Privacy-Preserving Participant Grouping for Mobile Social Sensing over Edge Clouds

Ting Li, Zhijin Qiu, Lijuan Cao, Dazhao Cheng, *Member, IEEE*, Weichao Wang, *Member, IEEE*, Xinghua Shi, *Member, IEEE*, Yu Wang, *Fellow, IEEE*

***Abstract*—Mobile social sensing leverages a large group of individuals having mobile devices capable of sensing and computing to perform intelligence sensing tasks. To perform smart participant selection (i.e. task assignment) for mobile social sensing, most of the cloud-based platforms often require mobile users to report their personal information, such as sensing cost, location and sensing quality. Therefore, the users might suffer from potential privacy breaches during the participant selection phase especially when they are not selected for performing the tasks. Existing solutions based on participant grouping can resolve this privacy leakage by leveraging secure sharing or group bidding within groups. However, the group formation problem has not be well studied, and the communication overhead over formed groups is usually ignored. To address these issues, in this paper, we consider a new set of privacy-preserving grouping problems over edge clouds to minimize the communication cost at edge clouds during secure sharing/bidding, while satisfying each participant's privacy requirement. Various grouping schemes are carefully designed to fulfill the optimization goal for two different scenarios: tree-based hierarchical edge clouds and graph-based interconnected edge clouds. Extensive simulations over both synthetic and real-life datasets are conducted to confirm the efficiency of all proposed schemes.**

***Index Terms*—Privacy preservation, participant grouping, mobile social sensing, mobile crowd sensing, edge computing.**

I. Introduction

With the rapidly growing number of mobile users who carry their smart phones or various wearables equipped with multiple low-power sensors, mobile social sensing (MSS) (or called mobile crowd sensing, MCS) becomes a promising sensing paradigm to collect observations of the physical world [1]–[3]. Compared to traditional static sensing, mobile social sensing provides better coverage at a lower cost by leveraging sensing capability (and/or human intelligence) from a large number of mobile users to accomplish large-scale sensing task. A mobile social sensing system normally includes three essential parts: a large pool of mobile participants, a set of sensing tasks, and a cloud-based platform which recruits specific participants for a given task. One of the key challenges in such a system is participant selection (also called task assignment) at the platform, i.e., how to select the appropriate mobile users to perform particular sensing tasks under certain constraints. Recently, various participant selection problems have been well studied from different aspects, such as coverage optimization [4], [5], energy efficiency [6]–[8], incentive mechanism [9]–[15], sensing quality [16]–[19], data collection [20]–[22] and truth discovery [23]–[25]. In this paper, we investigate another important but less studied issue in participant selection: *privacy-preserving participant grouping*.

To protect sensitive information of mobile participants in mobile social sensing, different privacy protection techniques have been proposed [17], [24], [26]–[32]. In a mobile sensing system, usually there are two types of sensitive information need to be protected from privacy breaches: sensing data and participant information. On one hand, sensing data collected from mobile participants may include various sensitive personal information, such as actual value of sensed data, location tag or user ID. To protect these sensitive information, various protection methods designed for data privacy, participant anonymity, and location privacy can be adopted [24], [29]–[33]. On the other hand, mobile users usually need to provide some participant information (e.g., bid value, location trace, or sensing quality) to the platform so that the platform can select the appropriate users. However, the submitted information could be sensitive. For example, the bid value may indicate a user's context (e.g., location or route) [26]–[28], or the sensing quality may leak the sensing ability of a user device (e.g., mobile device quality) [17]. Although data/location privacy of sensed data have been well studied, *privacy-preserving participant selection* to protect participant information during participant selection has rarely been discussed until quite recently [17], [26]–[28].

Current solutions for privacy-preserving participant selection usually have two steps: *participant grouping* and *secure bidding/sharing*. First, based on the concept of k-anonymity, the mobile users are assigned into small groups [27], [28], [47] where their privacy requirement could be satisfied (i.e. the group size of their group is larger than the desired group size k). Then a secure bidding or sharing [17], [27], [28] will be conducted within and across the formed groups so that the platform can select the participants to perform the task. Comparing to complex encryption methods over all users at the platform, grouping-based solutions not only protect participant information by guaranteeing k-anonymity but also make the secure sharing/bidding process more scalable by significantly reducing its computation or communication overheads.

T. Li is with Division of Natural Science and Mathematics, Oxford College of Emory University, Oxford, GA 30054, USA.

Z. Qiu is with Institute of Oceanographic Instrumentation, Qilu University of Technology, Qingdao, Shandong, China.

L. Cao, D. Cheng and W. Wang are with College of Computing and Informatics, University of North Carolina at Charlotte, Charlotte, NC 28223, USA.

X. Shi and Y. Wang are with Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania, PA 19112, USA.

T. Li (tli41@emory.edu) and Y. Wang (wangyu@temple.edu) are the co-corresponding authors. This work is partially supported by the National Science Foundation under Grant No. CCF-1908843.

The idea of privacy-preserving grouping for participant selection was first introduced by [27] but without discussion on how to form the groups. In an extended version, Li et al. [28] formally define and investigate the *privacy-preserving participant grouping* problem, where each participant should be put into a group whose size smaller than or equal to its privacy requirement. The overall goal of the grouping is to minimize the total communication cost during the secure bidding/sharing of participant selection. Two concise and straightforward solutions based on sorting or dynamic programming are then proposed. However, they did not consider the realistic communication delay/cost among group members, which makes them perform poorly in a real distributed cloud-based mobile sensing system. Therefore, in this paper, we focus on how to group users together in consideration of more realistic communication cost.

In existing solutions, participant selection are usually performed at the centralized platform located on a remote cloud server. Mobile users have to submit their participant information (such as bids, sensing qualities) to the remote server for participating the selection processing. This leads to long communication delays (might not acceptable for time-sensitive tasks) and privacy concerns (sending sensitive information to the cloud). To mitigate these issues, we adopt the edge computing [34]–[36] in our design by putting the grouping and selection processing over edge clouds (a group of small-scale servers at the edge of the network). Fig. 1 shows the overall architecture. Using edge clouds not only reduces the communication latency but also provides another layer of privacy protection [37] (since the sensitive information is kept within nearby edge servers which is closer to the users compared with the remote platform)[1]. In this paper, two types of edge cloud architectures (hierarchical or interconnected) will be considered.

We specifically investigate the *privacy-preserving participant grouping over edge clouds*, where each user's privacy requirement must be satisfied with the minimum communication cost among group members depending on their locations in the edge cloud. Privacy-preserving participant grouping over edge clouds has its own unique challenges. First, there is an trade-off to find the optimal *group location* for each group in the edge cloud. To reduce the communication latency, it prefers to place the group on the server closer to group members, while grouping with users from further servers could provide better privacy. Second, the *group size* matters too. The larger group size is, the better privacy protection is. However, the group with larger number of users leads to higher computation and communication cost during the secure sharing/bidding. Last, the overall costs and loads among the groups and edge servers should be balanced or optimized.

Our contributions of this work are summarized as follows.

[1]In general, edge computing provides better security and privacy protection. First, local data processing, filtering and anonymizing by an edge server can reduce the amount of sensitive and private information that is sent to the public cloud [37]. Second, compromising a single edge server will only affect a small amount of users, while a break at a centralized cloud will cause security breach at a very large scale [34]. Last, there are also significant existing efforts on secure edge servers via different protection techniques [38]–[40], including several dedicated for edge-computing-enabled mobile sensing [41]–[46].

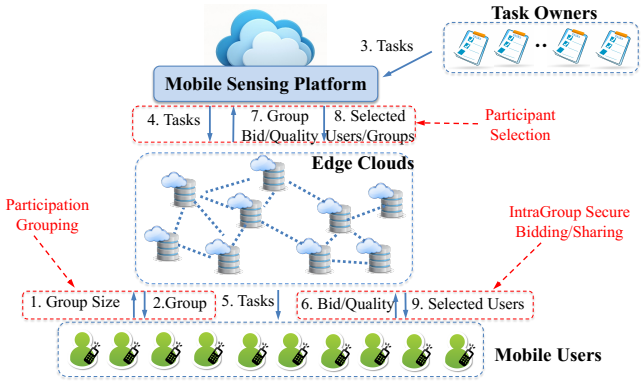
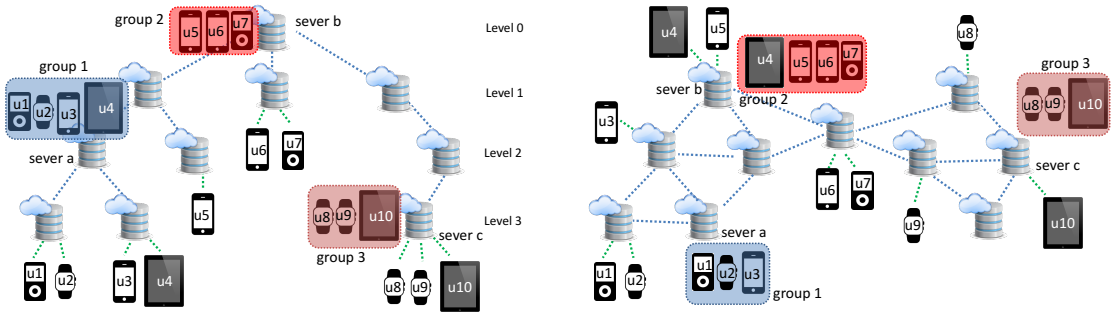


Fig. 1. **Mobile sensing system over edge clouds** mainly comprises three components: *mobile users* - participants, *edge clouds* - a third party for grouping and secure sharing/bidding, and the *mobile sensing platform* - receiving sensing tasks from task owners and performing the participant selection. Arrows with numbered labels (1 to 9) show the whole sequence of steps and interactions of participant selection over edge clouds.

- We mold privacy-preserving participant grouping (PPPG) over edge clouds into different optimization problems, *Min-Max* of group size (or its square) and *Min-Sum* of group size (or its square) reflecting the communication cost between hierarchical or interconnected edge servers. To the best of our knowledge, this is the first work coping with the privacy-preserving participant grouping problem over the mobile sensing architecture with edge clouds with considering the communication cost between edge servers. The problems in this paper are much more challenging and complex than a simpler version of participant grouping in our previous work [28] where the communication delay is ignored.
- We propose solutions for PPPG problems for two cases: with hierarchical edge clouds (PPPG-HEC) or with interconnected edge clouds (PPPG-IEC). Two heuristic algorithms *Top-Down* and *Bottom-Up* with time complexity $O(N \log N)$ are proposed to resolve the grouping problem with HEC. Two more efficient algorithms *Merge-Server* and *Merge-User* with time complexity $O(Y + X \log X)$ and $O(Y + N \log N)$ are proposed to cope with the more general grouping problem under IEC. Here N, X, and Y are the number of users, the number of edge servers, and the number of links among all servers, respectively.
- Simulations on both synthetic and real-life datasets are conducted to verify the proposed approaches. Simulation results confirm the efficiency of these methods to generate privacy-preserving groups over hierarchical or interconnected edge clouds architecture and minimize the communication cost.

The remainder of this paper is organized as follows. In Section II, we give the problem definition of *Privacy-preserving participant grouping over edge clouds* for two different situations: PPPG-HEC or PPPG-IEC. We then propose corresponding grouping algorithms for these two scenarios in Section III and Section IV, respectively. In Section V, we evaluate our algorithms using simulations based on both synthetic and real-world data traces. Finally, we present related work in Section VI and conclude in Section VII. A preliminary of this paper appeared in [47], and this version includes newly



(a) hierarchical edge clouds (HEC) (b) interconnected edge clouds (IEC)

Fig. 2. **Participant grouping over edge clouds:** An example where 10 users ($u1$ to $u10$) are connected to different edge servers, and formed into three groups at servers a, b & c for both HEC and IEC cases.

introduced grouping problem/schemes under interconnected edge clouds, additional sets of simulation results, more comprehensive related works, and new theoretical analysis.

II. Participant Grouping over Edge Clouds

A. MCS System with Edge Clouds

As shown in Fig. 1, the mobile sensing system with edge clouds typically has three components: mobile users as participants, the edge clouds near to mobile users, and the mobile sensing platform at a remote cloud. Here, the edge clouds (EC) play the role of a third party to divide all participants into groups and facilitate secure sharing/bidding within groups. The participants are still selected by the mobile sensing platform over the remote cloud. Here we assume that both the platform and the third party EC are *semi-honest*, i.e., they follow the protocol but are curious to learn the information of others. In this paper, we particularly concentrate on participant grouping over edge clouds. After such grouping, standard participant selection with secure sharing/bidding [17], [27] can be applied. Fig. 1 describes the overall flows in mobile sensing system with hierarchical or interconnected edge clouds.

Hierarchical vs Interconnected Edge Clouds: In this paper, we consider two types of edge clouds, *tree-based hierarchical edge clouds* and *graph-based interconnected edge clouds*. Recall that the edge servers (or called edge nodes) are semi-honest which only has influence on the secure bidding process but not grouping. As shown in Fig. 2(a), the **hierarchical edge clouds (HEC)** can be represented by a tree with multiple levels, connecting a set of X edge nodes (denoted by $V = \{v_1, v_2, \cdots, v_X\}$). The number of children for each node varies and is determined by the edge network architecture. For each node v in the tree, we use l_v, A_v and D_v to denote its level, its ancestor and descendant sets, respectively. Assume that the root node has level 0, and there are I levels in total. Then, the ancestor (or descendant) of node v at level l are defined as $A_{l,v}$ (or $D_{l,v}$). We assume that mobile users are directly connected to a leaf node in HEC where the delay between mobile user and its connected edge node could be neglected. In HEC, a mobile user can be grouped at any of its ancestor node in the tree. When a mobile user is grouped at a server at lower level (nearer to the root with a smaller l), more potential participants can be grouped together for better privacy but with the cost of longer communication delay. As shown in Fig. 2(b), the **interconnected edge clouds (IEC)** is represented by a graph $\mathbb{G}(V, E)$ where there could be a link between any two edge servers. We assume that X edge server nodes (denoted by $V = \{v_1, v_2, \cdots, v_X\}$) are connected via Y links among them (the link set is denoted by $E = \{v_i v_j | \textit{ servers } i \textit{ and } j \textit{ connected}\}$). For each link $v_i v_j$, there is an associated cost $c_{i,j} = c(v_i, v_j)$ (such as delay between the two edge servers). We can also use a cost matrix to represent all the cost, i.e., $C = \{c_{i,j}\}$. Note that HEC is a special case of IEC where graph $\mathbb{G}(V, E)$ is a tree.

Participant Grouping: To protect the participants' privacy during participant selection, we adopt the idea of *k-anonymity*, where the information of each individual contained in the release cannot be distinguished from at least $k-1$ individuals whose information also exist in the release, to perform *participant grouping*. Assume that there is a set of N mobile users, denoted by $U = \{u_1, u_2, \cdots, u_N\}$. During the grouping, mobile users are divided into small groups based on their privacy requirement (i.e. group size requirement $\gamma(u_i)$). (1) When user u_i joins in the system through a nearby edge server $s_0(u_i)$, it submits his registration request $(\gamma(u_i), s_0(u_i))$, where $\gamma(u_i)$ and $s_0(u_i)$ are his group size requirement and the edge node he is connected to, respectively. (2) During the grouping, an edge node v can put its user u_i to an edge node at a lower level if either $\gamma(u_i)$ cannot be satisfied locally at v or the node at the lower level needs more users. (3) After grouping, the system notifies u_i the grouping result $(s(u_i), g(u_i)))$ where $s(u_i)$ is the edge node where u_i's final group sits after grouping and $g(u_i)$ is the index of the final group of u_i. In other words, after the grouping process, each user u_i is put in a group $G_{g(u_i)}$ which sits on the server $s(u_i)$. Assume that we end up with x groups, denoted by $G_1, G_2, \cdots, G_x$. We know that $\sum_{i=1}^{x} |G_i| = N$ and $|G_i| \geq \max_{u \in G_i} \gamma(u)$ (i.e., in each group, the total amount of users is equal to or larger than the largest group size requirement in that group). For HEC, due to the tree structure of edge clouds, the users on $s_0(u_i)$ can only be put to one of its ancestors $A_{s_0(u_i)}$, i.e., $s(u_i) \in \{s_0(u_i)\} \cup A_{s_0(u_i)}$ and $0 \leq l_{s(u_i)} \leq l_{s_0(u_i)}$. However, for IEC, $s(u_i)$ can be any connected edge server in the edge clouds, i.e., $s(u_i) \in V$.

Participant Selection: As shown in Fig. 1, after grouping, privacy-preserving participant selection will be performed within each group and at platform level to select the participants for each sensing task (see details in [27], [28]). Note that the participant grouping performance has no influence on the participant selection decision and payment process. In this

paper, two secure bidding or sharing methods are considered: the secure bidding approach in [27], [28] with $O(|G|)$ messages exchanged (assuming user IDs are public) and the secure sharing method in [17] with $O(|G|^2)$ messages exchanged. For simplicity, we use $|G|$ and $|G|^2$ as the communication costs of a group G during secure bidding/sharing.

B. Privacy-Preserving Participant Grouping Problems

Now we formally formulate the optimization problem of *privacy-preserving participant grouping* (PPPG) over cloud edges. Its goal is to divide the participants into several groups at cloud edges, so that both the privacy requirements of all participants are satisfied and the total communication cost during the participant selection (secure bidding or sharing over groups) is minimized. In this paper, we investigate PPPG under two distinct scenarios: **PPPG-HEC** and **PPPG-IEC**, with different edge cloud topology. Before introducing them, we first review and extend our previous results on a simpler PPPG problem.

Scenario 0: PPPG w/o Delays: In [28], we have defined a simpler PPPG problem where *communication costs* among group members are the same. In other words, the delay between edge servers is negligible. We assume that the communication cost of secure bidding/sharing is only linear with the group size $|G|$. To perform the secure bidding/sharing within groups in parallel, we only care about the communication cost of the largest group. Then the participation grouping problem can be defined as follows:

$$f_1 = \min \max_{i=1}^{x} |G_i|, \quad \text{s.t.} \quad |G_i| \geq \max_{u_j \in G_i} \gamma(u_j).$$

The objective of this PPPG is to minimize the maximal group size, while the constraint makes sure that the number of participants in each group must be larger than or equal to the largest group size requirement of any participant in that group. For this problem, [28] has proposed two algorithms: one based simple sorting and the other one based on dynamic programming (DP). The sorting algorithm is proved to be a 2-approximation while the DP algorithm can find optimal solution of this version of PPPG.

We can further extend the optimization goal in this model to consider other three cases of PPPG: $f_2 = \min \max_{i=1}^{x} |G_i|^2$ or $f_3 = \min \sum_{i=1}^{x} |G_i|$ or $f_4 = \min \sum_{i=1}^{x} |G_i|^2$. Note f_1 and f_2 are for the *Min-Max* case when the groups are dealt with in parallel, while f_3 and f_4 are for the *Min-Sum* case when the groups are dealt with sequentially. Note that f_3 is meaningless since it is equal to N. For the other three optimization problems, the sorting and DP algorithms from [28] can be applied. The sorting algorithm can be proved to be a 2- or 4- approximation comparing to optimal solution for f_1 and f_2 with complexity $O(N \log N)$. The DP algorithm can find optimal solution for f_1, f_2 and f_4 with complexity $O(N^2)$.

However, [28] and above extension do not consider the realistic communication delay/cost among different group members over different edge servers, which makes these proposed algorithms perform poorly in a real edge cloud based mobile sensing system. Therefore, in this paper, we focus on **PPPG-HEC** and **PPPG-IEC**, where the delays among edge servers do matter and edge cloud topology plays an important role in the optimization.

Scenario I: PPPG-HEC: In this scenario, with the tree topology of HEC, the communication cost is considered as the delay among different levels of edge servers. Although the servers can provide better privacy resource at lower level, it may cause longer communication delays between levels. The delay for mobile user u_i is bounded by the level difference between $l_{s_0(u_i)}$ (where u_i originally sits) and $l_{s(u_i)}$ (where the final group he sits). For example, in Fig. 2(a), the level difference of u_1 is 1, while u_5's is 2 and u_8's is 0. Here, we assume that all group members communicate with each other via the edge server where the group sits. We define the four types of grouping optimization problems (two of *Min-Max* and two of *Min-Sum*) as following:

$$\begin{aligned}
f_{I_1} &= \min \max_{i=1}^{x} \sum_{u_j \in G_i} (l_{s_0(u_j)} - l_{s(u_j)}) \text{ or} \\
f_{I_2} &= \min \max_{i=1}^{x} (|G_i|^2 \max_{u_j \in G_i} (l_{s_0(u_j)} - l_{s(u_j)})) \text{ or} \\
f_{I_3} &= \min \sum_{i=1}^{x} \sum_{u_j \in G_i} (l_{s_0(u_j)} - l_{s(u_j)}) \text{ or} \\
f_{I_4} &= \min \sum_{i=1}^{x} (|G_i|^2 \max_{u_j \in G_i} (l_{s_0(u_j)} - l_{s(u_j)})) \\
\text{s.t.} \quad & |G_i| \geq \max_{u \in G_i} \gamma(u), \forall G_i, \\
& s(u_i) \in \{s_0(u_i)\} \cup A_{s_0(u_i)}, \forall u_i.
\end{aligned} \tag{1}$$

Besides each participant's privacy requirement needs to be satisfied (the first constraint), each user's final group could only sit on either the original server or its server's ancestor nodes (the second constraint). For simplicity, we assume that there is at most one group sitting on each edge node, and the maximal level difference among all the group members is utilized for f_{I_2} and f_{I_4} (such as 2 for $group$ 2 in Fig. 2(a). Note that the worst solution is that all users are grouped in a single group at the root node. The proposed methods can be extended to solve the optimization problems with more accurate delay definitions.

Scenario II: PPPG-IEC: In this scenario, the edge servers are interconnected and the delay between any two edge servers v_i and v_j can be modeled by $c(v_i, v_j)$ based on the graph $\mathbb{G}(V, E)$[2]. Again all in-group communications are via the edge server where the group sits, thus we consider the delay between the original server $s_0(u_j)$ of the group member u_j and his final group server $s(u_j)$, i.e., $c(s_0(u_j), s(u_j))$. Therefore, we

[2]Note that our model is general enough to also consider remote cloud. In our experiments, we include an additional node in V to represent the remote cloud and add links from it to all edge servers with a cost D or d.

define the optimization problems as follows:

$$
\begin{aligned}
f_{II_1} &= \min \max_{i=1}^{x} \sum_{u_j \in G_i} c(s_0(u_j), s(u_j)) \text{ or} \\
f_{II_2} &= \min \max_{i=1}^{x} (|G_i|^2 \max_{u_j \in G_i} c(s_0(u_j), s(u_j))) \text{ or} \\
f_{II_3} &= \min \sum_{i=1}^{x} \sum_{u_j \in G_i} c(s_0(u_j), s(u_j)) \text{ or} \\
f_{II_4} &= \min \sum_{i=1}^{x} (|G_i|^2 \max_{u_j \in G_i} c(s_0(u_j), s(u_j))) \\
\text{s.t.} \quad & |G_i| \geq \max_{u \in G_i} \gamma(u), \forall G_i \\
& c(s_0(u_j), s(u_j)) < \infty, \forall u_i.
\end{aligned} \tag{2}
$$

In this model the final group location $s(u_j)$ is not be restricted by the tree structure anymore.

Note that while PPPG is in P and can be optimally solved by DP algorithm, PPPG-HEC and PPPG-IEC are more complex and computationally challenging. We conjecture that both PPPG-HEC and PPPG-IEC are NP-hard, but formal proofs of their NP-hardness are still open.

III. Group Formation for PPPG-HEC

In this section, we present several grouping algorithms for PPG-HEC optimization problems. *Genetic* algorithms and *stimulated annealing* algorithms are not suitable here because there are limited feasible solutions in the grouping space so that the mutation of current solution (neighboring solution in the space) may be infeasible with high probability[3]. Instead, two heuristic algorithms are proposed here: *Top-Down* and *Bottom-Up* to solve f_{I_3}. Then we extend them for f_{I_2}. With further modifications they can be applicable for other optimization cases in Scenario I too.

A. Top-Down Algorithm

The main idea of our proposed Top-Down algorithm is to arrange the users with largest privacy requirements starting at the root, and try to push them towards the leaf edge nodes via the root's children. When the users on the root and its children are satisfied, repeat such a process to the next level. The whole procedure is done from top to bottom of the tree. Algorithm 1 shows the details of Top-Down algorithm. Initially, we first put all users at the root, and then consider to push them to its child nodes. At each child node of the root, we place all of the users whose original leaf edge server is the descendant of the current child node to that child. Then we check whether there are users who could not be satisfied at these child nodes, if so we have to place them back to the root node. After this step, the requirements of all users in groups on the child nodes are satisfied. However if at least one of the users at root node are not satisfied with the current grouping, we need to move more users from child nodes back to the root through certain exchanges (by calling Algorithm 2 at Line 16). After this, all of the users on both root node and its child nodes are satisfied. We then move to the next level, and repeat this process for each node at that level. Overall, this algorithm aims to retain the users towards as higher levels as possible. Fig. 3 shows a detailed example of the top-down algorithm, and Fig. 4 illustrates the corresponding exchange process of Algorithm 2. Since sorting the users' privacy requirement needs $O(N \log N)$ and Algorithm 1 just traverses all the nodes in the tree to check the satisfaction of each user only once with $\Theta(N)$, the overall time complexity of Algorithm 1 is $O(N \log N)$.

[3]Here infeasible solution means that the privacy requirement of certain mobile user is not satisfied in such solution. An upper bound on the probability of getting a feasible solution with randomly generated solutions is given in the *Appendix*.

Algorithm 1 Top-Down Algorithm for f_{I_3}

Input: each user's location $s_0(u_i)$ and requirement $\gamma(u_i)$
Output: each user's final group $g(u_i)$ and location $s(u_i)$

1: Create an empty group G_v for each node v
2: Place all users at the root
3: **for** $l = 0$ to $I - 1$ **do**
4: **for** all nodes v at level l **do**
5: TD(v, l)
6: **return** $g(u_i)$ and $s(u_i)$ for all users

7: **Function** TD(v, l)
8: **if** v is a leaf node **then**
9: **return**
10: **for all** $u \in G_v$ **do**
11: Place u in $G_{v'}$, where v' is the unique child of v that is an element $A_{s_0(u)}$
12: **for all** v' that are children of v **do**
13: **while** $\max_{u \in G_{v'}} \gamma(u) > |G_{v'}|$ **do**
14: Move all $u \in G_{v'}$ with $\gamma(u) > |G_{v'}|$ to G_v
15: **if** $|G_v| < \max_{u \in G_v} \gamma(u)$ **then**
16: Call Algorithm 2 with node v and level $l + 1$

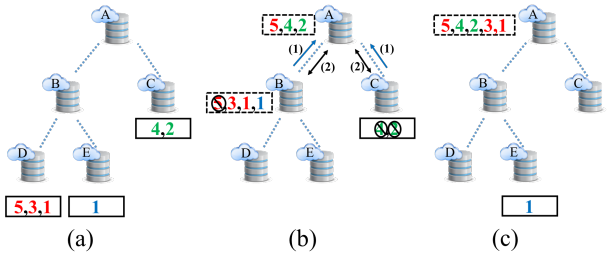


Fig. 3. **An example of Top-Down algorithm:** (a) initially, there are 6 users on leaf nodes of a 3-level edge tree; here, each number stands for the group size requirement of a user; (b) starting with the root node, 5, 4 and 2 cannot fit in the root's children groups, thus they are left at the root (shown as blue arrows); Then Algorithm 2 is called at the root (as shown in Fig. 4) to add 2 more users (related interactions shown as black arrows); (c) the left user with 1 will be remained at the leaf when TD is called at the next level.

B. Bottom-Up Algorithm

Different from Top-Down method, Bottom-Up algorithm starts from the bottom of the tree and has two steps. It first places the users on the highest level (lowest in the tree) where his privacy requirement could be satisfied by all users within the subtree at that node, and then moves them up as necessary to meet all users' privacy requirement. Algorithm 3 shows the details of Bottom-Up algorithm. In **Step 1** (Lines 2-9),

Algorithm 2 Update G_v at node v

Input: Node v and level l' (along with all user/group info.)

1: **while** $|G_v| < \max_{u \in G_v} \gamma(u)$ **do**
2: **for all** $v' \in D_{l',v}$ **do**
3: Define $S_{v'} = \emptyset$, and take any $u \in G_{v'}$ with $\gamma(u) = \max_{u' \in G_{v'}} \gamma(u')$ and place it into $S_{v'}$
4: **while** $\max_{u \in G_{v'} \setminus S_{v'}} \gamma(u) > |G_{v'} \setminus S_{v'}|$ **do**
5: Take any $u \in G_{v'} \setminus S_{v'}$ with $\gamma(u) = \max_{u' \in G_{v'} \setminus S_{v'}} \gamma(u')$ and place it into $S_{v'}$
6: Sort v' in order of increasing $|S_{v'}|$
7: **if** $\max_{v'} |S_{v'}| \leq \left(\max_{u \in G_v} \gamma(u) - |G_v|\right)$ **then**
8: Choose the last v' and move all $u \in S_{v'}$ to G_v
9: **else**
10: Choose the first v' s.t. $|S_{v'}| \geq \max_{u \in G_v} \gamma(u) - |G_v|$ and move all $u \in S_{v'}$ to G_v

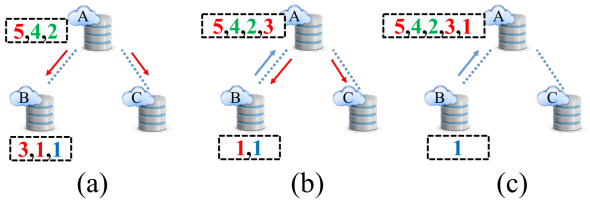


Fig. 4. **An example of Algorithm 2:** (a) to recruit more users, node A requests the help from nodes B and C; (b) node B has more users while C has no user to offer, and thus B offers the user with 3 to A and A accepts. A asks again since his group size does not meet the requirement yet; (c) after accepting one more user from B, A is satisfied and finished. Red arrow and blue arrow represent the user request and user offloading, respectively.

we place all users to the groups on leaves (where they are connected to the edge cloud). If the user cannot be satisfied at current node (considering a group with the users on this nodes and all of the descendant nodes), then this user will be moved up to the parent node. This process will be repeated for all nodes toward to the root node. After this, the users in the group located at leaf nodes are guaranteed to be satisfied. Then, in **Step 2** (Lines 10-17), we start at the root by applying Algorithm 2 to add more users on each node so that all the users on each node have satisfied group size. This step is similar to Top-Down method, but such interaction (Line 13) is performed with all descendants. In Top-Down, it is restricted to children nodes only (Line 12) An detailed example is shown in Fig. 5. The overall time complexity of Algorithm 3 is $O(N \log N)$ too. Bottom-Up usually takes longer time than Top-Down since it has the additional Step 1, but Bottom-Up performs slightly better than Top-Down as shown in Section V. Therefore, there is a trade-off between running time and performance.

C. Extended Algorithm for f_{I_2}

Solutions of Algorithm 1 and Algorithm 3 not only ensure that every user's requirement is satisfied but also aim to minimize the total level difference (which is relevant to the goal of f_{I_3}). However, they can be further extended to optimize f_{I_2} (where the square of group size matters). Algorithm 4 shows the details. Starting from a feasible solution by executing

Algorithm 3 Bottom-Up Algorithm for f_{I_3}

Input: each user's location $s_0(u_i)$ and requirement $\gamma(u_i)$
Output: each user's final group $g(u_i)$ and location $s(u_i)$

Create an empty group G_v on each node v
for all users u **do**
Place u into $G_{s_0(u)}$
for $l = (I-1)$ to 0 **do**
for all nodes v at level l **do**
if $|G_v| \neq 0$ **then**
Set $H_v = G_{v'} \cup \left(\bigcup_{v' \in D_v} G_{v'}\right)$
while $\max_{u \in G_v} \gamma(u) > |H_v|$ **do**
Move all $u \in G_v$ with $\gamma(u) > |H_v|$ from G_v to G_{v_p} where v_p is the parent of v
for $l = 0$ to $(I-1)$ **do**
for all v at level l **do**
if $G_v \neq \emptyset$ and $|G_v| < \max_{u \in G_v} \gamma(u)$ **then**
for $l' = l+1$ to $I-1$ **do**
if $\sum_{v' \in D_{l',v}} |G_{v'}| + |G_v| < \max_{u \in G_v} \gamma(u)$ **then**
Move all users from $\bigcup_{v' \in D_{l',v}} G_{v'}$ to G_v
else
Call Algorithm 2 at v and l', and **break**
return $g(u_i)$ and $s(u_i)$ for all users

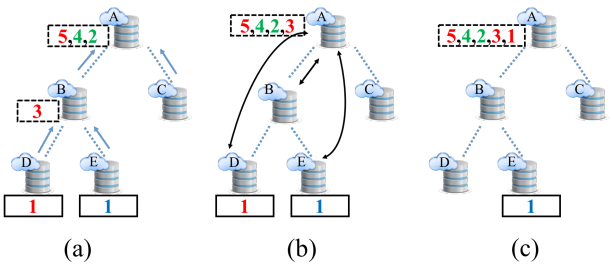


Fig. 5. **An example of Bottom-Up algorithm:** take the same example in Fig. 3(a); (a) first move each user to an ancestor node on highest level where its privacy requirement could be satisfied when considering all the users from descendant nodes; e.g. the user with 3 could be satisfied on node B since its descendants D and E have two more users; (b) starting from the root, run Algorithm 2 for each node whose users are unsatisfied yet; here the root gets the user with 3 from B and the user with 1 from D; (c) the algorithm ends when all users meet their group size requirements.

Algorithm 1 or Algorithm 3, we repeatedly find the edge node v with the highest cost (in term of the optimized function f_{I_2}) and distribute some of its users to other edge nodes so that the value of f_{I_2} decreases. During the redistribution procedure, all users' privacy requirements are kept satisfied. We first find all possible subsets of users in v's group, where the privacy requirement of v's group is still guaranteed after redistributing the uses in the subset to other nodes. Fig. 6 illustrates an example of possible partitions for these subsets. With a feasible partition, we move the users in the subset to one of their common ancestor nodes. To optimize the objective function, we either choose the ancestor with the least increasing cost (as shown in Line 6 of Algorithm 4) or select the ancestor that cause the most decreasing cost of current node v. We repeatedly decrease the maximal cost until no further reduction is possible. We denote such method (Algorithm 4) by Extended-LI (or the variation version of

Algorithm 4 Extended Algorithm for f_{I_2}

Input: each user's location $s_0(u_i)$ and requirement $\gamma(u_i)$
Output: each user's final group $g(u_i)$ and location $s(u_i)$

1: Get a feasible solution from Algorithm 1 or 3
2: $max = \max_{i=1}^{x}(|G_i|^2 \max_{u_j \in G_i}(l_{s_0(u_i)} - l_{s(u_i)}))$ and $v = \arg\max_{i=1}^{x}(|G_i|^2 \max_{u_j \in G_i}(l_{s_0(u_i)} - l_{s(u_i)}))$
3: **repeat**
4: Add all users $u \in G_v$ into a list L and sort L based on $\gamma(u)$
5: Find all possible user partitions of L and consider all subsets either from the head or tail (as long as the remaining users in L can be still satisfied), as shown in Fig. 6(b) and (c)
6: For each user partitions, we find their common ancestors and move these users to the ancestor node whose cost increases the least and all requirements of the users are still satisfied (after moving).
7: $max = \max_{i=1}^{x}(|G_i|^2 \max_{u_j \in G_i}(l_{s_0(u_i)} - l_{s(u_i)}))$ and $v = \arg\max_{i=1}^{x}(|G_i|^2 \max_{u_j \in G_i}(l_{s_0(u_i)} - l_{s(u_i)}))$
8: **until** max does not decrease anymore and all nodes v with max cost have been tried
9: **return** $g(u_i)$ and $s(u_i)$ for all users

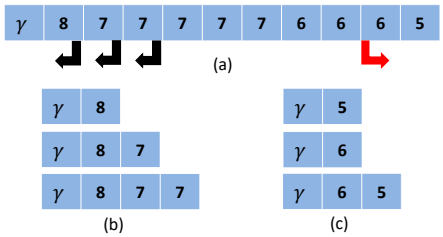


Fig. 6. **User partition in Algorithm 4:** (a) the original sorted user list L at v; (b) the subset of users by partitioning L from the head; (c) the subset of users by partitioning L from the tail. Algorithm 4 will try all possible subsets.

selecting the most decreasing cost of v by Extended-MD). Since there are only linear possible partitions, the overall time complexity of Algorithm 4 is still $O(N \log N)$, dominated by the complexity of Algorithm 1 or Algorithm 3.

IV. Group Formation for PPPG-IEC

In this section, we propose several group formation algorithms to solve the optimization problem in PPPG-IEC. As mentioned, the edge servers in this scenario are interconnected and can connect to several other edge servers at same time. In PPPG-HEC, edge server only connect to its ancestor or children in the tree. The optimization problem becomes more complicated in PPPG-IEC since users could group with any users as long as their edge servers are connected. Hence, derived from clustering and sorting, we propose two algorithms *Merge-Server* and *Merge-User* to solve the optimization problems f_{II_1} in PPPG-IEC. Note that they could be easily adopted for other optimization functions f_{II_2}, f_{II_3} and f_{II_4}.

A. Merge-Server Algorithm

Merge-Server method aims to merge edge servers with unsatisfied users together based on distance measurement until the requirements of all users get satisfied. Algorithm 5 shows the details. First, the pair-wise distances among any edge servers (including the remote cloud if there is one) are obtained via Dijkstra algorithm. Second, if there is an unprocessed server, we treat it as a new cluster and check whether the privacy requirements of its users are satisfied or not. If not, we group it with the nearest connected and unprocessed server. If there is no unprocessed server, we merge it with the nearest processed cluster. The merge process will repeat until all users on the current cluster get satisfied. Finally, the final location of each group (cluster) is decided based on the optimization goal (Line 13). The time complexity of this algorithm is $O(Y + X \log X)$, which is dominated by the Dijkstra algorithm. Here Y is the number of links among all servers and X is the number servers.

Algorithm 5 Merge-Server Algorithm for f_{II_1}

Input: each user's $\gamma(u_i)$ and $s_0(u_i)$, $\mathbb{G}(V, E)$
Output: each user's final group $g(u_i)$ and location $s(u_i)$

1: Compute a $X \times X$ cost matrix M, which has the least cost path among any two edge servers via Dijkstra algorithm
2: Sort the users on each edge server v with its group size requirement γ in descending order
3: Mark every edge server *unprocessed*
4: **while** there is an unprocessed server **do**
5: Consider an unprocessed server v_k as a new cluster
6: **while** the max group size requirement in v_k's cluster > the number of users of v_k's cluster **do**
7: **if** there exists unprocessed servers **then**
8: Find the closest unprocessed server v_j in M and merge v_j's users to v_k's cluster, mark v_j *processed*
9: **else**
10: Merge users of the closest processed server v_j, who holds a cluster, to v_k's cluster
11: Mark v_k *processed*
12: **for all** processed edge server who holds a cluster **do**
13: Find the optimal edge server within this cluster to group all users (regarding to the objective function), and make it as the location $s(u_i)$ of all users in this cluster
14: **return** $g(u_i)$ and $s(u_i)$ for all users

B. Merge-User Algorithm

Merge-User method is similar with *Merge-Server*, except for that clustering is over mobile users instead of edge servers. By merging users with unsatisfied privacy requirements, larger cluster is formed. Such merging process ends until all the users' requirements get satisfied. The details are given in Algorithm 6. The time complexity of this algorithm is $O(Y + N \log N)$. Notice that the algorithm needs to take an unprocessed user to process in Line 4. Different orderings to pick the user can be used, such as picking it randomly or picking the one with the largest/smallest group size requirement. We will test these methods in our experiments in the next section.

V. Performance Evaluations

In this section, we assess the proposed algorithms performance with one synthetic and two real-world datasets. We

Algorithm 6 Merge-User Algorithm for f_{II_1}

Input: each user's $\gamma(u_i)$ and $s_0(u_i)$, $\mathbb{G}(V, E)$
Output: each user's final group $g(u_i)$ and location $s(u_i)$

```
Compute a N×N cost matrix M, which has the least cost path among any two users via Dijkstra algorithm
Mark every user unprocessed
while there is unprocessed user do
    Consider an unprocessed user u_k as a new cluster
    while the max group size requirement in u_k's cluster > the number of users of u_k's cluster do
        if there exists unprocessed users then
            Find the closest unprocessed users u_j in M and merge u_j into u_k's cluster, and mark u_j processed
        else
            Merge users of the closest processed cluster to u_k's cluster
    Mark u_k processed
for all processed user cluster do
    Find the optimal edge server within this cluster to group all users (regarding to the objective function), and make it as the location s(u_i) of all users in this cluster
return g(u_i) and s(u_i) for all users
```

begin with a brief introduction of these datasets and our simulation settings, then we analyze the performances of our algorithms for both scenarios via extensive simulations.

A. Simulator, Datasets and Simulation Settings

All simulations are preformed within a mobile crowd sensing simulator developed by our team, which has been used in [5], [18], [19], [21], [27], [28], [47]. It is written in Java. The experiments are running on a computer with an Intel Core i7 processor at 3.4 GHz with 24 GB of RAM. All the topology of edge clouds, sensing tasks, and mobile users are simulated within the simulator based either a synthetic dataset and two real-world datasets. Next, we introduce these datasets and all simulation settings in detail.

Synthetic Dataset: This dataset is only used for the comparison between our proposed methods and optimal solution in Scenario I. We construct a 3-level balanced binary tree with 50 mobile users randomly distributed on the leaf nodes. In the experiment, the group size requirement of users are generated by a uniform distribution $U(0, r)$, and where $r = [10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30]$ respectively.

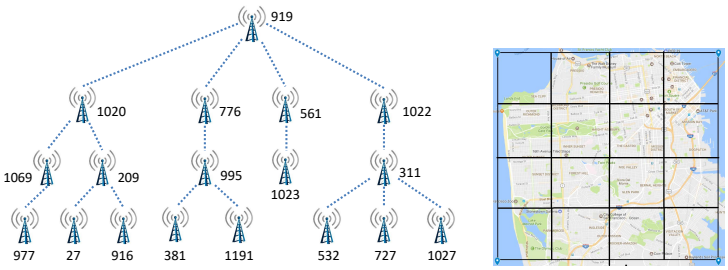
D4D Dataset: This dataset, from Orange's *Data for Development (D4D) challenge* [48], consists anonymous phone call records of 50,000 Orange mobile users in the Ivory Coast from December 1, 2011 to April 28, 2012. Here, we only use a two-week records (from December 5, 2011 and January 8, 2012), since the released data is anatomized based on two-week periods.

For Scenario I (f_I), in order to construct the hierarchical edge clouds, we chose 18 towers with highest call records as edge servers and organized them into a 4-level tree as shown in Fig. 7(a)[4]. We only choose the users who access the edge servers at those 9 leaf nodes, and there are the number of users are 6,880 and 50,898 call records. Without loss of the generality, we run our experiment with 34 distinct tasks and report the average of simulation results. For each experiment, the number of participating users is either 100-500 or 50. Besides, each user's group size (privacy requirement) is assigned by a uniform distribution $U(0, r)$, where $r = [30, 60, 90, 120, 150, 180]$ or $[2, 3, 4, 5, 6, 7]$.

For Scenario II (f_{II}), we evaluate our algorithms over two different graph-based interconnected edge clouds to cover broader application scenarios. Different graph models are used to generate the topology, and communication cost between edge servers are either identical or various in each model. In the first model (graph model (Δ, D)), we assume that the connection between two edge servers exists if there are more than Δ users visiting those two servers in certain time. Hence, for a certain dataset, we could construct various interconnected graph with different value of Δ. For the cost of each link, we set to 1 for links among edge servers and D for links from edge

TABLE I
PARAMETERS IN D4D AND SFC SIMULATIONS.

Dataset	Parameter	Value or Range
D4D	number of accessible towers	9 out 18 towers
	number of data records	50, 898
	total period of traces used	Dec 5, 2011 to Jan 8, 2012
	total number of tasks	34 or 100
	# of candidate N (f_{I_3})	100, 200, 300, 400, 500
	# of candidate N (f_{I_2})	50
	# of candidate N (f_{II_1},f_{II_3})	50 or 200, 300, ..., 800, 900
	max group size req. r (f_{I_3})	30, 60, 90, 120, 150, 180
	max group size req. r (f_{I_2})	2, 3, 4, 5, 6, 7
	max group size req. r (f_{II_1},f_{II_3})	60, 90, 120, 150, 180
	cost to cloud D (f_{II_1}, f_{II_3})	10, 20, 30, 40, 50
	link threshold Δ (f_{II_1}, f_{II_3})	2, 4, 6, 8, 10
	cost to cloud d (f_{II_1}, f_{II_3})	163 or 140, 150, 160, 170, 180
	distance threshold δ (f_{II_1}, f_{II_3})	100, 200, 300, 400, 500
SFC	number of regions	16
	number of data records	508, 979
	total period of traces used	June 6, 2008
	total number of tasks	310
	# of candidate N	50, 100, 150, 200, 250, 300
	max group size req. r for f_{I_3}	20, 40, 60, 80, 100, 120
	max group size req. r for f_{I_2}	2, 3, 4, 5, 6, 7



(a) D4D tree (b) SFC quad tree

Fig. 7. **Hierarchical edge clouds for Scenario I:** (a) a 4-level unbalanced tree with 18 edge servers (towers with showing their IDs) in D4D dataset; (b) a 3-level balanced quad tree with 16 grid cells as its leaves for SFC dataset.

[4]Hierarchical edge cloud could also be constructed with additional servers beyond cellular towers, and here we only use the towers since those are available in this dataset.

server to the remote cloud. We run and average our simulation results on 100 tasks with 50 or 200-600 users, 60-180 maximal group size requirement, and various values of Δ and D. In the second model (graph model (δ, d)), to have a more realistic network topology, we generate both the link between edge servers and its cost based on physical distance between edge servers (towers) in D4D dataset. A link would exist between two servers if the physical distance between them is smaller than a threshold δ. We set δ with various value from 100 to 500. The communication cost on each link is determined based on its distance with about 5ms per 100 units [49], [50]. The cost d from edge servers to the remote cloud is set at 163ms based on [51] or from 140ms to 180ms. We also run and take the average of 10 tasks with 200-900 users.

The parameters for D4D simulations with all different settings are summarized in the upper section of Table I.

SFC Dataset: Although D4D dataset provides a real-life large scale traces for human mobility, it does not have high spatial resolution (only at cellular tower level). Therefore, San Francisco Cab (SFC) dataset [52] is used for our simulations too. SFC dataset includes the GPS records of 536 taxi during May 17 to June 10, 2008 at San Francisco. Similarly, we choose a single day with the most users and data records (June 6 with 504 users and 508,979 data records). Without specific towers as edge servers, we create a 3-level balanced quad tree by dividing the area of San Francisco into 16 cells/leaves, as shown in Fig. 7(b)). We also run simulations multiple time (with 310 tasks and $50 - 300$ users) and take the average of results to avoid the simulation bias. The group size also follows the uniform distribution $U(0, r)$, where $r = [20, 40, 60, 80, 100, 120]$. SFC dataset is mainly used for Scenario I. The parameter settings of SFC simulations are summarized in the lower section of Table I.

B. Performance Metrics

To evaluate our proposed algorithms, we utilize the following performance metrics under various simulation settings.

Group Ratio: This metric (denoted by η) aims to describe the changes made by the grouping method, by measuring the moving distance of all the users between their original location and final group location after grouping process. In Scenario I, group ratio indicates the ratio between the total moving distance of all users and the total depth of all users in the hierarchical tree. It is defined as the ratio between the optimization function value after grouping and the original summation of levels (which is the largest possible function value where every one is grouped at the root), as follows.

$$\eta = \frac{f_I}{\sum_u l_{s_0(u)}} = \frac{\sum_u (l_{s_0(u)} - l_{s(u)})}{\sum_u l_{s_0(u)}} = 1 - \frac{\sum_u l_{s(u)}}{\sum_u l_{s_0(u)}}.$$

where $\eta \in [0, 1]$, and the larger moving distance leads to a larger value of η. It equals to 0, when everyone are grouped at their original server node. If all of the users are grouped to the group at root node, $\eta = 1$. For Scenario II, η is the ratio of average moving distance of each user and the distance to the remote cloud. Therefore, $\eta = \frac{f_{II}}{N \times D}$ or $\frac{f_{II}}{N \times d}$ for f_{II}.

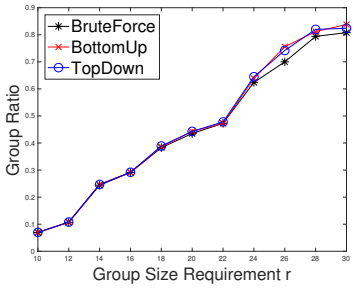


Fig. 8. **Synthetic data simulation for f_{I_3}:** group ratios of Top-Down, Bottom-Up and BruteForce with 50 participants and r = 10 to 30.

Objective Function: We use the function value f defined in Equation (1) and Equation (2) to evaluate the communication cost during the secure sharing/bidding.

Iteration Times of Algorithm 4: To optimize problem f_{I_2}, we repeatedly move a set of users to other nodes so that the objective function value can be further decreased (Lines 3-8 in Algorithm 4). Therefore, we count the number of iterations to measure the efficiency of that algorithm. The smaller value indicates the algorithm converges faster.

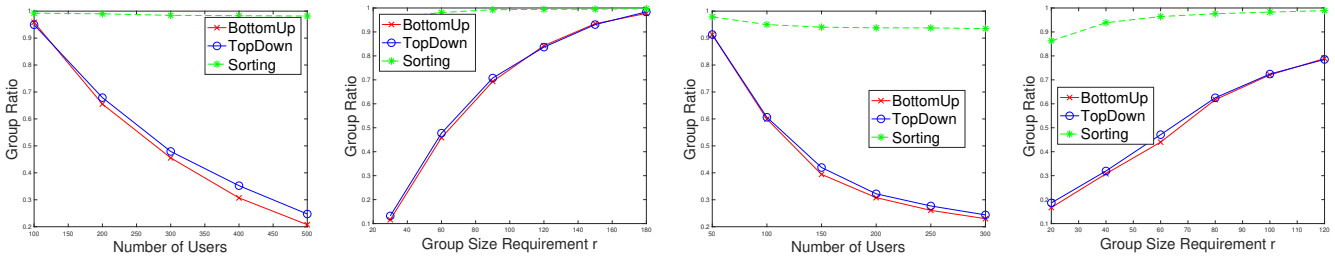
Notice that we report the average values of these metrics over multiple rounds of simulations.

C. Simulations for Scenario I

For Scenario I, due to space limitation, we only shows the performance results of our proposed methods for f_{I_3} and f_{I_2}.

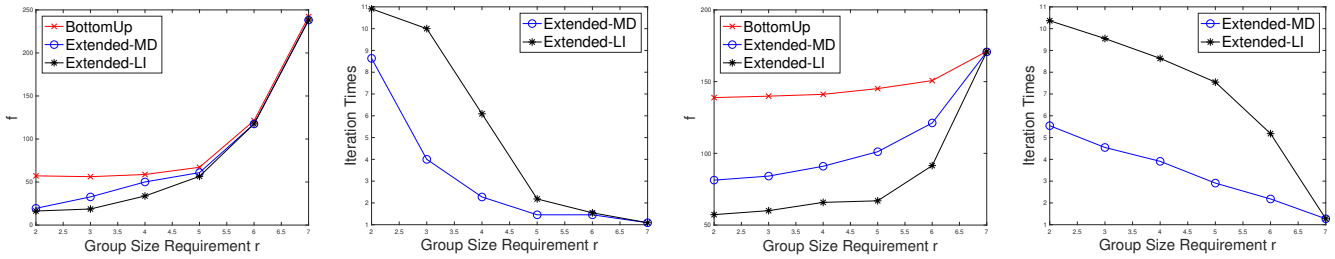
Simulation Results for f_{I_3}: To testify the efficiency of proposed algorithms, we first compare the performances of Top-Down (Algorithm 1) and Bottom-Up (Algorithm 3), against to a brute force solution (BruteForce) over a small synthetic dataset. Results on group ratio η of these three algorithms are reported in Fig. 8. Recall that η indicates the number of moves among levels during grouping and we try to keep it small. In Fig. 8, the group ratio η increases with the growth of r. It is obvious that with higher privacy requirement more group members (larger moving towards the root) are needed. In addition, the gaps between performances of our proposed methods and the one of BruteForce (which is the optimal solution) are relevantly small. This confirms the efficiency of Top-Down and Bottom-Up algorithms.

Then, we test our proposed methods over two real-life datasets with larger number of users and various group size requirements. BruteForce cannot be conducted on these datasets due to large search spaces. To demonstrate the improvement by considering communication costs in the hierarchical tree structure, we also compare our methods with the simple sorting method [28] (Sorting). To make it working for Scenario I, for each group, we set the common ancestor node of all users in the group as the final group location. If there are multiple groups on the same server, they are merged together. Fig. 9 shows the simulation results of D4D and SFC datasets. First, Sorting performs much worse than the two proposed methods for both datasets since it completely ignores the communication costs among groups caused by the level of edge structure. Second, all resulting curves have analogous



(a) D4D (b) D4D (c) SFC (d) SFC

Fig. 9. **D4D/SFC simulations for f_{I_3}:** (a) group ratios with maximal group requirement $r = 60$ and various number of users on D4D; (b) group ratios with 300 participants and different maximal group requirement r on D4D; (c) group ratios with maximal group requirement $r = 40$ and various number of users on SFC; (d) group ratios with 200 participants and different maximal group requirement r on SFC.



(a) D4D (b) D4D (c) SFC (d) SFC

Fig. 10. **D4D/SFC simulations for f_{I_2}:** (a) cost f_{I_2} and (b) the iteration times of Algorithm 4 with different maximal group size requirement r on D4D; (c) cost f_{I_2} and (d) the iteration times of Algorithm 4 with different maximal group size requirement r on SFC.

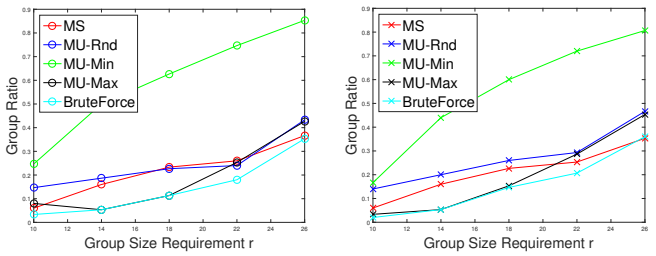
trends on both datasets, and this confirms the performances are consistent across different settings. Third, from Fig. 9(a)/(c), we can see that the group ratios decrease with more users. Clearly, with more users to select, the privacy requirement of a user can be satisfied at a higher level (lower in the tree near the leafs) with less moving. Fourth, from Fig. 9(b)/(d), it is obvious that the group ratios increase with the larger group size requirements. This is because that the users need to move towards the root to satisfy the requirements with larger group size requirement. Last, there is no significant difference between the performance of Top-Down and that of Bottom-up. Bottom-Up performs slightly better. Remember that after Step 1 (Lines 2-9) of Algorithm 3, Bottom-Up has already moved all users up to the level where their requirements can be satisfied. Most of the users with large requirements can stay at this level if there is no further grouping requests from ancestors. But in Top-Down, they may be retained at near the root to help requests from other users.

Simulation Results for f_{I_2}: We also evaluate the performance of two extended algorithms Extended-LI and Extended-MD (Algorithm 4 and its variation) with the feasible solution from Algorithm 3 for f_{I_2}. Fig. 10 presents the simulation results. Instead of measuring the group ratio, we observe the function value f directly. We also report iteration times to show the convergence speeds of these extended algorithms. From the results, we can have following observations. First, the value of f rises with larger group size requirement since users are move towards to the root and their group sizes become bigger. Second, both Extended-LI and Extended-MD can achieve better performance than Bottom-Up. This confirms the advantage of further improvement of extended algorithms for f_{I_2}. Third, even though the curves grows analogously, Extended-LI performs better than Extended-MD but with more iteration times. The reason may be that the longer the user list from partition is, the less opportunity that the participants has common ancestor and then it is more likely that vibration happens, *i.e.*, the receiver sends the same list of users to the sender. Last, the iteration times diminishes with larger group size requirement because there are less receivers with sufficient participants to accept user partitions while satisfying the privacy requirement.

D. Simulations for Scenario II

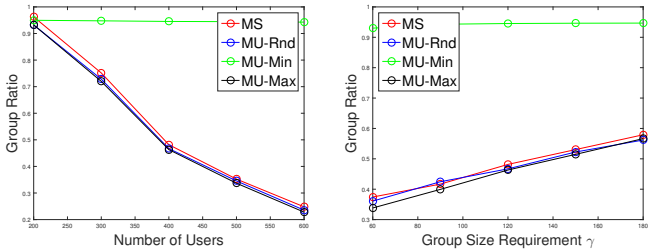
We now evaluate the performance with four different methods for both f_{II_1} and f_{II_3}: Merge-Server (MS), Merge-User with Random Picking (MU-Rnd), Merge-User with Minimum Requirement (MU-Min), and Merge-User with Maximum Requirement (MU-Max). Note that MU-Rnd, MU-Min and MU-Max are three variations of Algorithm 6 where in Line 4 it picks the user randomly or the one with minimum/maximum group size requirement.

Simulations over Graph Model (Δ, D): In this model, each link between edge servers has the identical communication cost and the topology is formed based on social ties between edge servers. We first compare the performances of our proposed algorithms with the optimal solution obtained from a brute force method (BruteForce) for both f_{II_1} and f_{II_3}, using a small D4D dataset with only 50 users. Results are reported in Fig. 11. Here, various maximal group size requirements $r = [10, 14, 18, 22, 26, 30]$ are used. With the increasing group size requirement, the group ratio grows since more users needs to move around and further. MU-Min performs the worst since the group is built with the user who has larger group size requirement. The new group member is barely



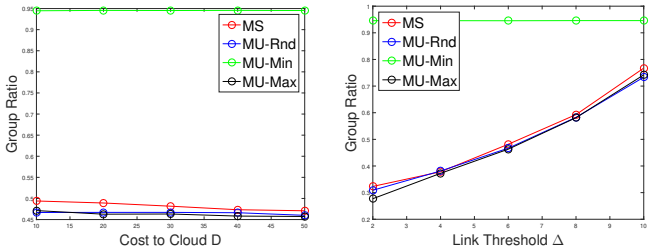
(a) (b)

Fig. 11. **D4D simulation for f_{II_1} (a) and f_{II_3} (b) - Graph Model (Δ, D):** group ratios of Brute Force, Merge-Sever and Merge-User with 50 participants and $r = 10$ to 26.



(a) (b)

Fig. 12. **D4D simulation for f_{II_1} - Graph Model (Δ, D):** (a) group ratios with various number of users N where $\gamma = 120$, $D = 30$, $\Delta = 6$; (b) group ratios with different γ where $N = 400$, $D = 30$, $\Delta = 6$.

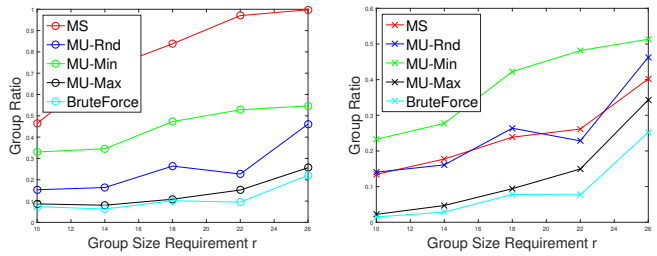


(a) (b)

Fig. 13. **D4D simulation for f_{II_1} - Graph Model (Δ, D):** (a) group ratios with various cost to the remote cloud D where $N = 400$, $\gamma = 120$, $\Delta = 3$; (b) group ratios with different distance threshold Δ where $N = 400$, $\gamma = 120$, $D = 30$.

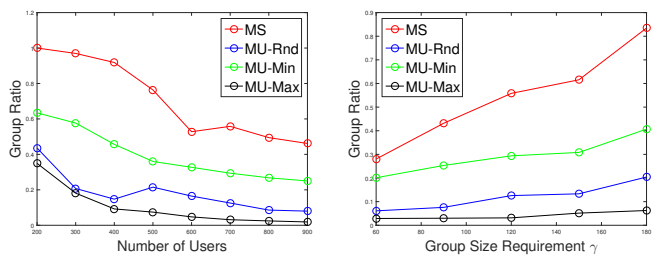
to be satisfied and the group need more users. MU-Rand performs poorly since we merge the users with random group size requirement. When the group size requirement is small, the performance of MU-Max is better than that of MS. MS performs closer to optimal solution obtained by BruteForce when the group size requirement is large. This is because MS performs grouping on servers where the users on a sever are handled together. Overall, the results confirm the efficiency of proposed solutions MU (mainly MU-Max) and MS.

Next, we evaluate the performance of proposed algorithms on a larger D4D dataset with various group size requirements and numbers of users. Due to space limit, we only present the results for f_{II_1} in Fig. 12. MU-Min again perfroms the worst among all the methods since it always groups with users who have larger group size requirements. With growing number of users, the group ratio drops since more groups could be formed locally and less users are shifted. With larger group size requirement, more users need to shifted to other servers so



(a) (b)

Fig. 14. **D4D simulation for f_{II_1} (a) and f_{II_3} (b) - Graph Model (δ, d):** group ratios of Brute Force, Merge-Sever and Merge-User with 50 participants and $r = 10$ to 26.



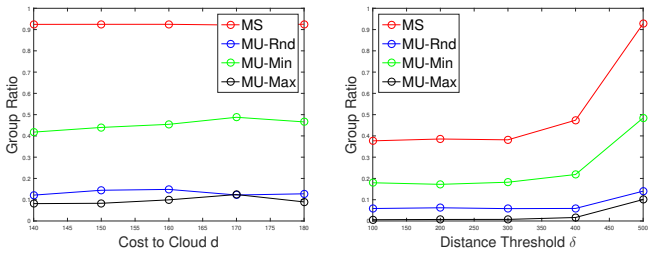
(a) (b)

Fig. 15. **D4D simulation for f_{II_1} - Graph Model (δ, d):** (a) group ratios with various number of users N where $\gamma = 120$, $d = 163$, $\delta = 500$; (b) group ratios with different γ where $N = 700$, $d = 163$, $\delta = 500$.

that the group ratio becomes larger. MU-Max always preforms the best.

Last, we demonstrate the influence of the cost to remote cloud D and the link threshold Δ on group ratios in Fig. 13. Notice that with rising cost to the cloud, the group ratio decreases. This fits the definition of group ratio with larger denominator. On the other hand, the larger Δ indicates the more isolation of each server nodes. Hence, the unsatisfied users are shifted to the remote cloud so that the group ratio becomes greater.

Simulations over Graph Model (δ, d): Different from graph model (Δ, D), graph model (δ, d) generates the edge cloud topology and its link cost based real distance between edge servers in D4D. The cost between different edge severs are various, and thus this is more realistic. Similarly, we first run the our proposed algorithms with small amount of data (50 users) to compare with the brute force solution, and results are given in Fig. 14 for f_{II_1} and f_{II_3}. We could see that the group ratio grows with larger group size requirement, just as in Fig. 11. Comparing with Fig. 11, the curves from different methods in Fig. 14 are more distinguishable, since link costs in this graph model are not identical anymore. Among all methods, MU-Max outperforms all other strategies with a clear margin. MS performs worse for f_{II_1} than for f_{II_3}, which is much higher than MU-Min. Note that in this graph model, the delay is based on physical distance between edge servers instead of an identical value (1), thus total group delay is not scale with group size any more. In addition, in f_{II_1}, it only considers the maximal total delay among groups other than the total delay in all groups. This observation (MS is worse than MU-Min) also exists in Fig.15 and Fig.16.



(a) (b)

Fig. 16. **D4D simulation for f_{II_1} - Graph Model (δ, d):** (a) group ratios with various cost to the remote cloud d where $N = 400$, $\gamma = 120$, $\delta = 500$; (b) group ratios with different link threshold δ where $N = 400$, $\gamma = 120$, $d = 163$.

Fig. 15 shows the simulation results for f_{II_1} with various number of users and group ratios. Clearly, compared with Fig. 12, the curves are more distinguishable and the group ratio of MS-Rnd is much higher than MU-Max as well. Similar to Fig. 14(a), MU-Max is the best and MS is the worst among all solutions. In Fig.15(a), with more users, the group size requirement gets satisfied easier. In Fig.15(b) with a certain number of users, the group ratio grows with larger group size requirement.

Finally, Fig. 16 show the results with various cost to cloud d and distance threshold δ. In Fig. 16(a), since the distance threshold δ keep as 500 whose delay is much smaller than the delay to cloud d, group ratio stays the same for most of the time. In Fig. 16(b), the group ratio raises with larger distance threshold δ. This is because users have more opportunities to be in a group on a further edge server.

VI. Related Works

A. Participant Selection in Mobile Sensing

Due to the large number of participants and the diverse sensing tasks in mobile sensing, the participant selection for different tasks becomes challenging. On one hand, assigning more participants for certain task can lead to better quality of the sensed data. On the other hand, mobile sensing platforms have to pay more rewards to the participants to cover their sensing cost. Recently, there are several studies on participant selection in mobile social sensing or mobile crowd sensing with various optimization goals such as coverage optimization [4], [5], energy efficiency [6]–[8], user incentive and truthfulness [9]–[15], [53], sensing quality [16]–[19], data collection [20]–[22] and truth discovery [23]–[25]. In this paper, we focus on the privacy protection during participant selection where secure bidding or sharing is used at edge clouds.

B. Mobile Sensing over Edge Clouds

Using edge computing with edge servers in mobile sensing could not only reduce the connection latency to enable timely performing larger-scale sensing tasks, but also better protect the privacy by keeping data locally at the edge (instead of sensing it to remote cloud). There are a few recent studies [44]–[46], [54] adopting this idea for tasking allocation, collusion resistance, incentive mechanism, and privacy protection. In [44], the edge node could allocate tasks locally based on the users' local information. The authors mainly focus on how to securely remove the duplicated date reports without revealing the user's information. Along the same line, Zhu *et al.* [46] leverage encryption techniques to prevent collusion attacks over fog nodes during the privacy-preserving data reporting and requesting. Li *et al.* [54] study how to better attract more participants for edge-based crowd sensing with the help of POI-tagging Apps. The authors model the interactions of users, platform, and Apps by a three-stage decision process and provide solutions for all reward, selection and payment decisions. In [45], edge nodes allocate the tasks to a targeted subset of users with high reputation obtained and maintained through the context online learning at cloud. The authors adopt the differential-privacy to protect the users' privacy by selecting the users according to a probability function. Note that all these studies are not on privacy-preserving grouping over edge clouds.

C. Privacy Protection in Mobile Sensing

To protect the users' privacy in mobile crowd sensing, many privacy-preserving techniques, such as data transform language [55], data aggregation [33], [56], [57], location obfuscation [30], [58], [59], cloaking [31], [60], k-anonymity [32], pseudonym [61], adding noise [29], and differential privacy [10], [26], [45], have been used in different procedures (e.g., data collection, data aggregation, location generation) to achieve the protection of the sensing data privacy, participants' anonymity or their location privacy.

D. Privacy-Preserving Participant Selection

In contrast to the data or location privacy solutions above, there are also recent efforts [17], [26]–[28] on protecting bid privacy or sensing quality privacy during the procedure of participant selection. Jin *et al.* [26] consider bid privacy in an aggregated MCS system. They define the bid privacy with differential privacy over the aggregated sensing data and assume that all sensing tasks are binary classification tasks. Li *et al.* [27], [28] consider a more general and direct model, where sensing tasks have sensing requests on both temporal and spacial domains and privacy is defined on the bids from participants. By leveraging Lagrange polynomial interpolation (LPI), their solution perturbs the participants' bids within groups so that bid information is protected during the group bidding and the final bidding at the platform. The communication cost of the LPI-based secure biding within a group with k members is $O(k)$. Xiao *et al.* [17] also address privacy-preserving participant selection by using secret sharing schemes. They consider quality-aware participant selection while trying to protect the inputs (sensing qualities) of each user from being revealed to the platform or to other users. The basic idea is to leverage the secure multi-party multiplication and secure multi-party comparison protocols. Their solution's communication cost is $O(k^2)$, where k is the number of participants. Note that secure sharing/bidding schemes has already been studied [17], [27], [28] and are discussed here. In this paper, we focus on privacy-preserving user grouping, which

can be combined with these secure sharing/bidding schemes to achieve overall privacy-preserving participant selection.

In [28], we did consider a simpler version of participant grouping which simplifies the communication cost between group members as the same value. Such problem can be easily solved by the proposed dynamic programming method. In this paper, we extend the problem into multiple optimization problems over edge clouds and consider the different types of communication costs and topologies of edge clouds. To the best of our knowledge, this is the first work tackling the privacy preserving participant grouping for MSS over edge clouds.

E. Grouping/Clustering for Privacy Protection

Grouping/clustering has been recently studied for privacy preserving [62]–[64] . Given a set of n points in general metric space and a value r, the r-gather problem is defined as clustering the points into groups at least r points each such that the largest diameter of clusters are minimized [62]. Aggarwal *et al.* [62] prove that there's a polynomial time algorithm that give a 2-approximation to the problem. Armon [63] extends such results and shows that it's NP-hard to approximate with a ratio better than 3 for $r > 2$ for general metric space. Zeng *et al.* [64] describes a distributed algorithm with an approximation factor of 4 for r-gather problem. All of these existing works cluster the points with only one parameter r, however, in our model, the privacy criteria r is different from each user and the optimization problem is formulated differently. Also, the existing algorithms (e.g. sweep algorithm) could not applied here in HEC architecture since the location of users' group could be only on its ancestor node but anywhere else. Last, none of algorithms are proposed for problem *Min-Sum*.

VII. Conclusion

In this paper, we form and investigate the participant grouping problems over hierarchical or interconnected edge clouds in mobile crowd sensing to protect participants' privacy. Via privacy-preserving grouping at edge clouds, participants' information is hidden in groups at local edge server and participant selection can be performed by secure bidding/sharing among group members efficiently. Overall both privacy and scalability are significantly improved. Particularly, we propose a set of grouping schemes to achieve our goal for different scenarios under various optimization functions. Extensive simulations over three datasets have been conducted to substantiate the efficiency of our proposed methods. We leave the NP-harness proof of PPPG-HEC/PPPG-IEC and theoretical analysis of our proposed methods as possible future works. In addition, we will further develop more privacy-preserving schemes for MSS in aim to balance privacy protection with participant selection performance.

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges," *IEEE Communications Magazine*, 49(11), 2011.

[2] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Computing Surveys*, 48(1), 2015.

[3] D. Wang, B.K. Szymanski, T. Abdelzaher, H. Ji, and L. Kaplan, "The age of social sensing," *IEEE Computer*, 52(1):36-45, 2019.

[4] D. Zhang, H. Xiong, L. Wang, and G. Chen, "Crowdrecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proc. of ACM UbiComp*, 2014.

[5] H. Li, T. Li, W. Wang, and Y. Wang, "Dynamic participant selection for large-scale mobile crowd sensing," *IEEE Trans. on Mobile Computing*, 18(12):2842-2855, 2019.

[6] H. Xiong, D. Zhang, L. Wang, and H Chaouchi, "EMC^3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint," *IEEE Trans. on Mobile Computing*, 14(7): 1355 -1368, 2014.

[7] H. Xiong, D. Zhang, L. Wang, J.P. Gibson, and J. Zhu, "EEMC: Enabling energy-efficient mobile crowdsensing with anonymous participants," *ACM Trans. on Intelligent Systems and Technology*, 6(3), 2015.

[8] D. Zhao, H. Ma, and L. Liu, "Energy-efficient opportunistic coverage for people-centric urban sensing," *Wireless Networks*, 20(6):1461-1476, 2014.

[9] D. Zhao, X.-Y. Li, and H. Ma, "Budget-feasible online incentive mechanisms for crowdsourcing tasks truthfully," *IEEE/ACM Transactions on Networking*, 24(2): 647 - 661, 2014.

[10] H. Jin, L. Su, and K. Nahrstedt, "Centurion: Incentivizing multi-requester mobile crowd sensing," in *Proc. of IEEE INFOCOM*, 2017.

[11] Y. Wu, F. Li, L. Ma, Y. Xie, T. Li, and Y. Wang, "A context-aware multi-armed bandit incentive mechanism for mobile crowd sensing systems," *IEEE Internet of Things Journal*, 6(3):5853-5863, 2019.

[12] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos, "TRAC: truthful auction for location-aware collaborative sensing in mobile crowdsourcing," in Proc. of IEEE INFOCOM, 2014.

[13] Y. Wang, Z. Cai, Z. Zhan, Y. Gong, and X. Tong, "An optimization and auction based incentive mechanism to maximize social welfare for mobile crowdsourcing," *IEEE Transactions on Computational Social Systems* 6(4):702-714, 2019.

[14] K. Sooksatra, R. Li, Y. Li, X. Guan and W. Li, "Fairness-aware auction mechanism for sustainable mobile crowdsensing," in *Proc. of WASA*, 2019.

[15] J. Li, Z. Cai, J. Wang, M. Han, and Y. Li, "Truthful incentive mechanisms for geographical position conflicting mobile crowdsensing systems," *IEEE Transactions on Computational Social Systems*, 5(2):324-334, 2018.

[16] B. Guo, H.i Chen, Q. Han, Z. Yu, D. Zhang, and Y. Wang, "Worker-contributed data utility measurement for visual crowdsensing systems," *IEEE Trans. on Mobile Computing*, 16(8): 2379-2391, 2017.

[17] M. Xiao, J. Wu, S. Zhang, and J. Wu., "Secret-sharing-based secure user recruitment protocol for mobile crowdsensing," in *Proc. of IEEE INFOCOM*, 2017.

[18] H. Li, T. Li, F. Li, S. Yang, and Y. Wang, "Multi-expertise aware participant selection in mobile crowd sensing via online learning," in *Prof. of IEEE MASS*, 2018.

[19] H. Li, T. Li, F. Li, Y. Wu, and Y. Wang, "Cumulative participant selection with switch costs in large-scale mobile crowd sensing," in *Prof. of IEEE ICCCN*, 2018.

[20] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proc. of IEEE INFOCOM*, 2015.

[21] Y. Wang, H. Li, and T. Li, "Participant selection for data collection through device-to-device communications in mobile sensing," *Personal and Ubiquitous Computing*, 21(1):31-41, 2017.

[22] L. Wang, D. Zhang, and H. Xiong, "effSense: energy-efficient and cost-effective data uploading in mobile crowdsensing," in *Proc. of ACM UbiComp*, 2013.

[23] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng, "Truth discovery on crowd sensing of correlated entities," In *Proc. of ACM Sensys*, 2015.

[24] H. Jin, L. Su, and K. Nahrstedt, "Theseus: Incentivizing truth discovery in mobile crowd sensing systems," in *Proc. of ACM MobiHoc*, 2017.

[25] C. Huang, D. Wang, and N. Chawla, "Towards time-sensitive truth discovery in social sensing applications," in *Proc. of IEEE MASS*, 2015.

[26] H. Jin, L. Su, B. Ding, K. Nahrstedt, and N. Borisov, "Enabling privacy-preserving incentives for mobile crowd sensing systems," in *Proc. of IEEE ICDCS*, 2016.

[27] T. Li, T. Jung, H. Li, H. Li, L. Cao, W. Wang, X. Y. Li, and Y. Wang, "Scalable privacy-preserving participant selection in mobile crowd sensing," in *Proc. of IEEE PerCom*, 2017.

[28] T. Li, T. Jung, Z. Qiu, H. Li, L. Cao, and Y. Wang, "Scalable privacy-preserving participant selection for mobile crowdsensing systems: participant grouping and secure group bidding," *IEEE Transactions on*

Network Science and Engineering, Volume: 7, Issue: 2, Pages: 855-868, April-June 2020.
[29] R.K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher, "Poolview: Stream privacy for grassroots participatory sensing," in *Proc. of ACM SenSys*, 2008.
[30] B. Agir, T. G. Papaioannou, R. Narendula, K. Aberer, and J.-P. Hubaux, "User-side adaptive protection of location privacy in participatory sensing," *GeoInformatica*, 18(1):165–191, 2014.
[31] L. Pournajaf, L. Xiong, V. S. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *Proc. of IEEE MDM*, 2014.
[32] K.L. Huang, S.S. Kanhere, and W. Hu, "Preserving privacy in participatory sensing systems," *Comput. Commun.*, 33(11):1266-1280, 2010.
[33] Y. Liu, H. Wang, M. Peng, J. Guan, J. Xu, and Y. Wang, "DeePGA: A privacy-preserving data aggregation game in crowdsensing via deep reinforcement learning," *IEEE Internet of Things Journal*, Volume: 7, Issue: 5, Pages: 4113 - 4127, May 2020.
[34] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things J.*, 2018
[35] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Transactions on Mobile Computing*, to appear.
[36] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet of Things Journal*, Volume: 6 , Issue: 3 , Pages 5853 - 5863, June 2019.
[37] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, 6, pp.18209-18237, 2018.
[38] Z. Qian, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *Proc. of ACM CCS*, 2016.
[39] M. Du, K. Wang, Y. Chen, X. Wang, and Y. Sun, "Big data privacy preserving in multi-access edge computing for heterogeneous Internet of Things," *IEEE Communications Magazine*, 56(8):62-7, 2018.
[40] X. Li, S. Liu, F. Wu, S. Kumari, and JJ. Rodrigues, "Privacy preserving data aggregation scheme for mobile edge computing assisted IoT applications," *IEEE Internet of Things Journal*, 6(3):4755-63 ,2018.
[41] J. B. Abdo, T. Bourgeau, J. Demerjian, and H. Chaouchi, "Extended privacy in crowdsourced location-based services using mobile cloud," computing," *Mobile Inf. Syst.*, 2016
[42] L. Ma, X. Liu, Q. Pei, and Y. Xiang, "Privacy-preserving reputation management for edge computing enhanced mobile crowdsensing," *IEEE Transactions on Services Computing*, 12(5):786-99, 2018.
[43] S. Basudan, X. Lin, and K. Sankaranarayanan, "A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing," *IEEE Internet of Things Journal*, 4(3):772-82, 2017.
[44] J. Ni, K. Zhang, Y. Yu, X. Lin and X. S. Shen, "Providing task allocation and secure deduplication for mobile crowdsensing via fog Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 581-594, 1 May-June 2020.
[45] P. Zhou, W. Chen, S. Ji, H. Jiang, L. Yu, and D. Wu, "Privacy-preserving online task allocation in edge-computing-enabled massive crowdsensing," *IEEE Internet of Things Journal*, 6(5):7773–7787, 2019.
[46] L. Zhu, M. Li, and Z. Zhang, "Secure fog-assisted crowdsensing withcollusion resistance: From data reporting to data requesting," *IEEE Internet of Things Journal*, 6(3):5473–5484, 2019.
[47] T. Li, Z. Qiu, L. Cao, F. Li, Z. Guo, X. Shi, and Y. Wang, "Participant Grouping for Privacy Preservation in Mobile Crowdsensing over Hierarchical Edge Clouds," in *IEEE IPCCC*, 2019.
[48] The Data for Development (D4D) Challenge, by Orange in 2013, http://www.d4d.orange.com.
[49] G. Ziegler, "Numerical Distance Protection: Principles and Applications", pp 51- 53, ISBN: 978-3-895-78667-9, 2011
[50] S. Cheng, Z. Chen, J. Li, H. Gao, "Task Assignment Algorithms in Data Shared Mobile Edge Computing Systems," in *Proc. of IEEE ICDCS*, 2019
[51] https://www.cloudping.info/
[52] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009-02-24)," Downloaded from http://crawdad.org/epfl/mobility/20090224, Feb. 2009.
[53] Z. Wang, J. Li, J. Hu, J. Ren, Z. Li, and Y. Li, "Towards privacy-preserving incentive for mobile crowdsensing under an untrusted platform," in *Proc. of IEEE INFOCOM*, 2019.
[54] Y. Li, F. Li, S. Yang, H. Chen, Q. Zhang, Y. Wu, and Y. Wang, "PTASIM: Incentivizing crowdsensing with POI-tagging cooperation over edge clouds," *IEEE Transactions on Industrial Informatics*, Volume: 16, Issue: 7, Pages: 4823 - 4831, July 2020..
[55] C. Cornelius, A. Kapadia, and N. Triandopoulos, "Anonysense: Privacy-aware people-centric sensing," in *Proc. of ACM MobiSys*, 2008.
[56] J. Shi, R. Zhang, Y. Liu, and Y. Zhang, "Prisense: Privacy-preserving data aggregation in people-centric urban sensing systems," in *Proc. of IEEE INFOCOM*, 2010.
[57] H. Duan, Y. Zheng, Y. Du, A. Zhou, C. Wang, and M.H. Au, "Aggregating crowd wisdom via blockchain: A private, correct, and robust realization," in *IEEE PerCom*, 2019.
[58] L. Wang, D. Yang, X. Han, D. Zhang, and X. Ma, "Mobile crowdsourcing task allocation with differential-and-distortion geo-obfuscation," *IEEE Transactions on Dependable and Secure Computing*, to appear.
[59] J. Ni, K. Zhang, Q. Xia, X. Lin, and X.S. Shen, "Enabling strong privacy preservation and accurate task allocation for mobile crowd-sensing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1317-1331, 1 June 2020.
[60] Y. Wang, D. Xu, and F. Li, "Providing location-aware location privacy protection for mobile location-based services," *Tsinghua Science and Technology*, 21(3):243-259, 2016.
[61] Q. Li and G. Cao, "Providing privacy-aware incentives for mobile sensing," in *Proc. of IEEE PerCom*, 2013.
[62] G. Aggarwal, R. Panigrahy, T. Feder, D. Thomas, K. Kethapadi, S. Khuller, and A. Zhu, "Achieving Anonymity via Clustering," *ACM Trans. on Algorithms*, 6(3):1549-6325, 2010.
[63] A. Armon, "On Min-max R-gatherings" *Theor. Comput. Sci.*, 412(7):573–582, 2011.
[64] J. Zeng, T. Gaurish, M.P. Johnson, R. Sarkar, J. Gao, E.M. Arkin, and J.S.B. Mitchell, "Mobile r-gather: distributed and geographic clustering for location anonymity," in *Proc. of ACM Mobihoc*, 2017.

APPENDIX

Theorem 1: Let N be the total number of participants and I be the number of levels in the hierarchical edge cloud tree. If η is the number of all feasible solution divided by all possible solution, then

$$\eta \leq \frac{1}{(I-1)^{N_n}} \left(\frac{I-1}{I}\right)^{N-N_I},$$

where N_n is the number of participants on the root node and N_I is the total number of participants on leaf nodes after running Lines 4-9 of Algorithm 3.

Proof: We begin by calculating the denominator. Since there are N participants and each can be at any one of I levels, there are I^N ways to place the participants when there are no restrictions. Hence, the denominator is I^N. We now turn to the numerator. Lines 4-9 in Algorithm 3 place participants in levels such that there does not exist a feasible solution with them at a lower level. Thus, the N_n placed at the root must stay at the root and so there is only one way to place them. N_I at the leaves can be placed at any level, thus there are I^{N_I} way to place them. The remaining $N-N_n-N_I$ cannot be placed at the leaves, thus there are at most $I-1$ possible levels for them. It follows that they can be placed in, at most, $(I-1)^{N-N_n-N_I}$ levels. Hence, the numerator is $I^{N_I}(I-1)^{N-N_n-N_I}$. Performing the division and simplifying gives the result. ■

We now plot the upper bound on η for various values of the parameters N_I, N, N_n, I. This shows that the upper bound, η, can be quite small. Notice that $0 \leq \frac{1}{(I-1)^{N_n}}(\frac{I-1}{I})^{N-N_I} \leq 1$, hence, its log will increase to 0 as the ratio approaches 1. With the prediction from the formulation, we can clearly notice that the value linearly decreases with growth of N in Fig. 17(a) and the same trend as with N_n in Fig. 17(c). The ratio rises with increasing I and growth slowly even decrease a little bit with

larger I in Fig. 17(b) because $\frac{I-1}{I}$ is getting close to 1 while at the same time $\frac{1}{I-1}$ is getting smaller. At last, Fig. 17(d) also testifies that the ratio increases with larger N_I.

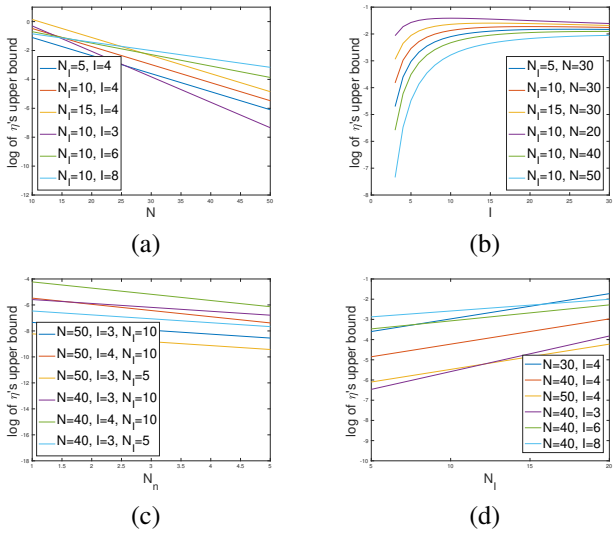


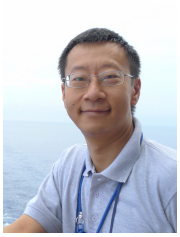
Fig. 17. Bound of η: (a) with different total number of users and (b) with different levels of tree structure, here $N_n = 1$ while N_I and I are set with various values. (c) with different number of users on the root node and (d) with different number of users on the leaf nodes when $N_n = 1$.

Ting Li received the B.S. degree in information engineering from Xidian University, China, in 2012, and her Ph.D. degree in computer science from the University of North Carolina at Charlotte in 2019. She is an assistant professor in the Division of Natural Science and Mathematics, Oxford College of Emory University. Her research interests include privacy, mobile crowdsensing and cognitive radio networks.

Zhijin Qiu received the B.S., M.S. and Ph.D. degrees from Qingdao University of Science and Technology, China, in 2011, 2013, and 2018, respectively, all in computer science. He is currently an assistant researcher in the Institute of Oceanographic Instrumentation at Qilu University of Technology, Qingdao, Shandong, China. He was a visiting Ph.D student at University of North Carolina at Charlotte from 2016 to 2017.

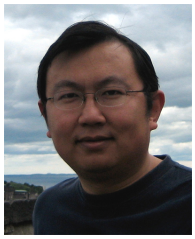
Lijuan Cao is a teaching associate professor in Department of Software and Information Systems at the University of North Carolina at Charlotte. She received her Ph.D. in Information Technology from UNC Charlotte in 2008, and her B.S. degree in Computer Science from the University of Electronic Science and Technology of China in 2003. Her research focuses on wireless networking and mobile computing.

Dazhao Cheng received the BS and MS degrees in electronic engineering from the Hefei University of Technology, in 2006 and the University of Science and Technology of China, in 2009, respectively. He received the PhD degree from the University of Colorado, Colorado Springs, in 2016. He is currently an assistant professor with the Department of Computer Science, University of North Carolina, Charlotte. His research interests include big data and cloud computing. He is a member of the IEEE and ACM.



Weichao Wang is a Professor at the Department of Software and Information Systems, the University of North Carolina at Charlotte. He received his Ph.D degree in Computer Science from Purdue University in 2005. He also holds Master's Degrees in Computer Science from Purdue University (2002) and Tsinghua University (2000), and the Bachelor's Degree in Computer Science from Tsinghua University (1998). His research interests are in designing protocols and mechanisms to secure pervasive systems, wireless networks, cloud computing environments, and critical infrastructures. He is a member of the IEEE, the ACM, and the ASEE.

Xinghua Shi (S'08-M'17) received the B.Eng. and M.Eng. degrees in computer science from Beijing Institute of Technology, China, and the M.S. and Ph.D. degrees in computer science from the University of Chicago. She is an Associate Professor in the Department of Computer and Information Science at Temple University. Previously, she was with the University of North Carolina at Charlotte, Brigham and Women's Hospital and Harvard Medical School. Her research interests include machine learning, bioinformatics, data privacy, and big data analytics in biomedical research. She is an ACM member and IEEE Member.



Yu Wang (S'02-M'04-SM'10-F'18) is a Professor in the Department of Computer and Information Science at Temple University. Previously, he was with the Department of Computer Science at University of North Carolina at Charlotte. He received his Ph.D. degree in Computer Science from Illinois Institute of Technology, his B.Eng. degree and M.Eng. degree in Computer Science from Tsinghua University, China. His research interest includes wireless networks, smart sensing, and mobile computing. He has published over 200 papers in peer reviewed journals and conferences, with four best paper awards. He is a recipient of Ralph E. Powe Junior Faculty Enhancement Awards from Oak Ridge Associated Universities (2006), Outstanding Faculty Research Award from College of Computing and Informatics at the University of North Carolina at Charlotte (2008), and Fellow of IEEE (2018). He is also a senior member of ACM and a member of AAAS.