Privacy-Aware Location Data Publishing

Haibo Hu, Jianliang Xu, Sai Tung On, Jing Du, Joseph Kee-Yin Ng
Department of Computer Science
Hong Kong Baptist University

This paper examines a new problem of k-anonymity with respect to a reference dataset in privacy-aware location data publishing: given a user dataset and a sensitive event dataset, we want to generalize the user dataset such that by joining it with the event dataset through location, each event is covered by at least k users. Existing k-anonymity algorithms generalize every k user locations to the same vague value, regardless of the events. Therefore, they tend to overprotect against the privacy compromise and make the published data less useful. In this paper, we propose a new generalization paradigm called *local enlargement*, as opposed to conventional hierarchy- or partition-based generalization. Local enlargement guarantees that user locations are enlarged just enough to cover all events k times, and thus maximize the usefulness of the published data. We develop an $O(H_n)$-approximate algorithm under the local enlargement paradigm, where n is the maximum number of events a user could possibly cover and H_n is the Harmonic number of n. With strong pruning techniques and mathematical analysis, we show that it runs efficiently and that the generalized user locations are up to several orders of magnitude smaller than those by the existing algorithms. In addition, it is robust enough to protect against various privacy attacks.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*; H.2.0 [**Database Management**]: General—*Security, integrity, and protection*

Additional Key Words and Phrases: k-anonymity, location privacy

1. INTRODUCTION

With the advent of location-positioning technologies, such as the global positioning system (GPS) and CDMA network-based positioning, mobile telephone subscribers can now be continuously tracked by mobile operators or service providers whenever they are making calls, enjoying value-added services, or even when they are just idle. With the surging market growth in this industry, we foresee that it will not be long before the operators or service providers begin to publish their archived user location data for researchers and commercial organizations to carry out academic and market research (e.g., customer data mining), just as several years ago when supermarkets and internet portals started to analyze their customer shopping records or web browsing logs. In addition, to protect the public's right to know, governments will also recommend or require that data of this kind be published, just as the public records of medical treatments, marriages or voter registration are made public today.

However, how to publish such data without compromising user privacy is always a critical problem. In relational databases, k-anonymity (and its variants l-diversity and t-closeness) has been advocated as the predominant measure for privacy protection [Sweeney 2002; Machanavajjhala et al. 2006; Li et al. 2007]. The main idea is to group users based on the values of quasi-identifier attributes, and generalize these values on a group basis so that the users in each group are indistinguish-

able from one another. This idea has also been adapted to location-based services where location is considered the quasi-identifier attribute. Specifically, the locations of k users are generalized (or more accurately, *cloaked*) to the same region so that a service provider cannot distinguish the requesting user from other $k-1$ users (hereafter called *single-dataset k-anonymity*) [Gedik and Liu 2005; Mokbel et al. 2006; Ghinita et al. 2007b].

In this paper, we study a new problem of location data publishing. Consider a service provider (e.g., a department store) that administrates certain territories and monitors the locations of subscribing users (e.g., customers). For research or publicity purposes, the service provider wants to or needs to publish both the user location data and records of services provided to these users. For example, the store may want to publish customers' location data to study their shopping behavior (e.g., how customers' profiles such as age and education affect their interest in different shopping categories, such as food and clothing, or interest in a particular product of consumer electronics). Similarly, for market basket analysis, the store also wants to publish the purchase records. Figure 1 shows an example of the two published datasets.[1] Thus, the identities of the service records can be discovered by joining the location attributes of the two published datasets and hence finding spatiotemporal coincidence of a user and a service record. Once the identities are discovered, sensitive information can be deduced from them. For example in Figure 1, there is a purchase record of drug "diazepam" for insomnia (e_1) and Michael (o_1) happened to be the only customer at the drug counter when this drug was purchased (his location overlaps with the event location). Then, an adversary can infer that Michael must have bought diazepam and therefore was likely to have insomnia. Similarly, the adversary can infer that "Sara (o_3) probably smokes because she bought cigarettes (e_2)". We argue that this is not a standalone case, and there are many service providers or government administrators who are publishing or about to publish records of services or incidents for various purposes. Typical examples include clinics publishing their medical consultation records, and traffic authorities publishing records of traffic violations. In this paper, we use the general term *event* to denote such a record. As shown above, to prevent against privacy attacks, we should anonymize the *spatiotemporal coincidence* of user locations and sensitive events, so that the genuine users of these events cannot be inferred. Fortunately, similar to location anonymity, such anonymization can also be performed by generalizing the user locations. In Figure 1, if the locations of Philip and Lincoln are generalized (i.e., enlarged) to overlap with e_1 and e_2, respectively, then either Michael or Philip might have bought diazepam, and either Lincoln or Sara might have bought cigarettes.

In general, in this paper we are interested in the following problem: given a user dataset which records user IDs (or quasi-IDs) and locations, and a sensitive event dataset which records events and their locations, how can we generalize the user dataset with respect to the event dataset in such a way that by joining these two datasets through location, every event is covered by at least k users? In this way, an adversary who has no background knowledge can only infer that each of the k users

[1] Although the Purchases dataset may not explicitly contain the locations, they can be inferred since the purchases are made at publicly-known locations, e.g., drugs are sold at the drug counter.

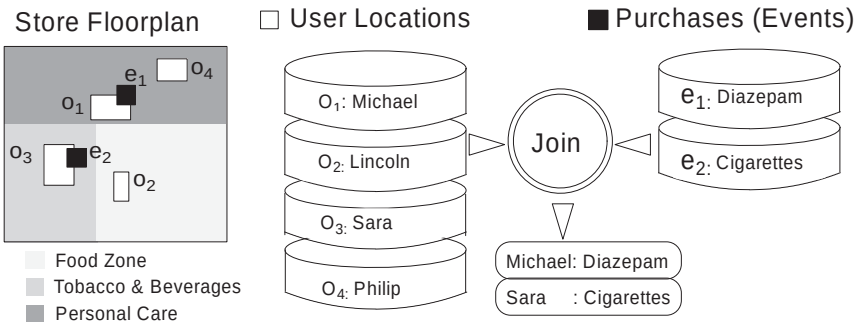
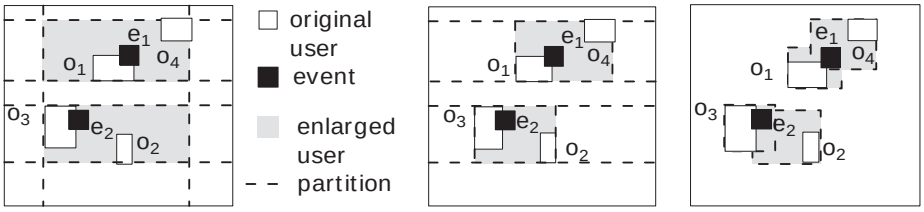


Fig. 1. Anonymize Spatiotemporal Coincidence

has the same $1/k$ probability of being the genuine user. This problem is termed as *k-anonymity with respect to a reference dataset*, in contrast to the previous single-dataset k-anonymity problem. The criterion of the published dataset, as in the single-dataset k-anonymity problem, is the *generalization cost* (in terms of the area or the perimeter of generalized locations). This cost is essentially the reverse of the *utility* — another commonly-used metric — of the published dataset. The lower the cost, the higher the utility, and thus the more details are preserved for analysis. Note that the generalization cost should not be confused with the CPU or I/O cost used for generalization operations. It is also noteworthy that in the presence of multiple event datasets, the problem can be recursively decomposed and reduced to the aforementioned standard form with only one event dataset, thanks to the associative and commutative properties of the join operation.

To solve the problem, the existing k-anonymity generalization algorithms (e.g., [LeFevre et al. 2005; Sweeney 2002; Samarati 2001; Fung et al. 2005; Wang et al. 2004; Iyengar 2002; Bayardo and Agrawal 2005]) can be applied. However, as these algorithm are not aware of event datasets, they may *overprotect* users' privacy and, hence, make published data less useful. According to generalization flexibility, the existing hierarchy-based or partition-based algorithms can be categorized as *single-dimension* or *multi-dimension* recoding[LeFevre et al. 2005]. The former means that generalization in one attribute is independent of the values in other attributes. For example, if the generalization rule of an attribute *zipcode* is "3317x", then any zipcode with the prefix "3317" must be generalized to "3317x", whatever the values of other attributes in this record. On the contrary, multi-dimension recoding allows more complex generalization rules across several dimensions. As a result, it might produce a better anonymization in terms of the generalization cost, but at the cost of higher computational complexity [LeFevre et al. 2006; Du et al. 2007; Ghinita et al. 2007b].

As complex generalization rules are shown to lead to a better anonymization, in this paper, we propose an innovative generalization paradigm called *local enlargement*, which goes one step further to allow each individual data record to have its own generalization rule. In the spatial context, this means that the location of each individual user is enlarged accordingly. Local enlargement can achieve higher flexibility and better generalization in many aspects than existing single- or multi-



(a) Single-Dimension Recoding (b) Multi-Dimension Recoding (c) Local Enlargement

Fig. 2. Local Enlargement vs Single- and Multi-Dimension Recoding

dimension recoding. First, the size (i.e., the cost) of the generalized location is much smaller. Figures 2(a), 2(b) and 2(c) show the original and generalized user locations for the motivating example in Figure 1, using single-dimension recoding, multi-dimension recoding, and local enlargement, respectively. Each achieves 2-anonymity with respect to events e_1 and e_2. In both single- and multi-dimension recoding, the generalization is based on space partition, where each partition that contains an event must contain two users and this partition becomes the generalized location of these two users. The difference between single- and multi-dimension recoding is the method of partition — whether to treat each attribute (i.e., dimension) independently or dependently. On the other hand, local enlargement guarantees that user locations are enlarged just enough to cover each event by two users. Therefore, it incurs a smaller generalization cost. In the figures, local enlargement has the smallest sum of areas of generalized user locations. Although this is at the cost of a higher computation overhead, with carefully designed algorithm and pruning techniques, the overhead can be significantly reduced and is thus justified for offline data publishing. Second, the existing generalization algorithm has no control of the generalized location of an individual user. However, as was pointed out in [Gedik and Liu 2005; Mokbel et al. 2006; Xiao and Tao 2006b], different people may have different degrees of privacy tolerance. Local enlargement allows user locations to enlarge according to their personalized privacy profiles and still achieves a low-cost generalization that conforms to both k-anonymity and these profiles. For example in Figure 2(c), if user o_1 is privacy-conservative and prefers a larger generalized location, it can be enlarged to cover both e_1 and e_2, and as a cost relief, o_2 can be shrunk to its original location if his or her privacy profile allows. Lastly, local enlargement handles updates efficiently. In case of a new user or event insertion, or an existing user or event removal, the update of enlargement is always restricted to the local area. However, in the existing generalization algorithms the events and users have to be re-partitioned, which means a single change may be propagated throughout the space.

Our contributions made in this paper are summarized as follows:

— We introduce a new problem of user location k-anonymity with respect to a reference event dataset, which arises from the observation that a join of the two datasets may associate users with sensitive events, leading to severe privacy compromise.

— We show that applying conventional single-dimension or multi-dimension k-anonymity algorithms on a user dataset only is inefficient in solving this problem. As such, we propose a new generalization paradigm, called *local enlargement*, that minimizes the generalization cost of the resulting dataset.

— We present an efficient and close-to-optimal algorithm for local enlargement. Furthermore, we formally prove its H_n approximation ratio, where n is the maximum number of events a user could possibly cover and H is the Harmonic number of n, that is, $H_n = \sum_{i=1}^{n} \frac{1}{i}$.

— We propose advanced pruning techniques for this algorithm. Furthermore, their pruning power is shown through analytical cost models and extensive experiment results, which exhibit their efficiency and feasibility.

— We extend the location k-anonymity problem to allow additional background knowledge of user-event correspondence. In particular, we formalize the privacy loss using entropy and prove that certain privacy threats can be prevented by further restricting each user to cover the same number of events. As such, we present a coverage adjustment algorithm as a post-generalization procedure in face of such threats. Our experimental results show that our algorithm can sustain various privacy attacks.

The rest of the paper is organized as follows. Section 2 reviews existing work on privacy-aware data publishing. Section 3 formally introduces the problem, followed by the local enlargement algorithm and the proof of its approximation ratio in Section 4. Section 5 proposes the pruning algorithms and Section 6 presents their cost models in terms of pruning power. Section 7 extends the problem by also allowing the adversary to have background knowledge and proposes a coverage adjustment algorithm. The experimental results are shown in Section 8.

2. RELATED WORK

Privacy-aware data publishing has been recently studied in RDBMS [Samarati 2001; Sweeney 2002; Machanavajjhala et al. 2006; Li et al. 2007; Xiao and Tao 2006b; 2006a]. Samarati and Sweeney studied the phenomenon of the *quasi-identifier*, a set of attributes that can distinguish any record from the others in the table. As a classic example, the zipcode and birthdate together may already identify every patient of a patient table within a hospital. They therefore proposed k-anonymity to generalize the values of quasi-identifier attributes in each record so that it was indistinguishable from at least $k-1$ other records with respect to the quasi-identifier [Samarati 2001; Sweeney 2002]. However, k-anonymity only prevents adversary A from associating an identity (e.g., a patient) with a complete record in the table. It cannot prevent A from associating the identity with the value of some attribute(s), which might be sensitive. For example, if all k indistinguishable records (which are called an equivalence class) in the patient table have a value of "cancer" in their "disease" attribute, then A can infer that a patient who falls in this class has cancer with 100% confidence, even though A may not know which record this patient is associated with. This privacy compromise is called *attribute disclosure*, as opposed to *identity disclosure*, which is solved by k-anonymity. To solve attribute disclosure, Machanavajjhala *et al.* proposed a new measure of privacy, called l-

diversity [Machanavajjhala et al. 2006], which requires any equivalence class to have at least l well-represented values in the sensitive attribute (e.g., disease).

However, l-diversity treats each value of the sensitive attribute equally and thus only prevents adversary A from associating an entity with any specific sensitive value. Li *et al.* pointed out that even A cannot make associations, A still gains additional information from the generalized table, as long as the value distribution of the sensitive attribute in an equivalence class is different from that in the global table [Li et al. 2007]. As a typical example, suppose that there are only two values for the disease attribute, "cancer" and "heart disease". If in the global table, only 1% of patients develop cancer while in an equivalence class 2 out of 4 patients develop cancer, then the posteriori probability of a patient in this equivalence class developing cancer is significantly higher than the prior probability (50% vs. 1%), which severely compromises the patient's privacy. As such, they proposed a t-closeness privacy measure. This requires the value distribution in any equivalence class to differ by at most t from that in the global table. t-closeness can be considered as the generalization of l-diversity as it measures the loss of privacy by the increment of the posteriori probability instead of the number of distinct values.

The same limitation of k-anonymity and l-diversity was also studied by Xiao and Tao, who proposed an alternative solution that allows each individual to specify the granularity of value(s) he or she tolerates to reveal in the sensitive attribute [Xiao and Tao 2006b]. For example, a user may feel alright if adversary A associates him/her with cancer, but he/she may feel his/her privacy is violated if A can associate him with "prostate cancer," a sub-category of "cancer." With this fine-tuned privacy measure in addition to k-anonymity or l-diversity, the user can reserve the privacy that is of concern to him/her while revealing the rest for data analysis. To satisfy this additional privacy measure, Xiao and Tao suggested a second sensitive attribute generalization step after the traditional quasi-identifier generalization.

Apart from elaborating the input privacy measures that apply to more general and practical privacy-preserving scenarios, researchers also attempted to elaborate the format of output — the published data table. Xiao and Tao recommended that instead of publishing one generalized table, it is advisable to anatomize it into two tables, one quasi-identifer table and another sensitive table, with an equivalence class ID (*group ID* as they call it) being the foreign key to join them [Xiao and Tao 2006a]. They showed that while preserving the same amount of privacy, the anatomized tables reserve more detailed facts from the original table for data analysis, especially in terms of aggregate queries.

Many generalization algorithms have been devised to achieve k-anonymity in a single table [LeFevre et al. 2005; Sweeney 2002; Samarati 2001; Fung et al. 2005; Wang et al. 2004; Iyengar 2002; Bayardo and Agrawal 2005]. Early attempts assume a generalization hierarchy on each quasi-identifier attribute. Some even require that all values of this attribute must be generalized to values of the same level in the hierarchy (*full-domain generalization*, as opposed to *full subtree generalization*) [LeFevre et al. 2005]. Sweeney proposed Datafly, a greedy algorithm that generates frequency lists and iteratively generalizes those combinations with less than k occurrences [Sweeney 2002]. She also proposed an exhaustive algorithm that examines all possible generalizations to find the optimal one. However, this

approach is shown to be impractical even on a medium table. Samarati proposed another optimal algorithm by exploiting binary search and pruning the search space with the monotonicity property on the generalization lattice, but the algorithm remains exponential in the worst or even average case [Samarati 2001]. LeFevre *et al.* proposed the Incognito framework, the core of which is a dynamic programming algorithm with a priori pruning [LeFevre et al. 2005]. The idea is to construct all possible $n+1$-attribute generalizations (which form a lattice) from the n-attribute lattice. Since those nodes in each lattice that cannot be the optimal generalization are pruned, the size of the lattice should not grow exponentially as n increases.

As for full-subtree generalization algorithms, greedy top-down and bottom-up generalization techniques were devised by Fung *et al.* [Fung et al. 2005] and Wang *et al.* [Wang et al. 2004]. To remedy the "localness" of the search space, Iyengar proposed using a genetic algorithm [Iyengar 2002]. It is noteworthy that this work belongs to another family of k-anonymity generalization algorithms that is based on partition instead of hierarchy [LeFevre et al. 2005]. In these algorithms, the domain of an attribute is a set of total order, and generalizations are defined by partitioning the set into disjoint ranges. Apart from [Iyengar 2002], Bayardo and Agrawal obtained the optimal partition-based generalization by providing a branch-and-bound search strategy in the power-set of all partition possibilities [Bayardo and Agrawal 2005]. In general, partition-based algorithms are most suitable for continuous or numeric-valued attributes, and the hierarchy-based algorithms are better suited for categorical values [LeFevre et al. 2005].

Location anonymity is regarded as the spatial equivalence of k-anonymity in RDBMS, and location is usually a quasi-identifier in a spatial dataset. Generalization on user locations is called *location cloaking*. The generalized location of a user, called a *cloaked region*, is usually a circle or a rectangle that encloses the genuine position of this user as well as at least other $k-1$ users. Gruteser and Grunwald[Gruteser and Grunwald 2003] were the first to propose spatio-temporal cloaking, where a trusted middleware generalizes (i.e., cloaks) the spatial and temporal extents of the location for the requesting user to achieve k-anonymity. More specifically, the middleware indexes all user locations using a Quadtree. Upon receiving a request, the middleware traverses the Quadtree until it finds a quadrant containing the user of this request and other $k-1$ users. This quadrant is the cloaked region for this user. Gedik and Liu considered a personalized k-anonymity model and proposed "Clique-Cloak," which constructs a clique graph to combine clients that can share the same cloaked region[Gedik and Liu 2005; 2008]. A grid-based cloaking algorithm was suggested in the *Casper* framework[Mokbel et al. 2006]. Iwuchukwu and Naughton proposed to leverage classical spatial indexing to achieve efficient k-anonymity[Iwuchukwu and Naughton 2007]. Ghinita *et al.* studied cloaking in a distributed environment and proposed *hilbASR* to sort all users and store this ordering in a distributed annotated B^+-tree index [Ghinita et al. 2007b]. In *hilbASR*, all users are sorted by Hilbert space-filling curve ordering according to their locations, and then every k users are grouped together in this order. They have also extended this framework to a Chord peer-to-peer environment and used distributed hash tables, instead of the hierarchical B^+-tree to store user locations[Ghinita et al. 2007a].

Our paper addresses a new problem of achieving k-anonymity in a dataset with respect to a reference dataset. Moreover, instead of being based on hierarchy or partition, the algorithm we propose belongs to a new generalization paradigm, which is called local enlargement.

3. PROBLEM DEFINITION

DEFINITION 1. *A location dataset is a collection of identifier-location pairs <* $o.obj\_id$,$o.b\_box$*>, where* $o.obj\_id$ *is the identifier of an object* o *(e.g., a user or an event) and* $o.b\_box$ *is the bounding box of* o*'s location.*

DEFINITION 2. *Given a location dataset* $\mathcal{D}$ *of users and a reference location dataset* R *of events, a* **k-anonymous** *dataset of* $\mathcal{D}$ **with respect to R** *is a location dataset* $\mathcal{D}'$ *of the same users as* $\mathcal{D}$ *such that: 1) for any user* o*,* $o.b\_box'$*, the bounding box in* $\mathcal{D}'$*, fully contains* $o.b\_box$*, the bounding box in* $\mathcal{D}$*, i.e.,* $o.b\_box \subseteq o.b\_box'$*; and 2) for any event* $e \in R$*, there are at least* k *users in* $\mathcal{D}'$ *whose bounding boxes overlap with* $e.b\_box$*.*

In other words, $\mathcal{D}'$ is obtained by enlarging the bounding boxes of some (or all) users in $\mathcal{D}$ so that there are always k or more users whose bounding boxes overlap with that of an event, or more formally, k or more users *cover* an event. Obviously, an intuitive solution to this problem is to pick k users independently for each event e and enlarge their b_box to cover $e.b\_box$. However, this may make the bounding boxes of the published dataset $\mathcal{D}'$ too large to be useful. To quantify the usefulness of the published data, we define the *generalization cost* of a bounding box as a monotonic increasing function over its dimensions, or more formally, $cost(\alpha\_box) \leq cost(\beta\_box)$ if $\alpha\_box \subseteq \beta\_box$, and the generalization cost of the entire dataset is the summation of the cost over the bounding boxes of all users, that is, $cost(\mathcal{D}') = \sum_{o \in \mathcal{D}'} cost(o.b\_box')$. A typical cost function could be the area/volume/perimeter of the bounding box. Therefore, our objective in the sequel is to find a k-anonymous $\mathcal{D}'$ with the minimum generalization cost.

There are four remarks that should be noted about the above definition. First, the threat model in this definition assumes an adversary knows only the published k-anonymous dataset $\mathcal{D}'$ and event dataset R. Without any background knowledge, the adversary can only infer that each of the k covering users has the same $1/k$ probability of being the genuine user. This threat model is stricter than what is assumed in the traditional k-anonymity model, where the adversary also knows the input dataset $\mathcal{D}$. This stricter model enables more flexibility in the design of generalization algorithm than the k-anonymity model and can thus achieve better utility (i.e., lower generalization cost) for the published dataset. On the other hand, however, this model does not fit to applications where some information about the input dataset $\mathcal{D}$ is known to the adversary. For example, if the input locations are known to be points or in fixed precision, the adversary might be able to guess the genuine user with confidence higher than $1/k$. We have a detailed analysis in Section 7 on the implication of such background knowledge.

Second, while two or more users may have the same b_box, two or more events may also share their b_box's. This usually happens when the location precision is coarse. For example, two patients have medical consultations at the same time but in different doctor's offices; and their b_box's are both "the clinic." In this

paper, we treat them as two distinct events because they have different genuine users. Third, the input datasets of users and events share the same context. That is, the user dataset is the exact population who carried out the events in the event dataset. As such, every event is already covered by at least one user — the genuine user of this event and possibly some more users; so actually the problem is to find some other users to have a total of k covering users. Note that this context is unique, so no other user or event dataset can join with the input datasets. Also note that each event may have a different k value, which is predetermined by factors such as the privacy preference of the genuine user and the sensitivity of this event. However, for ease of presentation, throughout the paper we use a unified notation k to denote the number of covering users required to find for each event. Fourth, normally "overlap" means intersection; that is, two boxes share some common area. However, in order to minimize the generalization cost, at the border the enlarged bounding box of a user should just "touch" the box of an event; that is, they only share their borders. As such, in the sequel we use "overlap" and "touch" interchangeably. It is noteworthy, however, that the adversary may exploit this by considering any user whose enlarged bounding box just touches an event as a non-genuine user of this event. Nonetheless, this exploit can be solved by a post-process that randomly expands those enlarged bounding boxes that touch events. More specifically, these boxes are expanded to intersect the events such that on average they will intersect the events by the same proportion of event area as the boxes of the genuine users do. Since this process is inexpensive and introduces very little increase of the generalization cost, we ignore this in the rest of the paper.

4. LOCAL ENLARGEMENT ALGORITHM

As mentioned above, the bounding box b_box of a user o should be enlarged to cover a number of events. Given a reference dataset R, in each dimension there are at most $2 * |R|$ ($|R|$ is the cardinality of R) coordinates from which the borders of the enlarged box $o.b\_box'$ can be selected. Figure 3 illustrates a 2D scenario with three events, e_1, e_2 and e_3. Given the bounding box o, in the x-dimension, the border on the right can be selected from x_3, x_4 and x_5 only, while the border on the left can be selected from x_1 and x_2 only. Similarly, the border on the top can be selected from y_1, y_2 and y_3 only, while the border on the bottom can be selected from y_4 and y_5 only.

By selecting from different x and y coordinates, we obtain a set of enlarged bounding boxes for o, which is subsequently called the *candidate set* of o and each is called a *candidate box*.[2] Therefore, the k-anonymity problem is equivalent to finding an extended bounding box Ω for each user so that each event is covered by at least k such Ω's and their total size is minimized.

This problem is similar to a set cover problem if we regard each event as an *element*, the entire event set R as the universe U, and each candidate box as a *subset* of U which contains the events it covers. The weight of a subset S is the generalization cost of the corresponding candidate box. Our problem is equivalent to finding a set of subsets that cover each event k times while the sum of weights is

[2]The candidate set also contains the current bounding box of the user, i.e., his or her original location before generalization.

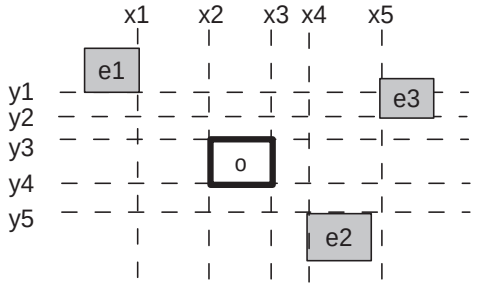


Fig. 3. At Most $2 * |R|$ Coordinates in Each Dimension

minimized. More accurately, this is a set multi-cover problem because each event must be covered multiple times. Furthermore, this problem is a unique variation of set multi-cover because there is a further restriction that at most one candidate box from each object's candidate set can be selected for the object to enlarge. That is to say, each candidate set serves as a *superset* in the set multi-cover problem from which at most one subset can be selected. We therefore call this problem as the *superset multi-cover* problem, which is formally defined as follows.

DEFINITION 3. **(Superset Multi-Cover Problem)** *Given a universe U of elements, a collection of supersets O, $O = \{O_1, ..., O_m\}$, each O_i is a collection of subsets of U; that is, $O_i = \{S_1^i, S_2^i, ...\}$, where S_j^i denotes the j-th subset in O_i, and a cost function $c : S_j^i \rightarrow Q^+$ (Q^+ is the domain of positive real numbers), choose at most one subset from each superset to form a collection $\{S_{j_1}^{i_1}, S_{j_2}^{i_2}, ...\}$ such that: 1) each element e is covered at least k times, and 2) the sum of costs is minimal.*

It is possible that none of the candidate boxes in a superset O_i is selected. In this case, the bounding box of the corresponding user o is simply not extended; that is, $o.b\_box' = o.b\_box$.

4.1 The Greedy Algorithm

For set cover and set multi-cover problems, a greedy algorithm can achieve an H_n approximation ratio to the optimal solution [Vazirani 2001], where n is the maximum cardinality of any subset S_j^i, and H_n is the Harmonic number of n; that is, $H(n) = \sum_{i=1}^{n} \frac{1}{i}$. The greedy algorithm chooses one subset at a time until each element is covered k times. In particular, it always chooses the subset Ω_* with the lowest *amortized cost*, i.e., the cost divided by the number of "living" elements that are covered by the subset. Element e is alive if so far the selected subsets cover it fewer than k times. Let C denote the set of elements that are no longer alive; then the amortized cost of subset S_j^i is $\frac{cost(S_j^i)}{|S_j^i - C|}$, and is hereafter called the c/s value. In our superset multi-cover problem, we can apply the same greedy algorithm, i.e., we repeatedly choose a subset (i.e., a candidate box) S_j^i with the minimum c/s value and let Ω_* denote it. The difference is that once this Ω_* is selected, only some candidate boxes in Ω_*'s corresponding superset O_* can continue to be selected while the rest should be removed from O_*. These candidate boxes are

those of which Ω_* is a subset; that is, each of them fully contains Ω_*. Therefore, whichever box is selected subsequently, it must cover the elements covered by Ω_*, which can thus be replaced by this new box. This modification guarantees that the algorithm always conforms to the restriction of at most one candidate box from each superset. Algorithm 1 shows the pseudo-code of this revised greedy algorithm. In each iteration, Ω_* is obtained in two steps. The algorithm first finds the candidate box with the minimum c/s value in each candidate set O_i (denoted by Ω_i), and then obtains Ω_* from all Ω_i's.

Algorithm 1 Algorithm for Set Multi-Cover Problem

Input: O_i: $1 \leq i \leq m$, m supersets
k: coverage requirement
Output: $\mathcal{S}_i$: $1 \leq i \leq m$, the selected subset in O_i
Procedure:
initialize $\mathcal{S} = C = \emptyset$;
while $C \neq U$ **do**
  for each O_i **do**
    Ω_i = the subset with the lowest c/s value in O_i;
  Ω_* = the subset with the lowest c/s value in all Ω_i's;
  $\mathcal{S}_* = \Omega_*$; // select this subset for superset O_*
  remove all subsets in O_* of which Ω_* is not a subset;
  for each element e in $\Omega_* - C$ **do**
    if e is covered k times **then**
      $C = C \cup \{e\}$; // e is no longer alive
      update c/s values of all subsets that cover e;
return all $\mathcal{S}_i$;

4.2 Proof of Approximation Ratio

We now prove that the greedy Algorithm 1 is an H_n-approximation algorithm for the superset multi-cover problem. The following proof is based on the linear programming (LP) duality theory [Vazirani 2001] but addresses our unique requirement of the superset. Let us assign a variable x_S for each set $S \in \bigcup O_j, 1 \leq j \leq m$. $x_S = 1$ if and only if set S is selected by the superset multi-cover; $x_S = 0$ if and only if it is not selected. Then the superset multi-cover problem is reduced to an integer programming problem which minimizes the cost: $\sum_{S \in \bigcup O_j} c(S)x_S$.

$$
\begin{aligned}
&\textit{minimize} && \sum_{S \in \bigcup O_j} c(S)x_S \\
&\textit{subject to} && \sum_{S:\, e \in S} x_S \geq k, && \forall e \in U \\
& && -\sum_{S \in O_j} x_S \geq -1, && \forall 1 \leq j \leq m \\
& && x_S \geq 0, && S \in \bigcup O_j
\end{aligned}
$$

Note that the constraints $x_S \leq 1$ are redundant because any set S must be contained in some O_j. These constraints are implicit in constraints $-\sum_{S \in O_j} x_S \geq -1$.

According to the LP-duality, the dual problem is

$$\begin{array}{ll} maximize & \sum_{e\in U} k y_e - \sum_{1\le j\le m} z_j \\ subject\ to & (\sum_{e:\ e\in S} y_e) - z_j \le c_S,\ S \in O_j \\ & y_e \ge 0, \qquad\qquad e \in U \\ & z_j \ge 0, \qquad\qquad 1 \le j \le m \end{array}$$

Then we construct a solution to this dual problem based on the greedy algorithm. Let the cost of selecting set S be evenly distributed among the living elements it covers, and let $price(e,t)$ denote the cost distributed to e when e is covered for the t-th time; that is, $price(e,t) = \frac{c(S)}{|S-C|}$, where S is the set to cover e, and C still denotes the set of elements that are no longer alive at that time. Obviously, according to the greedy algorithm, $price(e,t)$ is non-decreasing; that is, $price(e,t_1) \le price(e,t_2)$ if $t_1 \le t_2$. Therefore, the solution to the dual problem is set as follows: for each $e \in U$, $y_e = \frac{1}{H_n} price(e,k)$, and $z_j = \frac{1}{H_n} \sum_{e\ covered\ by\ S\in O_j} (price(e,k) - price(e,t_e))$, where t_e is the copy of e that is covered by $S \in O_j$; $z_j = 0$ if no $S \in O_j$ is selected by the greedy algorithm. The following lemma proves that y_e and z_j form a feasible solution to the dual problem.

LEMMA 1. *y_e and z_j is a feasible solution to the dual problem.*

PROOF. For any set $S \in O_j$, let l denote the cardinality of S. Without loss of generality, we further assume the elements in S are in the order of being covered for the last (i.e., k-th) time. There are three cases for S in the algorithm:

— S is not selected by the algorithm, and none is any set in O_j. Then when the algorithm is about to cover e_i for the k-th time, S contains at least $l-i+1$ elements to be covered. Since S is not selected, we have

$$price(e_i,k) \le \frac{c(S)}{l-i+1}$$

$$\begin{aligned} (\sum_{i=1}^{l} y_{e_i}) - z_j &= \frac{1}{H_n} \sum_{i=1}^{l} price(e_i,k) \\ &\le \frac{c(S)}{H_n}(\frac{1}{l} + \frac{1}{l-1} + \cdots + \frac{1}{1}) \le c(S) \end{aligned}$$

— S is selected by the algorithm. Then,

$$(\sum_{i=1}^{l} y_{e_i}) - z_j = \frac{1}{H_n}[\sum_{i=1}^{l'} price(e_i,k) + c(S)],$$

where $e_1, ..., e_{l'}$ have already been covered k times before S is selected. Therefore, for $1 \le i \le l'$, $price(e_i,k) \le \frac{c(S)}{l-i+1}$. So,

$$(\sum_{i=1}^{l} y_{e_i}) - z_j \le \frac{c(S)}{H_n}(\frac{1}{l} + \cdots + \frac{1}{l-l'+1} + 1) \le c(S)$$

— S is not selected by the algorithm but another set $S' \in O_j$ is selected by the algorithm. Let $\epsilon_1, ..., \epsilon_{l'}$ denote the elements in S'. Then,

$$(\sum_{i=1}^{l} y_{e_i}) - z_j = \frac{1}{H_n}[\sum_{i=1}^{l} price(e_i, k) - \sum_{u=1}^{l'} price(\epsilon_u, k) + c(S')]$$

According to the definition of *price*, $\sum_{i=1}^{l} price(e_i, k) - c(S)$ is the gain of cost reduction if S is selected after S' to cover all e_i's; similarly, $\sum_{u=1}^{l'} price(\epsilon_u, k) - c(S')$ is the gain of cost reduction if S' is selected after S to cover all e_u's. Since the greedy algorithm always selects the set with lowest cost, the former gain must be lower than the latter gain; that is, $\sum_{i=1}^{l} price(e_i, k) - c(S) \leq \sum_{u=1}^{l'} price(\epsilon_u, k) - c(S')$. So,

$$(\sum_{i=1}^{l} y_{e_i}) - z_j = \frac{1}{H_n}[\sum_{i=1}^{l} price(e_i, k) - \sum_{u=1}^{l'} price(\epsilon_u, k) + c(S')] \leq \frac{c(S)}{H_n} \leq c(S)$$

□

Next, we present the following theorem about the approximation bound.

THEOREM 2. *For the superset multi-cover problem, the greedy algorithm achieves an approximate bound of H_n.*

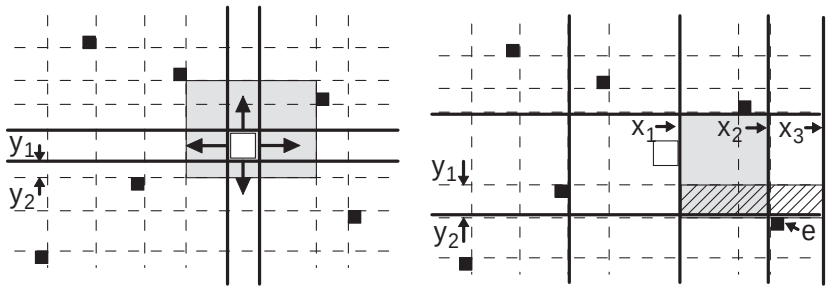
PROOF. Putting the dual solution y_e, z_j to its objective function, we have

$$\sum_{e \in U} k y_e - \sum_{1 \leq j \leq m} z_j = \frac{1}{H_n} \sum_{e \in U} \sum_{t=1}^{k} price(e, t) = \frac{GRD}{H_n},$$

where GRD is the total cost for the greedy algorithm. According to the LP-duality theorem, the above value must be a lower bound for the original superset multi-cover problem. That is, $\frac{GRD}{H_n} \leq OPT$; that is, $GRD \leq H_n \cdot OPT$. □

4.3 Fast Scan: Enhancing the Greedy Algorithm

The most time-consuming portion of the greedy algorithm involves finding Ω_i, the candidate box with the lowest c/s value in each candidate set. To simplify the notations in the sequel, we omit subscript i if Ω_i is unambiguous in the context. Intuitively, we have to compute the c/s values for all candidate boxes. These boxes can be enumerated by gradually scanning (and incorporating) the events from the user's bounding box to the top, left, bottom, and right in an iterative manner. Figure 4(a) shows the basic scanning algorithm. The initial locations of the top, left, bottom, and right scanning lines are the respective borders of the user's bounding box (white box). The enumeration is performed in a four-level loop, each of which



(a) Basic Scan (b) Fast Scan

Fig. 4. Implementation of the Greedy Algorithm

corresponds to one scanning line. Since the scanning lines pass through the borders of all events, all candidate boxes are enumerated, which justifies the correctness of the algorithm.

The basic scan algorithm examines all candidate boxes, whose number is $(|R|/2)^4$ in the worst case. On the other hand, since we are only interested in finding Ω, the one with the lowest c/s value, not all of them qualify. For example, the grey box in Figure 4(a) cannot be Ω because its bottom border y_2 does not touch any event and moving this border up to y_1, the bottom of the user's bounding box, results in a lower c/s value. In general, a possible candidate box must have each of its borders touch at least one event. Note that two borders may touch the same event; for example, the event in the top left corner of the grey box in Figure 4(a).

On the basis of this observation, we further propose a fast scan algorithm. This idea is illustrated in Figure 4(b). The improvement over the basic scan is that instead of resetting the corresponding scanning line to its initial location (the user's bounding box) in the beginning of each loop, the line is reset according to the scanning line of its immediate outer loop. Figure 4(b) shows the scenario when the algorithm is executed in the third loop, which corresponds to the bottom scanning line, and this line is moving from y_1 to y_2. Then the starting position for the right scanning line in the fourth loop should be x_2 rather than x_1 because any box scanned in between x_1 and x_2 is larger than it was in the previous loop when the bottom scanning line was at y_1, but it covers the same events as before. x_2 is the right-hand border of e, the event touched by the bottom scanning line when it moves to y_2. In general, the starting position of a scanning line is determined by the event that is touched by the scanning line in its immediate outer loop. The pseudo-code of the fast scan algorithm is described in Algorithm 2. Note that to scan sequentially, the algorithm should first sort the set of events R according to both the x- and y-axes. In addition, during the scan the algorithm always maintains the set of events covered by the current scanned area (i.e., the area formed by current scanning lines), so that when a candidate box B is formed, its c/s value is available immediately. It is noteworthy that each time a scanning line skips from the user bounding box to the location determined by e, a range query is needed to find the events covered by the skipped area, for example, the grey area in Figure 4(b), and add them to the set of events covered so far. This can be done by performing a

binary search first on either the x- or y-axis-sorted events, and then filtering those not inside this area.

Algorithm 2 Fast Scan Algorithm

Input: R: the set of events
o: the user
Output: Ω: the candidate box with the lowest c/s value
Procedure:

```
1: initialize scan lines S_top, S_left, S_bottom, S_right to o.b_box;
2: while S_top has not reached the end do
3:    S_left = min{e.right, o.b_box.left}
4:    while S_left has not reached the end do
5:       S_bottom = min{e.top, o.b_box.bottom};
6:       while S_bottom has not reached the end do
7:          S_right = max{e.left, o.b_box.right};
8:          while S_right has not reached the end do
9:             current scan lines form candidate box B;
10:            if B.cs < Ω.cs then
11:               Ω = B;
12:            move S_right;
13:         move S_bottom and get e;
14:      move S_left and get e;
15:   move S_top and get e;
16: return Ω;
```

The time complexity of the greedy algorithm (Algorithm 1) depends on that of the fast scan algorithm (Algorithm 2). A single run of Algorithm 2 or the basic scan algorithm costs $O(|R|^4)$ in the worst case. Therefore, it takes $O(|R|^4|\mathcal{D}|)$ time to find Ω's for all candidate sets in $\mathcal{D}$. In addition, each time an event e is covered for the k-th time and is thus removed from R, all Ω's need to be checked for whether they cover e, and those that do cover e have to be updated using Algorithm 2. Since there are $|R|$ event removals, and each removal incurs at most $|\mathcal{D}|$ runs of Algorithm 2, the total worst-case time complexity is $O(|R|^5|\mathcal{D}|)$. However, as will be shown in the next section, by applying appropriate pruning, the Ω's needed to be updated during each event removal can be restricted to a small number and can even be independent of $|R|$ or $|\mathcal{D}|$. Therefore, the practical time complexity is bounded by $O(|R|^4(|\mathcal{D}| + |R|))$.

5. ADVANCED PRUNING TECHNIQUES

Recall that a single run, even of the fast scan algorithm (Algorithm 2), still costs $O(|R|^4)$ in the worst case. To further reduce the total time cost, in this section we propose two pruning techniques: plane-sweep and index-based pruning. The former reduces the number of events ($|R|$) to be scanned during the fast scan algorithm; that is, during computing of a single Ω_i. The latter reduces the number of Ω_i's to compute to obtain Ω_*.

5.1 Plane-Sweep Pruning

In a given candidate set, not all candidate boxes can possibly be Ω (i.e., the candidate box with the lowest c/s value) because some events are too far away from

the user (i.e., too costly) to cover. We can therefore restrict the boundaries for the candidate boxes. However, we must assume the cost of a candidate box S is linear or hyper-linear to its area, or, more formally, there are constant values $c > 0, t \geq 1$ such that $cost(S) = c * Area(S)^t$, for any box S. Linearity or hyper-linearity guarantees the following lemma, which forms the ground of plane-sweep pruning.

LEMMA 3. *Suppose that a candidate box S is partitioned into several disjoint subspaces called "parts", $p_1, p_2, \cdots, p_i, \cdots$. If the c/s value of any p_i is larger than μ, then the c/s value of S is larger than μ.*

PROOF. According to linearity or hyper-linearity, $cost(S) \geq \sum_i cost(p_i)$. And since the c/s value of any p_i is larger than μ, we have

$$\mu \leq \frac{cost(p_i)}{|p_i - C|} \Longrightarrow |p_i - C| \leq \frac{cost(p_i)}{\mu}.$$

Summing up over all p_is, we obtain

$$\sum_i |p_i - C| = |S - C| \leq \frac{\sum_i cost(p_i)}{\mu} \leq \frac{cost(S)}{\mu}.$$

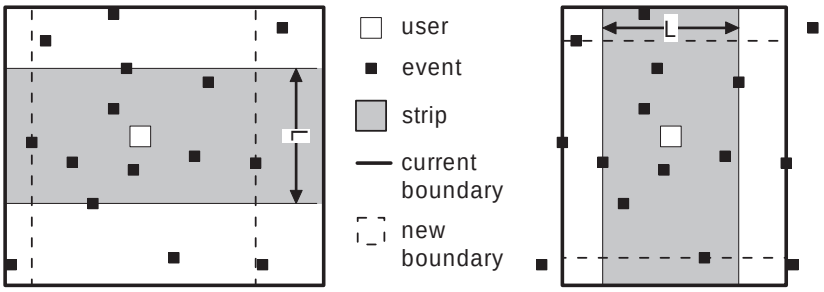
Finally, we have $\frac{cost(S)}{|S-C|} \geq \mu$. □

This lemma immediately enables further pruning in our existing fast scan algorithm. In Figure 4(b), as mentioned above, when the scanning line moves from y_1 to y_2, we can restrict the start location of the right scanning line to x_2. Now with Lemma 3, we can further restrict the stop location of this line to x_3, where x_3 is the solution to $(x_3 - x_1) * (y_2 - y_1) = \mu$ and μ is the c/s value of the best candidate box scanned so far. More formally:

Pruning Condition 1: In each loop of the fast scan algorithm, the stop location of the scanning line is $x_1 + \mu/(y_2 - y_1)$.

PROOF. When the right scanning line is at x, the difference between the two candidate boxes scanned in loop y_1 and loop y_2 is the shaded area whose width is $x - x_1$ and whose height is $y_2 - y_1$. This area covers 1 event (event e only) and thus the c/s value of this area is at least $(x - x_1) * (y_2 - y_1)$. If this value is larger than μ (i.e., $x > x_3$), then according to Lemma 3, the c/s value of the whole candidate box must exceed μ. This guarantees that no better candidate will be found beyond x_3. □

In the rest of this subsection, we propose a more powerful plane-sweeping technique that prunes those events that are ineligible for fast scan. The idea is to first compute Ω with the lowest c/s value among all candidate boxes for a sub-area (with less computational cost) and then prune the search space which cannot contain a candidate box with a c/s value lower than that of Ω. The pruning process is executed in each dimension repeatedly. Figures 5(a) and 5(b) illustrate the first two iterations. In each iteration, we compute Ω for a strip area (marked grey in the figures) centered at the user with height L. L is a parameter that depends on the current boundaries for candidate boxes and distribution of the events inside it. The optimal choice of L will be analyzed in Section 6.1. Let μ denote the c/s value of this Ω. Then we contract the x-axis boundaries (see Figure 5(a)) for candidate



(a) First Iteration (b) Second Iteration

Fig. 5. Pruning through Iterations

boxes (the dotted line) by applying the plane-sweep algorithm (to be detailed in the next paragraph) from the user toward the west and east, respectively. The same procedure is repeated in the second iteration (see Figure 5(b)), except that the strip is of width L and the contraction is on y-axis boundaries. The process is repeated until either the strip spans the entire interior of the current boundaries or none of the boundaries can be contracted anymore. If it terminates for the latter reason, Ω should be re-computed by enumerating the candidate boxes within the final boundaries.

In the plane-sweep algorithm, the sweeping line P limits the set of events seen so far. Each time P scans a new event, the algorithm updates the boundary B, which is parallel to P and guarantees that the lowest-c/s-value box of the events seen so far will not cross it. The area between B and P is always external to this box, so we call it the *external area* (see Figure 6). Initially, both B and P are located at the borders of the user's bounding box, so the external area is empty. Then P sweeps event-by-event until it reaches the end of the current boundary. Since all events are seen, the final external area can be pruned. To update B according to P, Figure 6 shows an example where P sweeps towards the west. Each time P sweeps to a new event, the algorithm checks if the new external area is still valid; that is, if there might be a lower-c/s-value box than Ω whose the left side is between B and P. To check this, let w denote the number of events in the external area; if the lower-bound cost of the part of any box in the external area, which is $L/2 * d$ (d is the distance between B and P), is larger than or equal to $w * \mu$, the c/s value of this part must always be larger than μ. Then, according to Lemma 3, the lowest-c/s-value box cannot overlap with the external area because removing that part would lead to an even lower c/s value. Now that the external area is still valid, B need not move. In contrast, if condition $L/2 * d \geq w * \mu$ does not hold, we move B to P to reset the external area to empty. The following pruning heuristic summarizes the plane-sweep technique.

Pruning Condition 2: No candidate box that overlaps with the external area, specified by $L/2 * d \geq w * \mu$, needs to be scanned.

Algorithm 3 shows the pseudo-code for the plane-sweep algorithm towards the west, where x_1 and x_2 denote the coordinates of the left and right borders of a box, respectively.

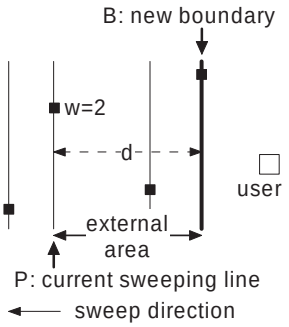


Fig. 6. Example of Plane Sweep Algorithm

Algorithm 3 Plane-Sweep Algorithm (westward)

Input: R: the set of events
o: the user
L: the L parameter
$\mathcal{B}_0$: current boundary for candidate boxes
Output: $\mathcal{B}$: new boundary for candidate boxes
Procedure:
1: initialize $\mathcal{B} = P = o.b\_box.x_1$;
2: $d = w = 0$;
3: **while** P has not reached $\mathcal{B}_0$ **do**
4: move P to the next event;
5: $d = \mathcal{B}.x - L.x$;
6: $w = w + 1$;
7: **if** $L/2 * d < w * \mu$ **then**
8: $\mathcal{B} = P$; // move new boundary to P
9: $w = 0$; // reset w
10: return $\mathcal{B}$;

Algorithm 4 shows the complete procedure that finds Ω for a candidate set. The algorithm integrates both fast scan and plane-sweep pruning. The while-loop iteratively applies the plane-sweep algorithm to the x-axis and the y-axis. As for the time complexity, the plane-sweep algorithm costs only $O(|R|)$ in each iteration. Therefore, the dominant complexity still lies in line 4 where the fast-scan algorithm is invoked in a strip, but the cost is much lower as there are fewer events in the strip. Therefore, Algorithm 4 is expected to be more efficient than Algorithm 2.

5.2 Index-Based Pruning

In this subsection, we study pruning in the presence of a spatial index. Without loss of generality, we use an R-tree as the running example throughout this paper, although the same algorithm can be applied to any hierarchy-based index, such as quad-tree or k-d tree.

The key observation is that, for a given user o and a given node n in the R-tree, if a candidate box covers any event contained in n, the minimum size (i.e., the lowest *cost*) this box can achieve is when it just touches the bounding box of n. Furthermore, the touching point must be the closest point on n's box to this box.

Algorithm 4 Finding Ω in One Candidate Set

Input: R: the set of events
o: the user
Output: Ω: candidate box with the lowest c/s value
Procedure:
1: initialize boundaries $\mathcal{B}$;
2: **while** make strip AND strip spans the entire interior of $\mathcal{B}$ **do**
3: $\Omega = argmin_{c/s}\{fast\_scan(strip), \Omega\}$;
4: $\mu = c/s$ value of Ω;
5: $\mathcal{B} = plane\_sweep(o, L, \mu)$;
6: remove events in R that are outside $\mathcal{B}$;
7: **if** strip is the same as the last in any dimension **then**
8: $\Omega = argmin_{c/s}\{fast\_scan(\mathcal{B}), \Omega\}$;
9: return Ω;
10: return Ω;

On the other hand, the highest $|S - C|$ value this box can achieve for node n is the number of events in n, denoted by $|n|$. Therefore, a lower bound of the c/s value for such a candidate box is obtained by imagining all events in n to concentrate at the touching point, and we call it a *super-event*.

DEFINITION 4. *A* super-event *corresponds to a node n in the spatial index of event dataset R. It is located at the point where the bounding box of node n is closest to that of user o. If the super-event is covered, all events in node n are covered.*

DEFINITION 5. *A* super-event dataset *R' of event dataset R consists of events E_i that are either an event or a super-event in R, and furthermore, R' is equivalent to R in terms of the events they contain; that is, $\cup_i E_i = R$ and $E_i \cap E_j = \emptyset$, $\forall i \neq j$.*

The aforementioned observation on a single super-event can be generalized to a super-event dataset: the lowest c/s value of a candidate box that covers event(s) in R is bounded by (i.e., always larger than) the lowest c/s value of a candidate box that covers event(s) in R' . The following lemma proves the correctness of this generalization.

LEMMA 4. *Let Ω denote the candidate box with the lowest c/s value of event set $R = \{e_1, e_2, ..., e_k\}$, and let Ω' denote the candidate box with the lowest c/s value of super-event set $R' = \{E_1, E_2, ..., E_l\}$ of R. Then $c/s(\Omega) \geq c/s(\Omega')$.*

PROOF. Prove by mathematical induction.

1. Assume that R' contains only one element E^*; then there are two cases about Ω: (i) Ω covers no event in E^*; or (ii) Ω covers some or all event(s) in E^*. For case (i), Ω is also a candidate box for R' with the same c/s value. Since Ω' is the one with the lowest c/s value, we have $c/s(\Omega) \geq c/s(\Omega')$. For case (ii), Ω is a candidate box for R' with a c/s value less than $c/s(\Omega)$ because now it covers all events in E^*. Since Ω' is the one with the lowest c/s value, we have $c/s(\Omega) \geq c/s(\Omega')$.

2. Assume the lemma holds when R' contains k elements; then if now there is one more element $E^* \in R'$, there are two cases about Ω: (i) Ω covers no event in E^*; or (ii) Ω covers event(s) in E^*. From (1), we know that in either case, $c/s(\Omega) \geq c/s(\Omega')$. Therefore, this lemma holds when R' contains $k + 1$ elements.

From (1) and (2), we conclude that the lemma holds in any case. □

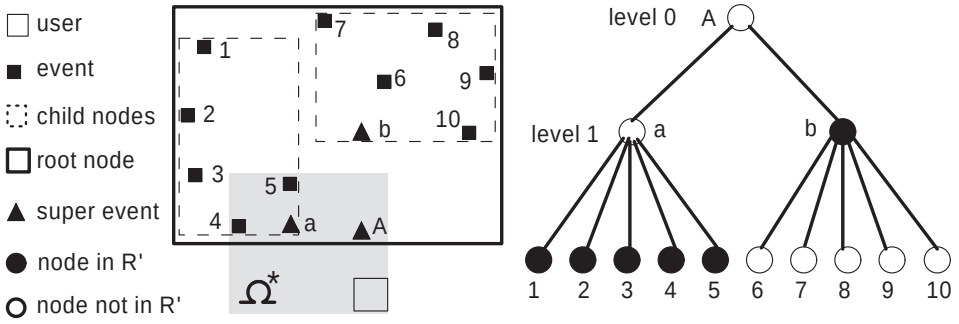


Fig. 7. Index-Based Pruning

Lemma 4 indicates that by computing Ω', which is significantly faster than computing Ω due to fewer events, we obtain a lower bound of the c/s value for Ω. For each user o, we can obtain a lower bound Ω'_o for its Ω_o, and intuitively Ω_* is more likely to come from the user with the lowest bound value. Furthermore, since the greedy algorithm only needs Ω_*, once the lower bound of a user is higher than the c/s value of the current best box, Ω_* can no longer come from this user, who can thus be safely pruned. Thus, the index-based pruning is summarized as follows.

Pruning Condition 3: Find Ω_* for the user whose c/s value of Ω'_o is the lowest. Prune those users whose c/s values of Ω'_o are higher than that of the current best candidate box.

In essence, index-based pruning uses as many super-events (i.e., intermediate R-tree nodes) as possible and uses as high-level tree nodes as possible. To achieve this, during the execution of the greedy algorithm, we use a priority queue to store triples of $<o, R'_o, \Omega'_o>$, where R'_o is the super-event set based on which Ω'_o is computed. The key to sort in the priority queue is the c/s value of Ω'_o. Initially, all users start from the root node r of the index; that is, $R'_o = \{r\}$ for any o, and $\Omega'_o = o.b\_box$. Then the algorithm pops up the top $<o, R'_o, \Omega'_o>$ triple and breaks all super-events in R'_o contained by Ω'_o into "smaller" super-events by replacing each such node with its child nodes. The Ω'_o of this new R'_o is then computed and a new triple $<o, R'_o, \Omega'_o>$ is inserted back in the queue with the updated key value.

When the Ω'_o of the popped-up triple contains no super-event in it, it is converged to Ω_o. Since its c/s value is the lowest in the priority queue, this Ω'_o can be returned as Ω_*. Figure 7 illustrates the execution of this algorithm. Initially, $R'_o = \{A\}$. Later, when this triple pops up from the priority queue, A is broken into super-events a and b; that is, $R'_o = \{a, b\}$. Then next time this triple pops up, super-event a is broken into events $1, 2, 3, 4$, and 5, $R'_o = \{1, 2, 3, 4, 5, b\}$; and Ω'_o is computed as the grey box. Next time this triple pops up again, Ω'_o contains no super-event, so it converges to Ω_o and is returned as Ω_*.

Furthermore, a spatial index usually preserves the spatial locality well, that is, events that are close to each other are likely to reside in the same subtree. In addition, the closer they are, the deeper the root of this subtree can be. As such, a bottom-up manner of finding Ω'_o in the super-event set R'_o can do further pruning by leveraging this same R-tree. The idea is to start finding Ω'_o in one of the lowest-

level subtrees (i.e., R-tree leaf nodes) and continues to move upwards the R-tree if this Ω'_o cannot be guaranteed for the entire super-event set R'_o. Since events that are closer to user o should be accessed earlier, we should pick the initial subtree as the R-tree node that is the closest to o. The right-hand side of Figure 7 illustrates this idea when $R'_o = \{1, 2, 3, 4, 5, b\}$ (the black dots). Ω'_o is first computed with (super-)events in node a (level 1) only, i.e., $\{1, 2, 3, 4, 5\}$. Then the plane-sweep algorithm (Algorithm 3) is invoked to test if this Ω'_o is also for the entire R'_o. The test is passed if the new boundary found by the plane-sweep algorithm is completely contained by node a. Otherwise, a new Ω'_o will be computed with (super-)events in the parent node of a, i.e., A, and the same test will be conducted. The correctness is guaranteed because ultimately, the search for Ω'_o will reach the root node, where the set of (super-)events becomes R'_o itself.

Algorithm 5 Complete Local Enlargement Algorithm

Input: r: root node of the R-tree index
k: coverage requirement
Output: $\mathcal{S}$: the selected subsets
Procedure:

```
1: initialize S = C = ∅;
2: initialize priority queue H with <o, {r}, o.b_box> for all users;
3: while C ≠ U do
4:    <o, R'_o, Ω'_o> = H.pop();
5:    if Ω'_o does not overlap with any super-event in R'_o then
6:       S = S ∪ Ω'_o;
7:       for each element e in Ω'_o do
8:          if e is covered k times then
9:             C = C ∪ {e}; // remove e
10:            update each Ω' in H that covers e;
11:   else
12:      replace nodes contained by Ω'_o with their child nodes;
13:      node n = the deepest subtree that is the closest to o;
14:      while Ω'_o is not for R'_o do
15:         Ω'_o = find_Ω(o, R'_o ∩ n);
16:         n = n.parent;
17:      insert <o, R'_o, Ω'_o> into H;
18: return S;
```

Finally, the pseudo-code of our local enlargement algorithm that integrates all pruning techniques is listed in Algorithm 5. Like Algorithm 1, the algorithm terminates when $C = U$; that is, when all events are fully covered. During the execution, when an event that has been fully covered is added to C, the algorithm needs to recompute those Ω'_o's in the queue that cover this event and update their key values. Note that as well as line 15, Algorithm 4 is invoked implicitly in line 10.

6. COST MODELS OF PRUNING ALGORITHMS

In this section, we study the optimal value of L in the plane-sweep pruning (i.e., Algorithm 4), as well as its pruning power. Obviously, the objective is to maximize α, the ratio of pruned candidate boxes to the total number of boxes in the candidate set. To simplify the analysis, we assume that the number of events covered by a

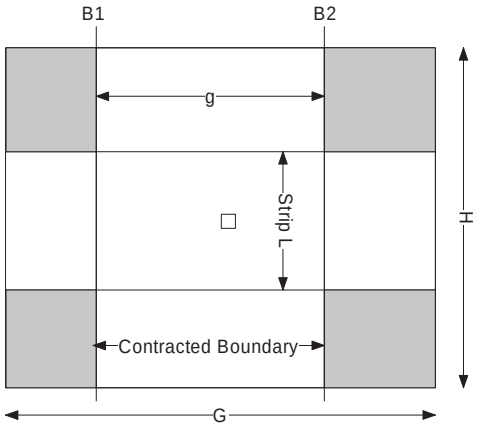


Fig. 8. One Iteration of Algorithm 4

candidate box is proportional to its area, and so is the cost of the box. This assumption is particularly reasonable when events are indexed by an R-tree, which groups them in such a way that they are distributed evenly in a node. Let β denote the ratio of events pruned by Algorithm 4 to the total number of events in R. The following lemma first shows that maximizing α is equivalent to maximizing β.

LEMMA 5. *Maximizing α is equivalent to maximizing β.*

PROOF. The pruning of β portion of events reduces β portion of coordinates for the candidate boxes. The remaining $(1-\beta)$ portion of coordinates only generates $(1-\beta)^2$ portion of candidate boxes. As such, the reduction of β area leads to a reduction of $2\beta - \beta^2$ candidate boxes, that is, $\alpha = 2\beta - \beta^2$. Since $2\beta - \beta^2$ is an ever-increasing function in the range $0 \leq \beta \leq 1$, maximizing α is equivalent to maximizing β. □

To derive β, we then analyze the area pruned by the plane-sweep algorithm. Figure 8 shows one iteration of Algorithm 4 where the current boundary of events is a G-by-H rectangle, and the L strip is horizontal. B_1 and B_2 are the resulting boundaries after this iteration. Therefore, the pruned area in this iteration is the four grey rectangles at the corners. Let $\mathcal{A}_i$ denote this area in the i-th iteration. Obviously, the total pruned area is the sum of $\mathcal{A}_i$ in all iterations, and maximizing β is equivalent to maximizing this sum. Furthermore, since the choice of L in each iteration solely depends on the settings in that iteration, we simplify the problem by finding the L that maximizes $\mathcal{A}$ in each iteration.

6.1 Derivation of L

According to Figure 8, $\mathcal{A} = (H-L)*(G-g)$, where only g depends on L. Figure 9 shows the execution of the plane-sweep algorithm where events are projected to the x-axis. Since events are assumed to be uniformly distributed, their projections should also be evenly distributed on the axis with interval $t = \frac{G}{N}$ (N is the cardinality of the event dataset). However, no area can be pruned under a perfectly uniform distribution, therefore, we model each projected event to have a probabil-

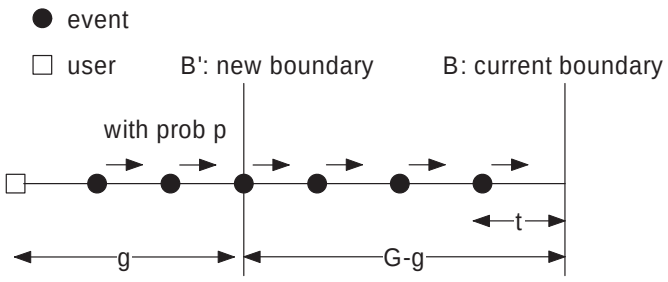


Fig. 9. Compute g

ity p of fluctuating from its position under uniform distribution. Without loss of generality, we assume that any fluctuation is rightward and does not change the order of projected events on the axis.

Then according to Algorithm 3, the plane-sweep pruning contracts the current boundary B to the farthest fluctuating event from the user and forms the new boundary B'. Figure 9 illustrates this contraction on the right-hand side of the user. The reason why all events on the right-hand side of B' can be pruned is that they do not fluctuate and therefore extending Ω to cover them only makes the low c/s value of Ω diluted by the high c/s value in this part. Then the probability of B' being at the event immediately to the left of B, denoted by $P(G-g=t)$, is p; likewise, the probability of B' being at the next left event, denoted by $P(G-g=2t)$, is $p(1-p)$; and so on. Therefore, the expected $G-g$ is

$$G-g=\sum_{i=1}^{N}p(1-p)^{(i-1)}it=\frac{G}{pN}[1-(1-p)^{N}(1+pN)]\approx\frac{G}{pN} \tag{1}$$

The above approximation is allowed because $pN>1$ (fluctuating events must exist) and $(1-p)^N\approx 1/(e^{pN})$ $(p\ll 1)$.

On the other hand, according to Algorithm 3, Equation (1) holds only if the following inequality holds.

$$L/2*d>w*\mu, \tag{2}$$

where $d=G-g$, $w=\frac{N(G-g)}{G}$, and μ is the c/s value of current Ω. To rewrite μ in terms of L and thus solve Inequality (2) about L, we first obtain the first derivative of μ with regard to L, $\mu'(L)$.

Without loss of generality, we set the width of the current Ω to G. Suppose that now a ΔL is added to this G-by-L strip; then two conditions must both hold in order for Ω to be extended to the enlarged strip and thus to have a lower μ. First, Ω must be adjacent to ΔL; that is, the height of Ω is L. Second, there must be an event in the enlarged G-by-ΔL area so that including it reduces μ. The probability of satisfying the first condition is $p(1-p)^{LN/H}$, using the same rationale as in Figure 9 where each event has a probability p of fluctuating. The probability of satisfying the second condition is $\frac{N\Delta L}{H}$ according to the assumption

of even distribution. Therefore,

$$\mu'(L) = \frac{d\mu(L)}{dL} = \frac{\mu(L+\Delta L)-\mu(L)}{\Delta L} = \frac{1}{\Delta L}\frac{G(L+\Delta L)-GL}{N\times\frac{L}{H}} * p(1-p)^{LN/H}$$
$$= \frac{\overline{\mu}}{L}p(1-p)^{LN/H} \approx \frac{p\overline{\mu}}{L}(1-pLN/H) = \frac{p\overline{\mu}}{L} - Gp^2, \quad (3)$$

where $\overline{\mu} = \frac{GH}{N}$ is the average c/s value in the strip. The above approximation is allowed because p is small and $LN/H \gg 1$. With Equation (3) and $\mu(H) = \overline{\mu}$, we obtain

$$\mu(L) = p\overline{\mu}\ln L - Gp^2L + C, \quad (4)$$

where $C = (Np^2 - p\ln H + 1)\overline{\mu}$. Using Equations (3) and (4), we can solve Inequality (2) as $L > L_0$, where L_0 is the solution to the following equation:

$$(\frac{\overline{\mu}}{H} + 2Gp^2)L = 2(p\overline{\mu}\ln L + C) \quad (5)$$

Finally, we conclude with the following theorem:

THEOREM 6. *The optimal L for the plane-sweep algorithm is the solution to equation $(\frac{\overline{\mu}}{H} + 2Gp^2)L = 2(p\overline{\mu}\ln L + C)$.*

6.2 Pruning Power

We continue to derive $\mathcal{A}_i$ and β, the pruning ratio of Algorithm 3. In the sequel, we use subscript i to denote the value of a parameter in the i-th iteration, for example, N_i, L_i; in particular, N_0, G_0 denote the initial values prior to the first iteration. With Equations (1) and $\overline{\mu} = \frac{GH}{N}$, we have

$$\mathcal{A}_i = (H_i - L_i) * (G_i - g_i) = \frac{G_i}{pN_i}(H_i - L_i) = \frac{\overline{\mu}}{p} - \frac{G_iL_i}{pN_i} \quad (6)$$

Let $\mathcal{B}_i$ denote the portion (in terms of area) of strip L to the initial unpruned area; that is, $\mathcal{B}_i = \frac{G_iL_i}{G_0H_0} = \frac{G_iL_i}{\overline{\mu}N_0}$. Then

$$\mathcal{A}_i = \frac{\overline{\mu}}{p} - \frac{\overline{\mu}\mathcal{B}_i}{p} = \frac{\overline{\mu}}{p}(1 - \frac{N_0}{N_i}\mathcal{B}_i), \quad (7)$$

Then we have

$$N_i - N_{i+1} = N_i * (G_i - g_i)/G_i = N_i/pN_i = 1/p,$$

which means the number of events pruned in each iteration is a constant; that is, $N_i = N_0 - i/p$. Summing up Equation (7) for all i with $N_i = N_0 - i/p$, the total pruned area $\mathcal{A}$ is

$$\mathcal{A} = \sum_i^I \mathcal{A}_i = \frac{\overline{\mu}}{p}(I - N_0\sum_{i=0}^{I}\frac{\mathcal{B}_i}{N_0 - i/p}),$$

where I denotes the number of iterations. And β is simply $\mathcal{A}$ divided by $\overline{\mu}N_0$

$$\beta = \frac{\mathcal{A}}{\overline{\mu}N_0} = \frac{I}{pN_0} - \sum_{i=0}^{I}\frac{\mathcal{B}_i}{pN_0 - i} \quad (8)$$

If we further assume L_i is proportional to H_i, that is, $L_i = \theta H_i$, we can obtain a closed form β from Equation (8), which is $\beta = \frac{I}{pN_0}(1-\theta)$. Finally, we have the following theorem on the pruning power of the plane-sweep algorithm (Algorithm 3).

THEOREM 7. *If strip L_i is proportional to height H_i, the ratio of pruned events, β, is proportional to I, $1/p$, and $1/N_0$, respectively.*

This theorem also backs our choice of L in Theorem 6 from a different aspect, because it shows that simply choosing an L proportional to H leads to an ever-decreasing pruning power β as the number of events (N_0) or fluctuated events (pN_0) increases. This trend is undesirable in practice when such a number is usually large.

6.3 Derivation of p

In previous cost models, we heavily use p, the probability of a projected event fluctuating away from its aligned location, that is, where it should be under a perfectly uniform distribution. However, in practice, every projected event deviates (even just a little) from its aligned locations, making $p = 1$. A possible remedy would be setting a threshold value for the deviation — only if the deviation is more significant than a threshold, is this event counted as fluctuating. However, this solution cannot handle scenarios where events are partially uniform in some local area but not globally, because aligned locations are calculated globally. In this subsection, we propose a more robust method to obtain p.

The idea is to calculate the deviation of a projected event from its local neighbors only, instead of the fixed aligned location. An event does not fluctuate if it is evenly located among its neighbors. To define *evenness*, we adopt the concept of a spatial index "quad-tree." In a 2D quad-tree, the space is recursively partitioned into four equal sub-spaces until there is no more than one event in the (sub-)space. Events in a denser area are indexed at a deeper level in the quad-tree, and vice versa. Furthermore, neighboring events can be accessed together, with an inorder traversal of the tree. Based on these properties of the quad-tree, we define an event as located evenly among its neighbors (i.e., in an area of uniform density) if it is indexed at the same level in the tree as its immediate next event during the inorder traversal.

The detailed algorithm to count fluctuating events is as follows. First, the events are projected onto the x-axis (or the y-axis if the strip is vertical). Then a 1D quad-tree is constructed to index them. Next the algorithm performs an inorder traversal of the quad-tree. Each tree node that indexes an event is compared with the first immediately following node that indexes an event: if they are of the same level, the former event is considered to be not fluctuating, and vice versa. As for the time complexity, the construction of a quad-tree costs only $O(n)$ time (n is the number of events); and the inorder traversal also costs $O(n)$ time because the quad-tree tree has at most $O(n)$ nodes. As such, this algorithm to obtain p only costs $O(n)$ time.

7. SECURITY ANALYSIS

In previous sections, we assumed that the adversary knows no more than the published location datasets of users and events. As such, the k-anonymity location publishing guarantees for each event, the probability of each covering user being

the genuine user equals to $1/k$. However, in reality the adversary may have other information sources about users and events, and even about the underlying generalization algorithms. The followings are several examples.

EXAMPLE 1. *User o cannot have made a purchase (event) in an ice-cream shop alone because he has diabetes.*

EXAMPLE 2. *User o must have made at least one purchase (event) as he drew money from an ATM.*

EXAMPLE 3. *User o, whose centroid is the closest to a purchase p (event), is the most likely to make this purchase, because some compromised user location dataset by the same generalization algorithm shows the genuine user tends to be closer to p than other covering users.*

Since such background knowledge is unknown to data publishers, it severely compromises the k-anonymity privacy. A formal and theoretical language that expresses and quantifies background knowledge about relational attributes was proposed by Martin *et al.* [Martin et al. 2007]. In addition, a more intuitive multidimensional formulation of similar background knowledge was proposed by Chen *et al.* [Chen et al. 2007]. In this section, however, we focus on the forms of background knowledge specific to our k-anonymity problem and study its privacy threat.

We assume the adversary possesses the published user location dataset $\mathcal{D}'$ (which already satisfies k-anonymity) and the reference event dataset R. The adversary's knowledge about the genuine users of events is modeled by an undirected and weighted graph $\mathcal{G}$, which is called the *knowledge graph*. The vertices in $\mathcal{G}$ are the W users in $\mathcal{D}$ and N events in R. There is an edge between two vertices in the graph if and only if: (1) one is a user and the other is an event, and (2) the user location covers the event. As such, the knowledge graph is a bipartite graph where users and events form two disjoint vertex sets. To simplify subsequent analysis, we assume every event is covered by exactly k users in $\mathcal{D}'$. The weight of this edge denotes the adversary's knowledge on the probability of this user being the genuine user of this event. If the adversary has no background knowledge, each edge has the same default weight, i.e., $1/k$. We differentiate three categories of background knowledge, namely, deterministic, constraint and probabilistic knowledge. The deterministic knowledge confirms or dismisses that a user is a genuine user of an event, or equivalently, sets the weights of corresponding edges to 0 or 1. On the other hand, the probabilistic knowledge sets the weights to other intermediate values. And constraint knowledge sets constraints for those events of which a specific user is the genuine user. Figure 10 shows an example of $\mathcal{G}$ where $\mathcal{D}'$ is a 3-anonymity dataset, $W = 5$ and $N = 3$. The adversary has deterministic knowledge that "o_3 did e_3 but not e_2", so the corresponding edge weights differ from the default edge weight 1/3.

A knowledge graph possesses several unique properties.

PROPERTY 8. *The degree of every vertex of event is k.*

PROPERTY 9. *The total number of edges is kN.*

PROPERTY 10. *The degree of every vertex of the user ranges from 0 to N.*

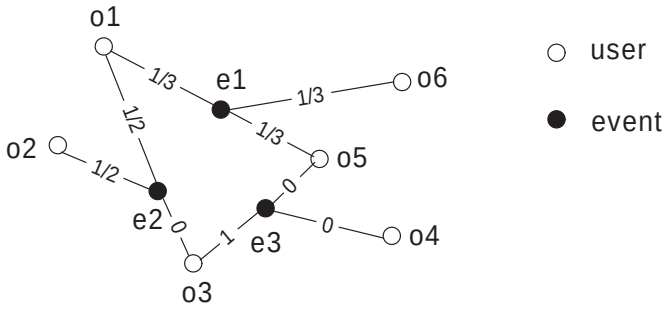


Fig. 10. An Example of a Knowledge Graph

PROPERTY 11. *The sum of weights of all edges that are adjacent to an event is 1.*

PROPERTY 12. *The girth, i.e., the number of edges of the shortest cycle, is at least 4.*

To quantify the adversary's gain of confidence through the background knowledge, we take a similar approach as in [Machanavajjhala et al. 2006] and adopt the *entropy* measure. More specifically, the entropy of an event is the uncertainty of its genuine user. Without background knowledge, each of the k covering users of an event has an equal probability of being the genuine user, so the entropy of this event is $k * \frac{1}{k} \log k = \log k$. The entropy of the knowledge graph is the sum of entropy of all events. In what follows, we study how the background knowledge affects the entropy of the knowledge graph.

7.1 Deterministic Knowledge

Without loss of generality, we categorize a piece of deterministic knowledge into two types, event-based knowledge and user-based knowledge, depending on whether it originates from an event or a user. For example, "event e must be done by user o" is event-based, and "user o must have done events e_1 and e_2" is user-based. A piece of event-based (or user-based) knowledge can further be categorized into *confirmative*, which confirms *genuine* edge(s), for example, "user o must have done event e", or *dismissive*, which dismisses *false* edge(s), e.g., "user o cannot have done event e."

7.1.1 *Event-Based Knowledge.* If the adversary has a piece of confirmative event-based knowledge, the loss of entropy is the entire entropy of this event, which is $\log k$. If the adversary has a piece of dismissive knowledge of m $(1 \leq m \leq k-1)$ edges, the loss of entropy is $\log k - \log(k-m)$. In both cases, the loss is independent of the k-anonymity algorithm used for $\mathcal{D}'$. Furthermore, the maximum entropy loss for a single event is $\log k$, when the genuine user is confirmed.

7.1.2 *User-Based Knowledge.* If the adversary has a piece of confirmative user-based knowledge of m edges, the loss of entropy is the entropy of the m events, which is $m \log k$. If the adversary has a piece of dismissive knowledge of m edges, the loss of entropy is the sum of the entropy loss of the m events, which is $m(\log k - \log(k-1))$.

Furthermore, the maximum entropy loss for a single user occurs when all edges adjacent to this user are either confirmed or dismissed. The loss of entropy is the sum of the entropy loss of each adjacent event, i.e., $c*\log k+d*(\log k-\log(k-1))$, where c and d are the numbers of edges confirmed and dismissed, respectively. By summing up this value over all users, we derive the maximum entropy loss for an average user as

$$\begin{aligned}&\frac{1}{W}\sum_{i=1}^{W}[c_i*\log k+d_i*(\log k-\log(k-1))]\\=&\frac{1}{W}[N*\log k+(kN-N)*(\log k-log(k-1))].\end{aligned}$$

From the above, the entropy loss increases as the number of users decreases or the number of events increases. In addition, this value is independent of the k-anonymity algorithm.

7.2 Constraint Knowledge

Besides confirming or dismissing a specific edge adjacent to a user, the adversary may know some constraints on the events that this user has carried out. Specifically, the adversary may know the maximum (M) or minimum (m) number of events. For example, credit card records may show that a user has used his credit card, so the minimum number of purchase events is $m = 1$. In this subsection, we study this special form of constraint knowledge, which is called user-based max-min knowledge.

With max-min knowledge, different events are correlated and so are different users; the privacy compromise of one event (user) may has chain effects that propagate to the rest part of the graph. For example, in Figure 10, if the m values of users o_1 and o_2 are both 1, then o_2 must carry out e_2 as this is the only event adjacent to o_2. This deduction in turn infers that o_1 carries e_1.

To analyze the entropy loss of user-based max-min knowledge, we make the following simplifications:

— The m and M values of each user are the same, and they are much smaller than k.

— Each edge has p_+ probability of being confirmed and p_- probability of being dismissed by the adversary.

— Both p_+ and p_- values are small; that is, the adversary has fair but not significant min-max knowledge.

Our objective is to prevent any event from being compromised, i.e., from losing all its entropy. To compromise an event, the adversary must know either that there are $k-1$ adjacent users each of whom already confirms M events, or there is one user who already dismisses $s-m$ adjacent edges, where s is the degree of this user. To reduce the probability of the former case, we must reduce the probability of an arbitrary user i confirming M events out of s_i-1, which is $\binom{s_i-1}{M}p_+^M(1-p_+)^{(s_i-1-M)} \approx \frac{(s_ip_+)^M}{M!}$, assuming $s_i \gg m > 1$. Taking the average over all users, this probability is

$$\frac{1}{W}\sum_i\frac{(s_ip_+)^M}{M!}=\frac{p_+^M}{WM!}\sum_i {s_i}^M.$$

Since $\sum_i s_i = kN$, $\sum_i {s_i}^M$ achieves the lowest value when $s_i = kN/W, \forall i$. That is, the degrees of users should be as even as possible; or in other words, in the k-anonymity location dataset $\mathcal{D}'$, each user should cover the same number of events.

To reduce the probability of the latter case, we must reduce the probability of an arbitrary user i dismissing exactly $s_i - m$ edges out of $s_i - 1$, which is $\binom{s_i - 1}{s_i - m} p_-^{(s_i - m)}(1 - p_-)^{(m-1)} \approx (1 + \frac{m^2}{s_i}) p_-^{s_i - m}$, assuming $s_i \gg m > 1$. Taking the average over all users, this probability is

$$\frac{1}{W} \sum_i (1 + \frac{m^2}{s_i}) p_-^{(s_i - m)} = \sum_i p_-^{(s_i - m)} + m^2 \sum_i \frac{p_-^{(s_i - m)}}{s_i}.$$

Both terms achieve the minimum value when $s_i = kN/W, \forall i$. Therefore, the conditions to minimize the probabilities in both cases are consistent, which is formally shown as the following theorem.

THEOREM 13. *The probability of a privacy compromise from user-based max-min knowledge is minimized when all users cover the same number of events.*

Our previous k-anonymity algorithm (Algorithm 5) only minimizes the generalization cost and cannot guarantee even distribution of event coverage. As such, we propose a post-processing algorithm which aims at adjusting the event coverage difference among all users. Algorithm 6 shows the pseudo-code of this algorithm. It maintains two priority queues, Q_H and Q_L, of users with higher and lower coverage than kN/W (the average coverage), respectively. In each iteration, the algorithm selects the top user o with the highest coverage from Q_H, and selects another user o' from Q_L to cover the farthest event e that is currently covered by o. Consequently, the bounding box of o shrinks to "uncover" e while that of o' expands to cover e. To reduce the generalization cost, this algorithm selects o' from Q_L that incurs the lowest increment of cost. It is noteworthy that such o' may not exist if all users in Q_L already cover e, in which case o is skipped from adjustment. Since the sizes of both Q_H and Q_L are non-increasing in each iteration, the algorithm is guaranteed to terminate when Q_H becomes empty.

7.3 Probabilistic Knowledge

Besides the specific knowledge regarding individual users or events, an adversary may learn some distinguishing feature of the genuine user among all covering users of an event. For example, the genuine user tends to locate closer to the event than others. Such knowledge is called "probabilistic knowledge" because the adversary can leverage this to speculate the genuine user with a probability higher than the default $1/k$. In practice, probabilistic knowledge may be deduced from information about the input dataset (e.g., user locations are points or in fixed precision), the property of a specific generalization algorithm (which is known to the public), or from those compromised events (whose genuine users are known).

In this paper, we model the adversary's probabilistic knowledge by a feature metric M of user. Typical examples of M include: (1) the distance between the centroids of the user and a covered event, (2) the size of the generalized user location, and (3) the number of covered events. The M distribution (frequency) of

Algorithm 6 Coverage Adjustment

Input: $\mathcal{D}'$: the k-anonymity user location dataset by Algorithm 5
R: the event dataset
Output: $\mathcal{D}'$: the updated dataset
Procedure:

```
1: compute event coverage of each user in D';
2: build priority queue Q_H of users with coverage > kN/W;
3: build priority queue Q_L of users with coverage < kN/W;
4: while Q_H is not empty do
5:    o = Q_H.pop();
6:    find e as the farthest event covered by o;
7:    find o' ∈ Q_L that incurs the lowest cost increment to cover it;
8:    if such o' exists then
9:       expand o' to cover e;
10:      shrink o to un-cover e;
11:      if the coverage of o > kN/W then
12:         Q_H.push(o);
13:      if the coverage of o' < kN/W then
14:         Q_L.push(o');
15: return D';
```

all users, denoted by $freq$, is already available to the adversary because the generalized user location dataset $\mathcal{D}'$ is published. The M distribution (frequency) of *genuine* users (denoted by $freq'$), on the other hand, is the adversary's probabilistic knowledge.

With probabilistic knowledge, the adversary can compute the probability of a user o being the genuine user of an event e by the Bayesian theorem. Recall that Bayesian theorem computes the posterior probability $P(H|E)$ of a hypothesis H given evidence E as follows.

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)},$$

where $P(E|H)$ is the conditional probability of seeing E if H is true, and $P(H)$ and $P(E)$ are the *a priori* probability of H and E, respectively. Consider evidence E as an observation of a covering user o having metric value $M(o)$, and consider hypothesis H as o is the genuine user of event e. Then the posterior probability $P(H|M(o))$ is

$$P(H|M(o)) = \frac{freq'(M(o))P(H)}{freq(M(o))} = \frac{freq'(M(o))}{k \cdot freq(M(o))} \tag{9}$$

For event e, the adversary can compute the posterior probability for each covering user o by Equation (9). Since this probability represents the adversary's knowledge of o being the genuine user of e, it is assigned as the weight of the corresponding edge in the knowledge graph. Note that a scaling factor will be appended so that the sum of weights for each event is normalized to 1.

From Equation (9), the adversary can speculate the genuine user as the one with the highest $freq'(M(o))$ to $freq(M(o))$ ratio. Furthermore, the overall loss of entropy due to this probabilistic knowledge depends only on this ratio. Specifically, the more widely this ratio distributes, i.e., the more $freq'(M(o))$ deviates

from $freq(M(o))$, the more the entropy loss is. This observation suggests that a secure and robust generalization algorithm should exhibit no feature metric whose distribution among all users differs significantly from that among the genuine users.

Since an exhaustive enumeration of all feature metrics is impossible, in what follows we focus on the three typical metrics mentioned above, namely, the distance between the centroids of the user and a covered event, the size of the generalized user location, and the number of covered events. As a counterexample, a naive k-nearest neighbor (KNN) generalization algorithm which always covers each event with the k-closest user locations has biased metric distributions of its genuine users. Specifically, the genuine user is closer to the event (because farther-away users are more likely to be KNNs that are chosen by the algorithm); the genuine user also covers fewer events (because it is less likely to be close to other events) and thus has a smaller size.

However, we argue that our local enlargement algorithm does not exhibit any biased distributions of these metrics: while KNN considers each event independently, our algorithm considers all events as a whole. Specifically, the genuine user of an event is still equally likely to be enlarged to cover other events, as long as this enlargement leads to the minimum c/s value. While this directly explains the metrics of user location size and number of covered events, the metric of centroid distance follows the same rationale. When the genuine user of an event is enlarged to cover other events, its centroid moves away from this event. By contrast, some other covering user of this event may have its centroid move towards this event by being enlarged to cover more events in the same direction. As such, the genuine user has no bias in terms of its centroid distance to the event. These arguments are verified in Section 8.2, where the distributions of the three metrics among both the genuine users and all covering users are shown.

Besides the generalization algorithm, information about the input dataset is also a cause of biased distributions of M metrics. As an extreme example, imagine there is only one event and input user locations are known to be points. Then the adversary can exclude any user whose generalized location is not a point to be the genuine user. Such biased distribution might be alleviated by a generalization algorithm that appends a postprocessing step to disguise any unenlarged user location by a random enlargement. Fortunately, in practice it is hard for the adversary to know all input user locations, which have different sizes and depend on factors such as the precision of user positioning device and user privacy preference. The unknownness of the input dataset, in turn, alleviates the biased distribution caused by the generalization algorithm. For example, even if the algorithm alone leads to a biased distribution on the generalized location size, by combining the fact that the input location size is unknown to the adversary, he/she will have less confidence on attributing the biased distribution to the algorithm alone.

8. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed local enlargement (denoted by *Local*) and coverage adjustment algorithms (denoted by *Adjust*) under various datasets and parameter settings. To conduct a comparative study, we design several non-naive competitive algorithms for *Local*. The first is based on *Incognito*,

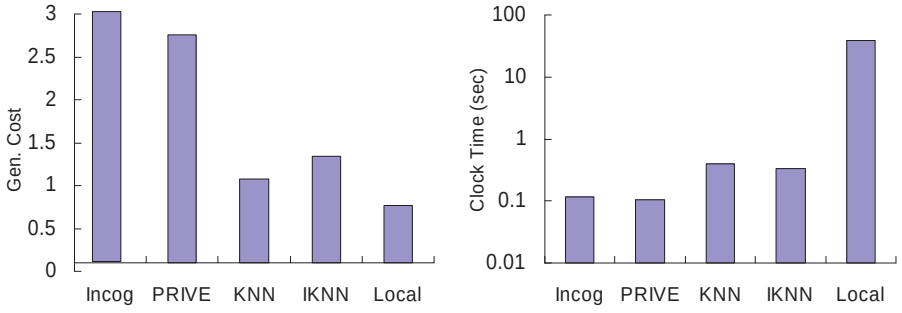
proposed by LeFevre *et al.*, the state-of-the-art k-anonymity framework on a single data table [LeFevre et al. 2005; LeFevre et al. 2006]. We adopt their latest recursive greedy partitioning algorithm Mondrian [LeFevre et al. 2006] that splits a partition at the median value in one preferred dimension until the partition is empty or has less than k users in it. Note that this algorithm is multi-dimension recoding, so it achieves a fairly low generalization cost in RDBMS. The second competitor is the $PRIVE$ framework, which is the latest k-anonymity partition algorithm for spatial databases [Ghinita et al. 2007b]. The $PRIVE$ framework uses a Hilbert curve to sort the users and then groups them into partitions, each of which contains k users. Note that both *Incognito* and $PRIVE$ are designed for only a single dataset, i.e., the user location dataset. To be fair, we adapt them to take advantage of the event dataset as well — only the partitions that contain event(s) need to be anonymized. The third and fourth competitors handle both datasets; in particular, they follow the same local enlargement paradigm as *Local*. They differ from *Local* in what user locations to enlarge to cover each event with at least k users. Both the k-nearest neighbor (KNN) and incremental KNN (IKNN) algorithms enlarge the k-closest user locations to cover each event. "Incremental" means the enlargement is effective immediately before the next event is considered and its kNN is searched. As such, IKNN is sensitive to the processing order of events, and in order to be neutral, we adopt the natural order, namely, their sequence numbers.

The main performance metrics are the total generalization cost of the published dataset and the clock time for running the algorithms. Two cost functions are used in the experiments, namely, LNR and QDR, which are respectively linear and quadratic to the area of a generalized location. More specifically, LNR equals to the area and QDR equals to the square of the area. QDR is used to emulate and approximate the scenarios where enlarging the location significantly reduces the utility of the published dataset. There are quite many applications where this assumption holds, particularly in data mining.

Since no real user location datasets are publicly available, we constructed synthetic datasets for both user locations and events using the state-of-the-art *Network-based Generator of Moving Objects* [Brinkhoff 2002]. This simulator is known to imitate the movement of automobiles and has been widely used in studies on real user movement. The road network fed into the generator was the Kowloon City area of Hong Kong, which is approximately a 2km-by-2km square with 115 main road segments. We randomly identified a maximum of 10,000 user locations, each of which was a 10m-by-10m square. We also obtained a maximum of 5,000 events from the same road network. These locations were the shops, telephone booths, and bus stops, and so on, where users may make purchases or other financial transactions. Like the user locations, each event also occurred within a 10m-by-10m square, and was initially covered by at least one user location. Without loss of generality, we randomly assigned one of these covering users as the genuine user. To achieve a fair comparison, all algorithms could access the R-tree index on both datasets. The testbed was implemented using Java (JDK 1.5) and all experiments were run on a desktop computer running Windows XP SP2 with an Intel Pentium 4 3.0GHz CPU and 512 MB memory. Table I summarizes the parameter settings for the datasets and experiments.

Parameter	Symbol	Default Value	Maximum Value
# of users	W	1,000	10,000
# of events	N	1,000	5,000
anonymity requirement	k	5	20
cost function	*cost*	LNR	

Table I. Simulation Parameters Settings



(a) Generalization Cost (b) Clock Time

Fig. 11. Overall Comparison

8.1 Overall Evaluation

Figures 11(a) and 11(b) show the generalization cost and clock time of all algorithms. Regarding the cost, we observe that: (1) both *Incognito* and $PRIVE$ have far higher costs than the others, amounting to over 6 (only shown partially in the figure) and 2.5, respectively, which confirms that k-anonymity on a single database is inefficient in solving this problem, even for multi-dimension recoding algorithms; (2) the *Local* algorithm leads on the cost and outperforms the first runner-up — KNN — by 30%; (3) KNN outperforms IKNN, which means that making the KNN algorithm incremental may not further improve its performance as we expected, probably because IKNN heavily relies on the processing order of events. As for the clock time, although *Local* is about two orders of magnitude higher than the others, the running time is still within 30 seconds, which is reasonable for offline and even realtime data publishing.

8.2 Security Test

In this experiment, we test whether these generalizing algorithms can sustain security threats that arise from the probabilistic background knowledge (see Section 7.3). Specifically, we compare the distributions of three feature metrics for all covering users and genuine users only, and show how their difference helps the adversary to speculate the genuine user from all covering users of an event.

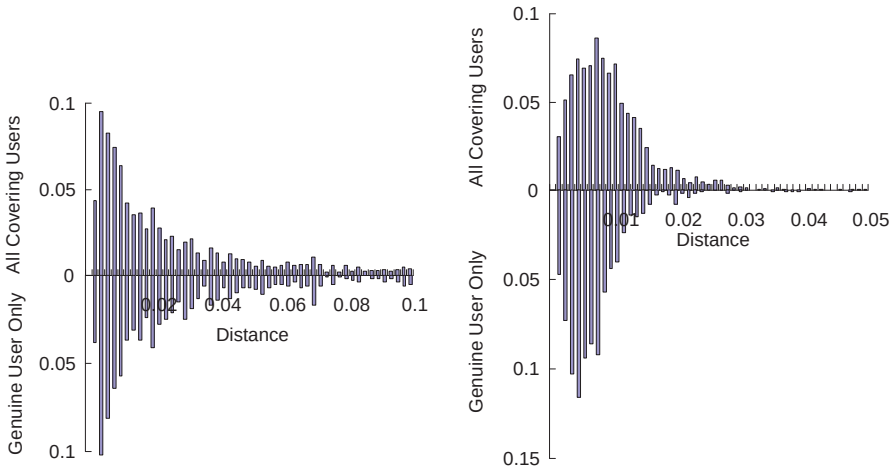
Figure 12(a) plots the distribution of the first metric for all covering users under *Local* algorithm. The metric is the distance between the centroids of a user and a covered event. The x-axis is the distance and the y-axis is the occurrence frequency. As a comparison, the lower half of the figure plots the distribution of the same metric for genuine users *only*. The symmetry of these two plots with respect to the x-axis confirms that *Local* does not have a biased distribution of centroid distance for the

genuine users. On the other hand, Figure 12(b) plots the distributions under the KNN algorithm, and we observe a distinct dissymmetry between the upper and lower distributions. In particular, the centroid distance of the genuine user tends to be shorter than the other covering users. This coincides with our analysis in Section 7.3. The distributions of the second metric, the areas of generalized user locations, are plotted in Figures 13(a) (under *Local* algorithm) and 13(b) (under KNN algorithm). *Local* has a slightly biased distribution for the genuine users only when the areas are extremely small ($\leq 0.2 \times 10^{-4}$). On the other hand, KNN exhibits a more heavily biased distribution across a wider range (0-0.5$\times 10^{-4}$), which shows the genuine user tends to have a smaller location size. This also coincides with our analysis in Section 7.3. Figures 14(a) and 14(b) plot the distributions of the third metric, i.e., the number of covered events, under the *Local* and KNN algorithms, respectively. Both algorithms show certain bias in the distributions of genuine users. However, under KNN, the genuine user has consistently fewer covered events than other covering users, because at $x = 1, 2$ and 3, the frequencies of the genuine users are consistently higher than the frequencies of all covering users. On the other hand, *Local* algorithm does not exhibit this property; in fact, only at $x = 2$ and 3 are the frequencies of the genuine users higher.

The above interpretation is reinforced by the success rates of the three corresponding attacks designed to exploit the biased metric distributions. To study the worst-case privacy loss, we assume the adversary has the complete knowledge of the three metric distributions for genuine users. For each event, the adversary computes the posterior probability of each covering user being the genuine user by Equation 9, and speculates the genuine user as the one with the highest probability. Figure 15 shows the success rates of attacks on the centroid distance metric (Attack 1), area metric (Attack 2) and number of covered events metric (Attack 3) under KNN, IKNN, and *Local* algorithms. *Local* consistently achieves the lowest rates, which range from 20% to 25%. Since $k = 5$, such rates are close to the success rate of a random guess, which is 20%. This shows *Local* algorithm is almost immune to these attacks and has little probabilistic knowledge to be exploited. In addition, Attack 2 has the highest rate because *Local* exhibits a bit more bias in the area metric than in the centroid distance, and the bias in the covered events is not consistently leftwards. KNN and IKNN suffer from higher success rates, especially under Attack 1. This can be explained by the wild bias in the distribution of centroid distance metric for the genuine users under KNN algorithm, as shown in Figure 12(b).

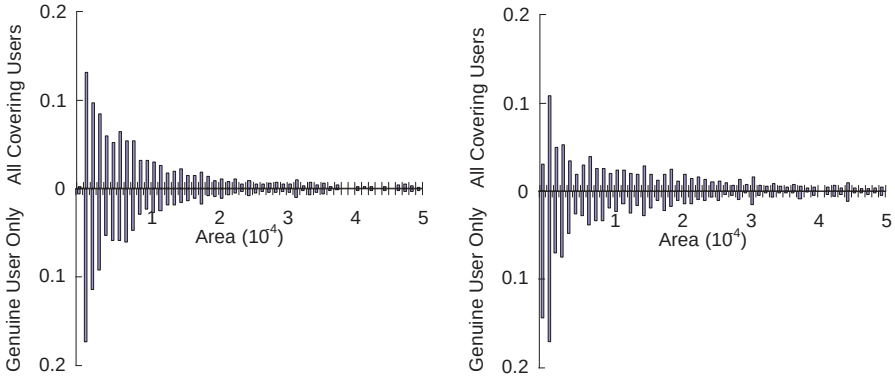
8.3 Effect of Cost Function

In this experiment, we change the cost function from LNR to QDR and the results are shown in Figures 16(a) and 16(b). The *Local* algorithm for QDR runs about four times faster than for LNR, while the other algorithms make no differences. This is because *Local* has strong pruning capability; therefore, while the cost function is QDR, the c/s values of larger candidate boxes increase significantly and thus are more vulnerable to pruning. In addition to speed boost, the cost gain of *Local* over the first runner-up algorithm is also more significant for QDR (by 70%) than for LNR (by 30%). This is due to the fact that only the *Local* algorithm aims at minimizing the generalization cost while the others follow heuristics that are



(a) Local (b) KNN

Fig. 12. Distribution of Centroid Distances between Covering Users and Events



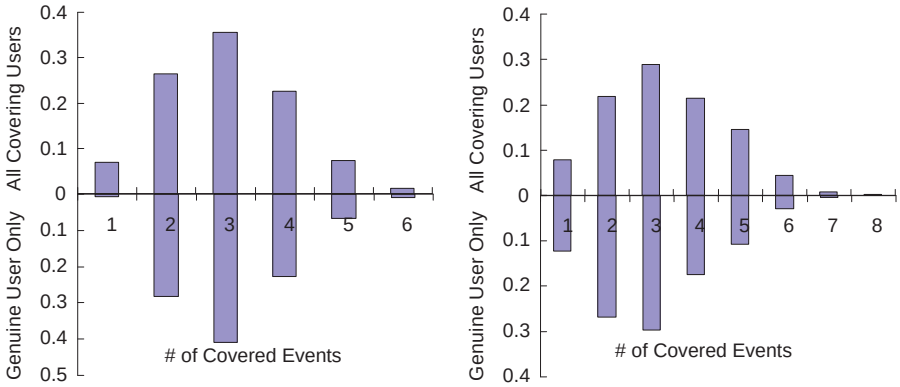
(a) Local (b) KNN

Fig. 13. Distribution of Areas of Covering Users

oblivious of whether the cost function is LNR or QDR. Both metrics consistently show that the *Local* algorithm performs even better with cost functions that are hyperlinear to the area. Such hyperlinear cost functions are common and useful in practice; for example, to suppress super-large user locations or totally eliminate oversized ones.

8.4 Effect of k

We vary the privacy requirement k from 1 to 20 and the results are shown in Figures 17(a) and 17(b). As k increases, the clock time for *Incognito* and $PRIVE$ decreases, because they are single-dataset-based k-anonymity algorithms, and higher k means fewer partitions. However, this is at the cost of increasing the generalization cost, and even worse, the cost gap with other algorithms also increases. On



(a) Local (b) KNN

Fig. 14. Distribution of Covered Event Numbers of Covering Users

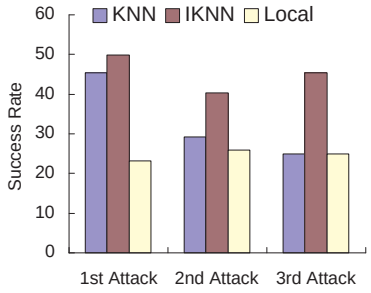
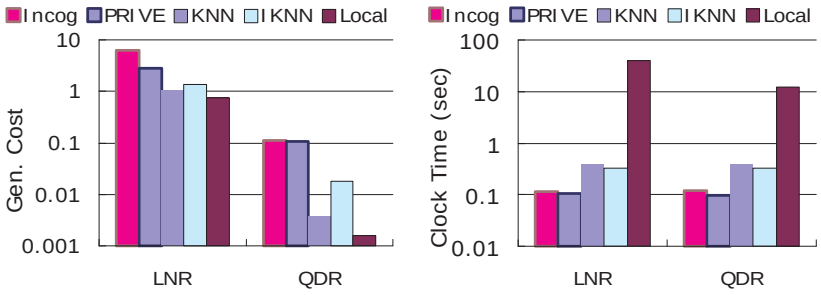
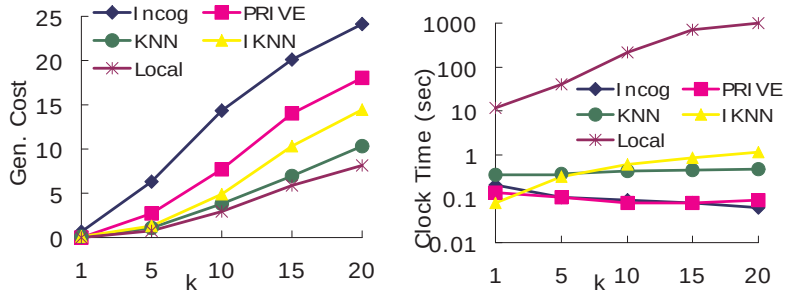


Fig. 15. Success Rates of Privacy Attacks



(a) Generalization Cost (b) Clock Time

Fig. 16. Comparison with Respect to Cost Function



(a) Generalization Cost (b) Clock Time

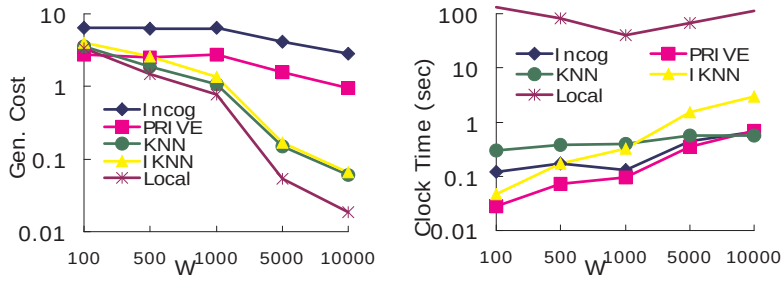
Fig. 17. Comparison with Respect to k

the other hand, KNN, IKNN, and *Local* require more clock time when k increases. In particular, the clock time for *Local* increases fast at first but tends to reach the saturation point afterwards, making its overall increasing rate similar to that of IKNN. The reason for this is that as the *Local* algorithm continues to run to achieve higher k-anonymity, the Ω's that pops up from the priority queue usually cover more events than earlier. KNN, IKNN, and *Local* maintain a relatively low generalization cost. In particular, *Local* outperforms the others even more significantly when k increases, which suggests that heuristics like KNN or IKNN may not scale well for larger k. Therefore, we can conclude that *Local* is a robust and low-cost algorithm under a wide range of k.

8.5 Scalability

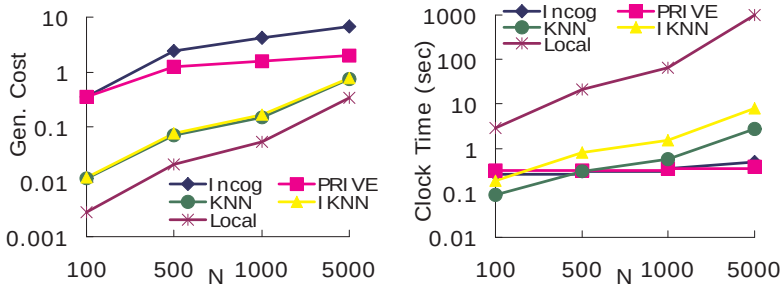
In this experiment, we evaluate the effect of increasing W (number of users) and N (number of events), respectively. Figures 18(a) and 18(b) show the comparison result when W increases from 100 to 10000 (fixing $N = 1000$). The generalization cost of our *Local* algorithm consistently outperforms the others in all W settings. Moreover, the trend of the curve shows that its performance gain over the others becomes even higher for larger W. This suggests that with more user locations available, *Local* algorithm still chooses them wisely to cover the events while heuristics like KNN or IKNN become less effective. As for the clock time, while all other algorithms increase monotonously as k increases, the *Local* algorithm decreases at first and increases afterwards. This is due to the fact that when W is small, each user location must cover more events, and computing Ω for larger candidate boxes is more time consuming. Interestingly, the turning point of this curve is at $W = 1000$, which coincides with N. This means that the *Local* algorithm performs best when the cardinality of two datasets is comparable.

Figures 19(a) and 19(b) show the comparison result when N increases from 100 to 5000 (fixing $W = 5000$). The generalization cost of *Local* is the least among all the algorithms. Moreover, the trend of the curve shows that the performance gain is still steady when N increases. The clock time shows a similar trend: although *Local* requires the highest clock time, the gap with KNN and IKNN is stable, suggesting that these three algorithms are almost linear to N. For the *Local* algorithm, this is a significant improvement over the asymptotic complexity $O(N^4(N + W))$



(a) Generalization Cost (b) Clock Time

Fig. 18. Comparison with Respect to W



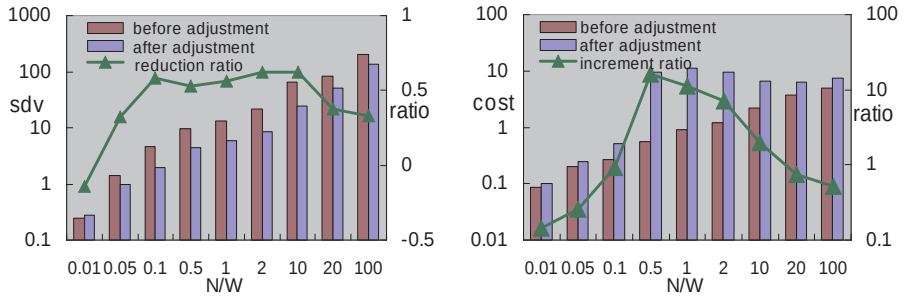
(a) Generalization Cost (b) Clock Time

Fig. 19. Comparison with Respect to N

(see Section 4.3), and indirectly verifies the effectiveness of the proposed pruning algorithms. On the other hand, the clock time of *Incognito* and *PRIVE* is almost invariable because the partition is performed on the user location dataset, and increasing N only slightly increases the number of partitions. Nonetheless, their generalization costs are consistently worse than those of the other three algorithms. Therefore, we can conclude that *Local* is a robust and scalable generalization algorithm for medium and reasonably large datasets.

8.6 Evaluation of Coverage Adjustment

In this experiment, we evaluate the effectiveness of the coverage adjustment algorithm (Algorithm 6). More specifically, we execute this algorithm on various combinations of user location and event dataset sizes (denoted by their N/W value), and plot the standard deviation (*std*) of the number of covered events by all users before and after the execution in Figure 20(a). In addition, we also plot in Figure 20(b) the increment of the generalization cost due to the adjustment. We observe that the algorithm reduces *std* in almost all cases; however, the ratio of reduction varies with respect to N/W — the density of events over users: the ratio always exceeds 50% when N/W is between 0.1 and 10, and decreases as N/W moves away from this range. The reason for this is that when events are too sparse, only a few users cover events, making the non-uniform coverage an intrinsic problem no matter how the algorithm adjusts; on the other hand, when events are too dense, almost all



(a) *std* of Event Coverage (b) Generalization Cost

Fig. 20. Effectiveness with Respect to N/W

users cover a large number of events, making the non-uniform coverage less of a problem. Figure 20(b) shows a similar trend for cost: its increment is the most significant when N/W is between 0.1 and 10, the same range where the highest *std* reduction ratio is achieved. However, even though the cost increases, the ratio is modest and always less than an order of magnitude in all cases. As for the computational cost, the algorithm takes no more than 2 seconds to execute in all cases. Therefore, we can conclude that the coverage adjustment algorithm is effective while the event density is moderate, but anyhow, uniform coverage is always at the cost of increasing generalization cost.

9. CONCLUSION

This paper presents the problem of k-anonymity with respect to a reference dataset and a solution that is based on local enlargement paradigm. We formally prove that the solution is an H_n approximation to the optimal. In addition, we propose plane-sweep and index-based pruning techniques that significantly reduce the search space of the solution. Through mathematical analysis and experimental results, our algorithm outperforms the conventional partition-based k-anonymity algorithm by several orders of magnitude and outperforms heuristic-based KNN algorithms by up to 60%. We also extend this problem to incorporate background knowledge and show through experiments that our algorithm can sustain various privacy attacks.

Regarding future work, we plan to find applications of this problem in relational databases where table joining is common and known to compromise privacy. We will still adapt a local enlargement paradigm to solve the problem; however, the challenge is that the cost function may no longer be as regular as the area or perimeter of a box, which makes some pruning techniques (Lemma 3) invalid. Meanwhile, we also plan to develop the incremental version of the local enlargement algorithm, and apply it to a dynamic environment when either the location dataset or the event dataset changes. Furthermore, in this paper, we do not consider the issue of continuous anonymity; that is, an adversary may infer information from a sequence of published locations of a user over time, particularly if the publishing cycles for these locations are short. To address this issue, we need to redefine the problem by either treating time as another dimension or adding an additional privacy requirement that involves time.

10. ACKNOWLEDGEMENT

The authors would like to thank the editor and anonymous reviewers for their valuable suggestions that significantly improved the quality of this paper. This work was supported by Research Grants Council, Hong Kong SAR, China, under Projects HKBU FRG/08-09/II-48, FRG2/09-10/047, FRG/07-08/II-23, GRF 210808, HKBU1/05C.

REFERENCES

Bayardo, R. and Agrawal, R. 2005. Data privacy through optimal k-anonymization. In *Proceedings of ICDE*. 217–228.

Brinkhoff, T. 2002. A framework for generating network-based moving objects. *Geoinformatica 6,* 2, 153–180.

Chen, B.-C., LeFevre, K., and Ramakrishnan, R. 2007. Privacy skyline: Privacy with multidimensional adversarial knowledge. In *Proceedings of VLDB*.

Du, Y., Xia, T., Tao, Y., Zhang, D., and Zhu, F. 2007. On multidimensional k-anonymity with local recoding generalization (poster paper). In *Proceedings of ICDE*.

Fung, C. M., Wang, K., and Yu, P. S. 2005. Top-down specialization for information and privacy preservation. In *Proceedings of ICDE*. 205–216.

Gedik, B. and Liu, L. 2005. Location-privacy in mobile systems: A personalized anonymization model. In *Proceedings of ICDCS*.

Gedik, B. and Liu, L. 2008. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing 7,* 1, 1–18.

Ghinita, G., Kalnis, P., and Skiadopoulos, S. 2007a. Mobihide: A mobile peer-to-peer system for anonymous location-based queries. In *Proceedings of the Int. Symposium in Spatial and Temporal Databases (SSTD)*. 221–238.

Ghinita, G., Kalnis, P., and Skiadopoulos, S. 2007b. Prive: Anonymous location-based queries in distributed mobile systems. In *Proc. of WWW '07*. 371–380.

Gruteser, M. and Grunwald, D. 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of MobiSys*. 31–42.

Iwuchukwu, T. and Naughton, J. F. 2007. K-anonymization as spatial indexing: toward scalable and incremental anonymization. In *Proceedings of VLDB*.

Iyengar, V. 2002. Transforming data to satisfy privacy constraints. In *Proceedings of ACM SIGKDD*. 279–288.

LeFevre, K., D.DeWitt, and Ramakrishnan, R. 2005. Incognito: Efficient full-domain k-anonymity. In *Proceedings of ACM Conference on Management of Data (SIGMOD05)*. 49–60.

LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. 2006. Mondrian multidimensional k-anonymity. In *Proceedings of ICDE*.

Li, N., Li, T., and Venkatasubramanian, S. 2007. t-closeness: Privacy beyond k-anonymity and *l*-diversity. In *Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE)*. 106–115.

Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. 2006. *l*-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE)*. 24–35.

Martin, D. J., Kifer, D., Machanavajjhala, A., Gehrke, J., and Halpern, J. Y. 2007. Worst-case background knowledge for privacy-preserving data publishing. In *Proceedings of ICDE*.

Mokbel, M. F., Chow, C.-Y., and Aref, W. G. 2006. The new casper: Query processing for location services without compromising privacy. In *Proceedings of VLDB*.

Samarati, P. 2001. Protecting respondents privacy in microdata release. *IEEE Transactions on Knowlowledge and Data Engineering 13,* 6, 1010–1027.

Sweeney, L. 2002. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10,* 5, 571–588.

Vazirani, V. V. 2001. *Approximation Algorithms*. Springer.

Wang, K., Yu, P. S., and Chakraborty, S. 2004. Bottom-up generalization: A data mining solution to privacy protection. In *Proceedings of ICDM*. 249–256.

Xiao, X. and Tao, Y. 2006a. Anatomy: simple and effective privacy preservation. In *Proceedings of the 32nd international conference on Very large data bases (VLDB '06)*. 139–150.

Xiao, X. and Tao, Y. 2006b. Personalized privacy preservation. In *Proceedings of ACM Conference on Management of Data (SIGMOD06)*. 229–240.