MissingFound: An Assistant System for Finding Missing Companions via Mobile Crowdsourcing

Weiqing Liu, Jing Li, Zhiqiang Zhou and Jiling He
School of Computer Science and Technology,
University of Science and Technology of China
Hefei, Anhui, 230027, P.R. China.
[E-mail : cslwqxx@mail.ustc.edu.cn, lj@ustc.edu.cn,
zzq0902@mail.ustc.edu.cn, hejl@mail.ustc.edu.cn]
*Corresponding author : Jing Li

Received April 6, 2016; accepted August 16, 2016; published October 31, 2016

Abstract

Looking for missing companions who are out of touch in public places might suffer a long and painful process. With the help of mobile crowdsourcing, the missing person's location may be reported in a short time. In this paper, we propose MissingFound, an assistant system that applies mobile crowdsourcing for finding missing companions. Discovering valuable users who have chances to see the missing person is the most important task of MissingFound but also a big challenge with the requirements of saving battery and protecting users' location privacy. A customized metric is designed to measure the probability of seeing, according to users' movement traces represented by WiFi RSSI fingerprints. Since WiFi RSSI fingerprints provide no knowledge of users' physical locations, the computation of probability is too complex for practical use. By parallelizing the original sequential algorithms under MapReduce framework, the selecting process can be accomplished within a few minutes for 10 thousand users with records of several days. Experimental evaluation with 23 volunteers shows that MissingFound can select out the potential witnesses in reality and achieves a high accuracy (76.75% on average). We believe that MissingFound can help not only find missing companions, but other public services (e.g., controlling communicable diseases).

Keywords: Mobile crowdsourcing, MapReduce, logical localization, WiFi RSSI fingerprint

This work is financially supported by the National High-Technology Research and Development Program (863 Program) of China under Grant 2014AA01A302.

1. Introduction

People commonly get separated in crowded public places and lose touch with their companions, especially vulnerable people such as young children and forgetful old people. For instance, there were reports that 72 children were found missing in a zoo during 5 days [1]. By broadcasting announcements, reviewing surveillance video or asking passers-by, the missing people are likely to be found, but the process costs hours or even days. Shortening the finding process can largely reduce the risk to missing vulnerable people, and alleviate the anxiety to be suffered. As the missing people are usually out of touch (e.g., their mobile phones are out of battery), crowdsourcing may help shorten the finding process. By asking mobile users, the location of the missing person might be reported in a short period of time. The system AMBER Alert [2] is originated in the United States and designed to instantly galvanize the entire community to assist in the search of abducted children. To avoid both false alarms and having alerts ignored as a "wolf cry", only abductions confirmed by polices can issue extensive alerts via emails, SMS text messages, social network sites (e.g., Facebook), etc. The strict criteria for issuing alerts makes AMBER Alert and similar systems achieve significant effect in abduction avoidance, but also makes them not suitable for ordinary missing cases.

In this paper, we design MissingFound, a mobile crowdsourcing system which helps find missing companions, in order to shorten the finding process and cover ordinary missing cases. However, the effect of crowdsourcing largely relies on its user base. To attract a large enough user base, we could integrate MissingFound with other popular mobile applications, such as UBER, WhatsApp, etc. More importantly, the system should be effective, user-friendly and widely applicable. Consequently, MissingFound should matches several principles as follows:

1. Only a limited number of users, who have a chance to encounter the missing person, are selected to be asked, in order to avoid frequently disturbing users. MissingFound selects the most valuable users by comparing the movement traces between the asker and other users.

2. Besides back-end servers and users' mobile phones, the system should not involve any other special equipments, in order to guarantee the applicability of MissingFound. Consequently, we use the surrounding WiFi Received Signal Strengths Indicator (RSSI) recorded by users' mobile phones to represent users' movement traces.

3. The energy & bandwidth consumptions on mobile phones should be little, which are the main overhead of MissingFound. It makes GPS-based approaches impracticable.

4. The users' privacy (e.g., their precise locations) should be protected, which increases the users' willingness to use MissingFound. Different from traditional WiFi RSSI based localization techniques such as Radar [3], MissingFound does not rely on or determine the placement positions of WiFi Access Points (APs), in order to protect users' privacy.

Following these principles, MissingFound can aware the historical proximity of different movement traces by calculating the similarity of two WiFi RSSI fingerprints. Given a time period of the person missing, MissingFound figures out the potential witnesses to the missing companion near the asker's movement trace. By asking them about the missing person and collecting their responses, MissingFound can be competent for assisting people to find their missing companions.

An effective MissingFound relies on a precise estimation of the probability of seeing the missing person, as well as a quick process of user selection. However, several challenges are arised during our implementation.

1. The various WiFi environments and drifty signal strengths make it difficult to estimate a user's probability of seeing the missing person. We cope with this challenge by designing a robust metric for the estimation to adapt various WiFi environments.

2. The seeing probabilities take too long to be sequentially calculated, which makes the strategy difficult to be applied in practice. Therefore, we modify the original sequential calculation to adapt MapReduce pattern for parallel calculation, and achieve good scalability.

3. The selection of most valuable users is essentially a nonlinear integer programming problem with a large solution space (the asker might pass through some crowded places), which is hard to seek its global optimum. A heuristic method is implemented to obtain a good enough result.

We note that MissingFound is a simple-to-install system – it does not rely on meticulous calibration, special hardware, GPS localization, or any forms of fixed reference frame (such as a building's floor plan). Besides back-end servers, the system just relies on a small number of nearby WiFi APs and users' smart phones. As a result, MissingFound can be deployed to most urban areas. In the view of this advantage, we believe MissingFound can be a useful system for not only finding missing people, but also lost pets and personal properties. Moreover, MissingFound can be applied to many other public services. For instance, if there is a newly confirmed patient with a dangerous communicable disease (e.g., SARS, H7N9). MissingFound can precisely issue alerts to the people who have been appeared in close proximity to this patient or visited some places after this patient in the past few days, thereby helping control the communicable disease.

We implement the system in a real-world environment and test its performance by 23 volunteers carrying Android phones. By arranging the movement traces of these volunteers in a teaching building over one hour, we simulate a realistic scenario of the crowdsourcing process for finding 4 imaginary missing people. The experiment result shows that MissingFound is capable of selecting out the potential witnesses in reality and achieves a high accuracy (0.7675 on average). One additional experiment is designed to evaluate the effect of MissingFound when only a long time range of the missing happened can be confirmed. Another experiment tests the scalability of MissingFound under different scales of user base and back-end servers. All the results of these experiments support that MissingFound is an effective assistant system for finding missing people in a real-world environment.

Our contributions can be summarized as follows. (1) We identify the feasibility of using mobile crowdsourcing with historical proximity sensing for many applications in public places. Based on our proposed method of proximity sensing, which protects users' location privacy, other public services & applications can also benefit from mobile crowdsouring. (2) We design and implement an assistant system MissingFound for finding missing companions covering ordinary missing cases, without any pre-deployment efforts and special hardware. Historical proximity sensing model using WiFi RSSI fingerprints, combined with parallel algorithm under MapReduce framework are implemented to support an effective assistant system for finding missing people. (3) We test MissingFound on a testbed of 23 Android phones and further evaluate its scalability. The experiment results show the feasibility of MissingFound on finding the most valuable users to ask about the missing people.

The rest of the paper is organized as follows. Section 2 presents a high level system overview, followed by the models and problem formulation in Section 3. The algorithm design is presented in Section 4, while the evaluation of these algorithms is described in Section 5. Section 6 further discusses some aspects of MissingFound, while Section 7 surveys the related work. The paper concludes with a brief summary in Section 8.

2. System Overview

Fig. 1 shows the system overview of MissingFound. We then briefly describe the system components. The MissingFound client periodically checks the WiFi scan readings from user-carried mobile phones and stores the WiFi RSSI fingerprints. A WiFi RSSI fingerprint includes the signal strength of all detectable WiFi APs. Since these signal strengths are different from one location to another, a series of WiFi RSSI fingerprints collected by a user's mobile phone can be used for representing this user's movement trace. A user's trace would be periodically uploaded to the back-end servers. Sorting in time sequence, the < time, user, wifi fingerprint > tuples for all users are stored in Hadoop Distributed File System (HDFS) on the back-end servers of MissingFound.

A user is recommended to send a report to MissingFound if he/she accidentally observes a potential missing person (e.g., a child crying alone). The system then matches the report to declared missing people on its server and notices the corresponding user if the match successed. Meanwhile, once a user realized that his/her companion is missing, this user (hereinafter called the *asker*) can submit information about his/her missing companion and the possible time range of the missing happened to MissingFound. We call the segment of the asker's movement trace in this time range as a *target trace*. By checking database, MissingFound lists all matchable reports of potential missing people for the asker. However, in most cases, there is no matchable report. In these cases, MissingFound analyzes relevant time-sequenced < time, user, wifi fingerprint > tuples to select *k* users to ask, who have the high probability of seeing the missing person (hereinafter called the *seeing probability*).

To select the *k* most valuable users, users' seeing probabilities are estimated according to their movement trace and the asker's target trace. The probability follows a simple assumption: if a user once was appeared in close proximity to the asker's target trace, this user might notice the missing person. But if a user left a place before the asker visited there, this user has no chance to see the missing person. Each data point (i.e., a record tuple) in the target trace represents the potential time and place of the missing happend. By estimating every user's seeing probability to a target point (i.e., a data point in the target trace), the seeing probability sum of any *k* combination of users to this target point can be estimated. Based on these seeing probability sum, an optimization problem are modeled to obtain high overall seeing probability and balance the seeing probability among different target points. MissingFound gathers the responses from these users and send them to the asker and/or a police station.

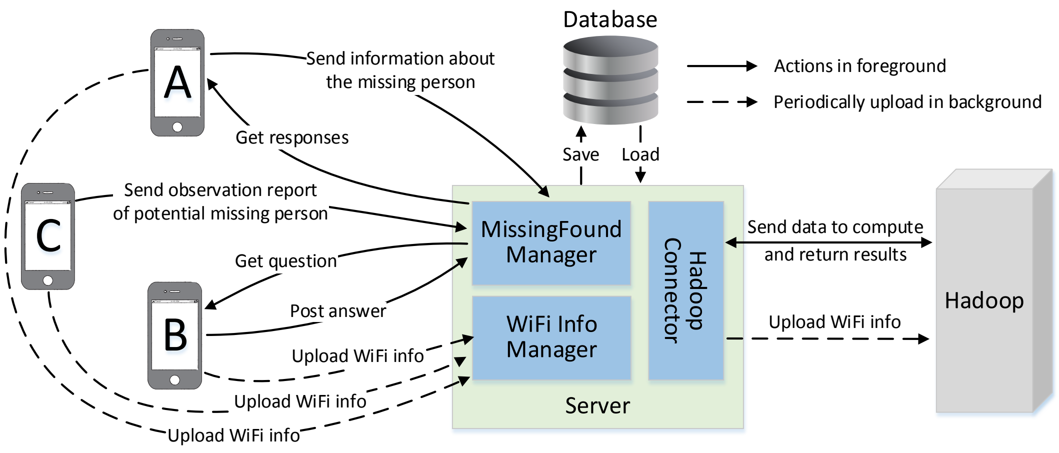


Fig. 1. System overview: A is the companion of the missing person, B is one of the selected user to be asked, C is the user who found a potential missing person

The time spent in selecting users should be as short as possible. However, the computation of seeing probability is too complex and it is an iterative process, which is hard to run in parallel. Moreover, the optimization problem for selecting *k* users is a nonlinear integer programming problem, which is NP-hard. A realistic user base and *k* are too large for obtaining a global optimum for the optimization. In our implementation, the computation of users' seeing probabilities are performed concurrently, by modifying the algorithm to suit MapReduce framework. A heuristic algorithm (greedy & simulated annealing), which is also parallelized under MapReduce framework, is designed to produce a good enough combination of *k* users within a reasonable time. Moreover, some target points indicating the same position are merged, in order to reduce the amount of computation. Thanks to these strategies, when an asker requests to find his/her companion on MissingFound, he/she only need to wait several minutes for the responses from system-selected potential witnesses. Continuing this process till the asker gets enough valuable responses, so that achieving the goal of MissingFound.

3. Models and Problem Formulation

As mentioned in the above section, all users' movement traces are periodically uploaded to the back-end servers of MissingFound. These records are sorted in chronological order on the server, and there is a fixed positive integral number that represents the gap between two successive records for all users. Based on these collected movement traces, we construct necessary models and formulate the optimization problem of selecting most valuable users.

3.1 The similarity between two records, representing the seeing probability when the user and the missing person at these two locations at the same time

Users are not willing to upload their precise locations of their daily life. Our system uses the signal strength of all detectable WiFi APs from mobile phones to represent their location. This representation is able to measure the distance between two locations, but hard to reproduce the original locations of users without the knowledge of the placement location of WiFi APs. Our model do not depend on this knowledge, in order to protect users' privacy. A movement trace of a user is composed of a sequence of data points. Each data point contains a timestamp and the strengths of all detectable WiFi APs (i.e., wifi fingerprint). Inspired by the log-distance path loss (LDPL) model of the propagation of radio frequency, we explore an appropriate definition of the similarity between two records based on their wifi fingerprints. After several failed attempts, we finally define the similarity as follows:

$$A_0 = \frac{\sum_{i \in W_a \cup W_b} e^{\frac{-1 \cdot |a_i - b_i|}{10 \cdot DF}}}{|W_a \cup W_b|} \tag{1}$$

$$A_n = A_{n-1}^2 / UB \tag{2}$$

$$s_{ab} = \begin{cases} A_{SL} & A_0 < UB \\ UB & \text{otherwise} \end{cases} \tag{3}$$

where W_a and W_b are the set of detectable WiFi APs in records *a* and *b*, a_i and b_i are the strength of a same WiFi AP in them. *SL* indicates the salient level of the missing person (i.e., how easily the missing person can be recognized from the crowd), where $SL \in \mathbf{N}$. *DF* is a decay factor and *UB* is the upper bound of the seeing probability. They are set as 3.0 and 0.82 respectively in our experiments. If an AP can only be detected by one of the two records, the strength of this AP is set as a minimum default value −96 dBm for the other record, Since the definition of similarity is an empirical formula, we explain the value of *SL* and *UB* with experiments in Section 5, as well as their meanings. The time cost on computing the similarity between two records is determined by the total number of detectable WiFi APs, which is relatively stable. Therefore, we regard the computation complexity of this part as a constant.

3.2 The probability of a user becoming the first witness

We assign *R* to represent the list of all uploaded records, sorted in chronological order. For an integer $i \in \mathbf{N}$, $T(i)$ represents the timestamp of the *i*th record in the list *R*. For any two records r_i, $r_j \in R$, where $i < j$, the timestamp of record r_i is smaller than r_j (i.e., $T(i) < T(j)$). For each user *h*, his/her movement trace X_h is composed of a set of records in the list *R*, and we use $r_i \in X_h$ to represent that the record r_i is one record in user *h*'s movement trace.

Suppose the asker *c*'s mobile phone were recording $r_i \in X_c$ when his/her companion separated with asker *c*. At the time of user *h*'s mobile phone collecting $r_j \in X_h$, the possibility of the missing person still at the location (represented by r_i), can be modeled as follows:

$$e_{ij} = \begin{cases} \alpha^{T(j)-T(i)} & T(i) < T(j); \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

where $\alpha \in (0,1]$ is the possibility of the missing person staying at the place for one second. As the similarity between r_i and r_j represents the seeing probability between these two locations, the probability of user *h* noticing the missing person at the time of r_j can be defined as follows:

$$p_{ij} = e_{ij} \cdot s_{ij}. \tag{5}$$

At some crowed public places where exist plenty users of MissingFound, an easily noticed missing person (e.g., a young child crying alone at the front gate of a park) has a good chance to be noticed and reported to MissingFound and/or a police station. In order to minimize the interference of users, we tend not to ask the users at these places. Consequently, we define the probability of user *h* to be the first to notice the missing person at the time of r_j as follows:

$$\hat{p}_{ij} = \left[\prod_{k=i+1}^{j-1} \left(1 - \eta \cdot p_{ik}\right) \right] \cdot p_{ij}, \tag{6}$$

where $\eta \in (0,1)$ is the possibility that a user reports to MissingFound after notice a potential missing person. Therefore, user *h*'s potential contribution on finding the missing person, who separates with the asker at the location represented by r_i, can be defined as follows:

$$v_{ih} = \sum_{j \in X_h} \hat{p}_{ij}. \tag{7}$$

Because the precise location where the asker and the missing person separated at is hard to know, the overall potential contribution of user *h* can be defined as follows:

$$V_h = \sum_{i \in Q} v_{ih}. \tag{8}$$

where *Q* is the target trace, a subsequence of the asker's movement trace X_c , covering the time duration of the separation of the asker and missing companion happened.

As the number of one user's records is $O(N)$ and the length of *R* is $O(M \cdot N)$, so the complexity of computing a particular v_{ih} is $O(M \cdot N^2)$, where *M* and *N* follow the definition in **Table 1**. Suppose a MissingFound client collects one record every 10 seconds and records of recent few days are kept in the system, the magnitude of *N* comes to 10,000. Since people do not move all the time, those successive data points with high similarity (larger than *UB*) can be merged, in order to reduce the amount of computation. In our experiment, about 90% records of the target trace can be merged. Consequently, the magnitude of *N* can be reduced to 1,000.

Table 1. Alphabet of Complexity Analyse

Descriptions	Magnitude	Notation
Number of records in target trace	1,000	N
Number of users in the system	10,000	M
Number of users to be selected	100	K

3.3 Selecting *k* most valuable users from all *M* users

After a person is found missing, the primary function of MissingFound is selecting users to ask whether they noticed the missing person. As the number of ordinary missing people may be huge, asking all users in the system will cause a heavy burden on them, leading to the drain of users. An effective strategy of selecting *k* most valuable users should be designed.

Since every point in the target trace might be the location of the missing person, simply choosing the top *k* valuable users by Equation (8) is not a wise strategy. The users selected by this simple strategy may have high seeing probability on a same part of the target trace, but none of them has the chance to encounter the missing person if he/she is at another part of the target trace. The objective of MissingFound is to maximize the probability that existing some selected users noticed the missing person, no matter where he/she is. We will further discuss these two strategies in Section 5. We formulate this optimization problem as follows:

Given:

Q, the set of data points in the target trace;

U, the set of all available users in the system;

v_{ih} , the potential contribution of user $h \in U$ for potential missing point $r_i \in Q$ in Equation (7);

l, the factor punishing that the potential contributions are crowed at a part of the target trace, satisfying $l \in (0,1)$;

k, the number of users preferred to be selected.

$$\max \quad \sum_{i \in Q} \left(\sum_{h \in U} x_h \cdot v_{ih} \right)^l, \tag{9}$$

$$\text{subject to:} \quad \sum_{h \in U} x_h = k, x_h \in \{0,1\}, \forall h \in U. \tag{10}$$

This is a non-linear integer programming problem, which is NP-hard. If every v_{ih} is calculated and cached, the complexity of the brute force algorithm is $O(M \cdot N^2 + C_M^k \cdot k \cdot N^2)$, which calculates the performance of all combinations and chooses the best one.

4. Algorithm Design

The computation of the optimization problem defined in the above section is composed of the calculation of all relevant v_{ih} and the selection of k users from all M users. The overall complexity is $O(M \cdot N^2 + C_M^k \cdot k \cdot N^2)$, while using non-parallel brute force algorithms. Obviously, it cannot support a real world system. In this section, we study these two parts respectively to reduce the complexity using MapReduce framework and heuristic algorithm.

4.1 Overview of MapReduce

MapReduce [4] is a programming model for data processing, with inherently parallel features. MapReduce works by breaking the processing into two phases: the map phase and the reduce phase. Each phase has key-value pairs as input and output, and the programmer rewrites the map function as well as the reduce function to complete the job. The output from the map function is processed by the MapReduce framework before sent to the reduce function. This processing (called shuffling) groups and sorts the key-value pairs, the rules of which can also be specified by programmer. After doing these, the data processing can be easily executed concurrently under MapReduce framework.

4.2 Parallelizing the calculation of v_{ih}

As shown in Equation (6), the value of every v_{ih} is based on all records between r_i and r_j . In order to parallelize the calculation of v_{ih} , we store the list R in Hadoop Distributed File System (HDFS). Once a target trace Q comes as a request, we generate the subsequence R' of sequence R which contains all the records whose timestamps are larger than the timestamp of the first record in the target trace Q.

The sequence R' is firstly separated into several parts with a fixed size. Each mapper takes charge of one of these parts and produces some intermediate results. The intermediate results calculated by all mappers are grouped by different target point $r_i \in Q$. Each reducer takes charge of several groups of intermediate results and produce final results of corresponding v_{ih} . We then explain the parallel calculation of v_{ih} in detail.

Assume R' has been separated into g parts, and the dth part of sequence R' start from the $start_d$th data point in R' and end with the end_dth data point. We then formally define the dth part of sequence R' (S_d), user h's partial movement trace in S_d ($X_h^{(d)}$), the partial target trace in S_d ($Q^{(d)}$), and the partial target trace containing all first d parts of R' ($\hat{Q}^{(d)}$) as follows:

$$\begin{aligned}
&S_d = \{r_j \mid \forall r_j \in R', start_d < j < end_d\},\\
&X_h^{(d)} = \{r_j \mid \forall r_j \in X_h \cap S_d\},\\
&Q^{(d)} = \{r_i \mid \forall r_i \in Q \cap S_d\},\\
&\hat{Q}^{(d)} = \textstyle\bigcup_{k=1}^{d} Q^{(k)}.
\end{aligned}$$

It is worth mentioning that the records in each S_d keep the original chronological order.

The basic idea is splitting the calculation of $\hat{p}_{ij}$ in Equation (6) into g independent parts, and then assembling corresponding v_{ih} based on the intermediate results produced by these g parts. Refer to Equation (6), the probability $b_i^{(d)}$ represents that all data points in S_d do not report the missing person at the potential missing point $r_i \in Q$. We express it as follows:

$$b_i^{(d)} = \prod_{o=start_d}^{end_d} \left(1 - \eta \cdot p_{io}\right). \tag{11}$$

Moreover, for each partial sequence S_d , we define a partial probability $\hat{p}_{ij}^{(d)}$ for each pair (r_i, r_j) $\in \hat{Q}^{(d)} \times S_d$ as follows:

$$\hat{p}_{ij}^{(d)} = \begin{cases} \left[\prod_{k=i+1}^{j-1} \left(1 - \eta \cdot p_{ik}\right)\right] \cdot p_{ij}, & r_i \in S_d \\ \left[\prod_{k=start_d}^{j-1} \left(1 - \eta \cdot p_{ik}\right)\right] \cdot p_{ij}, & \text{otherwise} \end{cases}. \tag{12}$$

This $\hat{p}_{ij}^{(d)}$ equals to $\hat{p}_{ij}$ in Equation (6) if r_i and r_j are in the same partial sequence. Suppose $r_i \in S_f$ and $r_j \in S_d$, where $i < j$, we can modify the $\hat{p}_{ij}$ in Equation (6) as the following:

$$\hat{p}_{ij} = \hat{p}_{ij}^{(d)} \cdot \prod_{k=f}^{d-1} b_i^{(k)}. \tag{13}$$

In addition, we let $\hat{v}_{ih}^{(d)}$ denote the probability that user h sees the missing person at the potential missing point $r_i \in \hat{Q}^{(d)}$, from partial trace $X_h^{(d)}$. We model this as follows:

$$\hat{v}_{ih}^{(d)} = \sum_{j \in X_h^{(d)}} \hat{p}_{ij}^{(d)}. \tag{14}$$

After that, for any target point $r_i \in Q^{(f)}$, we can modify its v_{ih} in Equation (7) as the following:

$$v_{ih} = \hat{v}_{ih}^{(f)} + b_i^{(f)} \cdot \hat{v}_{ih}^{(f+1)} + \ldots + b_i^{(f)} \cdot b_i^{(f+1)} \cdot \ldots \cdot b_i^{(g-1)} \cdot \hat{v}_{ih}^{(g)} = \sum_{j=1}^{g} \left(\prod_{d=f}^{j-1} b_i^{(d)} \right) \cdot \hat{v}_{ih}^{(j)}. \tag{15}$$

From the modified definition of v_{ih}, it is easy to notice that the $b_i^{(d)}$ is valid if and only if $r_i \in \hat{Q}^{(f)}$. Moreover, the calculation of $b_i^{(d)}$ in Equation (11) and $\hat{v}_{ih}^{(d)}$ in Equation (14) only involve the records in S_d. Therefore, the calculation of $b_i^{(d)}$ and $\hat{v}_{ih}^{(d)}$ can be parallelized, further parallelizing the calculation of v_{ih} in Equation (15). We then describe this algorithm under MapReduce framework in detail.

At the map-side (shown in Algorithm 1), each mapper takes charge of one subsequence S_d and calculates $\hat{v}_{ih}^{(d)}$ for each pair $(i, h) \in Q^{(d)} \times U$ and $b_i^{(d)}$ for all $r_i \in \hat{Q}^{(d)}$. At the end of the map-side job (cleanup function), these intermediate results are emitted to corresponding reducer by the key-value form $< (i, start_d), (b_i^{(d)}, \{\hat{v}_{ih}^{(d)} | \forall h \in U\}) >$ for all $r_i \in \hat{Q}^{(d)}$. Here, $(i, start_d)$ is a key pair where i indicates the target data point r_i and $start_d$ stands for the subsequence in charge. It is easy to verify that the computation complexity of the map-side job is $O(M \cdot N^2/map\_num)$,where map_num represents the number of map tasks running in parallel.

In the shuffle stage, all output data of mappers are partitioned and grouped by i in the key pair. All these variables about one particular target record $r_i \in Q$ are received by a single

```
Algorithm 1: Map-side Job (Probability Calculation)
  Input: $S_d$, asker, $Q$, $\eta$
  Output: $< (i, start_d), (b_i^{(d)}, \{\hat{v}_{ih}^{(d)} | \forall h \in U\}) >$ for all
          $r_i \in \hat{Q}^{(d)}$
// Configure phase begin
forall the $b_i^{(d)}$ do $b_i^{(d)} = 1.0$;
forall the $\hat{v}_{ih}^{(d)}$ do $\hat{v}_{ih}^{(d)} = 0.0$;
// Configure phase end
// Map phase begin
foreach $r_j \in S_d$ do
    Extract user $h$, $wifi\_map$ and $T(j)$ from $r_j$;
    if $h \neq asker$ then
        foreach $r_i \in \hat{Q}^{(d)}$ do
            if $T(j) < T(i)$ then break;
            $simi$ = SimiCalc($wifi\_map$, $j.wifi\_map$);
            $\hat{v}_{ih}^{(d)} = \hat{v}_{ih}^{(d)} + simi \cdot b_i^{(d)}$;
            $b_i^{(d)} = b_i^{(d)} \cdot (1 - \eta \cdot simi)$;
        end
    end
end
// Map phase end
// CleanUp phase begin
foreach $r_i \in \hat{Q}^{(d)}$ do
    output($< (i, start_d), (b_i^{(d)}, \{\hat{v}_{ih}^{(d)} | \forall h \in U\}) >$);
end
// CleanUp phase end
```

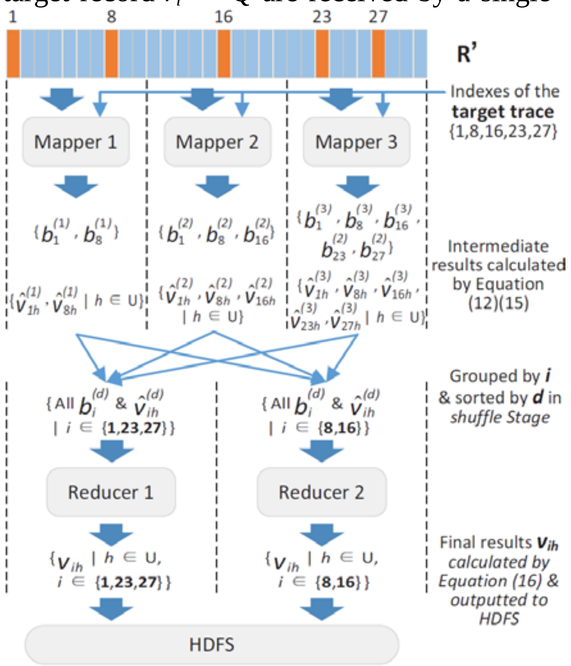

1
8
16
23
27
R'
Indexes of the target trace {1,8,16,23,27}
Mapper 1
Mapper 2
Mapper 3
Intermediate results calculated by Equation (12)(15)
Grouped by i & sorted by d in shuffle Stage
Reducer 1
Reducer 2
Final results v_{ih} calculated by Equation (16) & outputted to HDFS
HDFS

Fig. 2. An example of our parallelized calculation of v_{ih}

Algorithm 2: Reduce-side Job (Probability Calculation)

```
Input:
< $(i, start_d)$, $iterator$ < $(b_i^{(d)}, \{\hat{v}_{ih}^{(d)} \mid \forall h \in U\}$ >>
Output: < $i, \{v_{ih} \mid h \in U\}$ > for all $i \in Q$ which are
        assigned to this reducer

curr_prob = 1.0;
forall the $v_{ih}$ do $v_{ih} = 0.0$;
foreach $(b_i^{(d)}, \{\hat{v}_{ih}^{(d)} \mid \forall h \in U\}) \in iterator$ do
    // all records in the iterator have
       already sorted by d in
       increasing order
    foreach $\hat{v}_{ih}^{(d)}$ do
        $v_{ih} = v_{ih} + curr\_prob \cdot \hat{v}_{ih}^{(d)}$;
    end
    $curr\_prob = curr\_prob \cdot b_i^{(d)}$;
end
output(< $i, \{v_{ih} \mid h \in U\}$ > for all $i \in Q$ which are
assigned to this reducer);
```

Algorithm 3: Simulated Annealing (Selecting k Users)

```
Input: $k, U, t, r, \phi, \rho$
Output: set $X$ contains the selected $k$ users
$X \leftarrow$ an arbitrary solution such that, $|X| = k$;
changed = true;
while changed do
    changed = false;
    for $i = 1$ to $r$ do
        choose $x \in X$, $x' \in U - X$ randomly;
        $\delta$ = Value($X - x + x'$) − Value($X$);
        if $\delta < 0$ then
            generate random variable $\gamma$ uniformly
            distributed in (0, 1);
            if $\gamma > e^{\frac{\delta}{t}}$ then
                continue;
            end
        end
        $X = X - x + x'$; changed = true;
    end
    $t = t \cdot \phi$; $r = r \cdot \rho$;
end
output($X$);
```

reducer. For each target record r_i, all of its data are sorted by the $start_d$ in increasing order. The amount of transmitted data from one mapper to all reducer is $O(M \cdot N)$. As they are parallel transmitted in the shuffle stage, the time cost of this stage can be estimated as $O(M \cdot N / reduce\_num)$, where $reduce\_num$ is the number of reducers in this job.

At the reduce-side (shown in Algorithm 2), each reducer takes charge of map-side output for some $r_i \in Q$. As the records have already been sorted in the merge phase of shuffle stage, the elements of Equation (15) can be accumulated while reading records, which means the $\prod b_i^{(d)}$ need to be calculated only once. By assigning the calculation of the N target data points to reducers evenly, the reduce-side job can be accomplished in $O(map\_num \cdot N / reduce\_num)$.

Fig. 2 shows an example of our parallelized calculation of v_{ih}. Due to the difference of magnitude level, the time cost of all the v_{ih} calculation $O(M \cdot N^2/map\_num + M \cdot N/reduce\_num + map\_num \cdot N/reduce\_num)$ can be simplified into $O(M \cdot N^2/map\_num)$. Therefore, we can expect that, by parallelizing the calculation of v_{ih}, the complexity of this part reduces from $O(M \cdot N^2)$ to $O(M \cdot N^2/map\_num)$.

4.3 Algorithm for selecting *k* most valuable users

After calculating every v_{ih} , selecting k most valuable users from all M users is still a NP-hard problem. So we resort to heuristic algorithms.

4.3.1 Brute Force

In a small scale (e.g., selecting 3 users), it is possible to list all combinations. As stated in the complexity analysis for Function (9), the complexity of this algorithm is $O(M \cdot N^2 + C_M^k \cdot k \cdot N^2)$. However, k should be at least several dozens, which makes this method unpractical.

4.3.2 Greedy Algorithm

A greedy algorithm is designed for this problem. In the first iteration, we select the most valuable user from M candidates, by Objective Function (9) (i.e., k = 1 in constraint Equation (10)). In the following iterations, we select one user in each iteration, who contributes the largest increase of Objective Function (9), comparing with the last iteration. After k iterations, the selection of k users is accomplished. This algorithm is better than simply choosing top k

most valuable users and it runs in $O(k^2 \cdot M \cdot N)$ time, which is acceptable to MissingFound. But certainly, the greedy algorithm cannot guarantee a global optimum.

4.3.3 Simulated Annealing Algorithm

Besides the greedy algorithm, we also make use of simulated annealing to solve this problem. Simulated Annealing algorithm is an effective method to solve binary knapsack problem, which is similar to our problem. Initially, k users are randomly selected from all M candidates as the solution. In each iteration, the algorithm randomly chooses one of these k selected users and one of the M-k unselected users to exchange. If the overall value of the new solution (measured by the objective function (9)) is worse than the previous solution, there is a chance to recover from the exchange, with a certain probability. Along with the iteration processing, this probability of recovery grows bigger, which finally leads to a stable solution. This solution might jump out of a local optimum and achieve the global optimum. By executing the algorithm many times, we have a good chance to achieve the global optimum.

Algorithm 3 shows the approach. X is the set containing currently selected users. *Value(X)* is a function which returns the value of Objective Function (9) for a given solution set X. The parameter t (called temperature) is reduced by a positive factor $\varphi \in (0,1)$ during the iterations, such that the likelihood of accepting a deteriorated solution decreases as the algorithm progresses. Therefore, the simulated annealing algorithm has the opportunity to escape local optimum and achieve the global optimum. For any given value of t, some exchange trials r (repetitions) are performed. r is multiplied with a positive constant ρ, when t is decreased.

Simulated annealing is a kind of randomized local search algorithm, which was invented to avoid falling into the local optimum, but it still cannot guarantee the global optimum. Therefore, the algorithm need to be executed many times to increase the possibility of achieving the global optimum. As a probabilistic algorithm, its computation complexity is determined by the data and parameter t, r. The complexity of function *Value(X)* is $O(k \cdot N)$. By running the algorithm under MapReduce framework, the time cost can be reduced by *map_num* times (the algorithm runs in many mappers and one reducer merges their results).

MissingFound compares the results produced by greedy and simulated annealing algorithms, and chooses the best result as the final result. In Section 5, we will test the real time cost for these two algorithms.

5. Experiment and Evaluation

In this paper, we propose an assistant system for finding missing companions via mobile crowdsourcing. In our approach, the estimation of the probability of seeing the missing person is based on the similarity of two WiFi RSSI fingerprints. Meanwhile, a short period of time spent in the selection of users is crucially important for shortening the finding process. Therefore, we mainly evaluate MissingFound on the following aspects:

1. Does the similarity between two WiFi RSSI fingerprints fit the probability of seeing the missing person in reality?

2. How well does MissingFound perform in a real-world scenario?

3. How quickly does MissingFound select out most valuable users?

In this section, we examine MissingFound to answer these questions respectively.

5.1 Does the similarity between two records fit the probability of seeing the missing person?

To evaluate MissingFound, one primary work is to test and verify whether the similarity between two records (modeled in Equation (1)(2)(3)) has an appropriate definition. Since different locations have different WiFi environments, we try to design an uniform metric of the probability of seeing the missing person in various WiFi environments. We summarize 4 criteria that an appropriate metric should fulfill, listed as follows:

1. The probability of seeing the missing person has a negative correlation with the distance.
2. With the same distance to the missing person, a user in outdoor scenarios has higher probability to notice the missing person, compared with indoor scenarios.
3. If the user and the missing person are separated by an obstruction, the seeing probability should be lower than in a same distance but without the obstruction.
4. The probabilities of seeing different missing people from a same distance should be different, since some people might hard to be noticed among the crowd.

We choose 10 typical places in a campus to verify if the definition of the similarity fulfills these criteria. At each place, we slowly go through a straight line at a constant pace. By recording the WiFi RSSI fingerprints along each trace, we calculate the similarities from every data point to the very first data point in this trace. **Fig. 3** shows the distance-similarity curves of our collected data, as well as corresponding fittings and prediction bands. We can easily distinguish between the curves of indoor scenarios and outdoor scenarios. The similarity A_0 calculated by Equation (1) indeed fits the first two criteria.

In addition, we use the same method to collect data from the two sides of a wall in a building, in order to evaluate whether our approach fits the third criterion. As shown in **Fig. 4**, the similarity A_0 is sharply decreased when switching to the other side of the wall, which can demonstrate the decreasing of seeing probability (the 3rd criterion). This phenomenon is also the reason that the indoor decay rate of similarity is much higher than outdoor scenarios.

However, as shown in **Fig. 3** and **Fig. 4**, the values of A_0 is too large to indicate the probability of seeing the missing person. So the parameter *SL* is designed to indicate the salient level of the missing person. A higher *SL* represents that it is much more difficult to notice the missing person in the crowd. Calculated by Equation (2), the values can be projected into a reasonable range. Before that, the value of upper bound *UB* of similarity should be determined, which is used for judging whether two points represents the same location. We draw the time-similarity (A_0) curve for one fix point to decide the value of *UB*, showed in **Fig. 5**. The drifting of similarities at a fixed position is small and the period of waves is 24 hours, reflecting the turning on and off of some WiFi APs in daily life. According to this result, the upper bound *UB* is set at 0.82 in our experiments to guarantee the diversity of similarity.

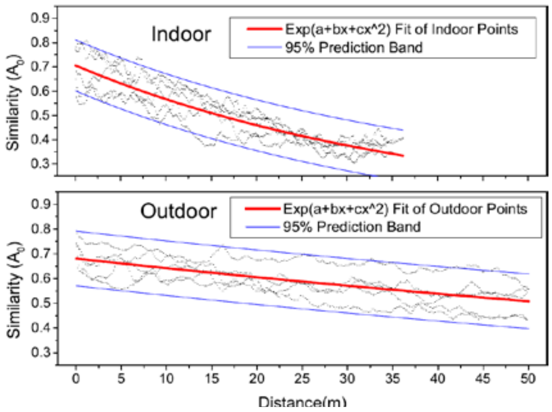


Fig. 3. Distance-similarity (A_0) curves and corresponding fits & prediction bands under *indoor* and *outdoor* scenarios.

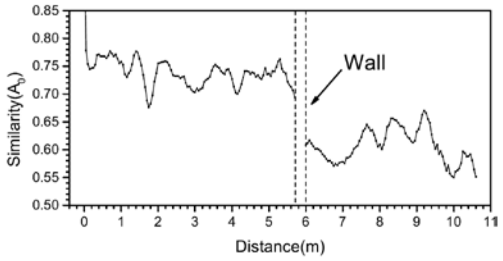


Fig. 4. Distance-similarity (A_0) curves for passing through a wall.

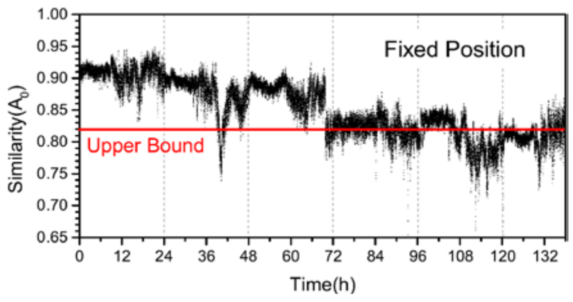


Fig. 5. Time-similarity (A_0) at a fixed position.

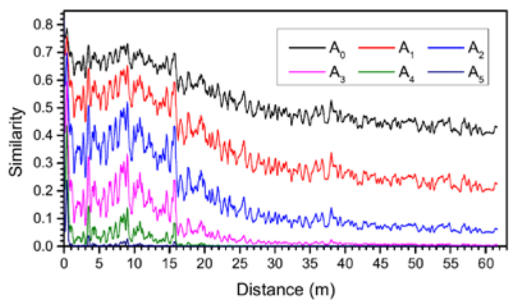


Fig. 6. Distance-similarity with different *SL*.

Based on the determined upper bound *UB*, we draw the distance-similarity curves for different *SL* values in an outdoor scene (shown in **Fig. 6**). We recommend to set $SL \in [3,5]$, in order to indicate the salient level of different missing people and fulfill the 4 th criterion. In summary, we believe that the definition of the similarity between two data points can properly indicate the probability of seeing the missing person with different salient levels under both indoor and outdoor scenarios.

5.2 How well does MissingFound perform in a real-world scenario?

After evaluating the similarity model between two data points, we test and verify the probability model of seeing the missing person in realistic scenarios. We recruited 23 volunteers and arranged their movement traces in the main teaching building of the campus over one hour, in order to simulate a real-world scenario. Each volunteer carries an Android phone with a MissingFound client pre-installed. These volunteers are initially distributed in different classrooms of the building when the experiment begins. As the experiment progresses, they move from one place to another along pre-arranged paths, and stay for a pre-determined time before next movement, according to their respective movement schedules. One of these volunteers plays the asker who can not find his/her companion, and this volunteer's movement trace is used as the target trace in MissingFound. **Fig. 7** shows a map of our experiment area. In this part of evaluation, we first assesses the performance of MissingFound when the asker knows the exact place where the missing person separated with the asker. After that, we explain the experiment used for testing the performance of our probability model while only a time range of the missing can be confirmed by the asker.

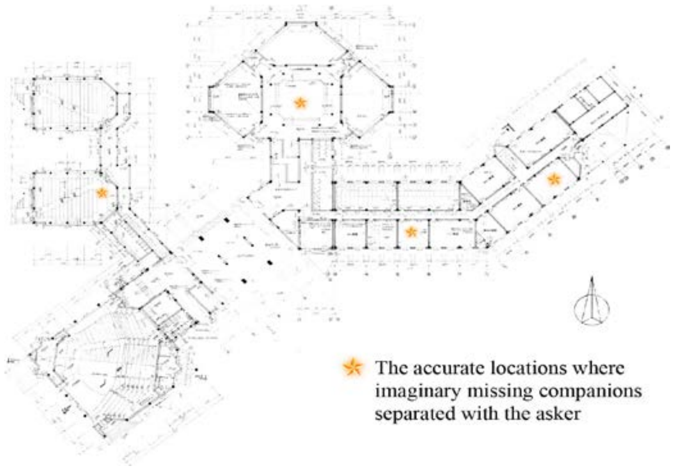

The accurate locations where imaginary missing companions separated with the asker

Fig. 7. Map of our experiment area.

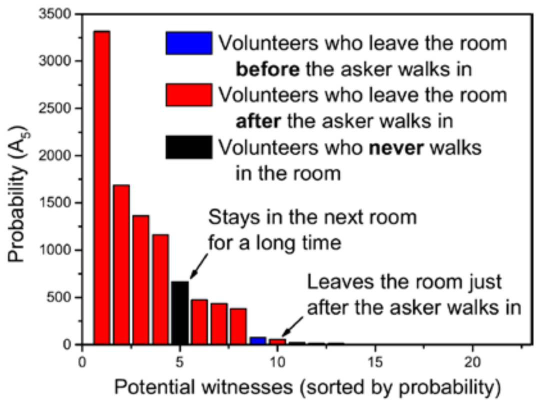


Fig. 8. Probability (A_5) histogram for each volunteer (sorted by value) when accurate losing place is confirmed.

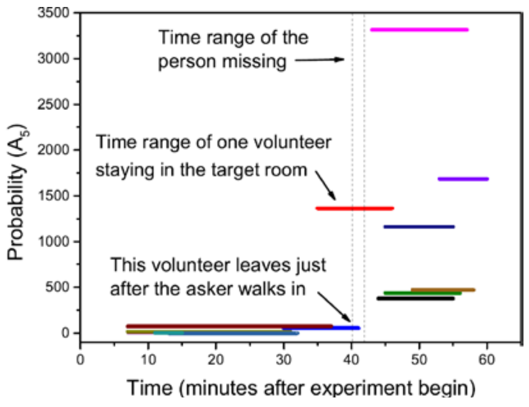


Fig. 9. Probability (A_5) for each volunteer in group 1 & 2, with the time ranges that they stay in the target place.

5.2.1 Accurate missing place can be confirmed

To evaluate the performance of MissingFound in this situation, we choose 4 locations from the asker's movement trace to represent the places where the asker separate with his/her 4 imaginary missing people respectively. After the experiment, we collect WiFi RSSI fingerprints from all 23 volunteers' mobile phones and calculate each volunteer's probability of seeing the imaginary missing people (V_h), using Equation (8). In the calculation, a short time range (two minutes) of the asker's trace was provided to stand for the accurate location where the asker separated with the imaginary missing person.

Excepting the asker, all 22 volunteers are classified into three groups according to their movement schedules. (1) Volunteers who left the place **before** the asker and the imaginary missing person walked in. They have no chance to see the missing person. (2) Volunteers who left the place **after** the asker and the imaginary missing person walked in. They have the chance to notice the missing person. (3) Volunteers who **never** visited this place according to their movement schedules. They also have no chance to see the missing person.

Fig. 8 and **Fig. 9** demonstrate the results of these three groups of volunteers for one imaginary missing people, who is supposed to be separated with the asker in a small classroom (8m*10m). Obviously, selecting out the second group of volunteers is the core goal of MissingFound. As shown in **Fig. 8**, 7 of the 8 volunteers who are in the second group have high probability value, compared with other two groups. By checking the movement schedules, we found that the volunteer, who has the lowest probability in the second group (the 10th user in **Fig. 8**), left the classroom just after the asker walked in. This small value indeed indicates the low probability to see the missing person in reality. The volunteer with 5th largest probability in **Fig. 8** was a member in group 3, who never visited the classroom during our experiment. We found that this volunteer stayed in the room next to this target classroom for all the second half of our experiment, which was at least twice longer than any volunteer in group 1 & 2 stayed in the target classroom. This volunteer has a small seeing probability from each of his data point, but these small probabilities are accumulated during the long time range and lead to an appreciable overall probability.

The staying time ranges of all volunteers who visited this target classroom during our experiment are demonstrated in **Fig. 9**, as well as their probability of seeing the imaginary missing person (A_5). The time range of the asker's target trace is expressed as the space between two dash lines. It is easy to notice that no matter how long a volunteer stays in the room before the asker walked in, the seeing probability is close to zero. The above mentioned volunteer in group 2, who left the room just after the asker walked in, is marked in this figure.

In this case, the potential witnesses, who have the chance to see the missing person in reality, can be selected out by simply choosing the top *k* users in **Fig. 8**. We design a variable *Acc* to indicate the accuracy of the result. The value of *Acc* for each missing person is defined as the quotient of the total number of volunteers in group 2 divided by the largest sequence number (sorted by probability in decreasing order) of all volunteers in group 2. The largest number here indicates the minimum number of users have to be selected out if we want to ask all potential witnesses. For example, the *Acc* equals to 8/10 = 0.8 for the missing case demonstrated by **Fig. 8** and **Fig. 9**. The *Acc* values for other imaginary missing people in our experiment are 0.833, 0.846, 0.591 respectively. Therefore, the average accuracy *Acc* of this part of experiment is 0.7675. Most of the 4 imaginary missing people have a good enough *Acc* for real world system if accurate missing place can be confirmed. In our experiment, the missing case with a low *Acc* valued 0.591 testified the limitation of our model, the corresponding missinglocation is semi-enclosed by wood screens and adjoins to a main passage of the floor. We will further discuss the limitations of our work in Section 6.

5.2.2 Only a time range of missing happened can be confirmed

The accuracy of MissingFound is fine when the accurate missing can be provided. However, only a time range of the person missing can be confirmed in a realistic scenario. By asking different users who have high probability of seeing the missing person at different potential locations, the missing person may be noticed and reported no matter where he/she exact separated with the asker. In our approach, we design an exponent *l* in Objective Function (9) to punish the selected user combination which have high seeing probability at a same potential location. We now testify the effect of the exponent *l* in Objective Function (9).

Fig. 10 shows the different results under different settings of *l*. Our task is selecting 2 volunteers from all 22 volunteers, according to the performance of Objective Function (9). The time range of the asker's target trace *Q* is set as 52 minutes in this experiment, almost covering the whole experiment. We draw the intermediate result $\sum_{h \in U} x_h \cdot v_{ih}$ in Objective Function (9) for the selected 2 volunteers in every target point $r_i \in Q$, in order to demonstrate the effect of different settings of *l*. Obviously, the exponent *l* in (0,1) punishes those data points in the target trace which have high probability. Therefore, the target points with low seeing probability may have a higher weight when selecting users.

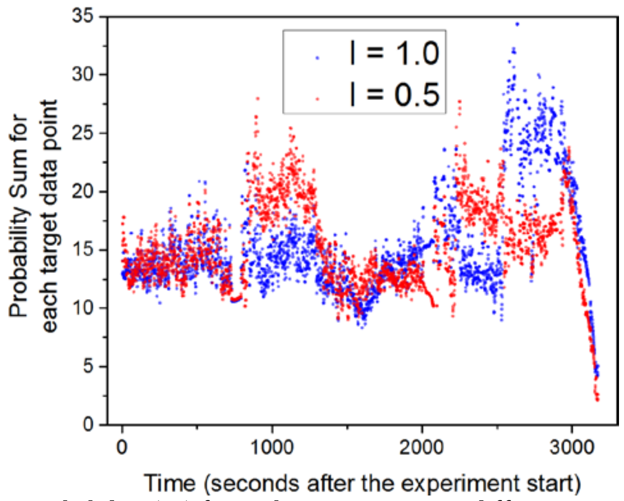


Fig. 10. Probability (v_{ih}) for each target point r_i in different exponent *l*.

As shown in **Fig. 10**, both of the two volunteers selected by the non-punishment version optimization (i.e., l = 1.0 in Function (9)) have high seeing probability in the last period of our experiment. This might cause that none of them has the chance to notice the missing person if the missing person separated with the asker in the front part of our experiment. By setting the factor l as 0.5 in Function (9), the best combination of two volunteers have high seeing probability in both the front part and the last part of our experiment. Therefore, we believe that the user selection in our approach with exponent l can balance the probability of seeing for different segments of the target trace, which is the mean concern in this part of evaluation.

5.3 How quickly does MissingFound select out most valuable users, even under a heavy workload?

To reduce the time cost to a reasonable level, a parallel algorithm and a heuristic method are designed in Section 4. We evaluate the probability-computing algorithm under MapReduce framework and the user-selecting algorithm using greedy and simulated annealing respectively.

5.3.1 Parallelization of the calculation of v_{ih} under MapReduce framework

To evaluate the performance of our parallel algorithm, a Hadoop cluster is set up. There are 10 blade servers, each with 2.4GHz*12 core CPU, 20G RAM, 270G hard disk. Each blade server installs 10 virtual machine with one core, 1G RAM and 16G hard disk. The experiments are executed on a 100 virtual machines cluster running Hadoop 2.6.0 [5]. All these physical machines are directly connected to the same fast network switch. Each machine runs at most one map task and one reduce task. The block size of this task is computed by the mappers number in demands.

The serial computation complexity of this part is $O(M \cdot N^2)$, and the time cost can be reduced to $O(M \cdot N^2/map\_num)$ by using MapReduce in theory. In **Fig. 11**, we use variable-controlling approach to test the time cost when N or M varies. After that, a figure shows several speed-up ratio curves in different data sizes, in order to illustrate the effect of parallelization. **Fig. 11** (a) and (b) testify the relationship between time cost and variables, showing the unacceptable high time cost of the serial algorithms. The fitting curves in these two figures fit well with our analyzed computation complexity $O(M \cdot N^2/map\_num)$. **Fig. 11** (c) illustrates the speed-up ratio in different situations. Due to the limitation of resources, the upper bound of map tasks is set as 64. As shown in this figure, the effect of parallelization comes better while increasing the scale of input data. Moreover, the system overhead of MapReduce can not be ignored. It includes the transmission time of non-local data, startup time of tasks, etc. Because of this, the speed-up ratios nearly reach their peaks when the scale of map tasks reaches 64. When using 64 map tasks for the designed input scale (M = 10,000, N = 1,000), the computing of all v_{ih} can be completed in **1** minute in our experiment, which is a reasonable time for a real world system.

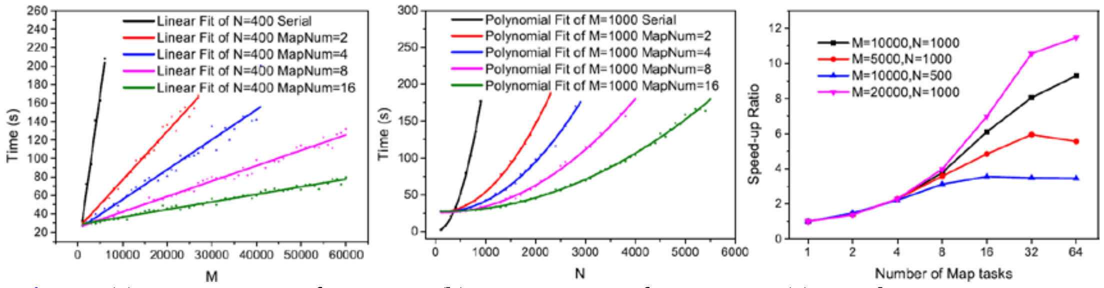


Fig. 11. (a) M-Time curves for N=400. (b) N-Time curves for M=1000. (c) Speed-up Ratio curves in different data sizes.

5.3.2 Selecting *k* most valuable users

There are three algorithms raised in Section 4 to solve this problem, Brute Force, Greedy and Simulated Annealing. It is easy to know that the Brute Force algorithm can be used to evaluate the performance of the other algorithms in small scale data, but can't be used in a real world system. When testing these algorithms using the data set of the field experiment with 22 volunteers and 1 asker, Greedy algorithm always returns the global optimum solutions (which is equal to the result produced by Brute Force), no matter what values are assigned to l and k. For a single execution of Simulated Annealing, the lowest percentage of returning the global optimum solutions is 65% in this test. Here, the values of parameters used in Simulated Annealing are described as follows: $r = M$, $t = 0.6$, $\varphi = 0.6$, $\rho = 1.2$.

When testing MissingFound under our simulated data in a large input scale ($M = 10{,}000$, $N = 1{,}000$, $k = 100$), the time cost for Greedy and Simulated Annealing are 29s and 26s respectively. By using several map tasks to run Simulated Annealing concurrently and then selecting a best result from both Greedy and Simulated Annealing in the reduce task, a good enough solution can be produced in **1** minute.

As evaluated above, we believe that MissingFound is able to select out a good enough group of users to the missing person in a realistic scenario, in a reasonable time.

6. Discussion

The evaluation of MissingFound shows that it is a sound assistant system for finding missing people in a realistic scenario. As a real world system, several more aspects should be considered by MissingFound. We briefly discuss them in this section.

6.1 Weak Wi-Fi environment

An effective MissingFound is based on sufficient number of WiFi APs, as well as their signal strengths. In our experiments, the minimum number of detectable WiFi APs is 3. For each data point we collected, there is at least one AP with the signal strength stronger than −86 dBm. MissingFound might produce a lower accuracy of results in a extreme weak WiFi environment. Fortunately, above limitations are fulfilled in the very majority of urban public areas.

6.2 Indoor space separated by "soft" obstacles

In Section 5, the result for one missing person's location, which is separated by wood screens, has much lower accuracy than others, because the similarity decay of passing through "soft" obstacles is not sharp enough. Only through an armored concrete wall will lead an sharp similarity decay (**Fig. 4**). However, this phenomenon avoids the distraction from phone placement – the results are similar no matter phones are placed in pockets or held in hands. This phenomenon results in a loss of accuracy to MissingFound, but luckily, the places separated by "soft" obstacles are relatively limited.

6.3 Bandwidth & energy consumptions

The energy & bandwidth consumptions of MissingFound on mobile phones should be very little, which are the main overhead of MissingFound on users' mobile phones. Suppose a MissingFound client collects one record every 10 seconds, there will be about 1 MB data collected on each mobile device per day. Since users are not moving all day long, those successive data points representing the same location ($A_0 > UB$) can be merged. Suppose these records are transmitted to the back-end server with a compression ratio 10% (7% can be achieved by our collected data), only 100 KB data need to be transmitted per day, which is affordable for a majority of users. Moreover, MissingFound client can store the sensor data for a period of time and upload them when a WiFi connection is available. Therefore, we believe that the bandwidth overhead of MissingFound is acceptable in a real world usage.

The MissingFound client on a mobile phone consumes little energy, which is suitable for running all day long in background. Even so, its energy efficiency can be further optimized. For example, the periodically WiFi scanning takes the majority part of the energy consumption. When the user is not moving (detected by accelerators on smart phones), MissingFound can slow down even stop WiFi scanning. Once the user moves again, the frequency of WiFi scanning would return to normal.

6.4 Behavior under heavy user load

We think the advantage from MissingFound will be more significant when it attracts a large number of users. We argue that under such scenarios, MissingFound users will have a higher chance to find another user who has noticed the missing person. However, the system burden will increase at the same time. To handle this challenge, area tags for user's trace may be helpful to keep the total number of candidates for one request in thousands. Only the users whose traces have the same tag or adjacent tag will be considered as a candidate.

7. Related Work

There are alternative strategies for MissingFound using floor plan or straight-line distance to estimate the probability of seeing after accurate positioning all data points, but none of them is practical. Because MissingFound client should be run on user's smart phone all day long, any high energy-consume methods are unacceptable, such as GPS technique. Moreover, a system without pre-deployment effort, knowledge about floor plan and special hardware can be widely applied. MissingFound is a prototype addressing these issue.

Some previous localization solutions rely on access points (AP) have already presented in the surroundings to enable localization, which is similar to our system, such as Radar [3], PlaceLab [6] and Active Campus [7]. SurroundSense [8] builds a map using several features found in typical indoor spaces such as ambient sound, light, color, etc., in addition to WiFi RSSI. These solutions require calibrating WiFi RSSI at many physical locations as pre-deployment effort, building RF signal maps. And in two recent works, FM radio [9] and Channel Frequency Response [10] are explored to use as fingerprints instead of RSSI. Theses calibration process is time-consuming and may not scale over large areas. Unlike these systems, MissingFound does not require signal calibration. Our approach is straightforward: we rely on WiFi RSSI fingerprints of users' trails to estimate the probability of seeing nearby missing people. Thus, MissingFound do not concern the users' physical coordinates (as in GPS) and do not require any pre-deployment effort.

Another line of research uses an RF propagation model (e.g., the log-distance path loss (LDPL) model) can be used to predict the RSSI at various locations, such as Lim et al. [11], TIX [12] and ARIADNE [13]. Using these models can reduce the pre-deployment effort dramatically but the accuracy decreased as cost. While these methods reduce the pre-deployment effort, they still require effort of placing infrastructure such as sniffers, obtaining information on the floor plans, and extending the capabilities of off-the-shelf APs, or at least knowledge of AP placement and power settings. MissingFound does not require these kinds of effort at all, and it takes use of the idea of LDPL model to design our computational formula for similarity (Equation (1)), which is related to the two data points' WiFi RSSI fingerprint independently.

In another thread of research, the localization schemes relied on deploying specialized infrastructure to assist localization. During a calibration phase, The user's position is estimated with the overheard signals and the data collected. For example, Cricket [14] and Bat [15] rely on ultrasound devices being deployed in the indoor environment as well as on the mobile devices. To perform localization, Active Badge [16] uses infrared (IR) beacons and receivers. Recently, some RFID based systems also have been proposed, such as LANDMARC [17]. Unlike these systems, MissingFound does not require any extra hardware or any beacons except the WiFi module on almost every smart phone and the WiFi APs already present in the surroundings.

Authors in [18–22] have paid attention to the tradeoff between energy and location accuracy. They have tried to use WiFi/GSM based schemes as an alternative of GPS to avoid the continuous usage of GPS drain the battery in a few hours. Users in EZ [23] only need to obtain a GPS lock at the edges of the indoor environment (e.g., the entrance or a window). The other features of this system is quite similar to our system, such as no pre-deployment effort, does not require knowledge of floor plan, special hardware and AP placemet or power settings. It only works with existing WiFi APs surrounding and WiFi module on users' smart phones, which are also the foundation of MissingFound. But dislike EZ, MissingFound does not concern the accurate location, does not determine the APs location and does not use GPS at all.

8. Conclusion

MissingFound is a novel, configuration-free assistant system to help find missing people via smart mobile crowdsourcing. Uploading users' physical locations is neither necessary for the system nor comfortable to users. Instead, we show that by recording WiFi RSSI fingerprints during users' movements, the probability of noticing the missing person can be calculated. Using this model, a group of users with highest probability of seeing the missing person can be selected out. To overcome the problem of high computational complexity in this calculation, a parallel and a heuristic algorithm in MapReduce are designed. We use these algorithms as components to develop MissingFound, and demonstrate the average accuracy of the selected users is high (0.7675) in our realistic experiments. Moreover, the time cost of selecting missing person for designed input scale (thousands of users with records of several days) is low (no longer than 2 minutes). We believe that MissingFound will not only significantly help find missing companions for people, but also boost a new trend in the design of mobile, multi-user collaboration and historical proximity sensing based applications.

References

[1] Xinhua News. http://news.xinhuanet.com/energy/2013-10/07/c_125489360.htm

[2] AmberAlert, http://www.amberalert.gov/

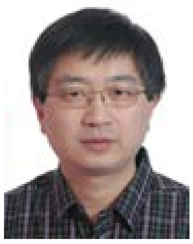
[3] Bahl P, Padmanabhan V N., "Radar: An in-building RF-based user location and tracking system," in *Proc. of 19th Annual Joint Conference of the IEEE Computer and Communications Societies*. pp. 775-784, March 26-30, 2000. Article (CrossRef Link)

[4] Dean J, Ghemawat S., "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, January, 2008. Article (CrossRef Link)

[5] Hadoop. http://hadoop.apache.org/

[6] LaMarca A, Chawathe Y, Consolvo S, Hightower J, Smith I, Scott J, Sohn T, Howard J, Hughes J, Potter F, et. al., "Place lab: Device positioning using radio beacons in the wild," *Pervasive computing*, vol. 3468, pp. 116-133, 2005. Article (CrossRef Link)
[7] Griswold W G, Shanahan P, Brown S W, Boyer R, Ratto M, Shapiro R B, Truong T M., "Activecampus: experiments in community-oriented ubiquitous computing," *Computer*, vol. 37, no. 10, pp. 73-81, October, 2004. Article (CrossRef Link)
[8] Azizyan M, Constandache I, Roy Choudhury R., "Surroundsense: mobile phone localization via ambience fingerprinting," in *Proc. of the 15th annual international conference on Mobile computing and networking*, pp. 261-272, September 20-25, 2009. Article (CrossRef Link)
[9] Chen Y, Lymberopoulos D, Liu J, Priyantha B., "FM-based indoor localization," in *Proc. of the 10th international conference on Mobile systems, applications, and services*, pp. 169–182, June 25-29, 2012. Article (CrossRef Link)
[10] Sen S, Radunovic B, Choudhury R R, Minka T., "You are facing the Mona Lisa: spot localization using PHY layer information," in *Proc. of the 10th international conference on Mobile systems, applications, and services*, pp. 183-196, June 25-29, 2012. Article (CrossRef Link)
[11] Lim H, Kung L C, Hou J C, Luo H., "Zero-configuration indoor localization over ieee 802.11 wireless infrastructure," *Wireless Networks*, vol. 16, no. 2, pp. 405-420, February, 2010. Article (CrossRef Link)
[12] Gwon Y, Jain R., "Error characteristics and calibration-free techniques for wireless LAN-based location estimation," in *Proc. of the 2nd international workshop on Mobility management & wireless access protocols*, pp. 2-9, October 01-01, 2004. Article (CrossRef Link)
[13] Haeberlen A, Flannery E, Ladd A M, Rudys A, Wallach D S, Kavraki L E., "Practical robust localization over large-scale 802.11 wireless networks," in *Proc. of the 10th annual international conference on Mobile computing and networking*, pp. 70-84, September 26 - October 01, 2004. Article (CrossRef Link)
[14] Priyantha N B, Chakraborty A, Balakrishnan H., "The Cricket location support system," in *Proc. of the 6th annual international conference on Mobile computing and networking*, pp. 32-43, August 06-11, 2000. Article (CrossRef Link)
[15] Ward A, Jones A, Hopper A., "A new location technique for the active office," *IEEE Personal Communications*, vol. 4, no. 5, pp. 42-27, October 1997. Article (CrossRef Link)
[16] Want R, Hopper A, Falcao V, Gibbons J., "The active badge location system," *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91-102, January 1992. Article (CrossRef Link)
[17] Ni L M, Liu Y, Lau Y C, Patil A P., "LANDMARC: indoor location sensing using active RFID," *Wireless networks*, vol. 10, no. 6, pp. 701-710, November, 2004. Article (CrossRef Link)
[18] Gaonkar S, Li J, Choudhury R R, Cox L, Schmidt A., "Micro-blog: sharing and querying content through mobile phones and social participation," in *Proc. of the 6th international conference on Mobile systems, applications, and services*, pp. 174-186, June 17-20, 2008.Article (CrossRef Link)
[19] Lin K, Kansal A, Lymberopoulos D, Zhao F., "Energy-accuracy trade-off for continuous mobile device location," in *Proc. of the 8th international conference on Mobile systems, applications, and services*, pp. 285-298, June 15-18, 2010. Article (CrossRef Link)
[20] Ra M R, Paek J, Sharma A B, Govindan R, Krieger M H, Neely M J., "Energy-delay tradeoffs in smartphone applications," in *Proc. of the 8th international conference on Mobile systems, applications, and services*, pp. 255-270, June 15-18, 2010. Article (CrossRef Link)
[21] Wang Y, Lin J, Annavaram M, Jacobson Q A, Hong J, Krishnamachar-i B, Sadeh N., "A framework of energy efficient mobile sensing for automatic human state recognition," in *Proc. of 7th Annual International Conference on Mobile Systems, Applications and Services*, pp. 179-192, June 22-25, 2009. Article (CrossRef Link)
[22] Zhuang Z, Kim K H, Singh J P., "Improving energy efficiency of location sensing on smartphones," in *Proc. of the 8th international conference on Mobile systems, applications, and services*, pp. 315-330, June 15-18, 2010. Article (CrossRef Link)
[23] Chintalapudi K, Padmanabha Iyer A, Padmanabhan V N., "Indoor localization without the pain," in *Proc. of the 16th annual international conference on Mobile computing and networking*, pp. 173-184, September 20-24, 2010. Article (CrossRef Link)

Weiqing Liu received the B.E. degree of Computer Science and Technology from the University of Science and Technology of China (USTC) in 2011. He is currently a PhD student at the School of Computer Science and Technology in USTC. His research interests include cloud computing, mobile computing and big data processing.



Jing Li received his B.E. degree of Computer Science from the University of Science and Technology of China (USTC) in 1987, and Ph.D. in Computer Science from USTC in 1993. Now he is a Professor at the School of Computer Science and Technology in USTC. His research interests include distributed systems, cloud computing, big data processing and mobile cloud computing.

Zhiqiang Zhou received the BS degree in computer science from University of Science and Technology of China (USTC), where he is currently working toward the doctor's degree. His research interests include cloud computing, cloud networks.

Jiling He received the MsC degree of Information Technology from the Hong Kong University of Science and Technology in 2014 and the B.E. degree of Computer Science from the University of Science and Technology of China in 2013. She is now working in Tencent.