On Designing Quality-Aware Steering Algorithms for Large-Scale Mobile Crowdsensing

Shuo Yang, Kunyan Han, Fan Wu, Guihai Chen

Shanghai Key Laboratory of Scalable Computing and Systems
Department of Computer Science and Engineering
Shanghai Jiao Tong University, China

June 12, 2018

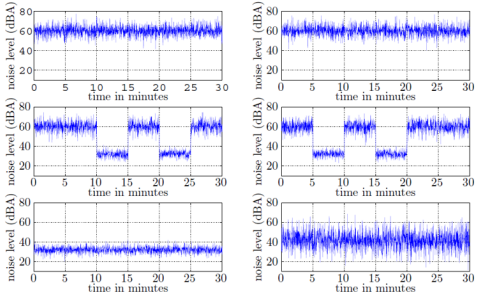
- Good afternoon, everyone. My name is XXX from XXX.
- The authors of this paper couldn't make it, so I am talking on their behalf.
- The title of this work is: on designing quality-aware steering algorithms for large-scale mobile crowdsensing.
- The authors of this work are Shuo Yang, Kunyan Han, Prof. Fan Wu, and Prof. Guihai Chen. They are from Shanghai Jiao Tong University.

Outline

- This is the outline of the presentation.
- I will first introduce the background and motivation of this work. Then, I will present the system model and the proposed algorithms. After that, I will present the evaluation, and finally conclude this work at the end of the presentation.
- First, let's take a look at the motivation.

Quality-Aware Mobile Crowdsensing

- Monitor a large sensing area
- Users' contributed data are of different quality



- A common task of mobile crowdsensing system is to monitor certain physical phenomenon of a sensing area, such as noise, temperature, or traffic congestion.
- Due to different sensing behaviors of humans, as well as heterogeneous sensing capabilities of mobile devices, the collected data are usually of uneven levels of data quality.
- An example is shown in the right figure. Six devices are asked to collect the noise information. The ground truth is about 60 dB. We can see that the top two devices contribute accurate noise estimation, while the data of other four devices are of low quality. The reasons for low quality data in the noise monitoring scenario may include: carelessly putting the phone in the pocket instead of exposing it to air, or intentionally contributing erroneous data.
- Thus, data quality is an important issue in mobile crowdsensing that should be carefully taken care of.

Quality-Aware Mobile Crowdsensing

- **Key problem:** accurate estimation for PoIs with
  - Data of uneven levels of quality
  - No ground truth knowledge

- Given the collected data of uneven levels of quality, the key problem is: how to estimate the monitored information of PoIs without the knowledge of ground truths. [Mouse Action!]
- To address the data quality issue, truth discovery problem has been widely studied. It iteratively calculates the data quality and truth estimation until convergence. It has many advantages, such as low computation complexity and accurate estimation results. Besides, it is purely unsupervised, so no prior knowledge of ground truths is needed. One disadvantage of truth discovery algorithms is that they rely on the usage of large amounts of data. Intuitively, having more data would result in higher confidence level of the estimation result, and vice versa. [Mouse Action!]
- This would cause a potential problem, that is, due to the limited number of data contributors in crowdsensing, we need to consider a tradeoff between estimation quality and coverage. Let's talk a bit more in the next slide.

Quality-Aware Mobile Crowdsensing

- **Key problem:** accurate estimation for PoIs with
  - Data of uneven levels of quality
  - No ground truth knowledge
- Truth Discovery Algorithms
  - Iterative quality estimation and data aggregation
  - Pro:
    - ☺ Efficient computation
    - ☺ Accurate estimation
    - ☺ Purely unsupervised
  - Con:
    - ☹ Need a lot of data

- Given the collected data of uneven levels of quality, the key problem is: how to estimate the monitored information of PoIs without the knowledge of ground truths. [Mouse Action!]
- To address the data quality issue, truth discovery problem has been widely studied. It iteratively calculates the data quality and truth estimation until convergence. It has many advantages, such as low computation complexity and accurate estimation results. Besides, it is purely unsupervised, so no prior knowledge of ground truths is needed. One disadvantage of truth discovery algorithms is that they rely on the usage of large amounts of data. Intuitively, having more data would result in higher confidence level of the estimation result, and vice versa. [Mouse Action!]
- This would cause a potential problem, that is, due to the limited number of data contributors in crowdsensing, we need to consider a tradeoff between estimation quality and coverage. Let's talk a bit more in the next slide.

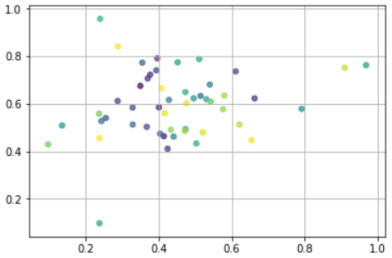
Quality-Aware Mobile Crowdsensing

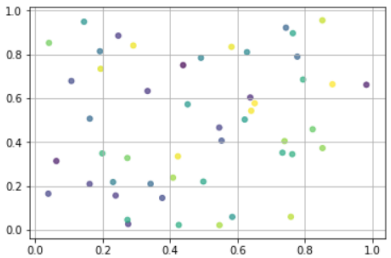
- **Key problem:** accurate estimation for PoIs with
  - Data of uneven levels of quality
  - No ground truth knowledge
- Truth Discovery Algorithms
  - Iterative quality estimation and data aggregation
  - Pro:
    - ☺ Efficient computation
    - ☺ Accurate estimation
    - ☺ Purely unsupervised
  - Con:
    - ☹ Need a lot of data
- **Potential problem:** Tradeoff between estimation quality and coverage (next page)

- Given the collected data of uneven levels of quality, the key problem is: how to estimate the monitored information of PoIs without the knowledge of ground truths. [Mouse Action!]
- To address the data quality issue, truth discovery problem has been widely studied. It iteratively calculates the data quality and truth estimation until convergence. It has many advantages, such as low computation complexity and accurate estimation results. Besides, it is purely unsupervised, so no prior knowledge of ground truths is needed. One disadvantage of truth discovery algorithms is that they rely on the usage of large amounts of data. Intuitively, having more data would result in higher confidence level of the estimation result, and vice versa. [Mouse Action!]
- This would cause a potential problem, that is, due to the limited number of data contributors in crowdsensing, we need to consider a tradeoff between estimation quality and coverage. Let's talk a bit more in the next slide.

Quality versus Coverage

- Tradeoff between *estimation quality* and *coverage*
  - **Densely distributed:** high estimation quality, low coverage
  - **Sparsely distributed:** low estimation quality, high coverage
  - Proof can be found in the paper.





- We observe that, for each PoI, having more data for the PoI is likely to produce an estimation result with a lower variance, and thus a higher confidence level. In this case, maximizing the coverage of the sensing area and improving the estimation result tend to be conflicting. Please refer to the paper for proof.
- Intuitively, when the user distribution is dense, especially when the users tend to aggregate towards a small number of locations, each of these locations should receive a plenty of data submissions and thus the estimation results of these locations can have low variances, but the coverage of the entire area is likely to be poor.
- On the contrary, when the users are sparsely distributed, the system should have a high coverage ratio, but the estimation result of each location may have high variance, causing the confidence of the estimation results unsatisfactory.
- Thus, specific coordination of the users is required to balance the estimation quality and coverage.
- Unfortunately, none of the previous works have considered such a quality-aware user coordination problem.

This Work

- Define *effectiveness* of crowdsensing system based on both *estimation quality* and *coverage*
- Take advantage of users' need of navigation, design sensing paths to maximize *effectiveness* (without causing detours)

- In this work, we consider the tradeoff of estimation quality and coverage. We tend to optimize the overall effectiveness of the location-dependent crowdsensing system, where the term "effectiveness" is measured by taking both the estimation quality and sensing coverage into consideration.
- We observe that people often need to use navigation tools, such as Google Map, Baidu Ditu, to find a path to somewhere. We take advantage of the users' frequent needs of these navigation paths, and propose to steer the users' movements to improve the system's overall effectiveness by designing each user a sensing path, without causing unnecessary detours to the users.

Outline

- Now, let's take a look at the model of the system.

Crowdsensing Model: Static

- For each user i:
  - **Reliability:** w_i degree of trustworthiness
  - **Error:** $\epsilon_i \sim N(0, \sigma_i^2)$ difference between data and ground truth

- Now, we take a look at our crowdsensing model. Let's forget the time factor for now. For each user, we model the data quality of the user i by considering her reliability w_i and error ϵ_i. The reliability metric measures the degree of quality of trustworthiness of a user. It can be calculated by applying existing truth discovery algorithms to users' historical data. The random variable ϵ_i measures the difference between the user i's data and the ground truth. We use the Gaussian distribution to model the errors. [Mouse Action!]
- We assume that for each Point of Interest, there is a set of users contributing data to it, which is denoted by $\mathcal{S}_j$. The data from user i to PoI j is denoted by x_j^i. [Mouse Action!]
- After having users' reliability information, we can calculate the estimation result for each PoI by using a widely-used weighted average method. [Mouse Action!]
- Then, we can calculate the estimation error for PoI j. It also follows a Gaussian distribution with mean 0 and variance σ_j^2. A smaller variance means a higher probability that the error is close to zero. In the rest of the presentation, we use the variance σ_j^2 to denote the estimation quality of PoI j.

Crowdsensing Model: Static

- For each user i:
  - **Reliability:** w_i degree of trustworthiness
  - **Error:** $\epsilon_i \sim N(0, \sigma_i^2)$ difference between data and ground truth
- For each PoI j
  - Set of data contributors: $\mathcal{S}_j$
  - Data from each contributor $i \in \mathcal{S}_j$: x_j^i

- Now, we take a look at our crowdsensing model. Let's forget the time factor for now. For each user, we model the data quality of the user i by considering her reliability w_i and error ϵ_i. The reliability metric measures the degree of quality of trustworthiness of a user. It can be calculated by applying existing truth discovery algorithms to users' historical data. The random variable ϵ_i measures the difference between the user i's data and the ground truth. We use the Gaussian distribution to model the errors. [Mouse Action!]
- We assume that for each Point of Interest, there is a set of users contributing data to it, which is denoted by $\mathcal{S}_j$. The data from user i to PoI j is denoted by x_j^i. [Mouse Action!]
- After having users' reliability information, we can calculate the estimation result for each PoI by using a widely-used weighted average method. [Mouse Action!]
- Then, we can calculate the estimation error for PoI j. It also follows a Gaussian distribution with mean 0 and variance σ_j^2. A smaller variance means a higher probability that the error is close to zero. In the rest of the presentation, we use the variance σ_j^2 to denote the estimation quality of PoI j.

Crowdsensing Model: Static

- For each user i:
  - **Reliability:** w_i degree of trustworthiness
  - **Error:** $\epsilon_i \sim N(0, \sigma_i^2)$ difference between data and ground truth
- For each PoI j
  - Set of data contributors: $\mathcal{S}_j$
  - Data from each contributor $i \in \mathcal{S}_j$: x_j^i
  - **Estimation result**:

$$\hat{x}_j = \frac{\sum_{i \in \mathcal{S}_j} w_i \cdot x_j^i}{\sum_{i \in \mathcal{S}_j} w_i}$$

- Now, we take a look at our crowdsensing model. Let's forget the time factor for now. For each user, we model the data quality of the user i by considering her reliability w_i and error ϵ_i. The reliability metric measures the degree of quality of trustworthiness of a user. It can be calculated by applying existing truth discovery algorithms to users' historical data. The random variable ϵ_i measures the difference between the user i's data and the ground truth. We use the Gaussian distribution to model the errors. [Mouse Action!]
- We assume that for each Point of Interest, there is a set of users contributing data to it, which is denoted by $\mathcal{S}_j$. The data from user i to PoI j is denoted by x_j^i. [Mouse Action!]
- After having users' reliability information, we can calculate the estimation result for each PoI by using a widely-used weighted average method. [Mouse Action!]
- Then, we can calculate the estimation error for PoI j. It also follows a Gaussian distribution with mean 0 and variance σ_j^2. A smaller variance means a higher probability that the error is close to zero. In the rest of the presentation, we use the variance σ_j^2 to denote the estimation quality of PoI j.

Crowdsensing Model: Static

- For each user i:
  - **Reliability:** w_i degree of trustworthiness
  - **Error:** $\epsilon_i \sim N(0, \sigma_i^2)$ difference between data and ground truth
- For each PoI j
  - Set of data contributors: $\mathcal{S}_j$
  - Data from each contributor $i \in \mathcal{S}_j$: x_j^i
  - **Estimation result**:

$$\hat{x}_j = \frac{\sum_{i \in \mathcal{S}_j} w_i \cdot x_j^i}{\sum_{i \in \mathcal{S}_j} w_i}$$

  - **Estimation error:**

$$\epsilon_j = \frac{\sum_{i \in \mathcal{S}_j} w_i \epsilon_i}{\sum_{i \in \mathcal{S}_j} w_i} \sim N(0, \sigma_j^2),$$

where $\sigma_j^2 = \frac{\sum_{i \in \mathcal{S}_j} w_i^2 \cdot \sigma_i^2}{(\sum_{i \in \mathcal{S}_j} w_i)^2}$ estimation quality

- Now, we take a look at our crowdsensing model. Let's forget the time factor for now. For each user, we model the data quality of the user i by considering her reliability w_i and error ϵ_i. The reliability metric measures the degree of quality of trustworthiness of a user. It can be calculated by applying existing truth discovery algorithms to users' historical data. The random variable ϵ_i measures the difference between the user i's data and the ground truth. We use the Gaussian distribution to model the errors. [Mouse Action!]
- We assume that for each Point of Interest, there is a set of users contributing data to it, which is denoted by $\mathcal{S}_j$. The data from user i to PoI j is denoted by x_j^i. [Mouse Action!]
- After having users' reliability information, we can calculate the estimation result for each PoI by using a widely-used weighted average method. [Mouse Action!]
- Then, we can calculate the estimation error for PoI j. It also follows a Gaussian distribution with mean 0 and variance σ_j^2. A smaller variance means a higher probability that the error is close to zero. In the rest of the presentation, we use the variance σ_j^2 to denote the estimation quality of PoI j.

Crowdsensing Model: Dynamic

- Manhattan Grid Model
- Each intersection as a PoI
- In each round t and for each PoI j
  - Data contributors: $\mathcal{S}_{j,t}$
  - Estimation quality: $\sigma^2_{j,t}$
- For each PoI j:
  - Non-negative **value:** v_j
  - **Required estimation quality:** ξ^2, so that: $\sigma^2_{j,t} \leq \xi^2$

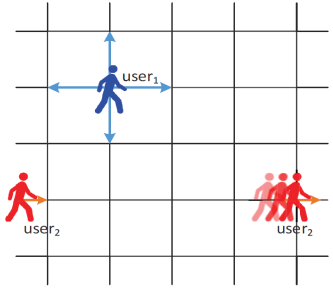


Figure: Manhattan Grid Map

- Now, let's consider a dynamic crowdsensing scenarios, where there are multiple rounds, and users can move within the sensing area.
- We use a Manhattan Grid Map to model the sensing area. As shown in the figure, the squared area is discretized into many small grids by a number of horizontal and vertical streets. The model has been widely used to simulate the map of an urban area, since the roads in well-designed metropolitan areas are likely to be perpendicular to each other. We treat each intersection of roads as a PoI.
- We assume that the crowdsensing campaign has T rounds, and the duration of each round is the same. The users may walk along the streets, and alter walking directions at intersections. For simplicity, we assume that all the users have the same walking speed, that is, a unit length per round.
- Different from the static model, now, we consider the data contribution, truth estimation, and estimation quality in each round. Thus, the notations are associated with an extra subscript t.
- Each PoI has a non-negative value, which means the utility of obtaining the information of the PoI. Along with the value, each PoI also has an estimation quality requirement ξ^2 (pronounced as xi), which is the maximum allowable variance of the estimation error to guarantee a required estimation confidence.

Crowdsensing Model: Effectiveness

- Let

$$u(\mathcal{S}_{j,t}) = \begin{cases} 1 & \text{if } \mathcal{S}_{j,t} \neq \emptyset \text{ and } \sigma^2_{j,t} \leq \xi^2, \\ 0 & \text{otherwise.} \end{cases}$$

covered & meet quality requirement

- Then, *effectiveness* of crowdsensing system is defined as the total value achieved during the entire sensing period, *i.e.*,

$$\sum_t \sum_j v_j \cdot u(\mathcal{S}_{j,t}).$$

- In each round t, the platform can obtain the PoI j's value v_j only if the PoI j is covered and the variance of the estimation error is no larger than ξ^2. We use the function $u(\mathcal{S}_{j,t})$ to denote if the platform can obtain the value.
- Given the function $u()$, we define the term of effectiveness as: the total value achieved in the sensing area during T rounds.

Quality-Aware Steering Problem

- *Path* of each user is a sequence of locations, where consecutive locations are either adjacent or the same.

Definition 1: (QASP)

The quality-aware steering problem (QASP) is to path p_i for each user $i \in \mathcal{N}$, so as to maximize the crowdsensing system's effectiveness, *i.e.*,

$$\max_{\{p_i\}} \sum_t \sum_j v_j \cdot u(\mathcal{S}_{j,t}),$$

- Next, we define the path of each user as the the sequence of locations, where the locations in consecutive rounds are either the same or adjacent.
- The objective of the crowdsensing platform is to design a path for each user so as to maximize the crowdsensing system's effectiveness. We refer to this problem as Quality-Aware Steering Problem (QASP). [Mouse Action!]
- Depending on the constraints of the users' paths, the problem can be further categorized into three different scenarios.
- We first consider QASP, where the users' initial locations and destinations are not specified. Then, we consider two special instances of QASP, namely, QASP-S and QASP-SD. QASP-S considers the scenarios where only the users' starting locations are specified beforehand, while QASP-SD assumes both the users' initial locations and destinations are specified.

Quality-Aware Steering Problem

- *Path* of each user is a sequence of locations, where consecutive locations are either adjacent or the same.

Definition 1: (QASP)

The quality-aware steering problem (QASP) is to path p_i for each user $i \in \mathcal{N}$, so as to maximize the crowdsensing system's effectiveness, *i.e.*,

$$\max_{\{p_i\}} \sum_t \sum_j v_j \cdot u(\mathcal{S}_{j,t}),$$

We consider three different scenarios:

	Specified **S**tarting locations	Specified **D**estinations
QASP	×	×
QASP-S	√	×
QASP-SD	√	√

- Next, we define the path of each user as the the sequence of locations, where the locations in consecutive rounds are either the same or adjacent.
- The objective of the crowdsensing platform is to design a path for each user so as to maximize the crowdsensing system's effectiveness. We refer to this problem as Quality-Aware Steering Problem (QASP). [Mouse Action!]
- Depending on the constraints of the users' paths, the problem can be further categorized into three different scenarios.
- We first consider QASP, where the users' initial locations and destinations are not specified. Then, we consider two special instances of QASP, namely, QASP-S and QASP-SD. QASP-S considers the scenarios where only the users' starting locations are specified beforehand, while QASP-SD assumes both the users' initial locations and destinations are specified.

Outline

- In this section, we present our solutions to QASP, QASP-S, and QASP-SD respectively.
- Let's first take a look at QASP.

QASP

Theorem 1

For QASP, there must exist an optimal user deployment, where users are static.

Thus, the solution to QASP can be found by solving the following problem.

- In QASP, users do not specify specify their initial locations or destinations, so that the users can start and end the crowdsensing task anywhere within the sensing area. This is a common assumption, since many platforms only require the users to be located within the sensing area before the crowdsensing task starts. We can prove that, in this case, there must exist an optimal user steering strategy where all the users are static. [Mouse Action!]
- In this case, the solution to the original QASP problem can be found by solving the simplified static QASP, which is to assign each user a location to maximize the platform's single-rounded effectiveness.
- We can prove that the simplified QASP is NP-hard by reducing it from the bin packing problem.

QASP

Theorem 1

For QASP, there must exist an optimal user deployment, where users are static.

Thus, the solution to QASP can be found by solving the following problem.

Definition 2: (Simplified QASP)

The simplified static QASP is to assign each user i a location z_i so as to maximize the platform's single-round effectiveness, *i.e.*,

$$\max_{\{z_i\}} \sum_j v_j \cdot u(\mathcal{S}_j)$$

Theorem 2

The simplified QASP is NP-hard.

- In QASP, users do not specify specify their initial locations or destinations, so that the users can start and end the crowdsensing task anywhere within the sensing area. This is a common assumption, since many platforms only require the users to be located within the sensing area before the crowdsensing task starts. We can prove that, in this case, there must exist an optimal user steering strategy where all the users are static. [Mouse Action!]
- In this case, the solution to the original QASP problem can be found by solving the simplified static QASP, which is to assign each user a location to maximize the platform's single-rounded effectiveness.
- We can prove that the simplified QASP is NP-hard by reducing it from the bin packing problem.

QASP Algorithm

Algorithm 1: Algorithm for Simplified QASP

Sort users and locations
Initial windows size $C \leftarrow 1$
repeat
  $\mathcal{W} \leftarrow$ leftmost C users
  repeat
    if *users in $\mathcal{W}$ meet quality requirement* **then**
      Let users in $\mathcal{W}$ to sense next location
      $\mathcal{W} \leftarrow$ next C users to the right
    else
      Move the window one user to the right
  until *the window reached the rightmost user*;
  $C \leftarrow C + 1$ increase windows size by 1
until *no further assignment can be made*;

- Greedy-based sliding window search
- **Time complexity:** $O(n^2)$

- We present an efficient algorithm to solve the problem.
- The intuitive idea of the algorithm is a greedy-sliding windows search.
- We first sort users and locations according to estimation quality and value respectively. Then, we use sliding window to find which users should cover which location. The initial size of the sliding windows is 1. We search from the leftmost user to see if one user alone can meet the estimation quality requirement of each location. After the window reaches the rightmost user, we increase windows size by 1. The search terminates when no further assignment can be made. Detailed explanations can be found in the paper.
- The time complexity of the algorithm is $O(n^2)$ time complexity, where n is the number of users.

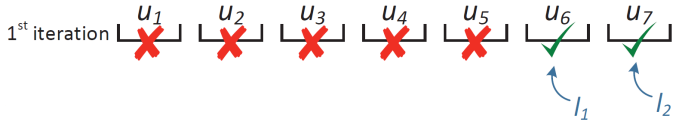
QASP Example

- Seven users, σ_i^2 are $7, 6, \ldots, 1$, and $\forall i, w_i = 1$.
- Four Pols $v_1 \geq v_2 \geq v_3 \geq v_4$, required estimation quality $\xi^2 = 2$.
- **Estimation quality requirement:** $\frac{\sum_{i \in \mathcal{W}} \sigma_i^2}{|\mathcal{W}|^2} \leq \xi^2$
- Execution process:

- Let's take a look at a simple example. Suppose there are seven users, and the values of their estimation quality are from 7 to 1. Their reliability parameters are 1. There are four locations, and the required estimation quality is 2. A user set can meet the estimation quality requirement if the sum of their estimation quality divided by the size square is no larger than the quality requirement.
- The algorithm works as follows: [Mouse Action]
- In the first iteration, the window size is 1. It searches from left to right to see which user alone can satisfy the quality requirement. It turns out that u_6 and u_7 satisfy the requirement, so we let u_6 to sense location 1 and u_7 to sense location 2. [Mouse Action]
- Then, we increase the window size by one, and searches from left to right to see which two users together can satisfy the quality requirement. We can see that only u_4 and u_5 together can do. So, we let u_4 and u_5 together sense location 3. [Mouse Action]
- After that, we increase the window size to 3 and redo the search. We find that u_1, u_2, and u_3 together can meet the quality requirement. So, we let these three users together sense the location 4.
- Till now, no further assignment can be made. Thus, the algorithm terminates.

QASP Example

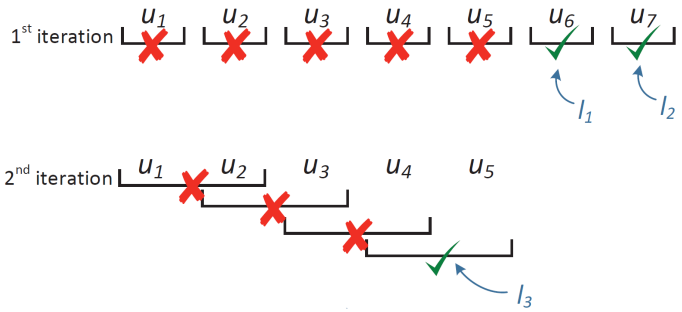
- Seven users, σ_i^2 are $7, 6, \ldots, 1$, and $\forall i, w_i = 1$.
- Four Pols $v_1 \geq v_2 \geq v_3 \geq v_4$, required estimation quality $\xi^2 = 2$.
- **Estimation quality requirement:** $\frac{\sum_{i \in \mathcal{W}} \sigma_i^2}{|\mathcal{W}|^2} \leq \xi^2$
- Execution process:



- Let's take a look at a simple example. Suppose there are seven users, and the values of their estimation quality are from 7 to 1. Their reliability parameters are 1. There are four locations, and the required estimation quality is 2. A user set can meet the estimation quality requirement if the sum of their estimation quality divided by the size square is no larger than the quality requirement.
- The algorithm works as follows: [Mouse Action]
- In the first iteration, the window size is 1. It searches from left to right to see which user alone can satisfy the quality requirement. It turns out that u_6 and u_7 satisfy the requirement, so we let u_6 to sense location 1 and u_7 to sense location 2. [Mouse Action]
- Then, we increase the window size by one, and searches from left to right to see which two users together can satisfy the quality requirement. We can see that only u_4 and u_5 together can do. So, we let u_4 and u_5 together sense location 3. [Mouse Action]
- After that, we increase the window size to 3 and redo the search. We find that u_1, u_2, and u_3 together can meet the quality requirement. So, we let these three users together sense the location 4.
- Till now, no further assignment can be made. Thus, the algorithm terminates.

QASP Example

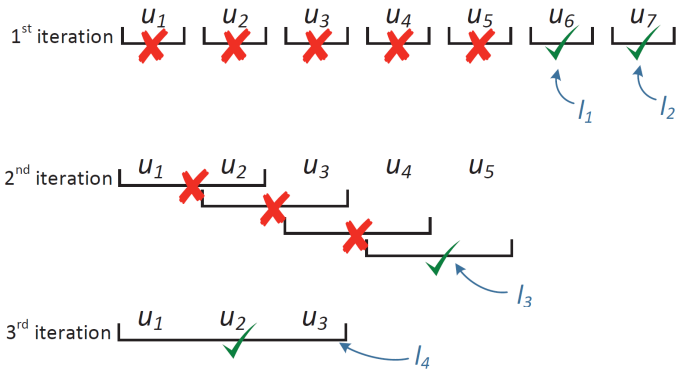
- Seven users, σ_i^2 are $7, 6, \ldots, 1$, and $\forall i, w_i = 1$.
- Four Pols $v_1 \geq v_2 \geq v_3 \geq v_4$, required estimation quality $\xi^2 = 2$.
- **Estimation quality requirement:** $\frac{\sum_{i \in \mathcal{W}} \sigma_i^2}{|\mathcal{W}|^2} \leq \xi^2$
- Execution process:



- Let's take a look at a simple example. Suppose there are seven users, and the values of their estimation quality are from 7 to 1. Their reliability parameters are 1. There are four locations, and the required estimation quality is 2. A user set can meet the estimation quality requirement if the sum of their estimation quality divided by the size square is no larger than the quality requirement.
- The algorithm works as follows: [Mouse Action]
- In the first iteration, the window size is 1. It searches from left to right to see which user alone can satisfy the quality requirement. It turns out that u_6 and u_7 satisfy the requirement, so we let u_6 to sense location 1 and u_7 to sense location 2. [Mouse Action]
- Then, we increase the window size by one, and searches from left to right to see which two users together can satisfy the quality requirement. We can see that only u_4 and u_5 together can do. So, we let u_4 and u_5 together sense location 3. [Mouse Action]
- After that, we increase the window size to 3 and redo the search. We find that u_1, u_2, and u_3 together can meet the quality requirement. So, we let these three users together sense the location 4.
- Till now, no further assignment can be made. Thus, the algorithm terminates.

QASP Example

- Seven users, σ_i^2 are $7, 6, \ldots, 1$, and $\forall i, w_i = 1$.
- Four Pols $v_1 \geq v_2 \geq v_3 \geq v_4$, required estimation quality $\xi^2 = 2$.
- **Estimation quality requirement:** $\frac{\sum_{i \in \mathcal{W}} \sigma_i^2}{|\mathcal{W}|^2} \leq \xi^2$
- Execution process:



- Let's take a look at a simple example. Suppose there are seven users, and the values of their estimation quality are from 7 to 1. Their reliability parameters are 1. There are four locations, and the required estimation quality is 2. A user set can meet the estimation quality requirement if the sum of their estimation quality divided by the size square is no larger than the quality requirement.
- The algorithm works as follows: [Mouse Action]
- In the first iteration, the window size is 1. It searches from left to right to see which user alone can satisfy the quality requirement. It turns out that u_6 and u_7 satisfy the requirement, so we let u_6 to sense location 1 and u_7 to sense location 2. [Mouse Action]
- Then, we increase the window size by one, and searches from left to right to see which two users together can satisfy the quality requirement. We can see that only u_4 and u_5 together can do. So, we let u_4 and u_5 together sense location 3. [Mouse Action]
- After that, we increase the window size to 3 and redo the search. We find that u_1, u_2, and u_3 together can meet the quality requirement. So, we let these three users together sense the location 4.
- Till now, no further assignment can be made. Thus, the algorithm terminates.

Outline

- Now, for the QASP-S problem...

QASP-S

Definition 3: (QASP-S)

QASP-S is to design a path p_i for each user $i \in \mathcal{N}$, with fixed starting locations, so as to maximize the crowdsensing system's total effectiveness.

Theorem 3

The QASP-S is NP-hard.

- In some cases, the users' initial locations cannot be arbitrary, since some users have certain regions where they appear frequently, and are less likely to appear in distant locations. To characterize this, the platform allows each user to specify her starting location before the sensing task starts. Under this scenario, the objective of the platform is to design a path for each user, so as to maximize the system's effectiveness, where the starting location for each user is fixed.
- We can see that QASP-S problem is also NP-hard, since it is a special instance of the original QASP.

QASP-S Algorithm

Algorithm 2: Algorithm for QASP-S

$\mathbf{z}_1 \leftarrow \langle z_1^{start}, z_2^{start}, \cdots, z_n^{start} \rangle$ fixed starting locations

foreach *round* $t > 1$ **do**
  $count \leftarrow 1$
  repeat
    foreach *user* i **do**
      Calculate user i's best movement based on others
    $count \leftarrow count + 1$
  until $count \geq \gamma$;

- Greedy-based repeative search
- **Time complexity:** $O(5\gamma nT)$

- The pseudo-code of the algorithm is provided. The intuition is to repeatedly find each user a better movement, which is her next round's location, under other users' movement choices. You may refer to the paper for more details.
- The time complexity of the algorithm is $O(5\gamma nT)$, where γ (pronounced as gamma) is a constant, n is the number of users, and T is the number of sensing rounds.

Outline

- As for the QASP-SD problem,

QASP-SD

Definition 4: (QASP-SD)

QASP-SD is to design a path p_i for each user $i \in \mathcal{N}$, with fixed starting locations and destinations, so as to maximize the crowdsensing system's total effectiveness.

Theorem 4

The QASP-SD is NP-hard.

- We consider a scenario where the crowdsensing platform asks each user to specify her starting locations and destinations. And then, the platform calculates each user a path without causing detour to the user.
- Since the QASP-SD is also a special instance of the original QASP, it is NP-hard.

QASP-SD Algorithm

Algorithm 3: Algorithm for QASP-SD

$\mathcal{P}^* \leftarrow \emptyset$, $count \leftarrow 0$;
repeat
  $\mathcal{P} \leftarrow \emptyset$;
  Randomly sample a user permutation o;
  foreach $i \in o$ **do**
    Calculate user i's path p_i^* that maximizes the effectiveness gain;
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_i^*\}$;
  if $U(\mathcal{P}) > U(\mathcal{P}^*)$ **then**
    $\mathcal{P}^* \leftarrow \mathcal{P}$;
  $count \leftarrow count + 1$;
until $count \geq \psi$;

- Greedy-based path search
- **Time complexity:** $O(\psi n T^2)$

- We propose a greedy-based searching algorithm. The idea is to sequentially calculate each user's optimal path based on the paths that are already been calculated.
- The time complexity is $O(\psi n T^2)$, where ψ (pronounced as sai) is a constant.

Outline

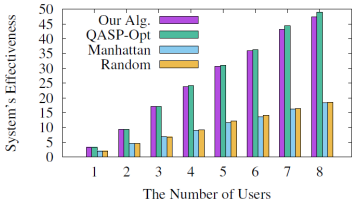
- Now, we present the evaluations of our proposed algorithms.

Evaluation Setting

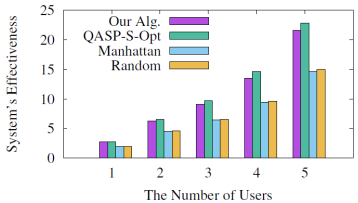
- **Synthetic dataset:** Manhattan Grid Model
  - Small-scale scenario: $n \in [1, 10], m \in [4, 64], T = 3$
  - Large-scale scenario: $n \in [50, 500], m = 10000, T \in [2, 20]$
- **Real dataset:** New York's Uber data with over 1M orders
- **Benchmarks:**
  - Random mobility model
  - Manhattan mobility model
  - Optimal solution (small-scale only)

- We use both a synthetic Manhattan model dataset and a real dataset from Uber to evaluate our algorithms.
- The benchmarks include: a random mobility model, a Manhattan mobility model, and the optimal solution.
- Since calculating optimal solutions requires exponential complexity, we only compare our algorithms to optimal solutions in small-scale synthetic dataset.

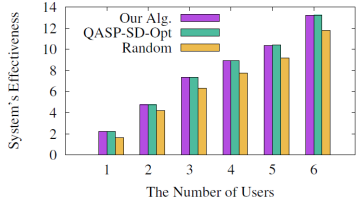
Synthetic Dataset: Small-Scale



(a) QASP



(b) QASP-S

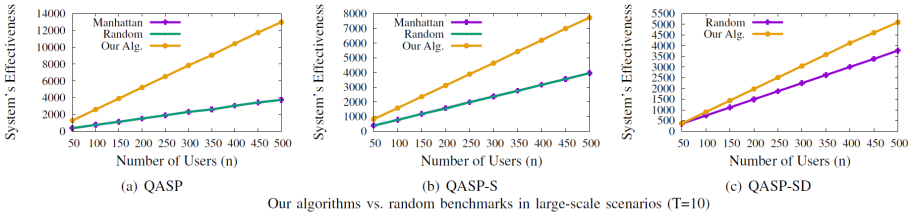


(c) QASP-SD

- Superior performance to the random mobility model and Manhattan mobility model, close to the optimal solution.

- This figure shows the evaluation result in the small-scale synthetic dataset.
- We can see that our algorithms achieves superior performance to the random and Manhattan models and close performance to the optimal solutions.

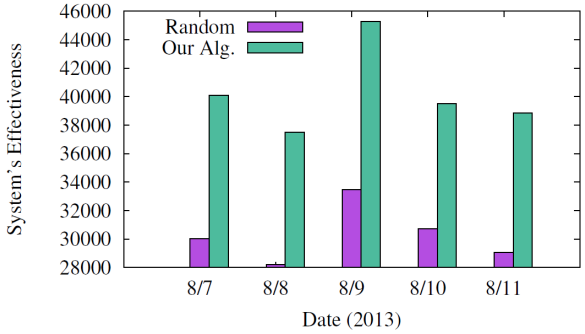
Synthetic Dataset: Large-Scale



- Dramatically improves the performance of benchmarks

- This slide shows the evaluation result in the large-scale synthetic dataset.
- The yellow line is our algorithms, and the other two are the benchmarks. We can see that for all the three scenarios, our algorithms dramatically improve the performance of the benchmarks.

Real Dataset



Evaluation with A Real Dataset

- Achieve 30% performance improvement to the benchmark

- This is the evaluation result in the real Uber dataset.
- We can see that our algorithm can achieve over 30% performance improvement to the random movement model, validating the superiority of our proposed algorithms.

Conclusion

- We consider the tradeoff between coverage and estimation quality, propose a characterization of the system's effectiveness based on both of them.
- We consider three different effectiveness maximization problems, prove the NP-hardness of them, and propose efficient algorithms for them respectively.
- Future work:
  - More practical mobility model
  - Further quantitative analysis

- Now, we conclude this work.
- In this work, we have considered the tradeoff between coverage and estimation quality. We have first defined the system's effectiveness based on coverage and estimation quality.
- Then, we have formalized the effectiveness maximization into three different problems according to three different path constraints. We have proved the NP-hardness of these problems, and proposed efficient algorithms for them respectively.
- As for future work, we tend to consider more practical mobility models of users, and provide further quantitative analysis of our algorithms.

Question?

- Thank you!
- More questions? Contact yangshuo9999@gmail.com

- That's all. Thanks for listening!
- If you have any questions, you may contact the first author via email.