the case that the worker has no targeting destinations (i.e., free to move), he/she can set $[\alpha_j^-, \alpha_j^+]$ to $[0, 2\pi]$.

After being assigned with a spatial task, a worker sometimes may not be able to finish the task. For example, the worker might reject the task request (e.g., due to other tasks with higher prices), do the task incorrectly (e.g., taking a wrong photo), or miss the deadline of the task. Thus, as given in Definition 2, each worker w_j is associated with a confidence $p_j \in [0, 1]$, which is the probability (or reliability) that w_j can successfully finish a task (inferred from historical data of this worker). In this paper, we consider the model of server assigned tasks (SAT) [19]. That is, we assume that once a worker accepts the assigned task, the worker will voluntarily do the task. Similar to spatial tasks, workers can freely register or leave the crowdsourcing system. Thus, the set of workers is also dynamically changing.

2.3 Reliable Diversity-Based Spatial Crowdsourcing

With the definitions of tasks and workers, we are now ready to formalize our spatial crowdsourcing problem (namely, RDB-SC), which assigns dynamically moving workers to time-constrained spatial tasks with high accuracy and quality.

Before we provide the formal problem definition, we first quantify the criteria of our task assignment during the crowdsourcing, in terms of two measures, the *reliability* and *spatial/temporal diversity*. The reliability indicates the confidence that at least some worker can successfully complete the task, whereas the spatial/temporal diversity reflects the quality of the task accomplishment by a group of workers, in both spatial and temporal dimensions (e.g., taking photos from diverse angles and at diverse timestamps).

Reliability. Since not all workers are trustable, we should consider the reliability, p_j, of each workers, w_j, during the task assignment. For example, some workers might take a wrong photo, or fail to reach the task location before the valid period. In such cases, the goal of our task assignment is to guarantee that tasks t_i can be accomplished by those assigned workers with high confidence.

Definition 3. (*Reliability*) *Given a spatial task t_i and its assigned set, W_i, of workers, the reliability, $rel(t_i, W_i)$, of a worker assignment w.r.t. t_i is given by:*

$$rel(t_i, W_i) = 1 - \prod_{\forall w_j \in W_i} (1 - p_j). \quad (1)$$

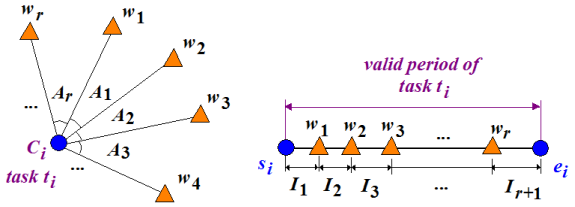
where p_j is the probability that worker w_j can reliably complete task t_i. ■

Intuitively, Eq. (1) gives the probability (reliability) that there exists some worker who can accomplish the task t_i reliably (e.g., taking the proper photo, or providing a reliable answer). In particular, the second term (i.e., $\prod_{\forall w_j \in W_i}(1 - p_j)$) in Eq. (1) is the probability that all the assigned workers in W_i cannot finish the task t_i. Thus, $1 - \prod_{\forall w_j \in W_i}(1 - p_j)$ is the probability that task t_i can be completed by at least one assigned worker in W_i. High reliability usually leads to good confidence of the task completion. In this paper, we aim to maximize the reliability for each task.

Possible Worlds of the Task Completion. Since not all the assigned workers in W_i can complete the task t_i, it is possible that only a subset of workers in W_i can succeed in accomplishing the task t_i in the real world. In practice, there are an exponential number (i.e., $O(2^{|W_i|})$) of such possible subsets.

Following the literature of probabilistic databases [16], we call each possible subset in W_i a *possible world*, denoted as $pw(W_i)$, which contains those workers who may finish the task t_i in reality. Each possible world, $pw(W_i)$, is associated with a probability confidence,

$$Pr\{pw(W_i)\} = \prod_{\forall w_j \in pw(W_i)} p_j \cdot \prod_{\forall w_j \in (W_i - pw(W_i))} (1 - p_j), \quad (2)$$



(a) Spatial Diversity (b) Temporal Diversity

Figure 2: Illustration of Spatial/Temporal Diversity.

which is given by multiplying probabilities that workers in W_i appear or do not appear in the possible world $pw(W_i)$.

Spatial/Temporal Diversity. As mentioned in Example 1 of Section 1 (i.e., take photos of a statue), it would be nice to obtain photos from different angles and at diverse times of the day, and get the full picture of the statue for virtual tours. Similarly, in Example 2 (Section 1), it is desirable to obtain photos of parking areas from different directions at diverse times, in order to collect/analyze data of available parking spaces during open hours. Thus, we want workers to accomplish spatial tasks (e.g., taking photos) from different angles and over timestamps as diverse as possible. We quantify the quality of the task completion by *spatial/temporal diversities*.

Specifically, the *spatial diversity* (SD) is defined as follows. As illustrated in Figure 2(a), let C_i be a point at the location l_i (or the centroid of a region) of task t_i. Assume that r workers $w_j \in W_i$ ($1 \le j \le r$) do tasks (i.e., take pictures) at l_i from different angles. We draw r rays from C_i to the directions of r workers who take photos. Then, with these r rays, we can obtain r angles, denoted as A_1, A_2, ..., and A_r, where $\sum_{j=1}^{r} A_j = 2\pi$. Intuitively, the entropy was used as an expression of the disorder, or randomness of a system. Here, higher diversity indicates more discorder, which can be exactly captured by the entropy. That is, when the answers come from diverse angles and timestamps, the answers have high entropy. Thus, we use the entropy to define the spatial diversity (SD) as follows:

$$SD(t_i) = -\sum_{j=1}^{r} \frac{A_j}{2\pi} \cdot log\left(\frac{A_j}{2\pi}\right) \quad (3)$$

Similarly, we can give the *temporal diversity* for the arrival times of workers (to do tasks), by using the entropy of time intervals. As shown in Figure 2(b), assume that the arrival times of r workers divide the valid period $[s_i, e_i]$ of task t_i into $(r + 1)$ sub-intervals of lengths I_1, I_2, ..., and I_{r+1}. We define the temporal diversity (TD) below:

$$TD(t_i) = -\sum_{j=1}^{r+1} \frac{I_j}{e_i - s_i} \cdot log\left(\frac{I_j}{e_i - s_i}\right) \quad (4)$$

Intuitively, larger SD or TD value indicates higher diversity of spatial angles or time distribution, which is more desirable by task requesters. We combine these 2 diversity types, and obtain the spatial/temporal diversity (STD) w.r.t. W_i, below:

$$STD(t_i, W_i) = \beta \cdot SD(t_i) + (1 - \beta) \cdot TD(t_i), \quad (5)$$

where parameter $\beta \in [0, 1]$. Here, β is a weight balancing between SD and TD, which depends on the application requirement specified by the task requester of t_i. When $\beta = 0$, we consider TD only; when $\beta = 1$, we require SD only in the spatial task t_i.

Therefore, under possible worlds semantics, in this paper, we will consider the *expected spatial/temporal diversity* (defined later) in our crowdsourcing problem.

The RDB-SC Problem. We define our RDB-SC problem below.

Definition 4. (*Reliable Diversity-Based Spatial Crowdsourcing, RDB-SC*) *Given m time-constrained spatial tasks in T, and n dynamically moving workers in W, the problem of reliable diversity-based spatial crowdsourcing (RDB-SC) is to assign each task $t_i \in T$ with a set, W_i, of workers $w_j \in W$, such that:*

1. *each worker $w_j \in W$ is assigned with a spatial task $t_i \in T$ such that his/her arrival time at location l_i falls into the valid period $[s_i, e_i]$,*

2. *the minimum reliability,* $\min_{i=1}^{m} rel(t_i, W_i)$*, of all tasks* t_i *is maximized, and*
3. *the summation,* $total\_STD$*, of the expected spatial/temporal diversities,* $E(STD(t_i))$*, for all tasks* t_i*, is maximized,*

where the expected spatial/temporal diversity $E(STD(t_i))$ *is:*

$$E(STD(t_i)) = \sum_{\forall pw(W_i)} Pr\{pw(W_i)\} \cdot STD(t_i, pw(W_i)), \text{ and} \quad (6)$$

$$total\_STD = \sum_{i=1}^{m} E(STD(t_i)). \quad (7)$$

The RDB-SC problem is to assign workers to collaboratively accomplish each task with two optimization goals: (1) the smallest reliability among all tasks is maximized (intuitively, if the smallest reliability of tasks is maximized, then the reliability of all tasks must be high), and (2) the summed expected diversity for all tasks is maximized. These two goals aim to guarantee the confidence of the task completion and the diversity quality of the tasks, respectively.

Answer Aggregation for a Spatial Task. After assigning a set, W_i, of workers to spatial task t_i, we can obtain a set of answers, for example, photos in Example 1 of Section 1 with different angles and at diverse timestamps. It is also important to present these photos to the task requester. Due to too many photos, we can apply aggregation techniques to group those photos with similar spatial/temporal diversities, and show the task requester only one representative photo from each group. Moreover, since we consider taking photos as tasks, the task requester can choose the photos with high quality (e.g., resolution or sharpness) from all the answers if he/she wants to. This is, however, not the focus of this paper, and we would like to leave it as future work.

2.4 Challenges

According to Definition 4, the RDB-SC problem is an optimization problem with two objectives. The challenges of tackling the RDB-SC problem are threefold. First, with m time-constrained spatial tasks and n moving workers, in the worst case, there are an exponential number of possible task-worker assignment strategies, that is, with time complexity $O(m^n)$. In fact, we will later prove that the RDB-SC problem is NP-hard. Thus, it is inefficient or even infeasible to enumerate all possible assignment strategies.

Second, the second objective in the RDB-SC problem considers maximizing the summed expected spatial/temporal diversities, which involves an exponential number of possible worlds for diversity computations. That is, the time complexity of enumerating possible worlds is $O(2^{|W_i|})$, where W_i is a set of assigned workers to task t_i. Therefore, it is not efficient to compute the spatial/temporal diversity by taking into account all possible worlds.

Third, in the RDB-SC problem, workers move towards some directions, whereas spatial tasks are restricted by time constraints (i.e., valid period $[s_i, e_i]$). Both workers and tasks can enter/quit the spatial crowdsourcing system dynamically. Thus, it is also challenging to dynamically decide the task-worker assignment.

Inspired by the challenges above, in this paper, we first prove that the RDB-SC problem is NP-hard, and design three efficient approximation algorithms, which are based on greedy, sampling, and divide-and-conquer approaches. To tackle the second challenge, we reduce the computation of the expected diversity under possible worlds semantics to the one with cubic cost, which can greatly improve the problem efficiency. Finally, to handle the cases that workers and tasks can dynamically join and leave the system freely, we design effective grid index to enable dynamic updates of worker/tasks, as well as the retrieval of assignment pairs.

3. PROBLEM REDUCTION

Table 1: Symbols and descriptions.

Symbol	Description
T	a set of m time-constrained spatial tasks t_i
W	a set of n dynamically moving workers w_j
$[s_i, e_i]$	the time constraint of accomplishing a task t_i
l_i (or l_j)	the position of task t_i (or worker w_j)
v_j	the velocity of moving worker w_j
p_j	the confidence of worker w_j that properly do the task
$[\alpha_j^-, \alpha_j^+]$	the interval of moving direction (angle)
$rel(t_i, W_i)$	the reliability that task t_i can be completed by workers in W_i
$R(t_i, W_i)$	a (equivalent) variant of reliability $rel(t_i, W_i)$
$STD(t_i, W_i)$	the spatial/temporal diversity of task t_i
$E(STD(t_i))$	the expected spatial/temporal diversity of task t_i
$total\_STD$	the sum of the expected spatial/temporal diversities for all tasks
K	the sample size

3.1 Reduction of Reliability

As mentioned in Definition 4, the first optimization goal of the RDB-SC problem is to maximize the minimum reliability among m tasks. Since it holds that the reliability $rel(t_i, W_i) = 1 - \prod_{w_j \in W_i}(1 - p_j)$ (given in Eq. (1)), we can rewrite it as:

$$R(t_i, W_i) = -ln(1 - rel(t_i, W_i)) = \sum_{w_j \in W_i} -ln(1 - p_j). \quad (8)$$

From Eq. (8), our goal (w.r.t. reliability) of maximizing the smallest $rel(t_i, W_i)$ for all tasks t_i is equivalent to maximizing the smallest $-ln(1 - rel(t_i, W_i))$ (i.e., LHS of Eq. (8)) for all t_i. In turn, we can maximize the smallest $\sum_{w_j \in W_i} -ln(1 - p_j)$ (i.e., RHS of Eq. (8)) among all tasks.

Intuitively, we associate each worker w_j with a positive constant $-ln(1 - p_j)$. Then, we divide n workers into m disjoint partitions (subsets) W_i ($1 \le i \le n$), and each subset W_i has a summed value $\sum_{w_j \in W_i} -ln(1 - p_j)$ (i.e., RHS of Eq. (8)). As a result, our equivalent reliability goal is to find a partitioning strategy such that the smallest summed value among m subsets is maximized.

3.2 Reduction of Diversity

As discussed in Section 2.4, the direct computation of the expected diversity involves an exponential number of possible worlds $pw(W_i)$ (see Eq. (6)). It is thus not efficient to enumerate all possible worlds. In this subsection, we will reduce such a computation to the problem with polynomial cost.

Specifically, in order to compute the expected spatial diversity, $E(SD(t_i))$, of a task t_i, we introduce a spatial diversity matrix, M_{SD}, in which each entry $M_{SD}[j][k]$ stores a value, given by multiplying the probability that an angle $A_{j,k} = (\sum_{x=j}^{k+r} A_{(x\%r)})\%2\pi$ exists (in possible worlds) and entropy $-(\frac{A_{j,k}}{2\pi}) \cdot log(\frac{A_{j,k}}{2\pi})$, where $x\%y$ is x mod y.

In particular, we have:

$$M_{SD}[j][k] = -(\frac{A_{j,k}}{2\pi}) \cdot log(\frac{A_{j,k}}{2\pi}) \cdot p_j \cdot p_k \cdot \prod_{x=j+1}^{(k+r-1)\%r} (1 - p_x). \quad (9)$$

The time complexity of computing $M_{SD}[j][k]$ is $O(r)$. Thus, the total cost of spatial diversity matrix is $O(r^3)$.

Similarly we can compute the temporal diversity matrix, M_{TD}, in which each entry $M_{TD}[j][k]$ ($j \le k$) is given by the multiplication of the probability that a time interval $I_{j,k} = \bigcup_{x=j}^{k} I_x$ exists in possible worlds and entropy $-(\frac{I_{j,k}}{e_i - s_i}) \cdot log(\frac{I_{j,k}}{e_i - s_i})$, for $j \le k$; moreover, $M_{TD}[j][k] = 0$,if $j > k$.

Formally, for $j \le k$, we have:

$$M_{TD}[j][k] = -(\frac{I_{j,k}}{e_i - s_i}) \cdot log(\frac{I_{j,k}}{e_i - s_i}) \cdot p_k \cdot \prod_{x=j+1}^{(k-1)} (1 - p_x). \quad (10)$$

To compute $E(STD(t_i))$, the expected spatial/temporal diversity, we have the following lemma.

Lemma 3.1. *(Expected Spatial/Temporal Diversity) The expected spatial/temporal diversity,* $E(STD(t_i))$*, of task* t_i *is given by:*

$$E(STD(t_i)) = \beta \cdot E(SD(t_i)) + (1-\beta) \cdot E(TD(t_i)) \quad (11)$$
$$= \beta \cdot \sum_{\forall j,k} M_{SD}[j][k] + (1-\beta) \cdot \sum_{\forall j,k} M_{TD}[j][k].$$

PROOF. Please refer to Appendix A of the technical report [6]. □

In this subsection, we prove that the hardness of our RDB-SC problem is NP-hard. Specifically, we can reduce the problem of the *number partition problem* [20] (which is known to be an NP-hard problem) to our RDB-SC problem. This way, our RDB-SC problem is also an NP-hard problem:

Lemma 3.2. *(Hardness of the RDB-SC Problem) The problem of the reliable diversity-based spatial crowdsourcing (RDB-SC) is NP-hard.*

PROOF. Please refer to Appendix B of the technical report [6]. □

From Lemma 3.2, we can see that the RDB-SC problem is not tractable. Therefore, in the sequel, we aim to propose approximation algorithms to find suboptimal solution efficiently.

4. THE GREEDY APPROACH

4.1 Properties of Optimization Goals

In this subsection, we provide the properties about the reliability and the expected spatial/temporal diversity. Specifically, assume that a task t_i is assigned with a set, W_i, of r workers w_j ($1 \leq j \leq r$). Let w_{r+1} be a new worker (with confidence p_{r+1} who is also assigned to task t_i.

Reliability. We first give the property of the reliability upon a newly assigned worker.

Lemma 4.1. *(Property of the Reliability) Let $R(t_i, W_i)$ be the reliability of task t_i (given in Eq. (8), in the reduced goal of Section 3.1), associated with a set, W_i, of r workers. If a new worker w_{r+1} is assigned to t_i, then we have:*

$$R(t_i, W_i \cup \{w_{r+1}\}) = R(t_i, W_i) - ln(1 - p_{r+1}). \quad (12)$$

PROOF. Please refer to Appendix C of the technical report [6]. □

From Eq. (12) in Lemma 4.1, we can see that the second term (i.e., $-ln(1-p_{r+1})$) is a positive value. Thus, it indicates that when we assign more workers (e.g., w_{r+1} with confidence $p_{r+1} \leq 0$) to task t_i, the reliability, $R(t_i, \cdot)$, of the task t_i is always increasing (at least non-decreasing).

Diversity. Next, we give the property of the expected spatial/temporal diversity, upon a newly assigned worker.

Lemma 4.2. *(Property of the Expected Spatial/Temporal Diversity) Let $E(STD(t_i))$ be the expected spatial/temporal diversity of task t_i. Upon a newly assigned worker w_{r+1} with confidence p_{r+1}, the expected diversity $E(STD(t_i))$ of task t_i is always non-decreasing, that is, $E(STD(t_i, W_i \cup \{w_{r+1}\})) \geq E(STD(t_i, W_i))$.*

PROOF. Please refer to Appendix D of the technical report [6]. □

Lemma 4.2 indicates that when we assign a new worker to a spatial task t_i, the expected spatial/temporal diversity is non-decreasing.

4.2 The Greedy Algorithm

As mentioned in Section 4.1, when we assign more workers to a spatial task, the reliability and diversity of the assignment strategy is always non-decreasing. Based on these properties, we propose a greedy algorithm, which iteratively assigns workers to spatial tasks that can always achieve high ranks (w.r.t. reliability and diversity).

Figure 3 illustrates the pseudo code of our RDB-SC greedy algorithm, namely RDB-SC_Greedy, which returns one best strategy, $\mathbb{S}$, containing task-and-worker assignments with high reliability and diversity. Specifically, our greedy algorithm iteratively finds one pair of task and worker such that the assignment with this pair can increase the reliability and diversity most.

Procedure RDB-SC_Greedy {
Input: m time-constrained spatial tasks in T and n workers in W
Output: a task-and-worker assignment strategy, $\mathbb{S}$, with high reliability and diversity
(1) $\mathbb{S} = \emptyset$
(2) compute all the valid task-and-worker pairs (t_i, w_j)
(3) for $i = 1$ to n
// in each round, select one best task-and-worker pair
(4) for each pair (t_i, w_j) $(w_j \in W)$
(5) compute the increase pair $(\Delta R(t_i, w_j), \Delta STD(t_i, w_j))$
(6) prune $(\Delta R(t_i, w_j), \Delta STD(t_i, w_j))$ dominated by others
(7) rank the remaining pairs by their scores (i.e., the number of dominated pairs)
(8) select a pair, (t_i, w_j), with the highest score and add it to $\mathbb{S}$
(9) $W = W - \{w_j\}$
(10) return $\mathbb{S}$
}

Figure 3: RDB-SC Greedy Algorithm.

Initially, there is no task-and-worker assignment, thus, we set $\mathbb{S}$ to empty (line 1). Next, we identify all the valid task-and-worker pairs (t_i, w_j) in the crowdsourcing system (line 2). Here, the validity of pair (t_i, w_j) means that worker w_j can reach the location of task t_i, under the constraints of both moving directions and valid period. Then, among these pairs, we want to incrementally select n best task-and-worker assignments such that the increases of reliability and diversity are always maximized (lines 3-9).

In particular, in each iteration, for every task-and-worker pair (t_i, w_j) ($w_j \in W$ is a worker who has no task), if we allow the assignment of worker w_j to t_i, we can calculate the increases of the reliability and diversity $(\Delta R(t_i, w_j), \Delta STD(t_i, w_j))$ (lines 4-5), where $\Delta R(t_i, w_j) = R(\mathbb{S} \cup \{(t_i, w_j)\}) - R(\mathbb{S})$, and $\Delta STD(t_i, w_j) = STD(\mathbb{S} \cup \{(t_i, w_j)\}) - STD(\mathbb{S})$. Note that, as guaranteed by Lemmas 4.1 and 4.2, here the two optimization goals are always non-decreasing (i.e., $\Delta R(\cdot)$ and $\Delta STD(\cdot)$ are positive).

Since some increase pairs may be *dominated* [12] by others, we can safely filter out such false alarms with both lower reliability and diversity (line 6). We say assignment $\mathbb{S}_i$ dominates $\mathbb{S}_j$ when $R(\mathbb{S}_i) > R(\mathbb{S}_j)$ and $STD(\mathbb{S}_i) \geq STD(\mathbb{S}_j)$, or $R(\mathbb{S}_i) \geq R(\mathbb{S}_j)$ and $STD(\mathbb{S}_i) > STD(\mathbb{S}_j)$. If there are more than one remaining pair, we rank them according to the number of pairs that they are dominating [21] (line 7). Intuitively, the pair (i.e., assignment) with higher rank indicates that this assignment is better than more other assignments. Thus, we add a pair (t_i, w_j) with the highest rank to $\mathbb{S}$, and remove the worker w_j from W (lines 8-9).

The selection of pairs (assignments) repeats for n rounds (line 3). In each round, we find one assignment pair that can locally increase the maximum reliability and diversity. Finally, we return $\mathbb{S}$ as the best RDB-SC assignment strategy (line 10).

The Time Complexity. The time complexity of computing the best task-and-worker pair in each iteration is given by $O(m \cdot n)$ in the worst case (i.e., each of n worker can be assigned to any of the m tasks). Since we only need to select n task-and-worker pairs (each worker can only be assigned to one task at a time), the total time complexity of our greedy algorithm is given by $O(m \cdot n^2)$.

4.3 Pruning Strategies

Note that, to compute the exact increase of the reliability $R(t_i, W_i)$, we can immediately obtain the reliability increase of task t_i: $\Delta R(t_i, w_j) = -ln(1 - p_j)$. For the diversity, however, it is not efficient to compute the exact increase, $\Delta STD(t_i, w_j)$, since we need to update the diversity matrices (as mentioned in Section 3.2) before/after the worker insertion. Therefore, in this subsection, we present an effective pruning method to reduce the search space without calculating the expected diversity for every (t_i, w_j) pair.

Our basic idea of the pruning method is as follows. For any task-and-worker pair (t_i, w_j), assume that we can quickly compute its lower and upper bounds of the increase for the expected spa-

tial/temporal diversity, denoted as $lb\_\Delta D(t_i, w_j)$ and $ub\_\Delta D(t_i, w_j)$, respectively.

Then, for two pairs (t_i, w_j) and (t'_i, w'_j), if it holds that $lb\_\Delta D(t_i, w_j) > ub\_\Delta D(t'_i, w'_j)$, then the diversity increase of pair (t'_i, w'_j) is inferior to that of pair (t_i, w_j).

We have the pruning lemma below.

Lemma 4.3. *(Pruning Strategy) Assume that $lb\_\Delta D(t_i, w_j)$ and $ub\_\Delta D(t_i, w_j)$ are lower and upper bounds of the increase for the expected spatial/temporal diversity, respectively. Similarly, let $\Delta min\_R(t_i, w_j)$ be the increase of the smallest reliability among m tasks after assigning worker w_i to t_i), respectively. Then, given two pairs (t_i, w_j) and (t'_i, w'_j), if it holds that: (1) $\Delta min\_R(t_i, w_j) \geq \Delta min\_R(t'_i, w'_j)$, and (2) $lb\_\Delta D(t_i, w_j) > ub\_\Delta D(t'_i, w'_j)$, then we can safely prune the pair (t'_i, w'_j).*

PROOF. Please refer to Appendix E of the technical report [6]. □

The Computation of Lower/Upper Bounds for the Diversity Increase. From Eq. (6), we can alternatively compute the lower/upper bounds of $E(STD(t_i))$ before and after assigning worker w_j to t_i. From Lemma 4.1, we know that the maximum diversity is achieved when the maximum number of workers are assigned to task t_i. Thus, in each possible world $pw(W_i)$, the upper bound of diversity $STD(t_i, pw(W_i))$ is given by $STD(t_i, W_i)$. Thus, we have $ub\_E(STD(t_i)) = STD(t_i, W_i)$.

Moreover, from Eq. (6), the lower bound, $lb\_E(STD(t_i))$, of the diversity is given by the probability that $STD(t_i, pw(W_i))$ is not zero in possible worlds times the minimum possible non-zero diversity. Note that, $STD(t_i, pw(W_i))$ is zero, when none or one worker is reliable. Thus, the minimum possible non-zero spatial diversity is achieved when we assign two workers to task t_i. In this case, two angles, $\min_{j=1}^{r} A_j$ and $(1 - \min_{j=1}^{r} A_j)$ can achieve the smallest diversity (i.e., entropy), which can be computed with $O(r)$ cost. The smallest non-zero temporal diversity is achieved when one worker is assigned to the task. The computation cost is also $O(r)$, where r is the maximum number of workers for task t_i.

After obtaining lower/upper bounds of the expected diversity, we can thus compute bounds of the diversity increase. We use subscript "b" and "a" to indicate the measures before/after the worker assignment, respectively. The bounds are:

$$lb\_\Delta D(t_i, w_j) = lb\_E_a(STD(t_i)) - ub\_E_b(STD(t_i)),$$
$$ub\_\Delta D(t_i, w_j) = ub\_E_a(STD(t_i)) - lb\_E_b(STD(t_i)).$$

Therefore, instead of computing the exact diversity values for all task-and-worker pairs with high cost, we now can utilize their lower/upper bounds to derive bounds of their increases, and in turn filter out false alarms by Lemma 4.3.

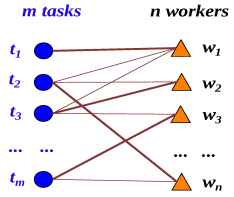


Figure 4: Illustration of the Task-and-Worker Assignment.

5. THE SAMPLING APPROACH

5.1 Random Sampling Algorithm

In this subsection, we illustrate how to obtain a good task-and-worker assignment strategy with high reliability and diversity by random sampling. Specifically, in our RDB-SC problem, all possible task-and-worker assignments correspond to the population, where each assignment is associated with a value (i.e., reliability or diversity). As shown in Figure 4, we denote m tasks t_i and n workers w_j by nodes of circle and triangle shapes, respectively. The edge between two types of nodes indicates that worker w_j can arrive at the location of task t_i within the time period (and with correct moving direction as well). Since each worker can be only assigned to one task, for each worker node w_j, we can select one of $deg(w_j)$ edges connecting to it (represented by bold edges in the figure), where $deg(w_j)$ is the degree of the worker node w_j. As a result, we can obtain n selected edges (as shown in Figure 4), which correspond to one possible assignment of workers to tasks.

Due to the exponential number ($O(\prod_{j=1}^{n} deg(w_j))$) of possible assignments (i.e., large size of the population), it is not feasible to enumerate all assignments, and find an optimal assignment with high reliability and diversity. Alternatively, we adopt the sampling techniques, and aim at obtaining K random samples from the entire population such that among these K samples there exists a sample with error-bounded ranks of reliability or diversity.

Figure 5 illustrates the pseudo code of our sampling algorithm, RDB-SC_Sampling, to tackle the RDB-SC problem. Specifically, we obtain each random sample (i.e., task-and-worker assignment), S_h ($1 \leq h \leq K$), from the entire population as follows. For each worker w_j ($1 \leq j \leq n$), we first randomly generate an integer x between $[1, deg(w_j)]$, and then select the x-th edge that connecting two nodes t_i and w_j (lines 5-7). After selecting n edges for n workers, respectively, we can obtain one possible assignment, which is exactly a random sample S_h ($1 \leq h \leq K$). Here, the random sample S_h is chosen with a probability $p = \prod_{j=1}^{n} \frac{1}{deg(w_j)}$. Given the sample (assignment) S_h, we can compute its reliability and diversity.

We repeat the sampling process above, until K samples (i.e., assignments) are obtained. After obtaining K samples, we rank them with the ranking scores [21] (w.r.t. reliability and diversity) (line 8). Let S_h be the sample with the highest score (line 9). We can return this sample (assignment) as the answer to the RDB-SC problem (line 10).

Procedure RDB-SC_Sampling {
Input: m time-constrained spatial tasks in T and n workers in W
Output: a task-and-worker assignment strategy, $\mathbb{S}$,with high reliability and diversity
(1) $\mathbb{S} = \emptyset$
(2) compute all the valid task-and-worker pairs (t_i, w_j)
(3) for $h = 1$ to K // *in each round, obtain one random sample* S_h
(4) $S_h = \emptyset$
(5) for each worker w_j ($\in W$)
(6) randomly select a task t_i with probability $\frac{1}{deg(w_j)}$
(7) $S_h = S_h \cup \{(t_i, w_j\}$
(8) rank S_h ($1 \leq h \leq K$) by dominating scores among samples
(9) let $\mathbb{S}$ be the sample, S_h, with the highest score
(10) return $\mathbb{S}$
}

Figure 5: RDB-SC Sampling Algorithm.

Intuitively, when the sample size K is approaching the population size (i.e., $\prod_{j=1}^{n} deg(w_j)$), we can obtain RDB-SC answers close to the optimal solution. However, since RDB-SC is NP-hard and intractable (as proved in Lemma 3.2), we alternatively aim to find approximate solution via samples with bounded rank errors. Specifically, our target is to determine the sample size K such that the sample with the maximum optimization goal (reliability or diversity) has the rank within the (ϵ, δ)-error-bound.

5.2 Determination of Sample Size

Without loss of generality, assume that we have the population of size N (i.e., $N = \prod_{j=1}^{n} deg(w_j)$), V_1, V_2, ..., and V_N, which correspond to the reliabilities/diversities of all possible task-and-worker assignments, where $V_1 \leq V_2 \leq ... \leq V_N$. Then, for each value V_i ($1 \leq i \leq N$), we flip a coin. With probability p, we accept value V_i as the selected random sample; otherwise (i.e., with probability $(1 - p)$), we reject value V_i, and repeat the same sampling process for the next value (i.e., V_{i+1}). This way,

we can obtain K samples, denoted as S_1, S_2, ..., and S_K, where $S_1 \leq S_2 \leq ... \leq S_K$.

Our goal is to estimate the required minimum number of samples, $\widehat{K}$, such that the rank of the largest sample S_K is bounded by ϵN in the population (i.e., within $((1-\epsilon)\cdot N, N]$) with probability greater than δ.

Let variable X be the rank of the largest sample, S_K, in the entire population. We can calculate the probability that $X = r$:

$$Pr\{X = r\} = \binom{r-1}{K-1} \cdot p^{K-1} \cdot (1-p)^{r-K} \cdot p \cdot (1-p)^{N-r} = \binom{r-1}{K-1} \cdot p^{K} \cdot (1-p)^{N-K}. \quad (13)$$

Intuitively, the first 3 terms above is the probability that $(K-1)$ out of $(r-1)$ values are selected from the population before (smaller than) S_K (i.e., $V_1 \sim V_{r-1}$). The fourth term (i.e., p) is the probability that the r-th largest value V_r $(= S_K)$ is selected. Finally, the last term is the probability that all the remaining $(N-r)$ values (i.e., $V_{r+1} \sim V_N$) are not sampled.

With Eq. (13), the cumulative distribution function of variable X is given by:

$$Pr\{X \leq r\} = \sum_{i=1}^{r} Pr\{X = i\}. \quad (14)$$

Now our problem is as follows. Given parameters p, ϵ, and δ, we want to decide the value of parameter K with high confidence. That is, we have:$Pr\{X > (1-\epsilon)\cdot N\} > \delta$.

By applying the combination theory and Harmonic series, we can rewrite the formula $Pr\{X > (1-\epsilon)\cdot N\} > \delta$, and derive the following formula w.r.t. K:

$$K > \frac{p \cdot M \cdot e - 1 + p}{1 - p + e \cdot p}, \quad (15)$$

where $M = (1-\epsilon)\cdot N$, and e is the base of the natural logarithm. Please refer the detailed derivation to Appendix F of the technical report [6]. Since $K \leq M$ holds, and the probability $Pr\{X \leq (1-\epsilon)\cdot N\}$ decreases with the increase of K, we can thus conduct a binary search for $\widehat{K}$ value within $\left(\frac{p\cdot M\cdot e-1+p}{1-p+e\cdot p}, M\right]$, such that $\widehat{K}$ is the smallest K value such that $Pr\{X \leq (1-\epsilon)\cdot N\} \leq 1-\delta$, where $p = \prod_{j=1}^{n} \frac{1}{deg(w_j)}$.

This way, we can first calculate the minimum required sample size, $\widehat{K}$, in order to achieve the (ϵ, δ)-bound. Then, we apply the sampling algorithm mentioned in Section 5.1 to retrieve samples. Finally, we calculate one sample with the highest reliability and diversity. Note that, in the case no sample dominates all other samples, we select one sample with the highest ranking score (i.e., dominating the most number of other samples) [21].

6. THE DIVIDE-AND-CONQUER APPROACH

6.1 Divide-and-Conquer Algorithm

We first illustrate the basic idea of the divide-and-conquer approach. As discussed in Section 5.1, the size of all possible task-and-worker assignments is exponential. Although RDB-SC is NP-hard, we still can speed up the process of finding the RDB-SC answers. By utilizing divide-and-conquer approach, the problem space is dramatically reduced.

Figure 6 illustrates the main framework for our divide-and-conquer approach, which includes three stages: (1) recursively divide the RDB-SC problem into two smaller subproblems, (2) solve two subproblems, and (3) merge the answers of two subproblems. In particular, for Stage (1), we design a partitioning algorithm, called BG_Partition, to divide the RDB-SC problem into smaller subproblems. In Stage (2), we use either the greedy or sampling algorithm, introduced in Section 4 and Section 5, respectively, to get an approximation result. Moreover, for Stage (3), we propose an algorithm, called SA_Merge, to obtain RDB-SC answers by combining answers to subproblems.

Procedure RDB-SC_DC {
Input: m time-constrained spatial tasks in T, n workers in W, and a threshold γ
Output: two sparse and balanced tasks-workers set pairs
(1) if $Size(T) \leq \gamma$
(2) solve problem (T,W) to get the result $\mathbb{S}$ directly
(3) else
(4) BG_Partition (T,W) to (T_1,W_1) and (T_2,W_2)
(5) RDB-SC_DC (T_1,W_1) to get answer $\mathbb{S}_1$
(6) RDB-SC_DC (T_2,W_2) to get answer $\mathbb{S}_2$
(7) SA_Merge ($\mathbb{S}_1$, $\mathbb{S}_2$) to get the result $\mathbb{S}$
(8) return $\mathbb{S}$
}

Figure 6: Divide and Conquer Algorithm.

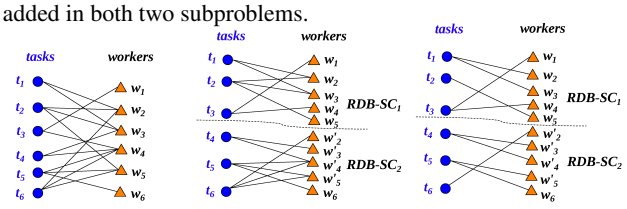
6.2 Partition the Bipartite Graph

As shown in Figure 4, the task-and-worker assignment is a bipartite graph. We first need to iteratively divide the whole graph into two subgraphs such that few edges crossing the cut (sparse) and close to bisection (balanced). However, this problem is NP-hard [11]. Here we just provide a heuristic algorithm, namely BG_Partition, which is shown in Figure 7. After running BG_Partition we can get two subproblems $RDB\text{-}SC_1$ and $RDB\text{-}SC_2$.

Procedure BG_Partition {
Input: m time-constrained spatial tasks in T and n workers in W
Output: two sparse and balanced tasks-workers set pairs
(1) $W_1 = \varnothing, W_2 = \varnothing$
(2) partition tasks into two even set T_1 and T_2 with KMeans
(3) for w_i in W
(4) if the tasks that w_i can do are all included in T_1
(5) put w_i into W_1 and $W = W - \{w_i\}$
(6) if the tasks that w_i can do are all included in T_2
(7) put w_i into W_2 and $W = W - \{w_i\}$
(8) add W into W_1 and W_2
(9) return (T_1, W_1) and (T_2, W_2)
}

Figure 7: Bipartite Graph Partitioning Algorithm.

First, we partition tasks into two almost even subsets, T_1 and T_2, based on their locations, which can be done through clustering the tasks to two set (i.e. KMeans.). Then we find out the workers who can reach tasks totally included in some subset, and add them to the corresponding worker subset, W_1 or W_2. By doing this, these workers are isolated in the corresponding subproblems. For the rest workers who can do tasks both in T_1 and T_2, we add them to both W_1 and W_2. As Figure 8 shows, w_1 and w_5 are isolated in $RDB\text{-}SC_1$ and $RDB\text{-}SC_2$ respectively while $w_2, w_3, w4$ are added in both two subproblems.



(a) Origin Problem (b) Partitioned Problem (c) ICW and DCW

Figure 8: Illustration of the Task-and-Worker Partitioning

Note that, even if some workers are duplicated and added to both W_1 and W_2, each of them can only be assigned to one task. Moreover, the duplicated workers in each subproblem can only do a part of the tasks that he can do in the whole problem. The complexity of $RDB\text{-}SC_1$ or $RDB\text{-}SC_2$ are much lower than $RDB\text{-}SC$. Each time before calling BG_Partition algorithm, we check whether the size of the tasks is greater than a threshold γ, otherwise that problem is small enough to solve directly. The threshold γ is set before running the divide-and-conquer algorithm.

6.3 Merge the Answers of the Subproblems

To merge the answers of the subproblems, we just need to solve the conflicts of the workers who are added to both two subproblems. Those duplicated workers in are called *conflicting workers*,

whereas others are called *non-conflicting workers*. We first give the property of deleting one copy of a conflicting worker.

A conflicting worker w_i is called *independent conflicting worker* (ICW), when w_i is assigned to tasks t_{i1} and t_{i2} in the optimal assignments for $RDB\text{-}SC_1$ and $RDB\text{-}SC_2$, respectively, and no other conflicting workers are assigned to either t_{i1} or t_{i2}. Otherwise, w_i is called *dependent conflicting worker* (DCW). For example, in Figure 8(c) worker w_5 is a ICW and worker w_2 is a DCW.

Lemma 6.1. *(Non-conflict Stable) The deletion of those conflicting workers' copies will not change the assignments of non-conflicting workers who are assigned to a same task with any deleted worker.*

PROOF. Please refer to Appendix G of the technical report [6]. □

Lemma 6.2. *(Deletion of copies of ICWs and DCWs) The deletion of copies of ICWs can be done independently while DCWs' deletion need to be considered integrally.*

PROOF. Please refer to Appendix H of the technical report [6]. □

With Lemmas 6.1 and 6.2, the algorithm for merging subproblems is shown in Figure 9, called SA_Merge.

Procedure SA_Merge {
Input: two subproblems (T_1, W_1) and (T_2, W_2), and their local answer $\mathbb{S}_1$ and $\mathbb{S}_2$
Output: one merged problem's answer $\mathbb{S}$ for $(T_1 \cup T_2, W_1 \cup W_2)$
(1) $W' = W_1 \cap W_2$
(2) while W' is not empty
(3) pick first worker in W' as w_t
(4) find out dependent workers for w_t as W_d
(5) add w_t, W_d into W_t
(6) remove one copy of each worker in W_t from $\mathbb{S}_1$ and $\mathbb{S}_2$ integrally
(7) $W' = W' - W_t$
(8) $\mathbb{S} = \mathbb{S}_1 \cup \mathbb{S}_2$
(9) return $\mathbb{S}$
}

Figure 9: Algorithm of Merging Answers to Subproblems.

7. COST-MODEL-BASED GRID INDEX

7.1 Index Structure

We first illustrate the index structure, namely RDB-SC-Grid, for the RDB-SC system. Specifically, in a 2-dimensional data space $[0, 1]^2$, we divide the space into $1/\eta^2$ square cells with side length η, where $\eta < 1$ and we discuss in Appendix H of the technical report [6] about how to set η based on a cost model.

Each cell has a unique ID, $cellid$, and contains a task list and a worker list which store tasks and workers in it, respectively. In each task list, we maintain quadruples (tid, l, s, e), where tid is the task ID, l is the position of the task, and $[s, e]$ is the valid period of the task. In each worker list, we keep records in the form $(wid, l, v, \alpha^-, \alpha^+, p)$, where wid is the worker ID, l and v represent the location and velocity of the worker, respectively, $[\alpha^-, \alpha^+]$ indicates the angle range of moving directions, and p is the reliability of the worker. For each cell, we also maintain bounds, $[v_{min}, v_{max}]$, for velocities of all workers in it, $[\alpha_{min}, \alpha_{max}]$, for all workers' moving directions, and $[s_{min}, e_{max}]$ of tasks' time constraints, where s_{min} is the earliest start time of tasks stored in the cell, and e_{max} is the latest deadline in the cell. In addition, each cell is associated with a list, $tcell\_list$, which contains all the IDs of cells that can be reachable to at least one worker in that cell.

Pruning Strategy on the Cell Level. One straightforward way to construct $tcell\_list$ for cell $cell_i$ is to check all cells $cell_j$, and add those "reachable" cells to $tcell\_list$. This is however quite time-consuming to check each pair of worker and task from $cell_i$ and $cell_j$, respectively. In order to accelerate the efficiency of building $tcell\_list$, we propose a pruning strategy to reduce the search space. That is, for cell $cell_i$, if a cell, $cell_j$, is in the *reachable area* (in workers' moving directions) within two rays starting from $cell_i$ (i.e., reachable by at least one worker in $cell_i$), then we add $cell_j$ to list $tcell\_list$ of $cell_i$. Therefore, our pruning strategy can effectively filter out those cells that are definitely unreachable.

Specifically, we can prune $cell_j$ as follows. First, we calculate the minimum and maximum distances, d_{min} and d_{max}, respectively, between any two points in $cell_i$ and $cell_j$. As a result, any worker who moves from $cell_i$ will arrive at $cell_j$ with time at least $t_{min} = \frac{d_{min}}{v_{max}(cell_i)}$, where $v_{max}(cell_i)$ is the maximum speed in $cell_i$. Thus, if $t_{min} > e_{max}(cell_i)$, we can safely prune $cell_j$, where $e_{max}(cell_i)$ represents the latest deadline of tasks in $cell_i$. After pruning these unreachable cells, we further check the rest cells one by one to build the final $tcell\_list$ for $cell_i$.

Please refer to Appendix I of the technical report [6].

Table 2: Experiments setting.

Parameter	Values
range of expiration time rt	[0.25, 0.5], [0.5, 1], **[1, 2]**, [2, 3]
reliability of workers $[p_{min}, p_{max}]$	(0.8, 1), (0.85, 1), **(0.9, 1)**, (0.95, 1)
number of tasks m	5K, 8K, **10K**, 50K, 100K
number of workers n	5K, 8K, **10K**, 15K, 20K
velocities of workers $[v^-, v^+]$	[0.1, 0.2],**[0.2, 0.3]**, [0.3, 0.4], [0.4, 0.5]
range of moving angles $(\alpha_j^+ - \alpha_j^-)$	(0, π/8], (0, π/7], **(0, π/6]**, (0, π/5], (0, π/4]
balancing weight β	(0, 0.2], (0.2, 0.4], **(0.4, 0.6]**, (0.6, 0.8], (0.8, 1)

7.2 Dynamic Maintenance

To insert a worker w_i into RDB-SC-Grid, we first find the cell $cell_k$ where w_i locates, which uses $O(1)$ time. Moreover, we also need to update the $tcell\_list$ for $cell_k$, which requires $O(cost_{update})$ time in the worst case. The case of removing a worker is similar.

To insert a task t_j into RDB-SC-Grid, we obtain the cell, $cell_k$, for the insertion, which requires $O(1)$ time cost. Furthermore, we need to check all the cells that do not contain $cell_k$ in their $tcell\_list$'s, which needs to check all workers in the worst case (i.e., $O(n)$). When removing a task from $cell_k$, we check all the cells containing it in their $tcell_list$s, which also requires to check every worker in the worst case (i.e., $O(n)$).

8. EXPERIMENTAL STUDY

8.1 Experimental Methodology

Data Sets. We use both real and synthetic data to test our approaches. Specifically, for real data, we use the POI (Point of Interest) data set of China [9] and T-Drive data set [22, 23]. The POI data set of China contains over 6 million POIs of China in 2008, whereas T-Drive data set includes GPS trajectories of 10,357 taxis within Beijing during the period from Feb. 2 to Feb. 8, 2008. We test our approaches in the area of Beijing (with latitude from 39.6° to 40.25° and longitude from 116.1° to 116.75°), which covers 74,013 POIs. After filtering out short trajectories from T-Drive data set, we obtain 9,748 taxis' trajectories. We use POIs to initialize the locations of tasks. From the trajectories, we extract workers' locations, ranges of moving directions, and moving speeds. For workers' confidences p, tasks' valid periods $[s, e]$, and parameter β to balance spatial and temporal diversity, we follow the same settings as that in synthetic data (as described below).

For synthetic data, we generate locations of workers and tasks in a 2D data space $[0, 1]^2$, following either Uniform (UNIFORM) or Skewed (SKEWED) distribution. In particular, similar to [17], we generate tasks and workers with Skewed distribution by letting 90% of tasks and workers falling into a Gaussian cluster (centered at (0.5, 0.5) with variance = 0.2^2). For each moving worker w_j, we randomly produce the angle range, $[\alpha_j^-, \alpha_j^+]$, where α_j^- is uniformly chosen within $[0, 2\pi]$ and $(\alpha_j^+ - \alpha_j^-)$ is uniformly distributed in a range of angle (e.g., $(0, \pi/6]$). Moreover, we also generate check-in times of each worker with Uniform or Skewed distribution, and compute one's confidence (reliability) following Gaussian distribution within the range $[p_{min}, p_{max}]$ (with mean

$\frac{p_{min}+p_{max}}{2}$, and variance 0.02^2). Furthermore, we obtain the velocity of each worker with either Uniform or Gaussian distribution within range $[v^-, v^+]$, where $v^-, v^+ \in (0, 1)$. Regarding spatial tasks, we generate their valid periods, $[s, e]$, within a time interval $[st, st+rt]$, where $st \in [0, 24]$ follows either Uniform or Gaussian distribution, and rt follows the Uniform distribution. To balance SD and TD, we test parameter β following Uniform distribution within $[0, 1]$. The case of rt or β following other distributions is similar, and thus omitted due to space limitations.

Configurations of a Customized gMission Spatial Crowdsourcing Platform. We tested the performance of our proposed algorithms with an incrementally updating strategy, on a real spatial crowdsourcing platform, namely gMission [7, 14]. In particular, gMission is a general laboratory application, which has over 20 existing active users in the Hong Kong University of Science and Technology. In gMission, users can ask/answer spatial crowdsourcing questions (tasks), and the platform pushes tasks to users based on their spatial locations. As gMission has been released, the recruitment, monitoring and compensation of workers are already provided. A credit point system of gMission can record the contribution of workers. Workers can use their credit point to redeem coupons of book stores or coffee shops. Moreover, when they are using gMission, their trajectories are recorded, which is informed to the users. To evaluate our RDB-SC model and algorithms, we will modify/adapt gMission to an RDB-SC system.

Specifically, in order to build user profiles, we set up the peer-rating over 613 photos taken by active users. They are asked to rate peers' photos based on their resolutions, distances, and lights. The score of each photo is given by first removing the highest and lowest scores, and then averaging the rest. Moreover, the score of each user is given by the average score of all photos taken by this user. Intuitively, a user receiving a higher score is more reliable. Thus, we set the user's peer-rating score as one's reliability value.

In addition, we also provide a setting option for users to configure their preferred working area. For example, a user is going home, and wants to do some tasks on the way home. In general, one may just want to go to places not deviating from the direction towards his/her home too much. Thus, a fan-shaped working area is reasonable. Due to the maximum possible speed of the user, this fan-shaped working area is also constrained by the maximum moving distance.

Furthermore, we enable gMission to detect the accuracy of answers. When a worker w_j is taking a photo to answer a task t_i, we record his/her instantaneous information, like the facing direction, the location, and the timestamp, through his/her smart device. By comparing the information with the required angle and time constraint of t_i, we can calculate the error of angle $\Delta\theta_{ij}$ and the error of time Δt_{ij}. Then, the accuracy of the answer is $Accuracy_{ij} = \beta_i \cdot \frac{\Delta\theta_{ij}}{\pi} + (1-\beta_i) \cdot \frac{\Delta t_{ij}}{e_i - s_i}$, where β_i is the balancing weight of t_i, and s_i and e_i are the starting and ending times, respectively. In particular, $0 \le \Delta\theta_{ij} \le \pi$ and $0 \le \Delta t_{ij} < e_i - s_i$. Then, the accuracy of a task is given by the average value of all accuracy values of the answers to the task. For the accuracy control, it could be quite interesting and challenging, and we would like to leave it as our future work.

In order to deploy our proposed algorithms, we implement the cost-model-based grid index, and apply the incremental updating strategy for dynamically-changing tasks and workers to our RDB-SC system. The framework for the incremental updating strategy is shown in Figure 10 below. In line 6 of the framework (Figure 10), we can use our proposed algorithms to assign the available workers to the opening tasks, where considering A and $\mathbb{S}_c$ means the reliability and diversity of a task t_i is calculated from the received answers, the workers assigned to t_i in $\mathbb{S}_c$, and newly assigned workers. In particular, we periodically update the task-and-worker assignments every $t_{interval}$ timestamps. For each update, those workers, who either have accomplished the assigned tasks or rejected the assignment requests, would be available to receive new tasks. In our experiments, we set this length, $t_{interval}$, of the periodic update interval, from 1 minute to 4 minutes, with an increment of 1 minute. We hired 10 active users in our experiments and chose 5 sites to ask spatial crowdsourcing questions (tasks) with 15 minutes opening time. The sites are close to each other, and in general a user can walk from one site to another one within 2 minutes.

Procedure RDB-SC_Incremental {
  Input: m time-constrained spatial tasks in T, n workers in W,
  Output: a updated task-and-worker assignment strategy, $\mathbb{S}$, with high reliability and diversity
  (1) $\mathbb{S} = \emptyset$
  (2) from W, retrieve all the available workers to W_a
  (3) from T, retrieve all the opening tasks to T_a
  (4) obtain the received answers of all the tasks in W_a, noted as A
  (5) obtain the current assignment, noted as S_c
  (6) assign workers in W_a to tasks in T_a considering A and S_c (new pairs are added to $\mathbb{S}$)
  (7) $\mathbb{S} = \mathbb{S} \cup \mathbb{S}_c$
  (8) return $\mathbb{S}$
}

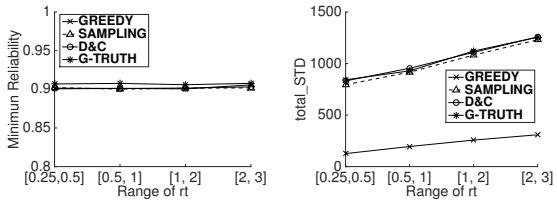
Figure 10: Incremental Updating Strategy.

RDB-SC Approaches and Measures. Greedy (GREEDY) assigns each worker to a "best" task according to the current situation when processing the worker, which is just a local optimal approach. Sampling (SAMPLING) randomly assigns all the available workers several times and picks the best test result, using our equations in Section 5.2 to calculate the sampling times to bound the accuracy. Divide-and-Conquer (D&C) divides the original problem into subproblems, solves each one and merges their results. To accelerate D&C, we use SAMPLING to solve subproblems of D&C, which will sacrifice a little accuracy. Nonetheless, this trade-off is effective, we will show SAMPLING has a good performance when the problem space is small in our experiments. To evaluate our 3 proposed approaches, we will compare them with the ground truth. However, since RDB-SC problem is NP-hard (as discussed in Section 3.2), it is infeasible to calculate the real optimal result as the ground truth. Thus, we use Divide-and-Conquer approach with the embedded sampling approach (discussed in Section 5) to calculate sub-optimal result by setting the sampling size 10 times larger than D&C (denoted as G-TRUTH).

Table 2 depicts the experimental settings, where the default values of parameters are in bold font. In each set of experiments, we vary one parameter, while setting others to their default values. We report $\min_{i=1}^{m} rel(t_i, W_i)$, the minimum reliability, and $total\_STD$, the summation of the expected spatial/temporal diversities. All our experiments were run on an Intel Xeon X5675 CPU @3.07 GHZ with 32 GB RAM.

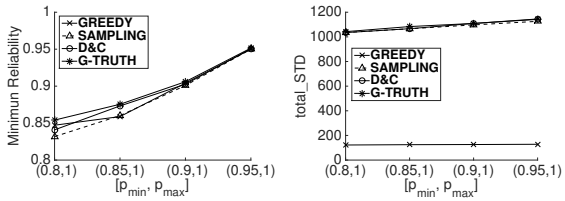
8.2 Experiments on Real Data

In this subsection, we show the effects of workers' confidence p, tasks' valid periods $[s, e]$ and balancing parameter β on the real data. We use the locations of the POIs as the locations of tasks. To initialize a worker based on trajectory records of a taxi, we use the start point of the trajectory as the worker's location, use the average speed of the taxi as the worker's speed. For the moving angle's range of the worker, we draw a sector at the start point and contain all the other points of the trajectory in the sector, then we use the sector as the moving angle's range of the worker. We uniformly sample 10,000 POIs from the 74,013 POIs in the area of Beijing and the sampled POI date set follows the original data set's distribution. In other words, we have 10,000 tasks and 9,748 workers in the experiments on real data.



(a) Minimum Reliability (b) Summation of Diversity

Figure 11: Effect of Tasks' Expiration Time Range of rt



(a) Minimum Reliability (b) Summation of Diversity

Figure 12: Effect of Workers' Reliability $[p_{min}, p_{max}]$

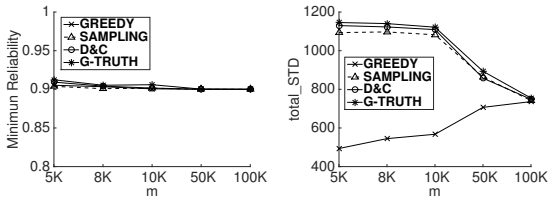
Effect of the Range of Tasks' Expiration Times rt. Figure 11 shows the effect of varying the range of tasks' expiration times rt. When this range increases from $[0.25, 0.5]$ to $[2, 3]$, the minimum reliability is very stable, and the diversities $total\_STD$ of all the approaches gradually increase. Intuitively, longer expiration time for a task t_i means more workers can arrive at location l_i to accomplish t_i. From the perspective of workers, each worker can have more choices in his/her reachable area. Thus, each worker can choose a better target task with higher diversity. Similar to previous results, SAMPLING and D&C approaches can achieve higher diversities than GREEDY, and slightly lower diversities compared with G-TRUTH. The requester can use this parameter to constrain the range of the opening time of a task. For example, if one wants to know the situation of a car park in a morning, he/she can set the time range as the period of the morning.

Effect of the Range of Workers' Reliabilities $[p_{min}, p_{max}]$. Figure 12 reports the effect of the range, $[p_{min}, p_{max}]$, of workers' reliabilities on the reliability/diversity of our proposed RDB-SC approaches. For the minimum reliability, the reliabilities of workers may greatly affect the reliability of spatial tasks (as given by Eq. (1)). Thus, as shown in Figure 12(a), for the range with higher reliabilities, the minimum reliability of tasks also becomes larger. For diversity $total\_STD$, according to Lemma 3.1, when the workers assigned to tasks t_i have higher reliabilities, the expected spatial/temporal diversity will be higher. Therefore, we can see the slight increases of $total\_STD$ in Figure 12(b). Similar to previous results, SAMPLING and D&C show reliability and diversity similar to G-TRUTH, and have higher diversities than GREEDY.

We test the effect of the requester-specified weight range. Due to space limitations, please refer to the experimental results with different β values in Appendix J of the technical report [6]. Requesters can use this parameter to reflect their preference. The valid value of β is from 0 to 1. The bigger β is, the more spatial diverse the answers are. The smaller β is, the more temporal diverse the answers are. If one has no preference, he/she can simply set β to 0.5.

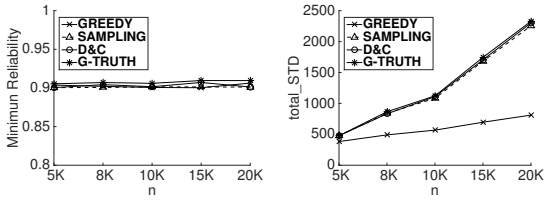
8.3 Experiments on Synthetic Data

In this subsection, we test the effectiveness and robustness of our proposed 3 RDB-SC approaches, GREEDY, SAMPLING, and D&C, compared with G-TRUTH, by varying different parameters. As we already see the effects of p and $[s, e]$, we will focus on the rest four parameters in Table 2 in this subsection. We first report the experimental results on Uniform task/worker distributions. Please refer to more (similar) experimental results over data sets with Uniform/Skew distributions in Appendix J of the technical report [6].



(a) Minimum Reliability (b) Summation of Diversity

Figure 13: Effect of the Number of Tasks m (UNIFORM)



(a) Minimum Reliability (b) Summation of Diversity

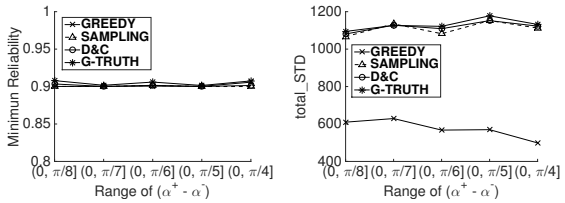
Figure 14: Effect of the Number of Workers n (UNIFORM)

Effect of the Number of Tasks m. Figure 13 shows the effect of the number, m, of spatial tasks on the reliability and diversity of RDB-SC answers, where we vary m from 5K to 100K. In Figure 13(a), all the 3 approximation approaches can achieve good minimum reliability, which are close to G-TRUTH, and remain high (i.e., with reliability around 0.9). The reliability of D&C is higher than that of the other two approaches. When the number, m, of tasks increases, the minimum reliability slightly decreases. This is because given a fixed (default) number of workers, our assignment approaches trade a bit the reliability for more accomplished tasks.

For the diversity, our 3 approaches have different trends for larger m. In Figure 13(b), for large m, the total diversity, $total\_STD$, of GREEDY becomes larger, while that of the other two approaches decreases. For GREEDY, more tasks means more possible task targets for each worker on average. This can make a particular worker to choose one possible task such that high diversity is obtained. In contrast, for SAMPLING, when the number of tasks increases, the size of possible combinations increases dramatically. Thus, under the same accuracy setting, the result will be relatively worse (as discussed in Section 5.2). In D&C, we divide the original problem into several subproblems of smaller scale. Since the number of possible combinations in subproblems decreases quickly, we can achieve good solutions to subproblems. After merging answers to subproblems, D&C can obtain a slightly higher $total\_STD$ than SAMPLING (about 3% improvement). We can see that, both SAMPLING and D&C have $total\_STD$ very close to G-TRUTH, which indicates the effectiveness of SAMPLING and D&C.

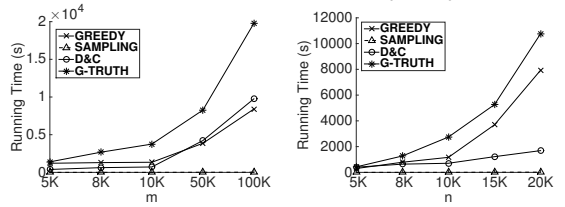
In Figure 13(b), when m is small, SAMPLING and D&C can achieve much higher $total\_STD$ than that of GREEDY. The reason is that GREEDY has a bad start-up performance. That is, when most reachable tasks of a worker are not assigned with workers (namely, empty tasks), he/she is prone to join those tasks that already have workers. In particular, when a worker joins an empty task, he/she can only improve the temporal diversity (TD) of that task, and has no contribution to the spatial diversity (SD), according to the definitions in Section 2.3. On the other hand, if a worker joins a task that has already been assigned with some workers, then his/her join can improve both SD and TD, which leads to higher STD. Since GREEDY always chooses task-and-worker pairs that increase the diversity most, GREEDY will always exploit those non-empty tasks, which may potentially miss the good assignment with high diversity $total\_STD$. Thus, $total\_STD$ of GREEDY is low when m is 5K - 10K.

Effect of the Number of Workers n. Figure 14 illustrates the experimental results on different numbers, n, of workers from $5K$ to



(a) Minimum Reliability (b) Summation of Diversity

Figure 15: Effect of the Range of Angles $(\alpha_j^+ - \alpha_j^-)$(UNIFORM)



(a) CPU Time (vs. m) (b) CPU Time (vs. n)

Figure 16: Comparisons of the CPU Time with RDB-SC Approaches

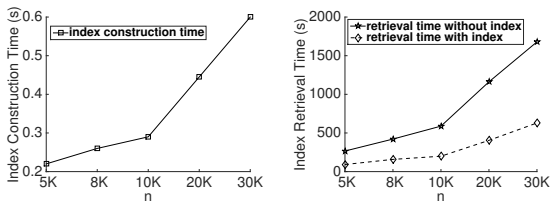
$20K$. In Figure 14(a), the minimum reliability is not very sensitive to n. This is because, although we have more workers, there always exist tasks that are assigned with just one worker. According to Eq. (1), the minimum reliability among tasks is very close to the lower bound of workers' confidences. Thus, the reliability slightly changes with respect to n.

On the other hand, the diversities, $total\_STD$, of all the four approaches increase for larger n value. In particular, as depicted in Figure 14(b), the diversity of SAMPLING increases more rapidly than that of GREEDY. Recall from Lemma 4.2 that, more workers means a higher diversity for each task. When the number of workers increases, the average number of workers of each task also increases, which leads to a higher $total\_STD$. Similar to previous results, SAMPLING and D&C have diversities very close to G-TRUTH, which confirms the effectiveness of our approaches.

Effect of the Range of Moving Angles $(\alpha_j^+ - \alpha_j^-)$. Figure 15 varies the range, $(\alpha_j^+ - \alpha_j^-)$, of moving angles for workers w_j from $(0, \pi/8]$ to $(0, \pi/4]$. From figures, we can see that the minimum reliability is not very sensitive to this angle range. With different angle ranges, the reliability of our proposed approaches remains high (i.e., above 0.9). Moreover, both SAMPLING and D&C approaches can achieve much higher diversities than GREEDY, and they have diversities similar to G-TRUTH, which indicates good effectiveness against different angle ranges of moving directions. On the other hand, $total\_STD$ of GREEDY drops when angle becomes larger. The reason is similar to the cause of GREEDY's bad start-up, which is discussed when we show the effect of the number of tasks. Larger angle range means more reachable tasks, then workers are more likely to find a task that has been assigned with workers and join that task, which leads to low diversity. On a real platform, the workers may set this parameter based on their personal interests. For example, if a worker would like to deviate more from his/her moving direction, he/she can set the range of his/her moving angle wider.

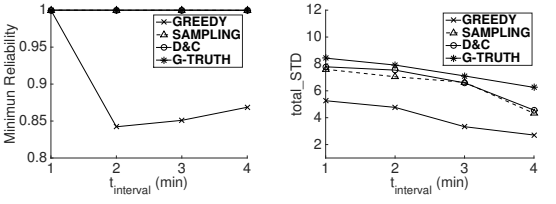
We test the effect of the range of workers' velocities. Due to space limitations, please refer to the experimental results with different ranges, $[v^-, v^+]$, in Appendix J of the technical report [6].

Running Time Comparisons and Efficiency of RDB-SC-Grid. We report the running time of our approaches by varying m and n in Figures 16(a) and 16(b), respectively. We can see that, when m increases, the running times of all approaches, except for SAMPLING, grow quickly. For GREEDY, when m increases, each worker has more reachable tasks, and thus the total running time grows (since more tasks should be checked). For large m, D&C needs to run more rounds for the divide-and-conquer process, which leads to higher running time. On the other hand, when n increases, only GREEDY's running time grows dramatically. This is because GREEDY needs to run more rounds to assign workers. Under both situations, SAMPLING only takes several seconds (due to small sample size). In contrast, D&C has higher CPU cost than SAMPLING, however, higher reliability and diversity (as confirmed by Figures 13 and 14). This indicates that D&C can trade the efficiency for effectiveness (i.e., reliability and diversity).



(a) Index Construction Time (b) W-T Pairs Retrieval Time

Figure 17: Efficiency of the RDB-SC-Grid Index



(a) Minimum Reliability (b) Summation of Diversity

Figure 18: Effect of the updating time interval $t_{interval}$

Figure 17 presents the *index construction time* and *index retrieval time* (i.e., the time cost for retrieving task-and-worker pairs, denoted as W-T pairs, from the index) over UNIFORM data, where $m = 10K$ and n varies from $5K$ to $30K$. As shown in Figure 17(a), the construction time of the RDB-SC-Grid index is small (i.e., less than 0.7 sec). In Figure 17(b), the RDB-SC-Grid index can dramatically reduce the time of finding W-T pairs (up to 67 %), compared with that of retrieving W-T pairs without index.

8.4 Experiments on Real RDB-SC Platform

Figure 18 shows the RDB-SC performance of GREEDY, SAMPLING, D&C, and G-TRUTH over the real RDB-SC system, where the length of the time interval, $t_{interval}$, between every two consecutive incremental updates varies from 1 minute to 4 minutes. In Figure 18(a), when $t_{interval}$ becomes larger, the minimum reliability remains high, except for GREEDY. This is because when $t_{interval}$ is larger than 1 minute, GREEDY assigns just one worker to some tasks, and it leads to sensitive change of the minimum reliability which is much more than that of other algorithms. Each user is assigned with fewer tasks in the entire testing period, when $t_{interval}$ increases. At the same time, GREEDY is prone to assign workers to those tasks already have workers or are answered, which has been discussed when we show the effect of the number of tasks in Section 8.3. When each user is assigned with fewer tasks in the entire testing period, it is more likely to assign only one worker to some task whose reliability will equal to the reliability of that worker. Thus, the minimum reliability of GREEDY varies much.

In Figure 18(b), we can see that for all the approaches, when $t_{interval}$ increases, the total spatial/temporal diversity $total\_STD$ decreases. This is reasonable, since each user is assigned with fewer tasks in the entire testing period. Meanwhile, SAMPLING and D&C are much better than GREEDY from the perspective of the diversity, and their diversities are close to that of G-TRUTH, which indicates the effectiveness of our RDB-SC approaches.

To show the potential value of our model, we present a 3D reconstruction showcase on gMission's homepage [7]. The showcase video can be found on Youtube [8]. Please refer to the figures of the showcase in Section 8.4 of the technical report [6].

9. RELATED WORK

Recently, with rapid development of GPS-equipped mobile devices, the spatial crowdsourcing [17, 19] that sends location-based requests to workers (based on their spatial positions) has become increasingly important in real applications, such as monitoring real-world scenes (e.g., street view of Google Maps [1]), local hotspots (e.g., Foursquare [3]), and the traffic (e.g., Waze [4]).

Some prior works [10, 13] studied the crowdsourcing problems which treat location information as the parameter, and distribute tasks to workers. However, in these works, workers do not have to visit spatial locations physically (in person) to complete the assigned tasks. In contrast, the spatial crowdsourcing usually needs to employ workers to conduct tasks (e.g., sensing jobs) by physically going to some specific positions. For example, some previous works [15, 18] studied the single-campaign or small-scale participatory sensing problems, which focus on particular applications of the participatory sensing.

According to people's motivation, Kazemi and Shahabi [19] classified the spatial crowdsourcing into two categories: reward-based and self-incentivised. That is, in the reward-based spatial crowdsourcing, workers can receive a small reward after completing a spatial task; oppositely, for the self-incentivised one, workers perform the tasks voluntarily (e.g., participatory sensing). In this paper, we consider the self-incentivised spatial crowdsourcing.

Furthermore, based on publishing modes of spatial tasks, the spatial crowdsourcing problems can be partitioned into another two classes: *worker selected tasks* (WST) and *server assigned tasks* (SAT) [19]. In particular, WST publishes spatial tasks on the server side, and workers can choose any tasks without contacting with the server; SAT collects location information of all workers to the server, and directly assigns workers with tasks. For example, in the WST mode, some existing works [10, 17] allowed users to browse and accept available spatial tasks. On the other hand, in the SAT mode, previous works [18, 19] assumed that the server decides how to assign spatial tasks to workers and their solutions only consider simple metrics such as maximizing the number of assigned tasks on the server side and maximizing the number of worker's self-selected tasks. In this paper, we not only consider the SAT mode, but also take into account constrained features of workers/tasks (e.g., moving directions of workers and valid period of tasks), which make our problem more complex and unsuitable for borrowing existing techniques.

Kazemi and Shahabi [19] studied the spatial crowdsourcing with the goal of static maximum task assignment, and proposed several heuristics approaches to enable fast assignment of workers to tasks. Similarly, Deng et al. [17] tackled the problem of scheduling spatial tasks for a single worker such that the number of completed tasks by this worker is maximized. In contrast, our work has a different goal of maximizing the reliability and spatial/temporal diversity that spatial tasks are accomplished. As mentioned in Section 1, the reliability and diversity of spatial tasks are very important criteria in applications like taking photos or checking whether or not parking spaces are available. Moreover, while prior works often consider static assignment, our work considers dynamic updates of spatial tasks and moving workers, and proposes a cost-model-based index. Therefore, previous techniques [17, 19] cannot be directly applied to our RDB-SC problem.

Another important topic about the spatial crowdsourcing is the privacy preserving. This is because workers need to report their locations to the server, which thus may potentially release some sensitive location/trajectory data. Some previous works [15, 18] investigate how to tackle the privacy preserving problem in spatial crowdsourcing, which is however out of the scope of this paper.

10. CONCLUSION

In this paper, we propose the problem of reliable diversity-based spatial crowdsourcing (RDB-SC), which assigns time-constrained spatial tasks to dynamically moving workers, such that tasks can be accomplished with high reliability and spatial/temporal diversity. We prove that the processing of the RDB-SC problem is NP-hard, and thus we propose three approximation algorithms (i.e., greedy, sampling, and divide-and-conquer). We also design a cost-model-based index to facilitate worker-task maintenance and RDB-SC answering. Extensive experiments have been conducted to confirm the efficiency and effectiveness of our proposed RDB-SC approaches on both real and synthetic data sets.

11. ACKNOWLEDGMENT

This work is supported in part by the Hong Kong RGC Project N_HKUST637/13; National Grand Fundamental Research 973 Program of China under Grant 2014CB340303; NSFC under Grant No. 61328202, 61325013, 61190112, 61373175, and 61402359; and Microsoft Research Asia Gift Grant.

12. REFERENCES

[1] https://www.google.com/maps/views/streetview.
[2] http://mediaqv3.cloudapp.net/MediaQ_MVC_V3/.
[3] https://foursquare.com.
[4] https://www.waze.com.
[5] https://www.mturk.com/mturk/welcome.
[6] http://arxiv.org/abs/1412.0223.
[7] http://www.gmissionhkust.com.
[8] http://youtu.be/FfNoeqFc084.
[9] Beijing city lab, 2008, data 13, points of interest of china in 2008. http://www.beijingcitylab.com.
[10] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *NordiCHI 2010: Extending Boundaries*, 2010.
[11] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *JACM*, 56(2):5, 2009.
[12] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. pages 421–430, 2001.
[13] M. F. Bulut, Y. S. Yilmaz, and M. Demirbas. Crowdsourcing location-based queries. In *2011 IEEE International Conference on PERCOM Workshops*, pages 513–518, 2011.
[14] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, and Y. Tong. gmission: A general spatial crowdsourcing platform. *Proceedings of the VLDB Endowment*, 7(13), 2014.
[15] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, 2008.
[16] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDBJ*, 16(4), 2007.
[17] D. Deng, C. Shahabi, and U. Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proceedings of the 21st SIGSPATIAL GIS*, pages 314–323, 2013.
[18] L. Kazemi and C. Shahabi. A privacy-aware framework for participatory sensing. *SIGKDD Explorations Newsletter*, 13(1):43–51, 2011.
[19] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 21st SIGSPATIAL GIS*, pages 189–198, 2012.
[20] S. Mertens. The easiest hard problem: Number partitioning. *Computational Complexity and Statistical Physics*, page 125, 2006.
[21] M. L. Yiu and N. Mamoulis. Efficient processing of top-k dominating queries on multi-dimensional data. In *VLDB*, pages 483–494, 2007.
[22] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th SIGKDD*, 2011.
[23] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL GIS*, pages 99–108, 2010.