COMPUTERS & SECURITY 73 (2018) 425–438

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Location privacy in public access points positioning: An optimization and geometry approach

Yunlong Mao [a], Yuan Zhang [a], Xiaoyan Zhang [b], Fengyuan Xu [a], Sheng Zhong [a,*]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, China
[b] School of Mathematical Science & Institute of Mathematics, Nanjing Normal University, China

ARTICLE INFO

Article history:
Received 16 March 2017
Received in revised form 29 October 2017
Accepted 5 November 2017
Available online 6 December 2017

Keywords:
Access point
Location privacy
Optimization
Belief propagation
Submodular

ABSTRACT

This study proposes a method to tackle a new threat in current location-based services (LBS) wherein access points (APs) use shared public IP addresses. In this scenario, if an LBS request originates from one user, the location of other users who are using the same public IP address will also be revealed, which is a significant threat to user privacy. In this paper, we propose a novel approach for the protection of location privacy when APs have shared public IP addresses. The basic idea behind ensuring location privacy is to divide APs into two exclusive groups, namely, one for LBS users, and the other for non-LBS users. Achieving this bipartition while keeping every user in the service region connecting to one AP is difficult. To resolve this challenge, we first propose two optimal algorithms to perform the bipartition based on different interests. We then use Loopy Belief Propagation to propose a min-sum belief propagation algorithm on a loopy graph model to guarantee the quality of service by complete coverage. Experimental results demonstrate the accuracy and feasibility of our algorithms.

1. Introduction

With the popularization of smart devices in recent years, people have become accustomed to accessing the Internet via Wi-Fi connections. Mobile data traffic is growing rapidly, a significant portion of which is contributed by Wi-Fi connections. According to a white paper from Cisco posted in February 2016 (Cisco visual networking index: Global mobile data traffic forecast update, 20152020 white paper, 2016), global mobile data traffic reached 3.7 exabytes per month at the end of 2015. Mobile offload exceeded cellular traffic for the first time in 2015, with 51% of total mobile data traffic offloaded onto fixed networks through Wi-Fi or femtocell. The significant volume of mobile data traffic is attributable to various applications. Among these are mobile applications in which location-based services (LBS) play an increasingly important role. Countless applications access users' location information. For example, location-based advertising within a map application has excited considerable interest among merchants.

A Wi-Fi positioning system (WFPS) is a system that works to locate users of wireless networks through their wireless access points. Service providers can obtain users' locations through IP-location services. Users can be located by utilizing IP-location mappings, sometimes called MAC-location mappings. Several Wi-Fi location databases exist, such as the

* Corresponding author.
E-mail address: zhongsheng@nju.edu.cn (S. Zhong).
https://doi.org/10.1016/j.cose.2017.11.005
0167-4048/

Combain Positioning Service (Covergage wifi – combain positioning service) and the Mozilla location service. Service providers can search users' Wi-Fi IP addresses in these databases. If records are found, users will be located by IP-location mappings. But these databases can only provide, at most, city-level accuracy, and most are commercial products. In order to satisfy market needs, many web companies such as Google are actively working on collecting IP-location mappings and constructing more accurate IP-location databases. Such efforts will have an impact on users' privacy.

Meanwhile, a new privacy threat (Vratonjic et al., 2013, 2014) aimed at Wi-Fi networks and their users, has been uncovered. We briefly describe it here to explain the motivations underlying this study. In general, the allocation of IP addresses can be static or dynamic. Dynamic allocation is primarily done through the dynamic host configuration protocol (DHCP). However, a reliable study (Casado, 2007) showed that the speed of IP reallocation reduces owing to DHCP. This makes it possible to build mappings between IP addresses and hosts. Additionally, network address translation (NAT) has been widely used for private address spaces. According to Casado (2007), a majority (roughly 60%) of hosts are behind NATs. When users connect to an AP while using NAT, they may share the same public IP address to access the Internet. We can assume that there are some users connecting to an AP, and they are responsible for the AP's NAT and access the Internet with a shared public IP address which may be assigned by ISP's DHCP service. If a user originates LBS requests to a curious provider, this user will disclose his exact location. Using this information, the service provider can build a mapping between the AP's public IP and its location. As long as AP's public IP remains unchanged, all its users are at risk of location exposure. When they send any authentication request to the server owned by the same provider, their identity and location will be connected and tracked, even though these are privacy-preserving, and even if the users have never used any LBS. Thus, even when protected by conventional techniques, non-LBS users' location privacy will still be exposed by this attack. The experimental results of Vratonjic et al. (2014) show that a service provider can learn the location of the AP only approximately an hour after users start connecting to it, and can locate up to 73% of the users' identities within 24 hours. Based on a real dataset, in a typical setting, Google learns the location of up to 90% of its users without explicitly disclosing such discovery. Intuitively, there may be several ways to address this problem, but as stated in Vratonjic et al. (2014), they will not work very well. We will discuss them in a related work.

As this threat cannot be solved simply or by conventional means, in this paper, we try to address this threat through optimization of AP utilization and modelling the geometry of AP coverage. Our goal is to protect the location privacy of AP[1] users who are not using LBS, without seeking an ISP's help or modifying their DHCP servers. Our solutions are based on the fact that there are normally several APs in one public location. The mainstream product usually has a practical covering radius of about 20 meters. The investigation performed here shows that when localization accuracy is under 50 meters[2], one public location (such as a library hall or a train station's waiting room) is always covered by multiple APs. We have proposed two bipartition algorithms focusing on accuracy and flexibility respectively. These two algorithms can be used to divide existing APs into two groups, with coverage maximized, so that LBS users and non-LBS users alike can access the Internet with isolated APs. What if these two groups cannot have their entire region covered by existing APs? To guarantee the quality of service with complete Wi-Fi signal coverage, we have also proposed a complete covering algorithm to add a minimum number of APs to cover entire regions. However, it is highly challenging to generate two groups for LBS requests and non-LBS requests respectively:

1. How can we divide existing APs into two groups so that both groups can cover the entire service region? The existing APs are fixed, and we need to divide them into two groups with original positions.
2. If it is impossible to divide the existing APs into two groups and satisfy each coverage requirement, then how can we add a minimum number of APs in order to form two such groups? The existing APs may be able to cover the entire region, but when we divide them into two groups, they may not cover the entire region. It is very difficult to deploy additional APs while original APs exist in this region.

In general, our contributions can be summarized as follows:

- We consider a new location privacy threat in more realistic scenes and propose two bipartition (BP) algorithms to divide existing APs into two groups to protect non-LBS users' location privacy. One algorithm adapts a BP algorithm to achieve more accurate partition and the other formalizes a submodular function to obtain efficient and flexible bipartition.
- To guarantee the quality of Wi-Fi service, we also propose a complete covering algorithm through which we can add a minimum number of APs to form two AP groups both covering the entire region. If the minimum number is zero, then we can divide existing APs without any additional APs. We achieve this algorithm by formalizing an optimization problem and adapting a Loopy Belief Propagation (LoopyBP) algorithm. Furthermore, we analyze the convergence of our LoopyBP algorithm, which is important for any such application. The formalization and analysis of LoopyBP are nontrivial.
- We verify the correctness and feasibility of all proposed algorithms through experimental evaluation, including real-life two scenarios. Also, by investigating the influence of practical parameters on our solutions, we find our solutions both feasible and effective.

[1] An AP which has shared public IP address, we will simply refer to this by AP instead in the rest of this paper if without any specific statement.

[2] This range is reasonable for most LBS applications.

2. Related work

Location privacy has been widely studied in various areas and circumstances. But the location privacy issue for public APs was firstly identified in Vratonjic et al. (2013, 2014). These authors introduced a solid threat model to estimate the probability of a user's location privacy being revealed by a service provider. This threat model is impressive, and the authors left its solution as an open question. Although they provide a discussion of several countermeasures, their conclusion was that existing methods did not sufficiently address the threat. Possible countermeasures can be classified as follows. First, cryptographic primitives are not suitable. If users' location coordinates are encrypted, the adversary will be stopped from eavesdropping. But this cannot stop curious LBS providers who are supposed to decrypt messages. If cryptographic protocols are used to protect a host's source IP, then equipment working at the data link layer will need to decrypt the traffic, causing widespread network latency because of resource-consuming decryption operations. Second, by using anonymous networks such as Tor (Dingledine et al., 2004) and Mix (Danezis et al., 2003), the service provider may not identify a user's source IP. But anonymous networks cannot guarantee QoS very well, and some relay nodes are not reliable or trustworthy. Third, techniques like VPN make a user's IP datagram appear to be sent from a remote network, which may be in a totally different domain compared to the source. But both anonymous networks and VPN are not widely deployed, especially for mobile communication. Some techniques (Ishikawa et al., 2011; Muir and Oorschot, 2009) have been proposed to locate the IP of a host while it is behind a NAT or proxy. As well, methods that mitigate the threat by reducing the accuracy of the user's location or not allowing the server to know the precise AP location have been well-studied, driven by other motivations (Ardagna et al., 2011; Gruteser and Grunwald, 2003). Such methods may be more feasible in comparison, but implementing them comes at the cost of location accuracy and service quality. We focus here on IP-based attacks, while attacks based on cookies or other Internet profiles (Felten and Schneider, 2000; Toubiana et al., 2012) are out of this paper's concern.

Prior work on developing algorithms to address coverage problems has mostly focused on ad-hoc or sensor networks. These algorithms (Wang et al., 2015; Yang et al., 2015; Yu et al., 2013) focus on single-layer coverage and energy costs in wireless sensor networks. K-cover algorithms (Abrams et al., 2004) have been proposed to utilize overlaps of coverage to achieve k-times coverage by sensor nodes. But this type of algorithm treats sensor nodes as providing the same role. All of these algorithms cannot be adapted to solve our challenge discussed here because we are faced with two different coverage layers which should be covered by different groups of nodes respectively. Moreover, existing APs in fixed positions induce complexity in our problem too high to be solved with prior algorithms.

In this paper, we use min-sum Belief Propagation (BP) to solve the problem. Gamarnik et al. give a min-sum BP algorithm on a tree-subject graph converging to an optimal solution of a min-cost network flow problem in pseudo-polynomial time (Gamarnik et al., 2010). Weiss and Freeman proved certain local optimality properties of the max-product BP for arbitrary graphs (Weiss and Freeman, 2006). Recently, Loopy Belief Propagation (LoopyBP), a direct application of BP to a graph model with cycles, has been widely applied to various areas. Although the convergence of LoopyBP is unsteady, it can return good results in many applications, and Ihler et al. provide a strong condition on convergence (Ihler et al., 2005). However, they provide no guarantee on the convergence of max-product BP for arbitrary graphs. Our challenge may correspond to a cyclic graph model, and we will discuss the convergence after introducing the algorithm.

3. Preliminary

We consider the APs in a public location, such as a public library or a station's waiting room. We assume that all APs work as wireless routers and are maintained by a trusted administrator. Without loss of generality, we regard a public region as a rectangular area of $a \times b$, but our solutions can be adapted to regions of any shape. We are interested in APs that are publicly shared. Portable APs or temporary individual hotspots for use will be ignored. Every AP in this region has its own public IP address on the Internet. This IP address can be static or dynamic. If it is dynamic, it may or may not be the same one as in the previous DHCP lease. While deploying an AP, we allow its position to have an accuracy range e_{pos} in orthogonal directions of the horizontal plane, so that we can assume every AP is at the center of a small square region $e_{pos} \times e_{pos}$. We assume an AP's coverage area is a disk of radius R. Each existing AP is located at fixed coordinates. Fig. 1 shows examples of this arrangement.

3.1. Threat model

When connected to the Internet, an AP is assigned with a public IP address $IP_{pub}(AP)$, and can provide Internet connectivity to its users by NAT. Any user u behind the NAT has a private IP address $IP_{priv}(u)$ assigned by the AP. We want to separate requests involving location from other requests, and because authenticated standard requests occur with almost the same frequency as unauthenticated requests (Vratonjic et al., 2014), we define two types of requests:

1. Req_{LBS}: originating LBS requests. The service provider will obtain the users location by embedded GPS or through users specification.
2. Req_{Std}: standard requests, which comprise all other requests with no LBS involved.

We consider the threat proposed in Vratonjic et al. (2014). The adversary is a service provider who can provide both LBS and account authentication, and is curious about its users' location information. Examples include providers such as Google (Google Map, Gmail, Google+), Microsoft (Bing, Outlook, Windows Phone) and Apple (map, iCloud, iPhone). An example of the threat is shown in Fig. 2. An AP connects to the Internet with $IP_{pub}(AP)$, and some users are behind NAT, including u_1 and u_2. Without loss of generality, we assume user u_1 has $IP_{priv}(u_1) =$

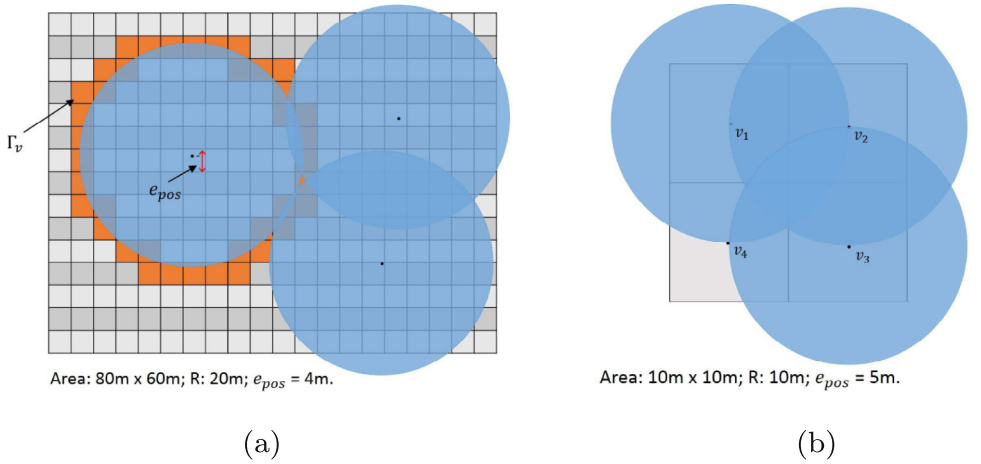

Fig. 1 – a) An example of an existing AP deployment with AP positioned within a small square region $e_{pos} \times e_{pos}$; b) An example of existing AP positions with greater e_{pos}.

(192.168.1.76) and sends a LBS request $Req_{LBS}(u_1)$ to a service provider $Server_{ad}$. $Server_{ad}$ receives u_1's location information loc from AP's address $IP_{pub}(AP)$, and then can create the mapping (loc, $IP_{pub}(AP)$). During the period when the AP's $IP_{pub}(AP)$ is kept consistent (i.e., in the same DHCP lease or assigned the same IP as the previous DHCP lease, which occurs with high probability), any user in this AP's private IP space ((192.168.1.76) & submask), such as u_2 who sends $Req_{Std}(u_2)$ to the server and is also controlled by $Server_{ad}$, will be located within an accuracy R according to the mapping (loc, $IP_{pub}(AP)$). It is assumed that the location privacy of a user not requesting LBS is already under the protection of an anonymizing technique or cryptographic tool. These users do not want to disclose their present location, but their location information is still exposed to others despite not using LBS. These users are the victims of interest in this study, while those who collude with the adversary by reporting the IP addresses of APs and their locations via some side channels are beyond this paper's discussion.

4. Bipartition algorithm

As we cannot easily modify existing infrastructure, our basic idea is to separate LBS users apart from non-LBS users. Thus, we divide deployed APs into two groups. G_{LBS} denotes the group for LBS requests, and G_{Std} denotes the other group for standard requests. Group G_{Std} will block improper LBS requests[3]. Users who use LBS connect to APs belonging to G_{LBS}. Users who only use standard requests use APs belonging to G_{Std}. When a user connecting to G_{Std} wants to originate any LBS request, he or she must manually switch to G_{LBS}. Users connecting to G_{LBS} have no need to switch to G_{Std} because G_{LBS}'s APs are fully functioning.

Recalling the example of the threat in Fig. 2, we divide APs so that Req_{Std} and Req_{LBS} are forwarded by two isolated groups of APs. Although $Server_{ad}$ can still build a mapping of LBS users, it cannot locate users that did not request LBS because their IP addresses are not included in the mapping.

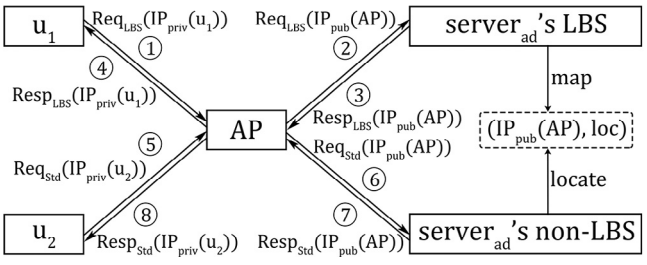

Fig. 2 – Instance of the threat.

4.1. Optimization problem

Let $\Pi = \{o_1, o_2, \ldots, o_m\}$ denote the set of m existing APs. Before introducing the BP algorithm, we clarify how to evaluate a partition. As the threat can be prevented completely, the net coverage of each group (excluding the overlapped and out-of-boundary areas) appears to be a good measurement. Note that we do not need to care about the area that lies outside of the region in bipartition algorithms, because that is determined by the fixed deployment. Thus, we aim to minimize the overlap while partitioning. The objective function for bipartition should be:

$$\min \sum_{o_i \in \Pi} overlap_{o_i}(x_i), \; subject\ to\ x_i > 0, \forall o_i \in \Pi, \tag{1}$$

where x_i is a variable within domain $\mathbb{X}$ indicating o_i's arrangement, and in order to explain the indication intuitively, we use binary to show its value:

[3] The blocking strategy can be implemented in different ways. To be low-cost, we choose a filtering method based on IP addresses and domain names which can be easily implemented by commercial APs' built-in firewall, filtering rules, or blacklist. More solutions will be discussed in Section 6.

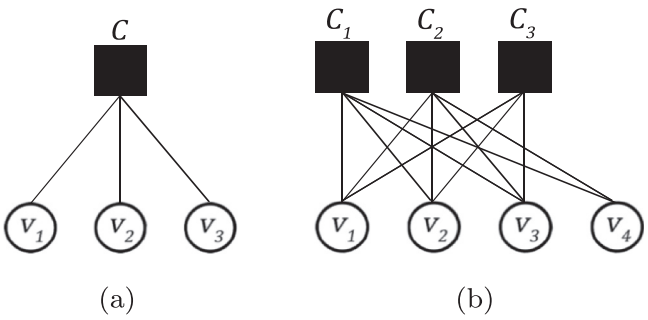

Fig. 3 – a) Factor graph of bipartition problem; b) Factor graph of complete covering problem.

$$x_i = \begin{cases} (01)_b, & o_i \in G_{Std}, \\ (10)_b, & o_i \in G_{LBS}, \\ (00)_b, & otherwise. \end{cases} \tag{2}$$

overlap is a basic operator, $overlap_{o_i}(x_i): \mathbb{R} \to \mathbb{R}$, returning the area in o_i's range but is overlapped by other APs' coverage based on x_i's value. There are many existing algorithms to solve this type of problem in both mathematics and computer science, such as Librino et al. (2009). This operator is computable and easy to implement, so we use it as a basic operator.

4.2. Factor graph

Before modeling our problem with a factor graph, let us define the objective function of our optimization problem in belief propagation style here:

$$\min \sum_{o_i \in \Pi} \Phi_{o_i}(x_i) + \sum_{\substack{o_i \in C, \\ C \in \mathcal{C}}} \Psi_C(x_i), \tag{3}$$

where $\Phi_{o_i}(x_i) = overlap_{o_i}(x_i)$ is a *variable function* calculating the overlapping area which is caused by o_i's arrangement. Ψ_C denotes a *factor function* corresponding to constraint C, which guarantees that all APs will be assigned into groups. $\mathcal{C}$ in this case there is only one constraint C in $\mathcal{C}$. The factor function of C is:

$$\Psi_C(x_i) = \begin{cases} 0, & if\ \forall o_i \in \Pi, x_i \geq 1, \\ \infty, & otherwise. \end{cases} \tag{4}$$

A factor graph (Kschischang et al., 2001) used for optimization problems is a bipartite graph with one partition V being the set of variable nodes, and the other partition $\mathcal{F}$ being the set of factor nodes corresponding to constraints. Moreover, there is an edge $(s, F) \in V \times \mathcal{F}$ if and only if $s \in F$. In our problem, every variable x_i represents a variable node, and constraint C represents a factor node. A factor graph example of our bipartition problem can be constructed as in Fig. 3a.

4.3. Message passing and BP algorithm

An important step of any BP algorithm is the encoding of the message function. As a kind of message-passing algorithm, the BP algorithm relies on exchanging messages between every two connected nodes to seek feasible solutions. In fact, the message contains the node's beliefs that are estimated about each possible state of each of its neighbors. There are two kinds of messages, message $m_{i \to C}$ from a variable node x_i to a factor node C, and $m_{C \to i}$ from a factor node C to a variable node x_i. Messages of our problem for any edge $(x_i, C) \in V \times \mathcal{F}$ are defined as:

$$m_{i \to C}(z) = \Phi_{o_i}(z) + \sum_{C' \in \mathcal{C} \setminus C} m_{C' \to i}(z), \tag{5}$$

$$m_{C \to i}(z) = \min_{\substack{y \in \mathbb{X}^{|C|} \\ y_i = z}} \Psi_C(y) + \sum_{o_j \in C \setminus \{o_i\}} m_{j \to C}(y_j), \tag{6}$$

where z is any possible value of x_i in $\mathbb{X}$. To implement the algorithm via dynamic programming, each time a specific value $z \in \mathbb{X}$ should be assigned to a variable o_i. Then the original optimization problem becomes:

$$\min \Phi_{o_i}(z) + \sum_{o_j \in \Pi \setminus \{o_i\}} \Phi_{o_j}(x_j) + \sum_{\substack{C \in \mathcal{C} \\ k \in C}} \Psi_C(o_k), \quad subject\ to\ x_i = z. \tag{7}$$

In this condition, let $b_{o_i}(z)$ denote the minimal cost of optimally assigning the rest of the problem variables with $x_i = z$. Then the optimal assignment of o_i can be found in $\arg\min_{z \in \mathbb{X}} b_{o_i}(z)$. By executing message passing until sufficient iterations have occurred, say N, the belief of each node o_i can be estimated by:

$$bel^N_{o_i}(z) = \Phi_{o_i}(z) + \sum_{C \in \mathcal{C}_{o_i}} m^N_{C \to i}(z). \tag{8}$$

For a tree-structured graph T as in Fig. 3a, dynamic programming of the optimal assignment of every variable can be decomposed into sub-problems on disconnected trees. Further, our optimization problem can be solved by updating messages recursively. As we defined two directional messages recursively, the update procedure for each edge $(x_i, C) \in V \times \mathcal{F}$ in the graph can be defined naturally in the same way:

$$m^{(t)}_{i \to C}(z) = \Phi_{o_i}(z) + \sum_{C' \in \mathcal{C} \setminus C} m^{(t-1)}_{C' \to i}(z), \tag{9}$$

$$m^{(t)}_{C \to i}(z) = \min_{\substack{y \in \mathbb{X}^{|C|} \\ y_i = z}} \Psi_C(y) + \sum_{o_j \in C \setminus \{o_i\}} m_{j \to C}(y_j)^{(t)}. \tag{10}$$

Our BP algorithm is shown in Algorithm 1. The optimal assignment of o_i can be obtained after recursion time t which should be greater than diameter D of tree T. Every node should be waiting until all the necessary messages have arrived. The initiation of message passing can be arbitrary, which means that in each iteration, the message sequence does not matter.

4.4. Weighted bipartition algorithm with submodular function

According to Vratonjic et al. (2014), standard and LBS requests are generated at different frequencies. In the authors' dataset, nearly 10% of users generated LBS requests overall, but this fraction is likely too complex for the BP algorithm to

Algorithm 1: BP algorithm for Bipartition

for any edge $(x_i, C) \in V \times \mathcal{F}$, initial $m^0_{i\to C}(z) = m^0_{C\to i}(z) = 0$, assign $N \geq D'$. **foreach** $t = 1, 2, \ldots, N$ **do**

  foreach *edge* (x_i, C) *and* $z \in \mathbb{X}$ **do**

$$m^{(t)}_{i\to C}(z) = \Phi_{o_i}(z) + \sum_{C' \in \mathcal{C}\setminus C} m^{(t-1)}_{C'\to i}(z).$$

$$m^{(t)}_{C\to i}(z) = \min_{\substack{\mathbf{y}\in\mathbb{X}^{|C|} \\ y_i = z}} \Psi_C(\mathbf{y}) + \sum_{o_j \in C\setminus\{o_i\}} m_{o_j\to C}(y_j)^{(t)}.$$

  $t = t + 1$

foreach $i = 1, 2, \ldots, m$ **do**

  calculate the belief function:

$$bel^N_{o_i}(z) = \Phi_{o_i}(z) + \sum_{C\in\mathcal{C}} m^N_{C\to i}(z),\ \forall o_i \in \Pi.$$

foreach $o_i \in \Pi$ **do**

  estimate o_i's optimal assignment $\arg\min bel^N_{o_i}(z)$.

consider. On the other hand, although the performance of the BP algorithm is typically good, large-scale computations will be resource-intensive. Therefore, we provide a difference bipartition algorithm which makes use of submodular functions that trade accuracy for efficiency and flexibility. This algorithm's core idea is to partition all existing APs into G_{Std} and G_{LBS} to receive nearly maximum utility, assuming that all APs are in the grid. We define a set function for the area, $\forall e \in \Pi$:

$$f(S) = \sum_{e\in S} cover(e) - overlap(S), \tag{11}$$

where $f(S)$ is a set function: $2^\Pi \to \mathbb{R}$, for any $S \subseteq 2^\Pi$, *cover* is an operator that gives e's coverage within the boundary, while *overlap* returns the sum of every overlapping area multiplied by the number of redundantly-covered areas. We assign a weight to the coverage of G_{Std}, denoted by w_{Std}. Meanwhile $w_{LBS} = \alpha w_{Std}$, $0 < \alpha < 1$. We assume users appear in the region uniformly. The deployment of existing APs cannot be modified, which means we can use the sum of the weighted coverage of two groups to evaluate one specific partition. Then, our goal is to maximize the expression:

$$\max_{S\subseteq 2^V} w_{Std} \times f(S) + w_{LBS} \times f(V - S). \tag{12}$$

In order to maximize the set function $f(S)$, we introduce the concepts of a discrete derivative and a submodular function, using the definition in Krause and Golovin (2012).

Definition 1. (discrete derivative). For a set function $f: 2^V \to \mathbb{R}$, $S \subseteq V$, and $e \in V$, let $\Delta_f(e\|S) f(S \cap e) - f(S)$ be the discrete derivative of f at S with respect to e.

Definition 2. (submodular function). A set function $f: 2^V \to \mathbb{R}$ is submodular if for every $A \subseteq B \subseteq V$ and $e \in V\setminus B$, $\Delta(e\|A) \geq \Delta(e\|B)$ holds.

According to the definition of a submodular function, our set function $f(S)$ is submodular and monotone. The maximization of many classes of submodular functions is NP-hard (Cornuejols et al., 1977; Nemhauser et al., 1978) when subject to constraints on S. However, an approximate solution can be obtained by various greedy algorithms. According to Nemhauser, an approximation to the optimal solution obtained by a greedy algorithm is at least $1 - 1/e$ (Nemhauser et al., 1978). We define Δ_{Std} and Δ_{LBS} to construct our constraint on set S:

$$\Delta_{Std}(e, S) = w_{Std} \times \Delta_f(e|S),\ \Delta_{LBS}(e, S) = w_{LBS} \times \Delta_f(e|S). \tag{13}$$

The detailed submodular algorithm is shown in Algorithm 2. This algorithm consists of two main steps, the first that finds all elements satisfying the constraint in the complement of G_{Std}, and then the second that selects one element with the greatest gain from the standard group and assigns it to G_{Std}. When no element can generate greater gain in G_{Std} than in G_{LBS}, we stop searching and assign the rest of the APs to G_{LBS}.

5. Complete covering algorithm

Let $P = \{v_1, v_2, \ldots, v_n\}$ denote the set of all possible positions where an AP can be installed. Every element in P has an error e_{pos} in directions orthogonal to the horizontal plane, as shown in Fig. 1a. Any newly added AP can be deployed in any position, and can have a distance difference of up to e_{pos}. Thus, any $v \in P$ may belong to G_{Std}, G_{LBS}, or G_{Std} and G_{LBS} together. In order to use fewer APs to cover every inch of an exposed area, it is equivalent to ensure that the overlapping area and area outside the boundary are minimal. Subject to the constraint collection $\mathcal{C}$, our objective function is:

$$\min \sum_{v_i\in P} overlap_{v_i}(x_i) + bound_{v_i}(x_i). \tag{14}$$

Variable x_i within domain $\mathbb{X}$ is an indicator of v_i. Little different from o_i's x_i, v_i's x_i is defined as:

Algorithm 2: Submodular Algorithm for Bipartition

initial $G_{Std} \leftarrow \varnothing$, $G_{LBS} \leftarrow \varnothing$, $S' \leftarrow S$, $C \leftarrow \varnothing$. **while** S' *is not empty* **do**
  foreach *element* $e \in S'$ **do**
    if $\Delta_{Std}(e, G_{Std}) \geq \Delta_{LBS}(e, G_{LBS}$ **then**
      $C \leftarrow C + e$
  $maxGain \leftarrow 0$, $maxE \leftarrow random\ e \in S'$
  if C *is empty* **then**
    foreach *element* $e \in S'$ **do**
      if $\Delta_{LBS}(e, G_{LBS}) \geq maxGain$ **then**
        $maxGain \leftarrow \Delta_{LBS}(e, G_{LBS})$
        $maxE \leftarrow e$
    $G_{LBS} \leftarrow G_{LBS} + maxE$
  else
    foreach *element* $e \in C$ **do**
      if $\Delta_{Std}(e, G_{Std}) \geq maxGain$ **then**
        $maxGain \leftarrow \Delta_{Std}(e, G_{Std})$
        $maxE \leftarrow e$
    $G_{Std} \leftarrow G_{Std} + maxE$
  $S' \leftarrow S' - maxE$

$$x_i = \begin{cases} (01)_b, & v_i \in G_{Std}, \\ (10)_b, & v_i \in G_{LBS}, \\ (11)_b, & v_i \in G_{Std} \cap G_{LBS}, \\ (00)_b, & otherwise. \end{cases} \tag{15}$$

$bound: \mathbb{R} \to \mathbb{R}$ is another basic operator. *bound* returns the area which is in v_i's range but outside the boundary of the region based on x_i's value. The *variable function* Φ_{v_i} calculates both the overlapping and out-of-boundary areas caused by v_i's assignment:

$$\Phi_{v_i}(x_i) = overlap_{v_i}(x_i) + bound_{v_i}(x_i). \tag{16}$$

Thus, the optimization problem in belief propagation should be:

$$\min \sum_{v_i \in P} \Phi_{v_i}(x_i) + \sum_{\substack{C \in \mathcal{C} \\ v_i \in C}} \Psi_C(x_i, \Gamma_{v_i}), \tag{17}$$

where Ψ_C denotes the *factor function* corresponding to constraint C belonging to the constraint collection $\mathcal{C}$. Γ_{v_i} denotes the neighbor positions that object to the boundary of v_i's disk (as shown in Fig. 1a)), and v_i belongs to C if v_i is in Γ_{v_j} ($\forall v_j \in P \setminus \{v_i\}$) or Π. The constraints in $\mathcal{C}$ are:

$$\Psi_{C_1} = \begin{cases} 0, & \begin{array}{l} if\ v_j\ is\ covered\ by \\ G_{Std}, \forall v_j \in \Gamma_{v_i}, \end{array} \quad , \quad \forall v_i \in G_{Std}, \\ \infty, & otherwise. \end{cases} \tag{18}$$

$$\Psi_{C_2} = \begin{cases} 0, & \begin{array}{l} if\ v_j\ is\ covered\ by \\ G_{LBS}, \forall v_j \in \Gamma_{v_i}, \end{array} \quad , \quad \forall v_i \in G_{LBS}, \\ \infty, & otherwise. \end{cases} \tag{19}$$

$$\Psi_{C_3} = \begin{cases} 0, & if\ \forall v_i \in \Pi, x_i \geq 1, \\ \infty, & otherwise. \end{cases} \tag{20}$$

Each Ψ_{C_i} function is $\mathbb{R}^{|C_i|} \to \bar{\mathbb{R}}$, $\forall C_i \in \mathcal{C}$. Constraints C_1 and C_2 guarantee that the region is completely covered by both G_{Std} and G_{LBS}. C_3 is required for existing APs. Still, x_i of v_i represents a variable node in V, and every constraint in $\mathcal{C}$ represents a factor node in $\mathcal{F}$. Finally, there is an edge $(v_i, C) \in V \times \mathcal{F}$ if and only if $v_i \in C$. A factor graph example constructed for the complete covering problem is shown in Fig. 3b.

Two directional messages for any edge $(v_i, C) \in V \times \mathcal{F}$ are defined as:

$$m_{i \to C}(z) = \Phi_{v_i}(z) + \sum_{C' \in \mathcal{C}_{v_i} \setminus C} m_{C' \to i}(z), \tag{21}$$

$$m_{C \to i}(z) = \min_{\substack{y \in \mathbb{X}^{|C|} \\ y_i = z}} \Psi_C(y) + \sum_{v_j \in C \setminus \{v_i\}} m_{j \to C}(y_j). \tag{22}$$

Then the original optimization problem becomes:

$$\begin{aligned} &\min \Phi_{v_i}(z) + \sum_{v_j \in P \setminus \{v_i\}} \Phi_j(x_j) + \sum_{\substack{C \in \mathcal{C} \\ k \in C}} \Psi_C(v_k), \\ &subject\ to\ v_i = z. \end{aligned} \tag{23}$$

The estimation of belief and message update procedure are the same as that of the BP algorithm.

5.1. *LoopyBP algorithm*

With our constraints, the factor graph contains cycles (as shown in Fig. 3b). To address issues caused by this kind of cyclic graph, LoopyBP is convenient to use because it can solve problems

Algorithm 3: LoopyBP for Complete Covering algorithm

for any edge $(v_i, C) \in V \times \mathcal{F}$, initial $m^0_{v_i \to C}(z) = m^0_{C \to i}(z) = 0$, assign $N \geq D'$. **foreach** $t = 1, 2, \ldots, N$ **do**

  foreach *edge* (v_i, C) *and* $z \in \mathbb{X}$ **do**

$$m^{(t)}_{i \to C}(z) = \Phi_{v_i}(z) + \sum_{C' \in \mathcal{C}_{v_i} \setminus C} m^{(t-1)}_{C' \to v_i}(z).$$

$$m^{(t)}_{C \to i}(z) = \min_{\substack{\mathbf{y} \in \mathbb{X}^{|C|} \\ y_i = z}} \Psi_C(\mathbf{y}) + \sum_{v_j \in C \setminus v_i} m_{j \to C}(y_j)^{(t)}.$$

  $t = t + 1$

  foreach $i = 1, 2, \ldots, n$ **do**

    calculate the belief function:

$$bel^N_{v_i}(z) = \Phi_{v_i}(z) + \sum_{C \in \mathcal{C}_{v_i}} m^N_{C \to i}(z), \forall v_i \in P.$$

  foreach $v_i \in P$ **do**

    estimate v_i's optimal assignment $\arg\min bel^N_{v_i}(z)$.

in the same way as BP regardless of the existence of cycles. Every node can perform its calculation as long as all required messages have been received. In LoopyBP, we assume that the recursion time t should be greater than the diameter D of tree T, which is the essential condition for enabling convergence. As it is not particularly efficient to find the exact value of an arbitrary graph's diameter, we use an upper bound D' of the actual graph diameter in place of D. D' can be found by using a depth-first search (DFS) to find the spanning tree of the graph, and then use breadth-first search (BFS) to determine the its diameter. This diameter will then be the upper bound of the original graph's diameter (Magnien et al., 2009).

Our LoopyBP algorithm is shown in Algorithm 3. Note that if nodes in a cycle cannot backtrack immediately, they should pass messages in the manner shown in Fig. 4. Although the LoopyBP algorithm is easy to use, it is difficult to analyze its convergence, so below we focus attention on the convergence of the algorithms presented here.

5.2. *Convergence of complete covering algorithm*

It is convenient to analyze the LoopyBP algorithm in terms of its computation tree, which differs little from the computation tree of an acyclic graph. We define a computation tree $T^N_{x_0}$, which is associated with any variable node x_0 and has N levels. Note that $T^N_{x_0}$ can be defined inductively corresponding to iterative message passing. $T^0_{x_0} = (V(T^0_{x_0}), E(T^0_{x_0}))$ is the initial tree with only one vertex x'_0 and an empty edge set. x'_0 is the replica of x_0 mapped by $\Gamma^0_{x_0}(x'_0) = x_0$. Suppose $T^1_{x_0} = (V(T^1_{x_0}), E(T^1_{x_0}))$ has vertex set $V(T^1_{x_0}) = \{x'_i | x_i \in C_j \text{ and } x_0 \in C_j, \forall C_j \in \mathcal{C}, 0 \leq i \leq n\}$, $E(T^1_{x_0}) = \{(x'_0, x'_i) | \forall x_i \in V(T^1_{x_0}), x'_i \neq x'_0\}$, x'_i and (x'_0, x'_i) are replicas of $x_i \in X$ and (x_0, x_i) mapped by $\Gamma^1_{x_0}(x^{(c)}_i) = x_i$, $\Gamma^1_{x_0}(x'_i) = \Gamma^1_{x_0}(x''_i) = \Gamma^1_{x_0}(x^{(c)}_i) = x_i$, for any c-th time x_i appears in a different path. We denote the set of vertices in the path from x'_i to the root (not included) as $B_{x'_i}$, and denote $L(T^i_{x_0})$ as the set of leaf nodes $T^i_{x_0}$. Then, we identify that $T^N_{x_0} = (V(T^N_{x_0}), E(T^N_{x_0}))$ is an N-level tree containing $T^{N-1}_{x_0}$ as a sub-tree. Therefore, $V(T^N_{x_0}) = V(T^{N-1}_{x_0}) \cup \{x'_s | x_s \in C_j \text{ and } x_t \in C_j, \forall C_j \in \mathcal{C}, \forall x_t \in V(T^{N-1}_{x_0}), x_s \notin \Gamma(B_{x_i}), \forall x_i \in V(T^{N-1}_{x_0})\}$ and $E(T^N_{x_0}) = E(T^{N-1}_{x_0}) \cup \{(x'_s, x'_t) | \forall x_t \in V(T^{N-1}_{x_0}), x'_i \neq x'_0\}$.

By constructing a computation tree in this way, we prevent the node from immediately backtracking a cycle while passing messages, and thus, confusion of old and new messages can be avoided. As the root choice does not make a significant difference, we choose it randomly. An example of a computation tree is shown in Fig. 4.

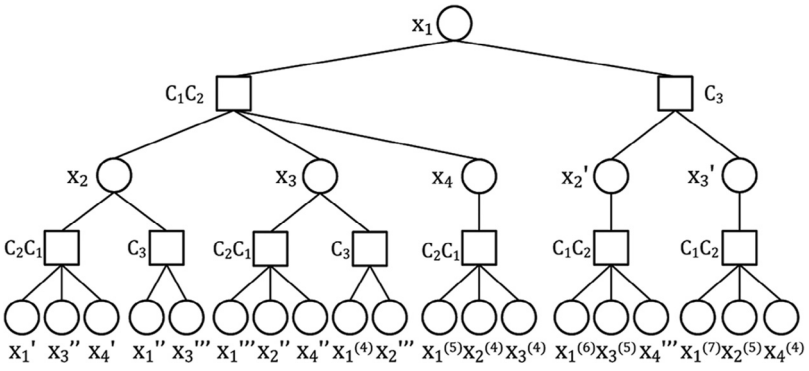

Fig. 4 – Computation tree of the factor graph shown in Fig. 3b.

It is clear that our problem has feasible solutions. Even in the worst case, assigning $(11)_b$ to every x_i of v_i in P, we will obtain a finite value instead of ∞. We analyze the convergence of our algorithm in the following steps. First, let us assume that there may be multiple fixed points in our problem. Then we measure the distance between fixed points with the measurement proposed in Ihler et al. (2005). If the distances between any pairwise fixed points are zero, it means that our assumption of the multiple fixed points is wrong, and the algorithm will converge to a unique fixed point. Otherwise, if there exist any non-zero distances, we provide a sufficiently-small distance bound. Finally, even if fixed-point convergence is not achieved, through iteration the algorithm can asymptotically approach the bounding ball with a diameter of the distance bound. We review fixed-point distances (Ihler et al., 2005) below:

Theorem 1. (Fixed-point distance bound). *Let* $\{M_t\}$ *and* $\{\tilde{M}_t\}$ *be the beliefs of any fixed points of LoopyBP. Then for any node* t *and for all* x, $\left|\log M_t(x)/\tilde{M}_t^n\right| \le 2\sum_{u\in\Gamma_t}\log\frac{d(\Psi_{ut})^2\epsilon+1}{d(\Psi_{ut})^2+\epsilon}$, *where* ε *is the largest value satisfying:* $\log\epsilon = \max_{(x,t)\in E}\sum_{u\in\Gamma_t\setminus s}\log\frac{d(\Psi_{ut})^2\epsilon+1}{d(\Psi_{ut})^2+\epsilon}$, $d(\Psi_{ut})^2 = \sup_{a,b,c,d}\frac{\Psi_{ut}(a,b)}{\Psi_{ut}(c,d)}$.

The error measurement introduced in Ihler et al. (2005) is multiplicative in message deviations and additive in the log domain. We define the approximate message in our problem in an exponential form so that we can convert the additive message deviations into multiplicative message deviations. Let the actual message $m_{ts}(x_s)$ be $e^{m_{C\to s}(v_s)}$, and the error function $e_{ts}(x_s)$ be $e^{e_{C\to s}(v_s)}$. Then the approximate message $\hat{m}_{ts}(x_s) = m_{ts}(x_s)e_{ts}(x_s) = e^{m_{C\to s}(v_s)+e_{C\to s}(v_s)}$. Thus, the dynamic measure will be re-defined as $d(e_{ts}) = \sup_{a,b} e^{(e_{C\to s}(a)-e_{C\to s}(b))/2}$. Because of the continuity and monotonicity of the exponential function, newly defined messages can be mapped perfectly to our original definition. In order to show that this definition is equivalent to the definition in Ihler et al. (2005), here we note one key theorem for our proof.

Theorem 2. (Additivity). *The log of a dynamic range measure is sub-additive:*

$$\log d\left(\hat{M}_{ts}^i/M_{ts}^i\right) \le \sum_{u\in\Gamma_t\setminus s}\log d\left(e_{ut}^i\right), \tag{24}$$

$$\log d\left(\hat{M}_t^i/M_t^i\right) \le \sum_{u\in\Gamma_t}\log d\left(e_{ut}^i\right). \tag{25}$$

Proof.

$$\begin{aligned}\log d\left(\hat{M}_{ts}^i/M_{ts}^i\right) &= \frac{1}{2}\log\sup_{a,b}\prod e_{ut}^i(a)\Big/\prod e_{ut}^i(b)\\ &= \frac{1}{2}\log\sup_{a,b} e^{\sum e_{u\to C}^i(v_u)}\Big/e^{\sum e_{u\to C}^i(v_b)}\\ &\le \log\sup_{a_u,b_u} e^{\frac{1}{2}\left(\sum e_{u\to C}^i(v_u)-e_{u\to C}^i(v_b)\right)}\\ &= \log\prod d\left(e_{ut}^i(x)\right) = \sum\log d\left(e_{ut}^i(x)\right).\end{aligned}$$

Other inequalities can be proved similarly.

The properties proposed in Ihler et al. (2005) can be verified easily with this exponential form. Because the verification is trivial, we do not go into details here. Similar to this new definition of a dynamic measure of a message, its extension can be defined as:

$$\begin{aligned}&d(\Psi_{ut})^2 = \sup_{v_a,v_b,v_c,v_d} e^{\Psi_{C\to s}(v_a,v_b)-\Psi_{C\to s}(v_c,v_d)},\\ &\forall v_b\in\Gamma_{v_a},\ \forall v_d\in\Gamma_{v_c},\ \forall C\in\mathcal{C}.\end{aligned} \tag{26}$$

At each iteration step, every C in $\mathcal{C}$ will determine a minimal belief $b_i^n(z)$ to v_i, and none of the variable nodes will receive ∞ as the incoming message. The distance bound of our algorithm can be calculated:

$$\begin{aligned}d(\Psi_{ut})^2 &= \sup_{v_a,v_b,v_c,v_d} e^{\Psi_{C\to s}(v_a,v_b)-\Psi_{C\to s}(v_c,v_d)} = 1.\\ \left|\log M_t(x)/\hat{M}_t^n(x)\right| &\le 2\log d\left(M_t/\hat{M}_t\right)\\ &\le 2\sum_{u\in\Gamma_t}\log\frac{d(\Psi_{ut})^2\epsilon+1}{d(\Psi_{ut})^2+\epsilon}\\ &= 2\sum_{u\in\Gamma_t}\log 1 = 0.\end{aligned}$$

Hence all fixed points to which our algorithm can converge are within zero distance. In other words, our complete covering algorithm obtains stable values for each variable after sufficient iterations.

6. Discussion

We have proposed two kinds of algorithms for bipartite APs, which can be used subject to how a network administrator may decide to deploy APs. This section discusses how users might access the Internet normally when bipartition has been carried out. There are two situations that a user may be faced with. One is that the administrator runs a Bipartition Algorithm and users may be in an area not covered by either G_{LBS} or G_{Std}, and the other is that users may find AP signals from both G_{LBS} and G_{Std}. The first situation may occur because of an insufficient number of existing APs. This can be fixed by our complete covering algorithm. In the second situation, users can use of any type of service. If both two groups can cover the region completely after bipartition, then the following settings should be adopted. First, Wi-Fi signs in public places should be accompanied by instructions explaining the two groups' SSIDs, clearly describing each group's use. Second, a privacy policy and instructions regarding AP use should be displayed on users' screens when they access the AP's welcome or login page, so that users can be aware of how to use it. Third, LBS requests should be blocked on G_{Std}'s APs. We suggest using an APs built-in firewall to implement filtering rules against IP addresses and domain names (or subdomains) for efficiency. If more fine-grained filtering is needed, we suggest applying deep packet inspection (De Carli et al., 2014; Sherry et al., 2015) aided by an auxiliary server or a third-party service provider. With deep packet inspection, fields like coordinates or longitude and latitude can be captured precisely if they appear in packet contents. APs' built-in blacklists or filtering rules may block some non-LBS services if these service hosts share the same IP or domain name with any LBS server.

Aided by regular expression matching during deep packet inspection (Xu et al., 2016), LBS servers filter more precisely because they can check not only IP addresses and domain fields but also key fields in packet contents.

The assumption of users' technical ability is dispensable. However, we should note that a collusion attack is not in the scope of the protection described here. To launch any collusion attack, the attacker must recruit a malicious user to track the target victim's geographical position. The malicious user can use encrypted messages to reveal the target's location or help the attacker to correlate the IP addresses of APs from different groups. The cost of this collusion attack is very high. This attack is out of our consideration because it is too difficult to ensure location privacy once the target has been identified physically.

7. Evaluation

In this section, we conduct experiments to evaluate the performance of our bipartition and complete covering algorithms. By design, privacy leakage is prevented after bipartition, so using the AP coverage ratio as a metric, we mainly investigate the quality of the partitions our bipartition algorithms produce. As the coverage ratio of the complete covering algorithm is always one, we focus instead on its convergence and iteration times. Each factor influencing the complete covering algorithm is evaluated and discussed in terms of its effects. A 50-m 50-m region is used by default throughout our study of real-world scenes. We vary R from 10 m to 30 m to investigate the influence of radius. In order to apply our work to general cases, we deploy all existing APs randomly and uniformly in the region for simulation. To demonstrate their practicality, we apply our bipartition and complete covering algorithms to two real-life scenarios.

7.1. Bipartition algorithms

We evaluate coverage and efficiency for both the BP and submodular (SM) algorithms. As shown in Fig. 5a, the number of APs is an important parameter for bipartition. Both algorithms' coverage area increases as the number of Aps increases. It is intuitive that more APs provide more coverage, but when the number is sufficiently large, the coverage increases slowly, because it becomes difficult to cover the missing area using randomly-deployed APs. Our BP algorithm provides better partitioning results than SM, but its running time is higher at larger scales, as seen in Fig. 5b. Generally, we consider offline situations, so this overhead is acceptable. For possible online situations, the use of the SM algorithm is recommended.

An additional variable that should be considered for inclusion in the SM algorithm is α (i.e., w_{Std} and w_{LBS}), as shown in Fig. 5c. It shows that α is a key parameter for submodular algorithm. With an increase of G_{LBS}'s weight, the partition significantly improves. When the weight is sufficiently large, the result approximates stable coverage because of the number of APs. Further, the value of R did not impact the results because its value primarily influences the bipartition's overhead.

7.2. Complete covering algorithm

Each x_i's initial value is $(11)_b$, which is the worst case. After sufficient iterations, the objective function (17) reaches a maximum value among all feasible solutions. As shown in Fig. 5d, for each assignment of R, the objective function decreased sharply in the first hundred iterations, and then became stable after sufficient additional iterations have occurred. Three objective functions with $R = 10$, 20, 30 are clearly convergent. To determine R's effect on necessary iteration times, we evaluate the iteration times in the same setting, but with R ranging from 10 to 30 in steps of 1. The result is shown in Fig. 5e. Mostly, the objective functions stabilize after 2500 iterations, but R's influence is not negligible. The fluctuation may be caused by several random factors in the algorithm, such as the random deployment of existing APs and the choice of the root for the computation tree. If we can remove these influences from the experiment, we would find a weak, nearly-linear connection between iteration time and AP radius.

In Fig. 5f, the values of the objective function differ greatly with different position errors, and the radius R and position error e_{pos} determine the granularity of the covering. Fig. 5f shows that if the positions of the AP are made more accurate, the objective function will perform better. However, a smaller e_{pos} will cause a larger computation overhead and longer iteration time. After considering this trade-off and practical needs, we suggest e_{pos} should be assigned with value 1.

7.3. Practical applications

To be more practical, we choose two real life scenarios to verify the correctness and feasibility of our algorithms. The first one is a rectangular floor plan with AP deployment as in Yang et al. (2012). The second scenario is an irregular polygonal office environment used in Zhao et al. (2014). Both floor plans have been used in real settings. We redraw the two floor plans using their original length-width ratio in Fig. 6. As there are many walls in an indoor office environment, we assume that the effective radius of every AP is 10 m. In the first floor plan, 14 APs are deployed. In Fig. 6c, we show the result of AP partitioning after the weighted bipartition algorithm was applied, with the coverage by G_{Std} denoted by blue circles and the coverage by G_{LBS} denoted by red circles. If we assign weights for G_{Std} and G_{LBS} as $w_{Std} = 0.6, w_{LBS} = 0.4$ respectively, the net coverage of the normally-functioning APs for the whole floor will be slightly reduced when compared to original AP deployment. The original APs cannot completely cover the area twice. As indicated by our evaluation, we set $e_{pos} = 1$ here. To achieve complete coverage for both groups, our LoopyBP algorithm indicates that the minimum number of additional APs should be 9. The complete coverage result for Fig. 6a is shown in Fig. 6e.

The second scenario is an irregular polygonal area with a sparse AP deployment. In this situation, we share a principle similar to Yu et al. (2013), and propose to use auxiliary areas and a divide-and-conquer approach. Specifically, we use auxiliary areas which are so small that they can be ignored when compared to the entire area, which allows us to reshape the irregular polygons. Then, we divide the whole area into mul-

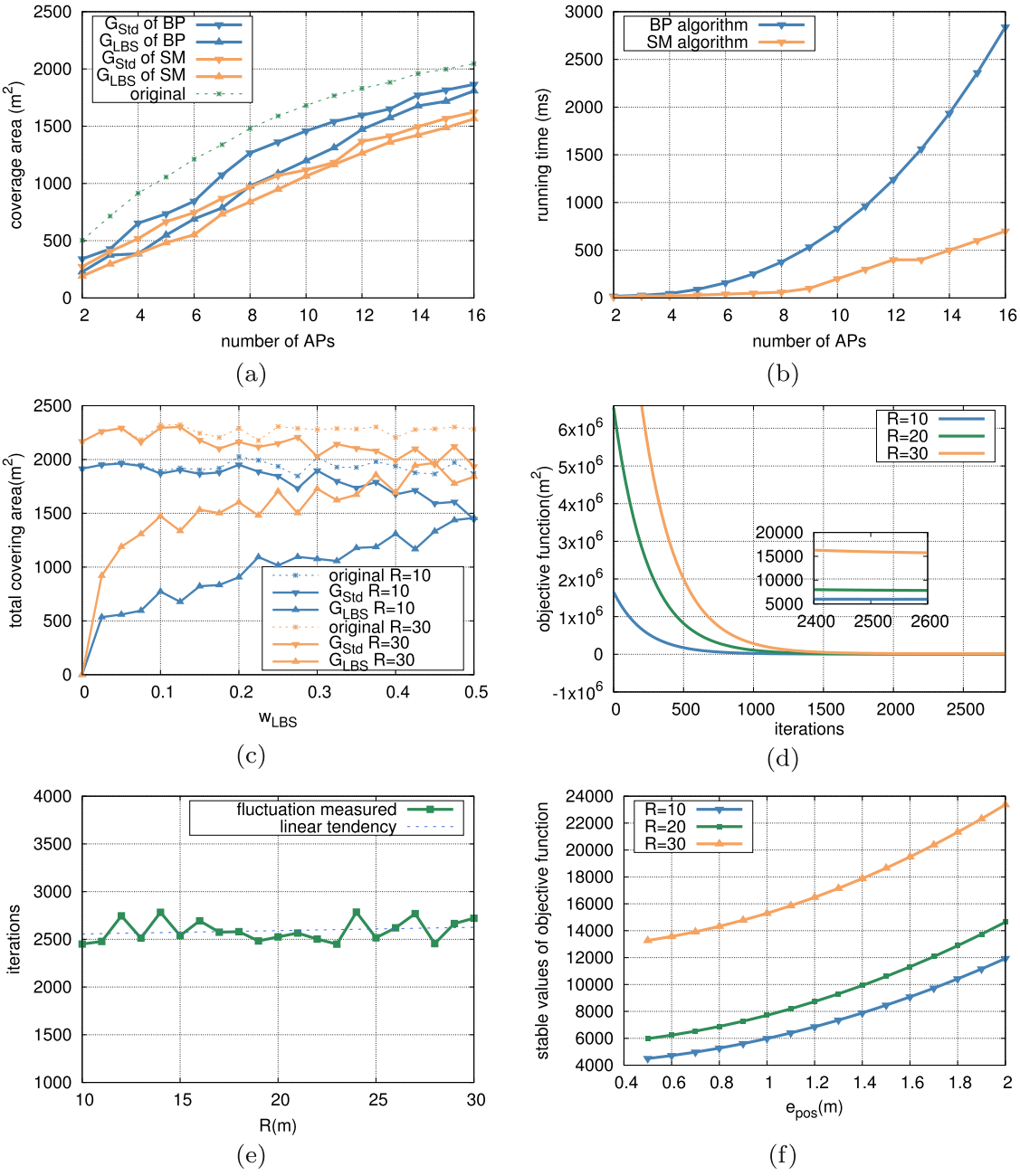

Fig. 5 – a) Coverage comparing the BP and SM algorithms, $R = 10$, $w_{Std} = w_{LBS}$; b) Running time results of two bipartition algorithms, $R = 10$, $w_{Std} = w_{LBS}$; c) Bipartition results with different weights for the submodular algorithm; d) Stable values of the objective function with different R; e) R's influence on iteration times; f) The tendency of e_{pos}'s influence on stable values of the objective function.

tiple rectangular areas to perform our algorithms. In practical usage, we can use some auxiliary areas to build larger rectangles instead of dividing them into several smaller rectangles. Note that auxiliary areas should be negligibly small relative to the total area. As shown in Fig. 6b, we reshaped the irregular polygonal area into three rectangular areas, after which we are able to apply our algorithms on each individual rectangular area. For each area, we considered all APs for coverage whether they are outside the boundary or not. When we partition APs using the same ratio of G_{Std} to G_{LBS} as in the first scenario, only two APs were assigned into G_{LBS} as shown in Fig. 6d. The original AP deployment was too sparse to create large overlaps. As the original 11 APs could not cover the entire area, additional APs were needed when the complete covering algorithm was used. As shown in Fig. 6f, when the area was completely covered by the two groups and $e_{pos} = 1$, 21 APs were added in total. In particular, six original APs and 10 newly-added APs composed G_{Std} to cover the whole area. Meanwhile, five original APs and 11 additional APs ensured the area could be completely covered by G_{LBS}. Even complete coverage was required from one group,

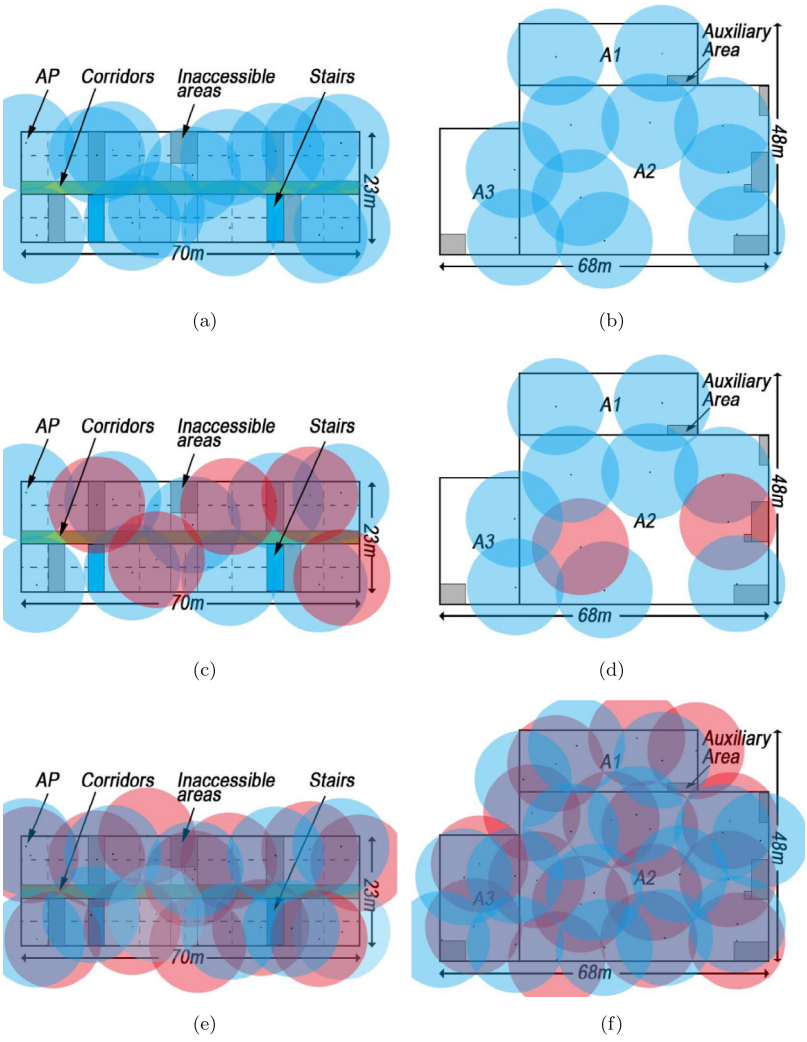

Fig. 6 – a) Original AP deployment of the rectangular area of Yang et al. (2012); b) Original AP deployment of the irregular polygonal area from Zhao et al. (2014); c) Submodular partition strategy for Fig. 6a, $w_{Std} = 0.6$; d) Submodular partition strategy for Fig. 6b, $w_{Std} = 0.6$; e) Complete covering strategy for Fig. 6a, with 9 APs added; f) Complete covering strategy for Fig. 6b, with 21 APs added in total. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

at least 5 APs were needed because of the sparse deployment and the irregularly-shaped polygonal space.

8. Conclusion

Location privacy threats related to public APs pose a significant potential threat to user privacy. As existing methods are not effective or feasible, this study takes the first feasible step in attempting to protect non-LBS users from the unintended exposure of their location. We proposed two bipartition algorithms. One BP algorithm yields an optimal partition result and can be extended into a complete covering algorithm, but has the disadvantage of high complexity. We also provide an approximation algorithm using submodular functions to deal with flexible and online situations. In the complete covering algorithm, we add a minimal number of APs to completely cover the region with the help of LoopyBP. To address common concerns about the convergence of LoopyBP algorithms, we show its convergence using a theoretical proof and experiment results.

Acknowledgement

The authors would like to thank the editors and reviewers for their valuable time and effort in reviewing this paper. Their insightful suggestions have helped in improving the quality of this paper significantly. Yuan Zhang was supported in part by NSFC-61402223. Xiaoyan Zhang was supported by NSFC-11471003 and Qing Lan Project. Fengyuan Xu was partially supported by CCF-NSFOCUS "Kunpeng" Research Fund, Alipay Research Fund and MSRA Visiting Young Faculty Program. This work was supported in part by NSFC-61425024, NSFC-61300235, the Jiangsu Province Double Innovation Talent Program, and NSFC-61321491.

REFERENCES

Abrams Z, Goel A, Plotkin S. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In: Proceedings of the 3rd international symposium on information processing in sensor networks. 2004. p. 424–32.

Ardagna C, Cremonini M, De Capitani di Vimercati S, Samarati P. An obfuscation based approach for protecting location privacy. In: Dependable and secure computing, IEEE transactions on. 2011. p. 13–27.

Casado M. Peering through the shroud: the effect of edge opacity on IP-based client identification. In: USENIX. 2007.

Cisco visual networking index: Global mobile data traffic forecast update, 20152020 white paper (2 2016). Available from: https://www.cisco.com/c/dam/m/en_in/innovation/enterprise/assets/mobile-white-paper-c11-520862.pdf.

Cornuejols G, Fisher ML, Nemhauser GL. Exceptional paper location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. Manage Sci 1977;789–810.

Covergage wifi – combain positioning service. Available from: https://combain.com/coverage-wifi-positioning-wifi-location/. [Accessed January 2016].

Danezis G, Dingledine R, Mathewson N. Mixminion: design of a type III anonymous remailer protocol. In: Symposium on security and privacy. 2003. p. 2–15.

De Carli L, Sommer R, Jha S. Beyond pattern matching: a concurrency model for stateful deep packet inspection. In: CCS '14. 2014. p. 1378–90.

Dingledine R, Mathewson N, Syverson P. Tor: The second-generation onion router. In: USENIX security. 2004. p. 21.

Felten EW, Schneider MA. Timing attacks on web privacy. In: CCS'00. 2000. p. 25–32.

Gamarnik D, Shah D, Wei Y. Belief propagation for mincost network flow: convergence & correctness. In: SODA '10. 2010. p. 279–92.

Gruteser M, Grunwald D. Anonymous usage of location based services through spatial and temporal cloaking. In: MobiSys'03. 2003. p. 31–42.

Ihler AT, Fischer JW III, Willsky AS. Loopy belief propagation: convergence and effects of message errors. J Mach Learn Res 2005;905–36.

Ishikawa Y, Yamai N, Okayama K, Nakamura M. An identification method of PCs behind NAT router with proxy authentication on HTTP communication. In: SAINT. 2011. p. 445–50.

Krause A, Golovin D. Submodular function maximization. In: Tractability: practical approaches to hard problems. 2012. p. 19.

Kschischang F, Frey B, Loeliger H-A. Factor graphs and the sumproduct algorithm. In: Information theory, IEEE transactions on. 2001. p. 498–519.

Librino F, Levorato M, Zorzi M. An algorithmic solution for computing circle intersection areas and its applications to wireless communications. In: Proceedings of the 7th international conference on modeling and optimization in mobile, Ad Hoc, and wireless networks. 2009. p. 239–48.

Magnien C, Latapy M, Habib M. Fast computation of empirically tight bounds for the diameter of massive graphs. J Exp Algorithm 2009;10:1.10–10:1.9.

Mozilla location service. Available from: https://location.services.mozilla.com/. [Accessed January 2016].

Muir JA, Oorschot PCV. Internet geolocation: evasion and counterevasion. ACM Comput Surv 2009;42(1):1–23.

Nemhauser G, Wolsey L, Fisher M. An analysis of approximations for maximizing submodular set functions I. Math Program 1978;265–94.

Sherry J, Lan C, Popa RA, Ratnasamy S. Blindbox: deep packet inspection over encrypted traffic. In: Proceedings of the 2015 ACM conference on special interest group on data communication. 2015. p. 213–26.

Toubiana V, Verdot V, Christophe B. Cookie based privacy issues on Google services. In: Proceedings of CODASPY '12. 2012. p. 141–8.

Vratonjic N, Huguenin K, Bindschaedler V, Hubaux J-P. How others compromise your location privacy: the case of shared public IPs at hotspots. In: PETS. 2013.

Vratonjic N, Huguenin K, Bindschaedler V, Hubaux J-P. A location privacy threat stemming from the use of shared public IP addresses. In: Mobile computing, IEEE transactions on. 2014. p. 2445–57.

Wang Y, Zhang Y, Liu J, Bhandari R. Coverage, Connectivity, and Deployment in Wireless Sensor Networks. In: Recent development in wireless sensor and Ad-hoc networks. 2015. p. 25–44.

Weiss Y, Freeman WT. On the optimality of solutions of the maxproduct beliefpropagation algorithm in arbitrary graphs. IEEE Trans Inform Theor 2006;736–44.

Xu C, Chen S, Su J, Yiu SM, Hui LCK. A survey on regular expression matching for deep packet inspection: applications, algorithms, and hardware platforms. IEEE Commun Surv Tutor 2016;18(4):2991–3029.

Yang Q, He S, Li J, Chen J, Sun Y. Energy-efficient probabilistic area coverage in wireless sensor networks. IEEE Trans Vehicular Technol 2015;64(1):367–77.

Yang Z, Wu C, Liu Y. Locating in fingerprint space: wireless indoor localization with little human intervention. In: Proceedings of the 18th annual international conference on mobile computing and networking, Mobicom '12. 2012. p. 269–80.

Yu Z, Teng J, Li X, Xuan D. On wireless network coverage in bounded areas. In: INFOCOM'13. 2013. p. 1195–203.

Zhao X, Xiao Z, Markham A, Trigoni N, Ren Y. Does BTLE measure up against WiFi? A comparison of indoor location performance. In: European wireless 2014; 20th European wireless conference. 2014. p. 1–6.

Yunlong Mao is pursuing his Ph.D. degree with the Department of Computer Science and Technology of Nanjing University. He received his B.S. degree in computer science from Nanjing University, in 2013. His research interests include data privacy, security and computer networks.

Yuan Zhang received the B.S. degree in automation from Tianjin University in 2005, the M.S.E. degree in software engineering from Tsinghua University in 2009, and the Ph.D. degree in computer science from the State University of New York at Buffalo in 2013. His current research interests include security, privacy, and economic incentives.

Xiaoyan Zhang received his Ph.D. degree in applied mathematics from Nankai University in 2006, Ph.D. degree in theoretical computer science from University of Twente in 2014. His current research interests include combination optimization, graph theory, algorithms and algorithms conpexity.

Fengyuan Xu is with the State Key Laboratory for Novel Software Technology, Nanjing University. He received his PhD degree, with Distinguished Dissertation award, from the College of William and Mary. His research interests are in the broad areas of systems and security, with a focus on big data analytics for security, wireless device protection, energy efficient mobile systems, and graph and stream processing platforms.

Sheng Zhong received the B.S. and M.S. degrees from Nanjing University in 1996 and 1999, respectively, and the Ph.D. degree from Yale University in 2004, all in computer science. He is interested in security, privacy, and economic incentives.