ENHANCING UTILITY IN PRIVACY PRESERVING DATA PUBLISHING

by

Junqiang Liu

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

In the School
of
Computing Science

SIMON FRASER UNIVERSITY

Summer 2010

Approval

Name: **Junqiang Liu**

Degree: **Doctor of Philosophy**

Title of Thesis: **Enhancing Utility in Privacy Preserving Data Publishing.**

Examining Committee:

Chair: **Dr. Janice Regan**

_____

Dr. Ke Wang
Senior Supervisor
Professor, School of Computing Science

_____

Dr. Wo-Shun Luk
Supervisor
Professor, School of Computing Science

_____

Dr. Binay Bhattacharya
Internal Examiner
Professor, School of Computing Science

_____

Dr. Li Xiong
External Examiner
Assistant Professor, Mathematics and Computer Science
Emory University

Date Defended/Approved: August 6th, 2010

Declaration of
Partial Copyright Licence

Abstract

In this age of universal information sharing, there is indeed no time at which information security and privacy protection does not matter. Information security mainly concerns the prevention of disclosure of data to unauthorized parties while privacy protection aims at protecting the disclosed data from unintended use, in particular, from inference about sensitive information of involved individuals. This dissertation focuses on privacy preserving data publishing, an important field in privacy protection.

Privacy protection, due to its distinctive objective, is more challenging than information security. One challenge that has not been well addressed is how to balance privacy and data utility in privacy preserving data publishing. This dissertation aims to address this challenge by introducing new privacy notions that are more general yet more flexible, and by proposing optimal solutions and efficient algorithms. Specifically, this dissertation makes three efforts, that is, to enhance data utility in preventing privacy attacks specific to relational data, set-valued data, and web search data respectively.

First, it proposes the l^+-diversity notion for relational data, which is more general than the classical l-diversity, together with an optimal algorithm, l^+-Optimize, which employs a flexible anonymization model. The solution is optimal in data utility, which has a significant gain comparing with a heuristic solution and a solution by a restrictive anonymization model.

Second, it proposes the KL(m, n)-privacy notion for set-valued data. While approaches for relational data are not applicable, only a few works are on set-valued data, and they provide either over-protection or under-protection. KL(m, n)-privacy with the proposed optimal algorithm provides sufficient protection and retains enough utility.

Third, it proposes the vocabulary k-anonymity notion for web search data, together with a semantic similarity based clustering approach. This approach preserves more data utility than the state of the art approaches.

Keywords: privacy; anonymity; relational data; set-valued data; web search data.

The journey is the reward.

To my wife, Xiaoying, and my son, Xunchao.

Acknowledgements

I would like to express my deep gratitude to my senior supervisor, Dr. Ke Wang, for his profound influence on my research, his patience in supervising me, and his valuable criticism of my work over the past few years. I am very thankful to my supervisor, Dr. Wo-Shun Luk, for his encouragement and insightful advice for my research and my future career. I would like to express my sincere thanks to my internal examiner, Dr. Binay Bhattacharya, and my external examiner, Dr. Li Xiong, for their patiently reading through my dissertation and providing valuable feedback. I would like to thank Dr. Janice Regan for chairing the examining committee.

I would also like to thank the faculty members and the support staff in the school of computing science at Simon Fraser University. A special acknowledgement goes to Dr. Martin Ester, Ms. Val Galat, Ms. Gerdi Snyder, Ms. Tracy Bruneau, and Mr. Ching Tai Wong.

Thanks full of love go to my wife, Xiaoying, and my son, Xunchao, for their understanding and encouragement during such a long journey.

Table of Contents

List of Figures

List of Tables

1: Introduction

The explosive growth of the internet has increased the dependence of both organizations and individuals on sharing information universally. This, in turn, has led to an ever-increasing awareness of the need to protect information from unintended use, and to guarantee the privacy of involved parties [15][71][85][92].

Privacy preserving data publishing [36][38], abbreviated as PPDP, is one of the important research fields of information security and privacy protection. PPDP addresses such a problem: A *trusted* data publisher collects personal data of individuals in the raw form, anonymizes the collected data, and releases the anonymized data to a data recipient or the public. The data publisher is responsible for preventing the data recipient from breaching the privacy of the individuals and for retaining usefulness of the anonymized data for data analysis. The challenge is that the legitimate data recipient could also be a privacy adversary, which makes the problem harder than information security. PPDP is motivated by real world applications that involve publishing and sharing relational data, set-valued data, web search data, etc.

Privacy protection in publishing relational data, especially in sharing patient medical records, has drawn significant attention from researchers and practitioners. Samarati and Sweeney in their pioneering work [85] showed that a publicly available, city's voter registration list could be used to easily re-identify individuals whose records were included in the medical data released as anonymous for research purpose. In the United States of America, the HIPAA Privacy Rule protects individually identifiable

health information in storage or in transmission, as part of a suite of regulations promulgated pursuant to the administrative simplification provisions of the Health Insurance Portability and Accountability Act. One of the requirements of the HIPAA Privacy Rule is the *de-identification* of protected heath information [96]. In Canada, the PIPED Act [74] as a federal law establishes ten principles that organizations must follow when collecting, using and disclosing personal information in the course of commercial activity. Many provinces and sectors all have similar laws or legislations. For example, de-identification is also one of the requirements of the personal health information protection act in Ontario [73].

Surprisingly, privacy protection in publishing set-valued data has received less attention. Nevertheless, this problem is important and even harder to address since unlike relational data, set-valued data have no fixed schema. For example, the world's largest online DVD rental service, Netflix, announced the $1-million prize for improving their movie recommendation service on October 2, 2006 [43], and publicly released a dataset containing 100,480,507 movie ratings, created by 480,189 Netflix subscribers between December 1999 and December 2005 for contestants to use. Narayanan and Shmatikov [71] demonstrated that 96% of Netflix subscribers can be uniquely identified with knowledge of no more than 8 movie ratings and dates. Unfortunately, the second Netflix Prize competition had to be called off at the last minute, after three and a half years since the first Netflix challenge, because the privacy issues were still not well addressed [117].

Most recently, privacy protection in publishing and sharing web query logs began to gain attentions. While the primary reason for search service providers to retain such data is to improve their own services [76][90][112], it turned out that such data have been

becoming the richest important source. Publishing or sharing such data benefits various applications. For example, Google shared its search engine query data with the US Centers for Disease Control and Prevention in detecting influenza epidemics [42]. Sometimes, such "sharing" is just to abide certain legal obligations. For example, in 2006 the US Department of Justice issued subpoenas to AOL, Google, Microsoft, and Yahoo, as part of its litigation of an Internet child safety law, for seeking several months' worth of query logs [80][27]. However, web query logs are extremely vulnerable to privacy attacks as highlighted by the AOL query log incident in 2006. Barbaro and Zeller [15] demonstrated the ease to determine the real identity of an "anonymous" AOL searcher by examining query terms even ignoring the existence of social security numbers, driver license numbers, and credit card numbers.

1.1 Motivations

One major challenge with privacy preserving data publishing is how to balance privacy and data utility, that is, how to transform raw data in such a way that the privacy is protected while the data utility is retained. While different types of data are subject to different privacy threats that are formulated by different attack models and are handled by different privacy principles, such a challenge is prevailing over privacy protection problems with all types of data, and has not yet been well addressed in general.

In the following, we discuss the specific challenge in publishing the three types of data, i.e., relational data, set-valued data, and web search data respectively.

Challenge with privacy preserving publishing of relational data. Relational data is a table that has a fixed schema with each column being an attribute and each row

being a record consisting of an individual's values on all attributes. The typical privacy attack on relational data is re-identification, i.e., disclosure of the identity or the sensitive attribute value of a target individual, resulted from linking the table via a fixed subset of attributes with external data, which is also known as the linking attack. The basic approach to thwart such linking attacks (a.k.a. re-identification) is *anonymization* (a.k.a. de-identification). Although there are many works on preventing such linking attacks on relational data, most of the works exerted a universal privacy requirement, employed a restrictive anonymization model, or proposed heuristic solutions.

Therefore, several questions remain open: can we get a significant gain in the data utility from an optimal solution compared to heuristic ones? Is it practical to find an optimal solution with a flexible anonymization model efficiently for real world datasets? Can we improve the data utility by a customizable privacy notion?

Challenge with privacy preserving publishing of set-valued data. Set-valued data is just a collection of sets with each set consisting of an arbitrary number of items from a domain. A typical example is the transaction data such as shopping basket data. Set-valued data is unstructured as it has no fixed schema, any combination of items may render clues that lead to privacy attacks. Thus, privacy protection notions and technique specific to set-valued data are in need. Comparatively, there are fewer works in this context. Among those works, some provide over-protection that introduces excessive distortion [41][48] by treating set-valued data as a relational table, some provide under-protection that ignores the adversary's knowledge about the absence of certain items [95][108] or only handles identity disclosure [48][95]. All the works only propose heuristic solutions that incur more information loss than necessary.

Therefore, a few concerns arise: can we propose a more flexible yet more general notion that retains enough data utility and provides sufficient privacy protection, i.e., no over-protection, nor under-protection? Can we present an optimal algorithm that guarantees the optimality of the solution on real world datasets and propose efficient approximate algorithms based on the optimal algorithm?

Challenge with privacy preserving publishing of web search data. Web search data, i.e., query logs, are different from relational data and set-valued data, and somehow are semi-structured. A query log usually has fixed fields, such as time stamps and URLs, with each field being of a special usage, and has multiple lines per web search user with each line being one of the search results returned for a query issued by the user. A web query log published in its entirety is extremely vulnerable to privacy attacks [58][54], and there is no one-size-fits-all scenario with web search data.

One interesting scenario is to publish vocabularies extracted from a query log, which is less vulnerable and also has a variety of applications [27][107]. However, vocabularies extracted from a query log are bag-valued data as the number of occurrences of each query-term in a vocabulary is important information to convey. The challenge is that such bag-valued data are extremely sparse as people hardly use the same query-terms even for the same concepts. Moreover, if we treat each occurrence of a query-term in a vocabulary as a different term, the data becomes even sparser because of the huge dimensionality. Due to the extreme sparsity, the conventional techniques for set-valued data will incur huge information loss in this context. Therefore, new anonymization techniques are in urgent need.

In short, there is a need for new privacy notions that allow specifying customized privacy requirement and avoid a universal protection that will cause much loss in data utility. There is also a need for optimal solutions and efficient, scalable algorithms that help retain data utility. Such needs have been observed with the privacy protection problems with relational data, set-valued data, and web search data respectively.

1.2 Objectives and Contributions

The main theme of this dissertation is to enhance data utitlity in privacy preserving data publishing, in particular, for publishing relational data, set-valued data, and web search data respectively. The concrete objectives include the following.

- to identify privacy threats in publishing relational data, set-valued data, and web search data, and to propose new privacy protection models that are flexible and customizable so that more data utility could be preserved;
- to study techniques for anonymizing relational data, set-valued data, and web search data to eliminate privacy threats, and to probe ways to measure the utility of anonymized data;
- to propose systematic approaches and/or optimal solutions to the problems of anonymizing relational data, set-valued data, and web search data to help retain data utility, and to develop efficient and scalable algorithms.

By achieving those objectives, this dissertation made a few contributions in anonymizing relational data, set-valued data, and web search data, namely the optimal l^+-*diverse* anonymization, the optimal KL(m, n)-privacy, and the vocabulary k-anonymous clustering respectively.

Optimal l^+-diverse anonymization for relational data. Anonymizing relational data to prevent the sensitive attribute value disclosure is a fundamental problem. The first contribution of this dissertation is to improve data utility in anonymizing relational data by avoid exerting the same amount of protection on different sensitive values, employing a flexible anonymization model, and proposing an optimal algorithm.

Specifically, this dissertation proposed a flexible privacy principle, l^+-*diversity*, for publishing relational data, and presented an optimal algorithm, l^+-Optimize, for improving the data utility by assuring the optimality of the solution with a flexible anonymization model; The optimal l^+-diverse solution has a significant gain comparing with a heuristic solution and a solution by a restrictive anonymization model.

Optimal KL(*m, n*)-privacy for set-valued data. Existing research mainly focuses on relational data, which is the context of the first contribution. Nevertheless, certain applications require privacy preserving publishing of set-valued data. Set-valued data, such as shopping transaction data, involves hundreds or even thousands of dimensions if treated as a relational table, rendering existing approaches either not working or introducing excessive data distortion. The second contribution of this dissertation is a new privacy notion together with an optimal solution that provides sufficient privacy protection and retains enough data utility in publishing set-valued data.

Specifically, this dissertation proposed the KL(m, n)-privacy notion that handles both identity disclosure and sensitive item disclosure from the adversary's knowledge about the presence of certain items and knowledge about the absence of certain items in publishing set-valued data. This dissertation also presented the first optimal solution in this context, and developed efficient algorithms based on the optimal solution.

Vocabulary *k*-anonymous clustering for web search data. Web search data, i.e., query logs, are semi-structured, and are extremely sparse. These features render the approaches for relational data and set-valued data inappropriate in this context. The third contribution of this dissertation is a new privacy notion and a new anonymization technique tailored for this context.

Specifically, this dissertation introduced the *vocabulary k-anonymity* principle that tackles the privacy attacks observed in the real world, and proposed the semantic similarity based clustering approach for anonymizing vocabularies, that is, bags of query terms extracted from a web query log with a given granularity. This approach overcomes the extreme sparcity and retains much more data utility than other approaches.

In short, this dissertation concentrated on enhancing the data utility in privacy preserving publishing by introducing new privacy notions, proposing new anonymization techniques, and presenting optimal solutions and efficent algorithms.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows.

Chapter 2 studies the related works in the literature of privacy preserving publishing of relational data, set-valued data, and web search data. Emphasis will be on the privacy principles, anonymization techniques, data utility metrics, and algorithms.

Chapter 3 studies the privacy preserving publishing of relational data, defines the l^+-diversity notion and the optimal anonymization problem, presents the strategy for searching an optimal solution, discusses cost lower bounding methods, and proposes dynamic pruning techniques.

Chapter 4 studies the privacy preserving publishing of set-valued data, defines the KL(m, n)-privacy notion and the optimal anonymization problem, describes an overall solution framework, discusses how to check the validity of a solution by mining the most general privacy threats, and presents an optimal solution.

Chapter 5 studies the privacy preserving publishing of web search data, defines the vocabulary k-anonymity principle, describes the semantic similarity based clustering approach, and presents a greedy algorithm.

Chapter 6 concludes this dissertation.

1.4 Non-Discrimination Statement

This dissertation contains some synthetic examples for illustrating technical concepts. The author has no intention to discriminate any group of people.

2: Related Works

This chapter first gives a brief description of the research fields relevant to privacy preserving data publishing (PPDP) [36][38], and then surveys three bodies of works related to this dissertation and discusses their connections to this dissertation and the differences.

Privacy preserving data mining (PPDM) is the most relevant field [7][10][97]. PPDM is to perturb data in such a way that the privacy at the level of individual records is preserved, while the data mining models over aggregate level can be accurately reconstructed by employing *customized* data mining algorithms. The main difference is that PPDM serves a particular data mining task while PPDP is of general purpose, and PPDM preserves the data semantics at the aggregate level while PPDP preserves the data semantics at the record level. Other three relevant fields are described as follows.

Privacy preserving data collection [11][20][110] is to make the data collection procedure anonymous so that the data owners are willing to submit the data while the data is inherently anonymous. The difference is that the data collector is *untrusted* in this field while *trusted* in PPDP. *Privacy preserving data sharing* [8][114][111] concerns multiple data collectors while PPDP concerns a single collector. Multipe collectors collaborate to answer a query or to compute a function across the private database owned by each collector. *Statistical disclosure control* [1][50] thwarts privacy attacks on the output, i.e., answers to queries on databases while PPDP prevents attacks on the database. Differential privacy [19][28][30][31][32][81] could be deemed as the latest development.

2.1 PPDP with Relational Data

In the literature, most of PPDP works consider the problem of protecting privacy in releasing relational data. Relational data is a table that has a fixed schema with each column being an attribute and each row being a record consisting of an individual's values on all attributes. The basic problem is how to publish an anonymous version of a relation table such that the identities or the sensitive attribute values of the individuals in the published table cannot be re-identified while the data remains practically useful.

It is common sense to remove all explicit identifying attributes, e.g. Name, from a relational table before publishing, which however is not enough. It is still possible that individuals can be re-identified by linking the published table with external data. For example, the patient data in Table 2.1(a) can be linked with external data in Table 2.1 (b) on the attributes Education and Country. By doing so, a patient, Ahmed, is uniquely re-identified from the patient data.

Table 2.1 Patient data (relation table) and external data

(a) patient table *T*

#	Education	Country	Disease
1	Junior	France	Flu
2	Junior	UK	Asthma
3	Senior	France	Flu
4	Senior	UK	Cancer
5	Bachelor	UK	Asthma
6	Bachelor	Canada	Asthma
7	Bachelor	USA	Flu
8	Graduate	USA	Asthma
9	Graduate	Canada	Cancer
10	Graduate	Canada	Cancer

(b) external data

Name	Education	Country
Ahmed	Junior	France
Brook	Senior	France
Clair	Senior	UK
Dave	Bachelor	UK
Evelyn	Bachelor	Canada
Jane	Graduate	Canada
Tom	Graduate	Canada

To address such linking attacks, various privacy protection principles and approaches have been proposed in the literature.

2.1.1 Privacy Principles

In general, a privacy principle defines what privacy is, how to quantify privacy, and how to specify the level of privacy protection.

The *k*-anonymity principle [85] deals with the *identity disclosure* problem in the foregoing example. The *k*-anonymity principle requires that each record in the published data is *indistinguishable* from at least *k*–1 other records on the so-called *quasi-identifier*. The quasi-identifier is a set of attributes that can be used to link the published relational table with external data, e.g., Education and Country in the foregoing example.

However, the *k*-anonymity principle does not address the so-called *sensitive attribute disclosure* problem, which is resulted from the homogeneity of sensitive attribute values. For example, record 9 and record 10 in Table 2.1(a) are indistinguishable from each other, so Jane's record is in a 2-anonymous group. However, all the values of Disease are Cancer in this group, which reveals Jane's sensitive value with a probability of 100%.

To handle the sensitive attribute disclosure problem, the *l*-diversity principle [67] was proposed, which requires that there are at least *l well-represented* values for a given sensitive attribute in each indistinguishable group of records. The most used instantiation of *l*-diversity requires that the probability for each sensitive attribute value being associated with an individual is no more than a unique privacy threshold 1 / *l*.

Chapter 3 proposes a new privacy principle, l^+-diversity, which is more general yet more flexible by allowing a different privacy threshold per sensitive value according to its sensitivity. Many other privacy notions also extend *k*-anonymity and *l*-diversity, but in a way different from what is proposed in Chapter 3, including privacy template [100],

(X,Y)-privacy [99], personalized anonymity [104], t-closeness [61], (k, e)-anonymity [113], (ε, m)-anonymity [62], δ-presence [72], and ε-differential privacy [30][31].

2.1.2 Anonymization Techniques

Anonymization techniques transform a relational table to enforce a particular privacy principle. There are two major categories of techniques: one is perturbation, and the other is generalization and suppression. Perturbation can retain data semantics *at the aggregate level*. There are a lot of perturbation techniques, including permutation or swapping [113], additive random noise [10][30][31], randomized response [29][47], clustering [4], synthetic data generation [6], and result sanitization [63].

Generalization and suppression can preserve data semantics *at the record level*. Generalization involves replacing specific values with more general values. Suppression involves deleting values or records. Generalization is made possible through the taxonomy tree, a.k.a. the generalization hierarchy, associated with the domain of every *quasi-identifier* attribute. If the node representing value x is the parent (ancestor) of the node representing value y on the taxonomy tree, x is called the *taxonomic parent* (*ancestor*) of y, and y is called a *taxonomic child* (*descendant*) of x.

The followings are several famous generalization models. The *full domain* generalization model [85] is restrictive in that all generalized values of an attribute in the anonymized data must be on the same level of the attribute's taxonomy tree. Such a restriction could exclude many natural generalization solutions. The *full subtree* generalization model [52] allows more natural generalization solutions, with which the generalized values can be at different levels of taxonomies. The two models require that

each generalization step is applied to all records in the table. Hence, all values in the anonymized data are exclusive of each other, which is called the *domain exclusiveness* property. The *multi-dimensional* generalization model [40][60] allows a generalization step to be locally applied to a multi-dimensional region (some records), which destroys the domain exclusiveness.

In Chapter 3, suppression is treated as an integral part of anonymization, and the equilibrium of suppression and generalization is achieved to help improve data utility, which is different from previous works where suppression is not considered [40][60] or is handled by an external constraint [59][84][85].

2.1.3 Data Utility Metrics

Data utility metrics are employed to measure the usefulness of the anonymized data. Alternatively, information loss metrics can serve the same purpose. There are two types of data utility (information loss) metrics.

General-purpose metrics capture the notion to keep the data as specific as possible. Such metrics include the generalization height [84][85][93], the average size of the indistinguishable groups [60], the loss metric LM [52], the normalized cardinality penalty NCP [109], the global certainty penalty GCP [40], the classification metric CM [52], the discernibility metric DM [17], KL-Divergence [55], and the mean square error [47].

Application-specific metrics capture the usefulness of the anonymized data in fulfilling a specific data mining task, which is usually expressed as the accuracy of the data mining model built on the anonymized data, including cassification eror [101][100] [37][6][94][10][29], error in answering aggregate queries [104][105][62][60][40][113].

In Chapter 3, a metric independent approach is adopted. That is, the proposed approach is generic in that it works with any reasonable data utility metric [17][52] and can easily adapt to a new metric. Such an approach is not emphasized by other works.

2.1.4 Algorithms

An algorithm solves such an anonymization problem, that is, to transform the raw data by an anonymization technique to enforce a privacy principle and to retain as much informaiton as possible in terms of a metric. In general, finding an optimal solution to an anonymization problem is computationally hard [5][60][70]. Therefore, most algorithms are heuristic based or approximate ones, e.g., TDS [37], Mondrian [60], space mapping [40] , space indexing [51], and approximate *k*-anonymity [70].

Optimal algorithms, such as Incognito [59], MinGen [93], and BinarySearch [84], find an optimal *k*-anonymous solution with a quite restrictive anonymization model, the full domain generalization model [85]. So far, *k*-Optimize [17] is the only algorithm that finds an optimal *k*-anonymous solution with a flexible anonymization model, the partition-based single dimension recoding with suppression, which is suitable for attributes whose domains have a total order, e.g. numerical attributes. The bucketization algorithm [105] is optimal when the quasi-identifier is the key which but is a rare case. In a usual case, it may cause information loss higher than heuristic algorithms [40].

In Chapter 3, the first optimal algorithm for finding an *l*-diverse solution with a flexible anonymization model is proposed. Beyond this work, all the prior works employ the *l*-diverse variant of the optimal *k*-anonymous algorithm, Incognito, which employs a very restrictive anonymization model.

2.2 PPDP with Set-valued Data

Set-valued data is a collection of sets, and each set consists of an arbitrary number of items from a domain. A typical example of set-valued data is the transaction data such as shopping basket data as shown in Table 2.2. Set-valued data are different from relational data in that set-valued data have no fixed schema. Nevertheless, set-valued data have a wide range of applications. For example, customer behaviour study and marketing analysis are made possible by mining the shopping basket data. However, set-valued data could contain significant amount of personal and sensitive information. The release of such data to a third party could breach privacy.

Table 2.2 Shopping Basket Transactions

TID	Transactions
t_1	Wine, Milk, Yogurt, AdultToy
t_2	Beer, Jacket, Pants, AdultToy, Viagra
t_3	Yogurt, Jacket, Hose, Shoe, Viagra
t_4	Milk, Yogurt, Jacket, Geta, PregancyTest
t_5	Beer, Wine, Milk, Jacket, Pants

Surprisingly, privacy preserving publishing of set-valued data has received less attention comparing with relational data. There are only a handful of works.

Transactional k-Anonymity *with* ***Local Generalization***. He and Naughton [48] proposed the transactional *k*-anonymity which requires that each transaction is *identical* to at least *k*-1 other transactions in the data. In other words, this notion takes all items together as the quasi-identifier. Therefore, one argument against this work is that such a treatment is too restrictive, i.e. provides over-protection while only considering the

identity disclosure. Chapter 4 proposes a privacy notion that avoids such over-protection and considers both the identity disclosure and the sensitive item disclosure.

He and Naughton [48] presented a top-down, partitioning algorithm by adapting the multi-dimensional partitioning algorithm Mondrian [60]. They called the algorithm local generalization [48], abbreviated as LG. The other argument against LG is that the domain exclusiveness is not kept in the generalized data. For example, LG allows both "apple" and its taxonomic ancestor "fruit" to co-occur in the data. With such co-occurrences, a counting query on the number of transactions containing "apple" cannot be answered accurately because some occurrences of "apple" may have been generalized to "fruit" while some may not. The limitation is that standard data mining algorithms no longer apply. Chapter 4 proposes an optimal algorithm with the full subtree generalization, which preserves the domain exclusiveness.

k^m-Anonymity with Global Generalization. Terrovitis, et al. [95] proposed the k^m-*anonymity* notion, which requires that for any combination X of no more than m items, at least k transactions in the data contain X. The rationale is that a realistic adversary is limited by the number of items that can be obtained as background knowledge. The k^m-anonymity notion treats all items as non-sensitive items. The authors of [95] presented a heuristic, global generalization algorithm AA for achieving k^m-anonymity. This problem actually is to mine *infrequent* itemsets [64], which is extremely challenging. AA is not efficient as it employs a breadth-first search strategy, counts all itemsets of sizes up to m, and only has the validity based pruning which is not powerful enough. Nevertheless, as AA is a heuristic algorithm, it has no guarantee on the data utility. The question is whether an optimal generalization algorithm can greatly improve the data utility.

Chapter 4 is different from [95] as follows. Chapter 4 considers both the presence and absence of certain items while [95] ignores the absence of certain items, Chapter 4 considers both the identity disclosure and the sensitive item disclosure while [95] only considers the identity disclosure, and Chapter 4 proposes an optimal algorithm while [95] focuses on heuristic algorithms.

(k,h,p)-Coherence with Total Suppression. Xu and Wang [108] proposed the (*k,h,p*)-*coherence* notion. They divided items into non-sensitive items and sensitive items. Non-sensitive items are those that potentially identify individual transactions, and sensitive items are those that should be protected. The (*k,h,p*)-coherence notion requires that for any combination *X* containing up to *p* non-sensitive items, there are at least *k* transactions containing *X* and no sensitive item occurs in more than *h* percentage of such transactions. [108] developed a heuristic, total suppression algorithm for achieving coherence by removing certain items from all transactions. This approach preserves the domain exclusiveness in the anonymized data, so we can use standard algorithms to analyze the anonymized data. The authors also pointed out that suppression is not good at handling sparse data, which may result in high information loss.

Chapter 4 is different from [108] in the following. Chapter 4 considers both the presence and absence of certain items while [108] ignores the absence of certain items, Chapter 4 employs generalization while [108] employs suppression, and Chapter 4 proposes an optimal algorithm while [108] proposes a heuristic algorithm.

Privacy Degree with Band Matrix Technique. Ghinita, et al. [41] proposed the *privacy degree* notion that is an adoption of the *l*-diversity principle. This notion differentiates items between sensitive and non-sensitive and limits all inferences about a

sensitive item to a low certainty ($\leq 1/p$). The authors extended the bucketization approach [105] to this case by treating a set-valued dataset as a high dimensional table consisting of binary attributes. The basic idea is to split such a table into two, one contains all non-sensitive items, and the other contains all sensitive items. For any transaction, its non-sensitive part is associated through a group-id with its sensitive part. To lower the information loss, the authors first employed the band matrix technique to arrange transactions such that adjacent transactions are highly correlated on the non-sensitive attributes. The authors then group adjacent transactions to observe the privacy principle.

The argument against bucketization [105] also applies to this approach, that is, publishing the exact non-sensitive items confirms the participation of individuals, which itself is a privacy breach. Another argument against this approach is that it modifies all inferences about a sensitive item to a low certainty, even though a realistic adversary may never be able to acquire the background knowledge to make such inferences, which is also a kind of over-protection. Chapter 4 proposes a privacy notion that has no such limitations.

Privacy Preserving Data Mining on Set-valued Data. A few works in this field are quite interesting although *not* directly related to our focus. [34] presented randomization operators to limit privacy breaches. [102] proposed generating a synthetic database from frequent itemsets discovered in the original database. Hiding sensitive association rules were considered in [87][98] by reducing confidences and supports of sensitive rules. [13] considered privacy threats embedded in data mining results and suggested ways to eliminate such threats by means of pattern distortion.

2.3 PPDP with Web Search Data

Web search data, i.e., web query logs retained by search engines, are different from relational data and set-valued data. For example, Table 2.3 shows such a query log. A log usually has fixed fields with each field having a special usage. Each web search user usually submitted multiple queries to a search engine. For each query, the engine usually returns multiple search results.

Web query logs have become more and more valuable to various applications because of the unparalleled volume of the data and the richness of the information. However, web query logs also present unparalleled privacy risks as every search service users left behind a trail of information about her/himself in the query log. Therefore, privacy protection in sharing / publishing web query logs has been becoming more and more important.

Table 2.3 An example query log

AnonID	Query	QueryTime	ItemRank	ClickURL
0001	care packages	1:00 Jan 1	3	a.com
0001	movies for dogs	2:00 Jan 1	8	b.com
0001	big cuddly dog	2:20 Jan 1	3	c.com
0001	movies on bipolar	1:00 Jan 2	5	d.com
0001	rescue of older dogs	1:00 Jan 1	8	e.com
0001	blue book	1:15 Jan 1	4	f.com
0001	school supply for children	1:15 Jan 1	1	g.com
0001	blue fingers	2:05 Jan 1	7	h.com
…	…	…	…	…

Xiong and Agichtein [107] discussed query log ananlyis applications and possible privacy risks in publishing query logs, summarized query log anonymization techniques along two dimensions, i.e., query log grouping granularity and query de-identification strictness, and pointed out research directions. Chapter 5 proposes a privacy notion in this context and developes an approach along one of such directions [107].

Cooper [27] provides a survey from a policy perspective on the query log privacy enhancing techniques, including log deletion, hashing queries, identifier deletion, and hashing identifiers. According to Cooper, *k*-anonymity has yet to be formally applied in the context of query logs. Chapter 5 is an effort to extend *k*-anonymity to such a context.

Trace Attack and Person Attack. Jones, et al. [54] showed how to build SVM classifiers for launching two types of attacks against Yahoo query logs. The first is the trace attack, in which an adversary studies a privacy-enhanced version of a sequence of searches (trace) made by a particular user, and attempts to discover information about that user, such as gender, age, and location, which allows identifying a small number of users containing the true user who generated the trace. The second is the person attack, in which an adversary attempts to discover the traces in a search engine log that correspond to a particular known user given some personal/background information of the user.

Attacks on Hashing Technique. Kumar, et al. [58] showed that statistical techniques can be applied to compromise the anonymization by a token based hashing technique. Such attacks are based on crypto-analysis: comparing frequencies of tokens in an original log and frequencies of hashes in an anonymized log to figure out the mapping between tokens and hashes. From the authers' point of view, there is no satisfying framework for proving privacy properties of a query log anonymization scheme.

The implication with the foregoing two works is that publishing a query log in its entirety especially with the subtle session information has significant privacy risks. Chapter 5 publishes the vocabularies extracted from a query log rather than the query log itself. Nevertheless, such vocabularies have a variety of applications.

p-Linkability with user profiles. Zhu and Xiong and Verdery 2010 [116] studied the problem of anonymizng user profiles so that the probability of linking a user to an individual term does not exceed a privacy threshold *p*. They augment the profiles with synonyms and hypernyms, cluster the augmented profiles by similarity, and publish the centroid of each cluster. Chapter 5 proposes the vocabulary *k*-anonymity and a clustering approach different from [116].

Differential Private Query-Click Graph. Korolova, et al. [57] proposed to publish a query-click graph, i.e., the result derived from the data, instead of the data itself. This is the case where the differential privacy principle applies. To achieve the differential privacy, i.e., ensuring the participation of one individual has no significant effect on the result, their algorithm limits the maximum number queries and clicks of a user that can be released, and adds noises to the number of times each query posted in the log and adds noises to the number of each URL clicked. However, only 1% distinct queries can be published by such an approach. Chapter 5 presents a publication scenario different from that in [57]. The next three works publish a query log in its entirety that is definitely different from Chapter 5.

Cryptographic Approach. Adar [2] presented a "cryptography" system to deal with dynamic publishing of queries, and proposed to split queries of a user into different groups by clustering queries by "interests" to prevent privacy adversaries based on aggregate analysis. The "cryptography" system works as follows. For a distinct query with occurrences less a given threshold *k*, the system hash the query to generate a key, encrypt the query by the key, and attaches one piece of the key to the query. The key is split into *k* pieces, and the *i*th occurrence of the distinct query gets the *i*th piece of the key.

When a query occurs no less than k times, the key is fully released and can be used to decrypt the query.

Website Privacy. Poblete, et al. [78] proposed the website privacy notion. That is, publishing query logs may reveal confidential information about traffic of particular websites. In particular, an adversary that owns a certain website may identify its competitors by obtaining auxiliary information. Therefore, their method anonymizes URLs and removes vulnerable queries, e.g., those in the intersections of queries retrieving competitors' websites.

k^{δ}-anonymity. Hong, et al. [46] proposed the k^{δ}-anonymity for anonymizing query logs. The similarity between two queries is defined as the degree of their overlapping on clicked URLs. Queries are merged into clusters by the similarity parameter δ. A query-cluster clicked-URL matrix is constructed per user, which represents the normalized number of entered queries in each cluster that link to each clicked-URLs. Then, users are clustered based on the similarity among their query-cluster clicked-URL matrices. A user is made indistinguishable from other users in the same cluster by adding or suppressing query-objects, i.e., queries and URLs. Their experiments reported utility in terms of the number of query-objects, and the recall of top 50 queries / URLs under a small of 1K users. It is not clear if they publish the representative of each query-cluster or the individual queries in the query-cluster, and how the number of query-clusters other than the number of query-objects related to the parameter δ.

Works on text document sanitization are seemingly relevant to Chapter 5.

K-safety. Chakaravarthy et al. [25] proposed a principled approach to sanitization of unstructured text documents. A document is treated as a set of terms, and public knowledge is modeled as a context database of entities. Each entity could be sensitive or non-sensitive, and the context of each entity is also a set of terms. The privacy principle, namely K-safety, is to ensure that the maximum subset of each sensitive entity's context contained in a document is also contained by the contexts of at least K other entities, i.e., the context exposed in the document is indistinguishable from K other contexts given a user-specified context database and a collection of sensitive entities. They enforce K-safety by suppression of terms from the document.

t-plausibility. Jiang, et al. [53] proposed to sanitize a single text document by generalizing sensitive words. A generalized document observes *t*-plausibility if at least *t* base text can be generalized to such a sanitized document (based on a set of ontologies derived from WordNet).

Named Identity Removal. Saygin et al. [86] proposed an approach for sanitizing text documents by utilizing automatic detection methods to find sensitive named entities, such as persons, locations, and organization names, and numeric values, such dates, and credit card numbers, and then using value distortion, value dissociation, and value-class membership to hide the personal information, i.e., enforcing *k*-anonymity at individual term level.

Chapter 5 has a fundamental difference from text document sanitization. That is, while Chapter 5 makes multiple bags of terms indistinguishable of each other, the latter sanitizes one set or one bag of terms, i.e., a single document, and focuses on detecting sensitive named identities and removing each named entity from the document.

3: Optimal l^+-Diversity for Publishing Relational Data

The wide availability of personal data has made the privacy protection problems in publishing and sharing data of vital interest. A fundamental problem is how to prevent an adversary from identifying a target individual's record or sensitive attribute value in a published relational table by linking with an external table. For example, a hospital may be required to release the patient data in Table 3.1(a) to a medical research institute. Patient Ahmed could be re-identified by a curious researcher in the institute who received the data and happened to have a voter registration list inTable 3.1(b).

To prevent such linking attacks, the *k*-anonymity principle [85] and the *l*-diversity principle [67] are proposed in the literature. The former tackles linking attacks by ensuring that each record in the published data is indistinguishable from at least *k*–1 other records. The latter improves the former by preventing homogeneity attacks, i.e., ensuring that there are at least *l* well-represented values for a given sensitive attribute in each indistinguishable group of records. A lot of works have been done to extend *k*-anonymity or *l*-diversity to different scenarios and to develop anonymization techniques and algorithms for enforcing these principles.

However, two challenges prevail over such a fundamental problem and have not been well addressed yet.

The first challenge is how to avoid a universal protection that could incur excessive data distortion. So far, the state of the art privacy principle is *l*-diversity. The widely used instantiation of *l*-diversity requires that the probability for an individual

being linked to a sensitive value is less than a *uniform* threshold 1 / *l*. As different sensitive values have different sensitivity, such a universal protection could incur excessive data distortion. For example, among various diseases, Flu is less sensitive than Cancer. Providing Flu with the same amount of protection as Cancer will inevitably cause over-protection. There is a need for flexibilities in specifying the privacy requirement.

Table 3.1 Patient data (relation table) and external data

(a) patient table *T*

#	Education	Country	Disease
1	Junior	France	Flu
2	Junior	UK	Asthma
3	Senior	France	Flu
4	Senior	UK	Cancer
5	Bachelor	UK	Asthma
6	Bachelor	Canada	Asthma
7	Bachelor	USA	Flu
8	Graduate	USA	Asthma
9	Graduate	Canada	Cancer
10	Graduate	Canada	Cancer

(b) external data

Name	Education	Country
Ahmed	Junior	France
Brook	Senior	France
Clair	Senior	UK
Dave	Bachelor	UK
Evelyn	Bachelor	Canada
Jane	Graduate	Canada
Tom	Graduate	Canada

The second challenge is that both the optimal *k*-anonymity problem and the optimal *l*-diversity problem are NP-hard in general [5][60][70]. Optimal algorithms proposed so far only solve the optimal *k*-anonymity problem with a quite restrictive anonymization model. Examples are Incognito [59], MinGen [93], and BinarySearch [84]. The only exception is *k*-Optimize [17], which finds an optimal *k*-anonymization with a flexible anonymization model, but only works with numeric attributes. Most algorithms are heuristic based or approximate ones, e.g., Mondrian [60], TDS [37], space mapping [40], space indexing [51], and approximate *k*-anonymity [70]. To our knowledge, no work has been reported on an optimal algorithm that ensures the optimality of data utility with a flexible anonymization model to enforce *l*-diversity.

Therefore, several questions remain unanswered regarding the optimal *l*-diversity problem: Is there a significant gain in the data utility from having an optimal solution compared to heuristic solutions, and from searching a flexible large solution space compared to a restricted solution space? Can we improve the utility by setting a distinct privacy threshold per sensitive value? Can a large portion of the search space be pruned without missing the optimal solution? Is it practical to find an optimal solution for real world datasets?

This chapter answers the foregoing questions. In particular, this chapter introduces an extended form of the classic *l*-diversity problem [67], called optimal l^+-diversity problem: find such an anonymization of a relational table, that ensures no individual can be linked to a sensitive value s_i with a probability higher than θ_i, and that incurs the minimum information loss. This problem extends the most used instantiation of *l*-diversity in that the latter is a special case of the former by setting $\theta_i = 1 / l$ for every sensitive value s_i. By allowing a different privacy threshold for a different sensitive value based on its sensitivity [26], this notion helps preserve more data utility and provides more protection for sensitive values.

The main challenge for this problem is the huge search space consisting of all possible ways of anonymization that is achieved by generalization and suppression. This problem is much more difficult than the optimal *k*-anonymity problem in [17]. *k*-anonymity observes a natural pruning property, called *monotonicity of suppression*: if a indistinguishable group *G* of records violates *k*-anonymity and should be suppressed, then all subgroups derived by partitioning *G* violate *k*-anonymity and should all be suppressed. l^+-diversity does not conform to such a monotonicity, so the amount of suppression is not

monotone in a top-down search. Hence, the pruning techniques in [17] are not applicable to this problem.

The main contributions of this chapter are the l^+-diversity notion and an efficient algorithm for finding the optimal l^+-anonymization. The novelties of the optimal algorithm are as follows:

- It adopts the full subtree generalization model [52] integrated with various suppression schemes, which results in a much larger search space than the full domain generalization model [85], and hence helps improve data utility. It handles suppression as an integral part of anonymization and finds the equilibrium of suppression and generalization, which is different from previous works where suppression is not considered [40][60] or is handled by an external constraint [59][84][85].
- It enumerates all solutions by the depth-first traversal of a novel *cut enumeration tree*. It estimates a *cost lower bound* for the solutions in the subtree rooted at each node, and prunes the subtree if the lower bound exceeds the current best cost.
- It employs an enumeration order that well suits the cost based pruning: finding a solution with a small cost quickly and packing solutions with large costs into subtrees for pruning.
- It is generic in that it works for any reasonable cost metric [17][52] and can easily adapt to a new metric. Extensive experiments show that the approach presented in this chapter outperforms the state of the art works.

The rest of this chapter is organized as follows. Section 3.1 defines the l^+-diversity notion and the optimal anonymization problem, Section 3.2 presents our search strategy and the overall description of the l^+-Optimize algorihtm, Section 3.3 discusses methods for lower bounding information loss, Section 3.4 proposes dynamic pruning techniques that help tighten lower bounds, Section 3.5 instantiates our approach with various information loss metrics, Section 3.6 evaluates our approach, Section 3.7 illustrates the choice of the anonymization model and extensions of the l^+-Optimize algorihtm, and Section 3.8 summarizes this chapter.

3.1 Problem Definition

A data holder wants to publish a person-specific relational table T which contains records $\{t_1,..,t_n\}$ and attributes $\{A_1,..,A_m\}$. As in the literature, we assume that each record represents a distinct individual. One attribute is called the *sensitive attribute*, denoted by SA. Any individual is unwilling to reveal his / her SA value. A subset of the attributes is called the *quasi-identifier*, denoted by QI, which can be used to link with external data. In other words, the QI values of an individual are publicly available. For any $t \in T$, $t[A_i,..,A_j]$ denotes the value sequence of t on attributes $A_i,..,A_j$. t[QI] is called the QI *value sequence* of t, and $t[A_k]$ is called a QI *value* of t for $A_k \in$ QI.

3.1.1 The l^+-Diversity Notion

The data holder publishes an anonymized version T^* of T. An adversary has access to T^*, but not T. The adversary knows a target individual's QI value sequence qi and wants to infer his/her SA value by analysing T^*. The set of anonymized records in T^* whose QI value sequences are qi or can be derived from qi is called an anonymized

partition $<qi>^*$. The corresponding set of the original records in T is called an original partition $<qi>$. Clearly, the adversary can determine $<qi>^*$ and is sure that his/her target transaction is in $<qi>^*$. However, the adversary cannot determine $<qi>$ in general.

Therefore, the *confidence* of the adversary in inferring that s_i is the target individual's SA value is bounded by the percentage of the records that contain s_i in the partition $<qi>^*$, which is denoted as $conf(s_i \mid <qi>^*)$. Our privacy goal is to limit $conf(s_i \mid <qi>^*)$. We also call $conf(s_i \mid part)$ the confidence of s_i in *Part* where *Part* is a partition.

Definition 3.1 (l^+-diversity): Let $\theta_i \in [0, 1]$ be a privacy threshold for each SA value s_i. We say that T^* satisfies l^+*-diversity* if for every QI value sequence qi in T and every SA value s_i, $conf(s_i \mid <qi>^*) \leq \theta_i$. We say that s_i is violated in a partition *part* if $conf(s_i \mid part) > \theta_i$. ■

The key idea of l^+-diversity is to avoid a universal protection, that is, to set a privacy threshold per sensitive value according to its sensitivity.

Table 3.2 Example of an anonymized partition and corresponding original partition

(a) Anonymized partition <Graduate, America>*

8	Graduate	America	Asthma
9	Graduate	America	*
10	Graduate	America	*

(b) Original partition <Graduate, America>

8	Graduate	USA	Asthma
9	Graduate	Canada	Cancer
10	Graduate	Canada	Cancer

Example 3.1: As Cancer, Asthma, and Flu have different sensitivities, the publisher may wish to select 30%, 40%, and 50% as their privacy thresholds respectively. The anonymized partition $<$Graduate, America$>^*$ shown in Table 3.2(a) observes such a privacy requirement while the corresponding original partition <Graduate, America> in Table 3.2(b) does not. ■

An anonymized partition as well as the anonymized version T^* of T is derived by a generalization model with a few suppression schemes as in Section 3.1.2 and 3.1.3.

3.1.2 Generalization Model

The *full subtree generalization* model [52] is selected because it keeps the domain exclusiveness in that the generalized values are exclusive of each other in their domains while multi-dimensional generalization model does not. Although the *full domain generalization* model [85] also keeps the domain exclusiveness, it is quite restrictive and excludes many natural solutions, i.e., its solution space is just a small subset of that of the full subtree generalization model. In other words, the full subtree generalization model [52] considers more solutions thus helps retain more data utility.

With this model, a generalization solution corresponds to one *cut* through every QI attribute's taxonomy tree, where *exactly one* node on each root-to-leaf path in the taxonomy trees is contained in the cut.

Example 3.2: The taxonomy trees for Education and Country are in Figure 3.1. And cut_a is a cut, and so is cut_b. Hereafter, we use Sec as the abbreviation of Secondary, Jr of Junior, Sr of Senior, Uni of University, Ba of Bachelor, Grad of Graduate, Eu of Euope, UK of UK, Fr of France, Am of America, USA of USA, and Ca of Canada. ■

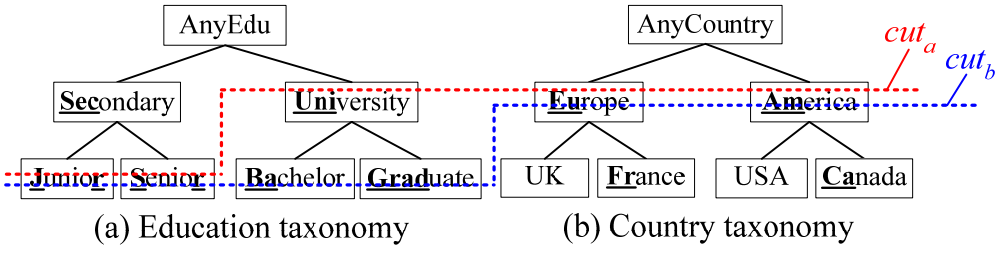


Figure 3.1 Taxonomy trees for QI attributes

Let *Cut* be such a cut. Then, the generalized data is obtained by replacing every QI value in *T* with its taxonomic ancestor in *Cut*. The partition <*qi*> is the set of records in *T* whose generalized QI value sequences are *qi* or a generalization of *qi*. We say <*qi*> is *induced* by *Cut*, and use *Cut.parts* to denote the set of all partitions induced by *Cut*. Partitions in *Cut.parts* are pair-wise disjoint and their union equals to *T*.

Table 3.3 Generalized table by different cuts

(a) Generalized table by Cut_a {Jr, Sr, Uni, Eu, Am}

#	Education	Country	Disease
1	Junior	Europe	Flu
2	Junior	Europe	Asthma
3	Senior	Europe	Flu
4	Senior	Europe	Cancer
5	University	Europe	Asthma
6	University	America	Asthma
7	University	America	Flu
8	University	America	Asthma
9	University	America	Cancer
10	University	America	Cancer

(b) Generalized table by Cut_b {Jr, Sr, Br, Grad, Eu, Am}

#	Education	Country	Disease
1	Junior	Europe	Flu
2	Junior	Europe	Asthma
3	Senior	Europe	Flu
4	Senior	Europe	Cancer
5	Bachelor	Europe	Asthma
6	Bachelor	America	Asthma
7	Bachelor	America	Flu
8	Graduate	America	Asthma
9	Graduate	America	Cancer
10	Graduate	America	Cancer

Example 3.3: The generalized tables in Table 3.3(a)-(b) are derived by generalizing the relational table in Table 3.1(a) to cut_a and cut_b in Figure 3.1(a)-(b). ■

3.1.3 Suppression Schemes

Suppression is integrated with generalization since there are outliers in the data, which can cause excessive generalization and deteriorates data utiltiy. By integrating suppression, outliers can be removed and data utility can be enhanced greatly.

Suppression involves deleting values or records. Suppressing a value can be deemed as generalizing the value to an unknown value *. In particular, we remove the violation by suppressing some records or SA values from *Part* if a partition *Part* violates

the privacy requirement by Definition 3.1. Suppression is independently applied to each partition. We consider several suppression schemes.

The first scheme, denoted by *vioSA*, deletes *all violating SA values* (i.e., values whose confidence exceeds the threshold) and replaces each deleted occurrence with the special *unknown* value *. In the extreme case that all SA values but one occur in *Part*, the adversary can infer that the *unknown* value * stands for the only missing SA value.

The second scheme, denoted by *allSA*, deletes *all SA values* from *Part* if any SA value' confidence in *Part* exceeds its threshold. This scheme incurs more distortion than *vioSA*, but the adversary can not infer what the *unknown* values stand for since all SA values in *Part* are suppressed.

The third scheme, denoted by *vioRec*, deletes a *minimum number of records* for a violating SA value so as to remove the violation in *Part*. If the adversary knows the size of *Part*, he/she may know that some records were deleted.

The fourth scheme, denoted by *allRec*, deletes *all records* from a violating *Part*. This scheme is very safe but it causes the most distortion.

The schemes, *allSA* and *allRec*, are coarse-grained suppression schemes that provide more privacy protection, whereas *vioSA* and *vioRec* are fine-grained suppression schemes that provide less protection. We include all schemes so that the data holder can make a choice based on trade-off between privacy and utility, or choose no suppression at all, denoted by *NoSupp*, in which case only generalization will be performed.

Example 3.4: As shown in Table 3.3(a), cut_a = {Jr, Sr, Uni, Eu, Am} induces 4 partitions, <Jr, Eu> containing record 1 and 2, <Sr, Eu> containing record 3 and 4, <Uni,

Eu> containing only record 5, and <Uni, Am> containing record 6 to 10. Given a uniform privacy threshold $\theta = 50\%$, partition <Uni, Eu> violates the privacy requirement.

And cut_b is derived from cut_a by specializing Uni to Ba and Grad, which splits the partition <Uni, Am> into sub-partitions <Ba, Am> and <Grad, Am> as shown in Table 3.3(b). <Grad, Am> contains records 8, 9 and 10, where Cancer has a confidence = 2/3 > 50%. With the *vioSA* scheme, Cancer is suppressed from records 9 and 10. With the *allSA* scheme, both Cancer and Asthma are suppressed from records 8, 9 and 10. With the *vioRec* scheme, it suffices to suppress either record 9 or 10. With the *allRec* scheme, all 3 records are deleted. ■

In sum, given *Cut* and a suppression scheme, we can produce the *anonymized* T^* from T in two steps. First, we generalize all QI values in T to their taxonomic ancestors in *Cut*. Then, for every QI value sequence *qi* in the generalized data, we apply the chosen suppression scheme to $<qi>^*$ to remove violations.

3.1.4 Optimal Anonymization

3.1.4.1 Information Loss Denoted by Cost Function

We want to find an optimal anonymization solution, defined by a cut and the chosen suppression scheme, that maximizes the data utility, i.e., minimizes the information loss incurred by the anonymization process.

Our approach for finding the optimal solution is generic, as it will be shown later, in that it does not depend on the metric used in measuring the information loss. Therefore, we only introduce a notation in the following and will discuss concrete metrics later.

Given *Cut* and the chosen suppression scheme, let *cost*(•, *Cut*) denote the *anonymization cost*, i.e., the information loss incurred by generalization and suppression, where • can be a table, a partition, a record, or a value, and the chosen suppression scheme is omitted from the parameter list for brevity. Section 3.5 will discuss the cost metrics for computing *cost*(•, *Cut*).

3.1.4.2 Optimal l^+-Anonymization

Definition 3.2 (Optimal l^+-anonymization): Given a table T, the taxonomy trees H for the QI attributes of T, a chosen suppression scheme, and the privacy threshold θ_i for each SA value s_i, find a cut Cut_{best} on H such that T^* defined by Cut_{best} and the chosen suppression scheme satisfies l^+-diversity by Definition 3.1 and $cost(T, Cut_{best}) = min_{cut}$ $cost(T, cut)$ for all possible cuts. The chosen suppression scheme and Cut_{best} comprise an optimal solution. ■

The number of all possible solutions (cuts) is huge as it is exponential to the number of internal nodes on the taxonomy trees according to the following formula. Given a taxonomy tree with a root R, the number of possible cuts is recursively computed by $\#cuts(R) = 1 + \prod_{r \in children(R)} \#cuts(r)$, where $children(R)$ is the set of child nodes of R.

Therefore, it is computationally hard to find an optimal solution. What make our approach practical are the three key elements: the search strategy, cost lower bounding methods, and dynamic pruning techniques.

3.2 Search Strategy and the l^+-*Optimize* Algorithm

In this section, we focus on how to organize and search all possible anonymizations, and present the generic algorithm.

3.2.1 Cut Enumeration Tree

Given a chosen suppression scheme, an anonymization is solely represented by a cut through the taxonomy trees of all QI attributes, which is a set of generalized values. We introduce the *cut enumeration tree* to enumerate all cuts. In the cut enumeration tree, each node represents a cut and is denoted by the set of its constituent values. The root represents the most generalized cut consisting of only the root values of the QI taxonomy trees. There is an edge from a parent node *Cut* down to a child node Cut_{child} if Cut_{child} is derived by specializing exactly one constituent value of *Cut*. *subtree*(*Cut*) denotes the subtree rooted at the node *Cut*. The term "cut" and the representing "node" are interchangeable.

To avoid duplicate enumeration of cuts, constituent values of *Cut* are divided into two types, namely, open values and locked values. An open value will be further specialized in *subtree*(*Cut*) and a locked value will not. The list of open values is denoted by *Cut.openVs*, and the list of locked values is denoted by *Cut.lockedVs*, which are determined as follows.

For the cut represented by the root of the cut enumeration tree, all constituent values of the cut are open values. Let $Cut.openVs = \{v_1, ..., v_m\}$. Then, the *i*-th child of *Cut*, denoted by Cut_i, is derived by specializing the *i*-th open value v_i to its taxonomic child values, $children(v_i)$. For Cut_i, $Cut_i.lockedVs$ contains all values in *Cut.lockedVs*, all values in $\{v_1, .., v_{i-1}\}$, and values in $children(v_i)$ that are leaf values on the QI taxonomy trees; $Cut_i.openVs$ contains all values in $\{v_{i+1}, .., v_m\}$ and values in $children(v_i)$ that are non-leaf values on the QI taxonomy trees. We will discuss the impact of value ordering on the search space pruning in Section 3.4.

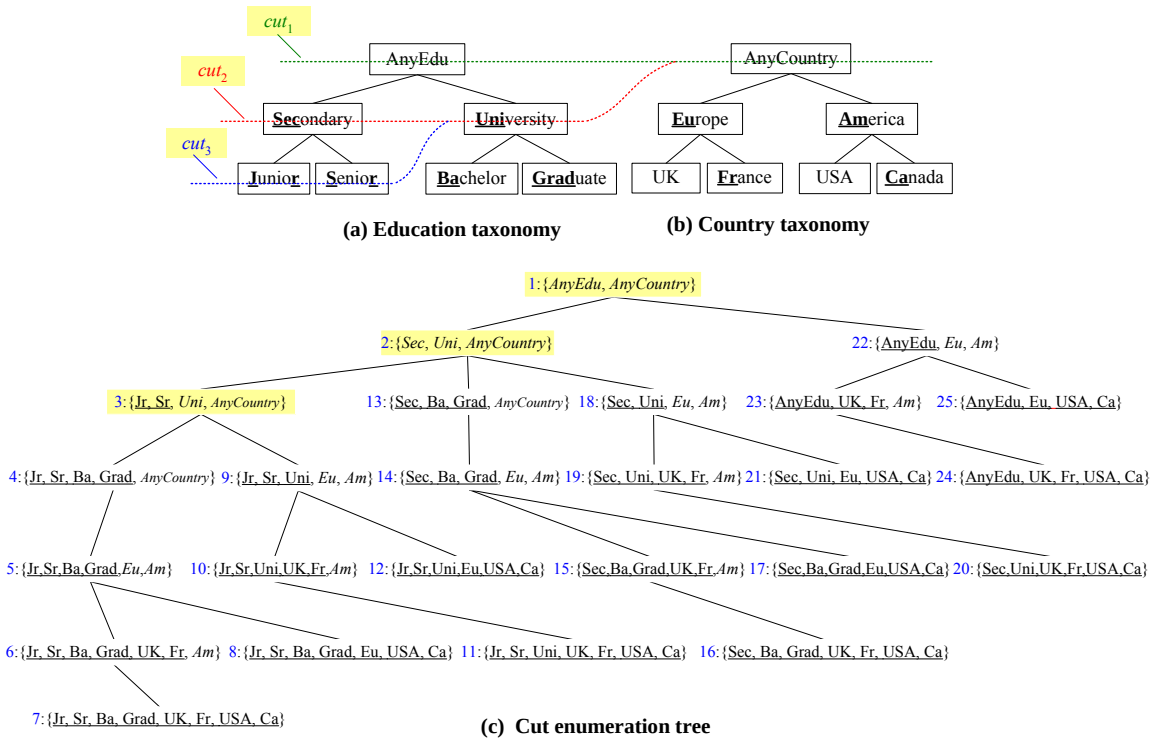


Figure 3.2 Cut enumeration tree enumerating all cuts on QI taxonomy trees

Example 3.5: The cut enumeration tree in Figure 3.2(c) represents all the 25 possible cuts on the taxonomy trees in Figure 3.2(a)-(b). Notice that Figure 3.2(a)-(b) reproduce Figure 3.1(a)-(b) just for convenience. All values in a cut are listed in the left-to-right order on the taxonomy trees. Open values are in italic and locked values are underlined. All constituent values of Cut_1 are open. Cut_2 is derived from Cut_1 by specializing AnyEdu to Sec and Uni. In Cut_2, Sec and Uni are open since they are non-leaf values on the Education taxonomy tree, and AnyCountry remains open. Cut_{22} is derived from Cut_1 by specializing AnyCountry to Eu and Am. In Cut_{22}, AnyEdu is locked since it is listed before AnyCountry in $Cut_1.openVs$. ■

Theorem 3.1 (Completeness and compactness): Every cut is enumerated exactly once by the cut enumeration tree.

Proof: All the cuts form a lattice according to the generalization relationships. By the construction of the cut enumeration tree, the edge between every cut node and its left most parent node on the *lattice* is retained on the *tree*, so every cut node is retained on the tree. In other words, the cut enumeration tree is complete. Any traversal algorithm will visit every cut node just once. In that sense, the cut enumeranstruction tree is compact. ■

3.2.2 Search Strategy

A complete traversal of the cut enumeration tree would yield an optimal cut, but it is computationally infeasible because the search space is huge. For example, the entire search space of the Adult dataset [12][52] consists of 356 million cuts. It is critical to prune any subspace if it contains no optimal cut.

We observe the following property of the cut enumeration tree. First, cuts are arranged from general to specific in the top-down direction. Second, cuts are arranged from specific to general in the left-to-right direction because for any cut, its i-th subtree specializes one more `open` value than its (i+1)-th subtree, e.g., in Figure 3.2(c), Cut_2 and Cut_{22} are children of Cut_1, *subtree*(Cut_2) specializes one more `open` value, AnyEdu, than *subtree*(Cut_{22}). Therefore, specific cuts are at the lower left portion of the cut enumeration tree. Usually, specific cuts have smaller costs than general cuts.

To make small the running cost that will be used for pruning, we should search specific cuts, i.e., those at the lower left portion of the tree, as early as possible. The depth-first search suits exactly this strategy.

The depth-first search with pruning. We start from the most generalized cut. Let *Cut* be the cut that we are currently visiting, and Cut_{best} be the best cut, i.e., the cut

with the smallest cost (a.k.a. information loss), examined so far. We first estimate a *lower bound* on the costs of the cuts in *subtree*(*Cut*), denoted by $LB_{cost}(T, Cut)$, that must be no more than the actual cost, $cost(T, U)$, for *every* cut U in *subtree*(*Cut*). The estimation of $LB_{cost}(T, Cut)$ will be discussed in Section 3.3. If the *pruning condition*

$$LB_{cost}(T, Cut) \geq cost(T, Cut_{best})$$

holds, we can prune the entire *subtree*(*Cut*) without missing an optimal cut. If the condition does not hold, we update Cut_{best} if applicable, and then get to the next child Cut_{child} of *Cut*, in the left-to-right order, in a specialization step. After visiting *subtree*(Cut_{child}), we return to *Cut* in a backtracking step. We will see in Section 3.4 that a proper order of the values in a cut helps reduce $cost(T, Cut_{best})$ and also helps tighten up $LB_{cost}(T, Cut)$, and hence maximizes the chance to satisfy the pruning condition.

Specialization Step. A specialization step brings us from *Cut* down to the next child Cut_{child} by specializing the next open value v in *Cut.openVs* into *children*(v). The partitions (induced by *Cut*) containing value v are split into sub-partitions (induced by Cut_{child}) containing values in *children*(v). And $cost(T, Cut_{child})$ can be incrementally calculated based on $cost(T, Cut)$ and the costs of those sub-partitions. $LB_{cost}(T, Cut_{child})$ is computed incrementally, too.

Backtracking Step. After searching *subtree*(Cut_{child}), the search backtracks to the parent *Cut*. Recall that Cut_{child} was derived by specializing an open value v of *Cut*. In the backtracking, we generalize *children*(v) to v and restore the partitions induced by *Cut*. In addition, the value v becomes locked in the rest of *subtree*(*Cut*). This change helps tighten up $LB_{cost}(T, Cut)$.

Example 3.6: Consider Figure 3.2(c). Cut_1 induces one partition, <AnyEdu, AnyCountry>, consisting of all records of *T*. The step from Cut_1 down to Cut_2 specializes AnyEdu to Sec and Uni, and splits <AnyEdu, AnyCountry> into sub-partitions <Sec, AnyCountry> and <Uni, AnyCountry>. After traversing *subtree*(Cut_2), the depth-first search backtracks to Cut_1 by generalizing Sec and Uni to AnyEdu, and merging the sub-partitions <Sec, AnyCountry> and <Uni, AnyCountry> to restore the partition <AnyEdu, AnyCountry>. In the rest of *subtree*(Cut_1), AnyEdu becomes locked. ■

3.2.3 Search Algorithm

Our search algorithm is listed as Algorithm 3.1. It searches *subtree*(*Cut*) and updates Cut_{best}. *Cut* and Cut_{best} are initialized to the most generalized cut, *Cut.openVs* contains all root values on the QI taxonomy trees, and *Cut.lockedVs* is empty. Without confusion, we abbreviate *Cost*(*T*, *Cut*) to *Cost*(*Cut*), and abbreviate $LB_{cost}(T, Cut)$ to $LB_{cost}(Cut)$ in the algorithm. Moreover, the chosen suppression scheme and the privacy threshold θ_i for each SA value s_i are omitted from the parameter list for brevity.

Basically, for each open constituent value *v* of *Cut* in the listed order (line 1), the algorithm performs the following: first check whether *subtree*(*Cut*) can be pruned (line 2), if yes, it breaks the for-loop, backtracks (line 8) and returns Cut_{best}; if not, it derives the child cut, Cut_{child}, by specializing *v* (line 3), and checks the pruning condition at Cut_{child} (line 4). If the condition holds, *subtree*(Cut_{child}) is pruned and the algorithm backtracks to *Cut* (line 4), otherwise *subtree*(Cut_{child}) is searched recursively (line 5-7). When finishing the search of *subtree*(*Cut*), the algorithm backtracks to the parent of *Cut* (line 8) and returns the best cut searched so far (line 9). There are two ways to make the

pruning effective. One is to have a large $LB_{cost}(T,Cut)$, and the other is to have a small $cost(T,Cut_{best})$. These are the topics of the next two sections.

Algorithm 3.1 The optimal anonymization algorithm for l^+-diversity

---

l^+-*Optimize*(Cut, Cut_{best})

Input: Cut is the current cut whose subtree is going to be searched
Output: Cut_{best} is the cut with the minimum cost searched so far

1. **for each** value v **in** $Cut.openVs$ **do**
2. **if** $LB_{cost}(Cut) \geq cost(Cut_{best})$ **then goto** 8
3. $Cut_{child} \leftarrow$ Specialization(Cut, v)
4. **if** $LB_{cost}(Cut_{child}) \geq cost(Cut_{best})$ **then** Backtracking(Cut_{child})
5. **else**
6. **if** $cost(Cut_{child}) < cost(Cut_{best})$ **then** $Cut_{best} \leftarrow Cut_{child}$
7. $Cut_{best} \leftarrow l^+$-*Optimize*($Cut_{child}$, Cut_{best})
8. Backtracking(Cut)
9. **return** Cut_{best}

3.3 Cost Lower Bounding

In this section, we first present a generic lower bound, and then tighten it up for individual suppression schemes.

3.3.1 Introduction to Cost Lower Bounding

A lower bound on costs of *subtree*(Cut), $LB_{cost}(\bullet,Cut)$, must be no more than the actual cost, $cost(\bullet,U)$, for *every* cut U in *subtree*(Cut), where $\bullet$ is a set of records in T. Let $Cut.parts$ be the set of partitions induced by Cut. Notice that costs can be decomposed as

$$cost(T, Cut) = \sum_{Part \in Cut.parts} cost(Part, Cut)$$

and

$$cost(Part, Cut) = \sum_{t \in Part} cost(t, Cut).$$

Therefore, we can also decompose the lower bounds as follows.

$$LB_{cost}(T, Cut) = \sum_{Part \in Cut.parts} LB_{cost}(Part, Cut).$$

In other words, we determine a cost lower bound for each partition *individually* and add up all partitions' lower bounds. To estimate the lower bound, we exploit the following property satisfied by most reasonable cost metrics.

Observation 3.1 (Metric monotonicity): We say that a cost metric satisfies the *metric monotonicity* if (a) the generalization cost decreases when specializing constituent values of a cut; (b) the cost on generalizing a record is no more than the cost on suppressing the record. ■

From now on, we estimate $LB_{cost}(Part, Cut)$ with a cost metric satisfying the metric monotonicity. A simple case is that, for every open value v in Cut, all taxonomic leaf descendants of v occurring in $Part$ are the same. In this case, we say that $Part$ is *homogeneous*. Essentially, a homogeneous $Part$ never splits if specialized to any cut in $subtree(Cut)$, therefore, the suppression cost for $Part$ is the same at every cut in $subtree(Cut)$. So we can estimate $LB_{cost}(Part, Cut)$ by $cost(Part, Cut_{spec})$, where Cut_{spec} denotes the *most specific cut* in $subtree(Cut)$ and is derived by specializing all open values in Cut to the leaf level on the QI taxonomy trees.

Now we consider the case that $Part$ induced by Cut is not homogeneous. Then, $Part$ may have a different suppression cost at each cut in $subtree(Cut)$. So, it does not work to just consider Cut_{spec}. To estimate $LB_{cost}(Part, Cut)$, we estimate the lower bound for each record in $Part$ and sum up the lower bound over all records in $Part$.

Notice that different suppression schemes have different interaction with generalization. Thus, we have different ways to aggregate these two types of cost

components (information loss) as summarized by Table 3.4. Simply put, for *vioSA* and *allSA*, the cost is the sum of generalization cost and suppression cost. And for *vioRec* and *allRec*, the cost is either suppression cost or generalization cost, depending on if the record is suppressed.

Specifically, for a single record t in *Part*, the components of $cost(t, U)$ for a cut U in $subtree(Cut)$ has two possible compoents. The first component, $cost\_s(t,U)$, is the suppression cost related to t. For the *vioSA* or *allSA* scheme, $cost\_s(t, U)$ is the cost for suppressing t's SA value, and for the *vioRec* or *allRec* scheme, $cost\_s(t, U)$ is the cost for suppressing the record t. The second component, $cost\_g(t, U)$, is the cost for generalizing t to the cut U. With the *vioSA* or *allSA* scheme, t may be generalized while t's SA value is suppressed, and with the *vioRec* or *allRec* scheme, t is generalized only if t is not suppressed. The last column of Table 3.4 summarizes the cost of t for each case. In all cases, if we can estimate *cost_g* and *cost_s* separately, we can estimate the lower bound of the cost of t.

Table 3.4 Possible components of *cost*(*t, U*)

Suppression scheme	Condition	Components
vioSA, *allSA*	SA value of t not suppressed	*cost_g*
	SA value of t is suppressed	*cost_g* + *cost_s*
vioRec, *allRec*	Record t not suppressed	*cost_g*
	Record t is suppressed	*cost_s*

3.3.2 Lower Bound on Generalization Cost

The metric monotonicity property (b) in Observation 3.1(b) implies that generalizing a record has no more cost than suppressing the record. So, we can estimate $LB_{cost}(Part, Cut)$ by treating each suppressed record in *Part* as if it were generalized (i.e.,

no record was suppressed), and examining the most specific cut in *subtree*(*Cut*), i.e., Cut_{spec}. The next lemma formalizes this idea.

Lemma 3.2 (LB on generalization cost): For *Part* induced by *Cut*, *cost_g*(*Part*, Cut_{spec}) is an estimate of LB_{cost}(*Part*, *Cut*), i.e., $cost\_g(Part, Cut_{spec}) \leq cost(Part,U)$ for *every* cut *U* in *subtree*(*Cut*).

Proof: For any cut *U* in *subtree*(*Cut*) and any record $t \in Part$, as summarized by Table 3.4, *cost_g*(*t*, *U*) is always a component of *cost*(*t*, *U*) except when *t* is suppressed with the *vioRec* or *allRec* scheme. In this exceptional case, *cost_s* is the only component, and by the metric monotonicity (b), $cost\_s(t, U) \geq cost\_g(t, U)$. For all cases, $cost(t, U) \geq cost\_g(t, U)$, so $cost(Part,U) \geq cost\_g(Part,U)$. By the metric monotonicity property (a) in Observation 3.1(a), $cost\_g(Part, U) \geq cost\_g(Part, Cut_{spec})$. The lemma holds. ■

3.3.3 Lower Bound on Suppression Cost

Similarly, let us estimate LB_{cost}(*Part*, *Cut*) by only counting the suppression cost. For this goal, given the unit cost for suppressing an occurrence of a SA value s_i, it suffices to estimate the minimum number of occurrences of s_i that must be suppressed by any cut in *subtree*(*Cut*), in order to satisfy l^+-diversity. The following observation enables us to do so.

Observation 3.2 (Confidence monotonicity): If the confidence of a SA value s_i in *Part* exceeds the threshold θ_i, then s_i's confidence in some sub-partitions split from *Part* must exceed θ_i. ■

This observation discusses the case when an indistinguishable group of records is partitioned into pair-wise disjoint subsets because of specialization. A similar observation is proved in [67][100].

Lemma 3.3 (Minimum number of suppression): For a partition *Part* of size n, induced by *Cut*, if a SA value s_i with n_i occurrences in *Part* violates the privacy threshold θ_i, i.e., $n_i / n > \theta_i$, then to satisfy θ_i, at least

$$\#MinRm(Part, s_i) = \lceil (n_i - \theta_i \cdot n) / (1 - \theta_i) \rceil$$

occurrences of s_i in *Part* must be suppressed in *subtree*(*Cut*).

Proof: Consider any cut U in *subtree*(*Cut*) and *Part*'s sub-partitions induced by U. Let $\#rm(s_i)$ be the total number of occurrences of s_i that must be suppressed from these sub-partitions in order to satisfy the privacy threshold θ_i.

First, let us consider the *vioSA* and *allSA* schemes. Let $SPs^{(1)}$ denote the union of the sub-partitions that violate the threshold θ_i, and $SPs^{(2)}$ denote $Part - SPs^{(1)}$. Then, $SPs^{(2)}$ holds $n_i - \#rm(s_i)$ occurrences of s_i. Since every sub-partition in S$Ps^{(2)}$ satisfies the threshold θ_i, so does their union by Observation 3.2, that is, $(n_i - \#rm(s_i)) / |SPs^{(2)}| \le \theta_i$. Note that $|SPs^{(2)}| = n - |SPs^{(1)}|$ and $|SPs^{(1)}| \ge \#rm(s_i)$. So $|SPs^{(2)}| \le n - \#rm(s_i)$ and $(n_i - \#rm(s_i)) / (n - \#rm(s_i)) \le \theta_i$. Solving this equation, we get $\#rm(s_i) \ge \lceil (n_i - \theta_i \cdot n) / (1 - \theta_i) \rceil$.

Now, let us consider the *vioRec* and *allRec* schemes. The occurrences of s_i after the suppression is $n_i - \#rm(s_i)$, and the total size of all sub-partitions after suppressing records holding s_i is $n - \#rm(s_i)$ for the *vioRec* scheme and at most $n - \#rm(s_i)$ for the *allRec* scheme. Since after suppression, all remaining sub-partitions satisfy θ_i, $(n_i - \#rm(s_i)) / (n - \#rm(s_i)) \le \theta_i$. Therefore, we get $\#rm(s_i) \ge \lceil (n_i - \theta_i \cdot n) / (1 - \theta_i) \rceil$. ■

With the *vioRec* or *allRec* scheme, suppressing a record holding s_i increases confidences of other SA values. Lemma 3.3 should be iteratively applied until there is no violation. The next example illlustrates this.

Example 3.7: Consider again *T* in Table 3.1(a). Partition <Uni, AnyCountry> consists of records 5 to 10, with Asthma occurring 3 times, Cancer twice, and Flu once. Given a uniform threshold $\theta = 1 / 3$, Asthma violates the threshold. By Lemma 3.3, *#MinRm* for Asthma is $\lceil (3 - 6 \cdot 1 / 3) / (1 - 1 / 3)) \rceil = 2$. If the *vioSA* or *allSA* scheme is used, the estimation is done. If the *vioRec* or *allRec* scheme is used, suppressing 2 records holding Asthma causes Cancer to violate the threshold. So, we recursively apply Lemma 3.3 for Cancer, and get its $\#MinRm = \lceil (2-4 \cdot 1/3) / (1-1/3)) \rceil = 1$. ■

3.3.4 Integrated Lower Bounds

Putting Lemma 3.2 and Lemma 3.3 together yields the first lower bound integrating both generalization and suppression costs.

Theorem 3.4 (Generic LB): Let $\#MinRm(Part, s_i)$ be determined by Lemma 3.3, and c_i be the unit cost for suppressing one occurrence of s_i. Then,

$$max \{ \sum_i \#MinRm(Part, s_i) \cdot c_i,\ cost\_g(Part, Cut_{spec}) \}$$

is an estimate of $LB_{cost}(Part, Cut)$. ■

This lower bound applies to all suppression schemes. Below, we derive a tighter, i.e., a larger, lower bound for each suppression scheme by exploiting the interaction of the cost for generalization and suppression as shown in Table 3.4. With the *vioSA* or *allSA* scheme, *cost_g*(*t*, *Cut*) is *always* a component of *cost*(*t*, *Cut*), and if the SA value in

t is suppressed, $cost\_s(t, Cut)$ is an *additional* component. So, we can add up these two components to get a tighter lower bound.

Theorem 3.5 (LB for *vioSA* and *allSA*): Let $\#MinRm(Part, s_i)$ be determined by Lemma 3.3, and c_i be the unit cost for suppressing one occurrence of s_i. If the *vioSA* or *allSA* scheme is used, then

$$\sum_i \#MinRm(Part, s_i) \cdot c_i + cost\_g(Part, Cut_{spec})$$

is an estimate of $LB_{cost}(Part, Cut)$. ■

With the *vioRec* or *allRec* scheme, for any record t, $cost(t, Cut)$ is equal to $cost\_g(t, Cut)$ if t is not suppressed, or equal to $cost\_s(t, Cut)$ if t is suppressed. In other words, t has *exactly one* cost component, not both. Theorem 3.5 does not apply to this case. Theorem 3.4 applies, but gets a lower bound that is not tight enough for some cost metrics.

To derive a tighter lower bound for the *vioRec* or *allRec* scheme, we shall estimate the lower bound for the suppressed records and the remaining records *separately* and *add them up*. For any cut U in $subtree(Cut)$, at least $d = \sum_i \#MinRm(Part, s_i)$ records in *Part* will be suppressed by Lemma 3.3. Let $n = |Part|$, for the remaining $n - d$ records in *Part*, their *cost* is no less than their $cost\_g$, which reaches the minimum at Cut_{spec} by the metric monotonicity (a). As each record may have a different $cost\_g$ at Cut_{spec}, we consider the set of $n - d$ records in *Part* that have smallest $cost\_g$ at Cut_{spec}. This set is defined by

$$SP_{spec} = arg\ min_{SP \subset Part \wedge |SP|=n-d}\ cost\_g(SP, Cut_{spec}).$$

The *cost_g* of SP_{spec} gives a lower bound on *cost_g* for the remaining $n - d$ records in *Part*.

Theorem 3.6 (LB for *vioRec* and *allRec*): Let $n = |Part|$, $d = \sum_i \#MinRm(Part, s_i)$ be determined by Lemma 3.3, and c_i be the unit cost for suppressing one record holding s_i. If the *vioRec* or *allRec* scheme is used, then

$$\sum_i \#MinRm(Part, s_i) \cdot c_i + cost\_g(SP_{spec}, Cut_{spec})$$

is an estimate of $LB_{cost}(Part, Cut)$. ■

In general, to determine SP_{spec} requires examining every size $n - d$ subset of *Part*, which is computationally expensive. However, if the following property holds, we have an efficient computation.

Observation 3.3 (Independence property): We say that the *independence property* is observed if suppressing a record from *Part* induced by *Cut* does not affect *cost_g*(*t, Cut*) for any other record *t* in *Part*. ■

The independence property is always observed with the *allRec* scheme because suppressing any record *t* entails suppressing all records in *t*'s partition. With the independence property, *cost_g*(SP_{spec}, Cut_{spec}) equals to the sum of the smallest $n - d$ terms in *cost_g*(*Part*, Cut_{spec}) for $n - d$ records.

Corollary 3.7 (Simplifying LB for *vioRec* and *allRec*): If the independence property holds, *cost_g*(SP_{spec}, Cut_{spec}) in Theorem 3.6 can be replaced by the sum of the smallest $n - d$ terms in *cost_g*(*Part*, Cut_{spec}) for $n - d$ records, where $n = |Part|$ and $d = \sum_i \#MinRm(Part, s_i)$. ■

3.4 Techniques for Improving the Pruning

The previous section presented methods for estimating a cost lower bound with a *given* order for enumerating cuts. The effectiveness of the pruning, i.e., the likelyhood for the pruning condition, $LB_{cost}(T,Cut) \geq cost(T,Cut_{best})$, being satisfied also depends on such an order for enumerating cuts and the way for updating the lower bound. In this section, we discuss the details of these issues.

3.4.1 Ordering Values in Cuts

Recall that *subtree*(*Cut*) is pruned if $LB_{cost}(T, Cut) \geq cost(T, Cut_{best})$. We can maximize this pruning in two ways: (1) having a small $cost(T, Cut_{best})$, which can be achieved by examining cuts with small costs as early as possible, and (2) having a large $LB_{cost}(T, Cut)$, which can be achieved by packing cuts with large costs into *subtree*(*Cut*). Notice that the order of values in *Cut.openVs* determines the order of examining cuts in *subtree*(*Cut*), and hence has a great impact on pruning the search space.

First, let us consider (1). Roughly speaking, the more specific a cut is, the smaller its cost is. Thus, to reduce $cost(T, Cut_{best})$ we should first specialize general values which usually have high occurrences. So, we should order the values of *Cut.openVs* in the descending order of their occurrences. For example in Figure 3.3, value v_1 is arranged before value v_2 in Cut_1. So, $Cut_2 = \{children(v_1), v_2, \ldots\}$ is examined before $Cut_3 = \{v_1, children(v_2), \ldots\}$. Let $O(v)$ denote the occurrences of a value v. If $O(v_1) > O(v_2)$, it is more likely that Cut_2 has a smaller cost than Cut_3, and hence examining Cut_2 first has a better chance to reduce $cost(T, Cut_{best})$.

Now we consider (2). If a child cut is derived by specializing a value v in *Cut.openVs*, v becomes locked when backtracking to *Cut* from the child cut.

Subsequently, in the rest of *subtree*(*Cut*), *v* remains locked and hence contributes to $LB_{cost}(T, Cut)$ a cost component proportional to $O(v)$. Therefore, if we arrange the values *v* in *Cut.openVs* in the descending order of O(*v*), *v* with a larger O(*v*) will become locked earlier, thus contribute a larger cost component to $LB_{cost}(T, Cut)$ earlier. For example, at Cut_3 in Figure 3.3, value v_1 changes from open to locked. With all other things being equal, $LB_{cost}(T, Cut_3)$ is proportional to $O(v_1)$. If $O(v_1) > O(v_2)$, $LB_{cost}(T, Cut_3)$ is likely tighter than if $O(v_1) < O(v_2)$. We summarize the above discussion as follows.

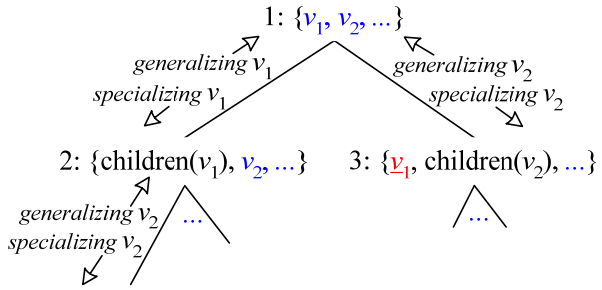


Figure 3.3 Value ordering vs. pruning and splitting / merging of partitions

Observation 3.4 (Ordering values in a cut): Arranging the values of *Cut.openVs* in the descending order of occurrences helps produce a small $cost(T, Cut_{best})$ and a large $LB_{cost}(T, Cut)$, thus improving the cost lower bounding based pruning. ■

3.4.2 Updating Lower Bounds

Upon backtracking to *Cut*, one more constituent value of *Cut* becomes locked, making the most specific cut Cut_{spec} in the rest of *subtree*(*Cut*) more general. Also, having more locked values will increase the chance for partitions to be homogenous. We can use this chance to tighten up the lower bound for the rest of *subtree*(*Cut*).

Example 3.8: Consider Figure 3.2(c) again. When first visiting Cut_{18} = {Sec, Uni, Eu, Am}, Eu is open and the most specific cut in *subtree*(Cut_{18}) is Cut_{20}. The partition

<Sec, Eu> consists of records 1 to 4 and is not homogenous. When backtracking from Cut_{19} to Cut_{18}, Eu becomes locked, so the most specific cut in the rest of *subtree*(Cut_{18}) is Cut_{21}, more general than Cut_{20}. The partition <Sec, Eu> is now *homogeneous*, its lower bound is its actual cost. The rest of *subtree*(Cut_{18}) may be pruned without visiting Cut_{21}.■

Observation 3.5 (Updating the lower bound when backtracking): Starting from *Cut*, we get to a child cut by specializing an open value *v* of *Cut*. When backtracking to *Cut*, *v* becomes locked in the rest of *subtree*(*Cut*). The lower bound for each partition containing *v* induced by *Cut* can be tightened up as *v* is locked. ■

In general, *Cut* has multiple children, $Cut_1,\ldots,Cut_k$. Each time backtracking to *Cut* from a child Cut_i, $LB_{cost}(T, Cut)$ is updated by one additional locked value in *Cut* as discussed above. In these updates, $LB_{cost}(T, Cut)$ is monotonically increasing, and $cost(T, Cut_{best})$ is monotonically decreasing. So, each update improves the chance to satisfy the pruning condition $LB_{cost}(T, Cut) \geq cost(T, Cut_{best})$. This idea is incorporated in the backtracking step of l^+-Optimize where $LB_{cost}(T, Cut)$ are incrementally updated by recalculating the lower bound of each partition containing value *v*, and the pruning condition is always rechecked before specializing to a new child. Experiments show that with all other techniques combined, 99% of the search space is pruned, and with this technique in addition, $99.{}_{97}$% to $99.{}_{99}$% is pruned. This additional 0.97% to 0.99% pruning rate is critical for pruning a huge search space.

3.4.3 Reducing Data Scan

The cut enumeration tree is just a conceptual vehicle for describing the search space and strategy; it is never materialized in its entirety in the memory. In the depth-first

search, we only materialize the partitions induced by the current cut. So, each specialization step is accompanied by splitting partitions, and each backtracking step is accompanied by merging partitions. Consider the example in Figure 3.3 again, where value v_1 involves 1 specialization / generalization step, while value v_2 involves 2 specialization / generalization steps. Thus the overhead of splitting / merging partitions is proportional to $1 \cdot O(v_1) + 2 \cdot O(v_2)$, where $O(v)$ denotes the occurrences of value v. Clearly, $1 \cdot O(v_1) + 2 \cdot O(v_2)$ is smaller if $O(v_1) > O(v_2)$ than if $O(v_1) < O(v_2)$. Therefore, with the values in *Cut.openVs* being arranged in the descending order of occurrences, the overhead of splitting / merging partitions is reduced.

Observation 3.6 (Reducing data scan): Examining the constituent values of a cut in the descending order of occurrences reduces partition splitting / merging overhead. ■

3.5 Instantiation with Cost Metrics

Our approach is generic in that it is independent of the cost metric used. The only requirement for our lower bounding method is that the metric observe the monotonicity property (Observation 3.1) and the independence property (Observation 3.3) required in estimating the lower bound on anonymization cost, i.e., information loss as discussed in Section 3.3. In this section, we instantiate our approach with several widely used cost metrics by establishing these properties.

3.5.1 Loss Metric, LM

Iyengar [52] presented the *loss metric*, LM, to quantify the information loss by measuring the ambiguity introduced by generalization.

3.5.1.1 Generalization Cost with LM

If a leaf value v is generalized to a value v^* in the QI taxonomy tree rooted at R,

$$cost\_g(v, Cut) = (\#leaves(v^*) - 1) / (\#leaves(R) - 1),$$

where $\#leaves(v^*)$ and $\#leaves(R)$ denote the numbers of leaves in $subtree(v^*)$ and $subtree$(R) respectively. And $cost\_g(t, Cut)$ is the sum of $cost\_g(t[A], Cut)$ for all QI attributes A,

$$cost\_g(t, Cut) = \sum_{A \in QI} cost\_g(t[A], Cut)$$

3.5.1.2 Suppression Cost with LM

Suppressing a value can be deemed as generalizing the value to an unknown value *. It is reasonable to assume that each SA value s_i is at least as important as all QI values in the record t holding s_i. In other words, suppressing s_i has at least as much cost as generalizing all QI values in t to the most generalized ones. Therefore, with the *vioSA* or *allSA* scheme,

$$cost\_s(t, Cut) = c_i \geq |QI|;$$

with the *vioRec* or *allRec* scheme, both the QI values and the SA value are suppressed, so

$$cost\_s(t, Cut) = c_i \geq 2|QI|.$$

3.5.1.3 Properties of LM

LM complies with the metric monotonicity (a) presented in Observation 3.1 because $\#leaves(x)$ increases when generalizing value x, and observes the metric monotonicity (b) because $cost\_g(t, Cut) \leq |QI| \leq cost\_s(t, Cut)$. LM observes the independence property since $cost\_g$ of t is independent of other records.

3.5.2 Discernibility Metric, DM

Bayardo and Agrawal [17] proposed the *discernibility metric*, DM, which attempts to capture the desire to maintain *discernibility* between records as much as is allowed.

3.5.2.1 Generalization Cost with DM

DM assigns a penalty to each record t based on the size of the indistinguishable group holding t, i.e., based on how many records are indistinguishable from t on QI,

$$cost\_g(t, Cut) = |Part^*|$$

where $Part^*$ is the anonymized partition holding t.

3.5.2.2 Suppression Cost with DM

DM considers only record suppression, so we only use DM with the *vioRec* and *allRec* schemes. DM assigns a penalty, equal to the size of the relational table T, to a suppressed record t,

$$cost\_s(t, Cut) = c_i = |T|.$$

3.5.2.3 Properties of DM

DM observes the metric monotonicity (a) because the partition size increases when generalizing values, and observes the metric monotonicity (b) because $cost\_g(t, Cut) = |Part^*| \leq |T| = cost\_s(t, Cut)$. DM with the *vioRec* scheme does not observe the independence property since suppressing a record from a partition reduces the size of the partition, which affects the generalization cost of the remaining records in the partition.

However, the independence property always holds for the *allRec* scheme regardless of the cost metric. So, Corollary 3.7 is applicable to DM if the *allRec* scheme is used.

3.5.3 Classification Metric, CM

Iyengar [52] presented the *classification metric*, CM. The purpose is to optimize the anonymized data for training a classifier, where each record in the relational table belongs to a particular class indicated by a class label. CM assigns no penalty to an unsuppressed record that belongs to the majority class in its indistinguishable group.

3.5.3.1 Generalization Cost with CM

Let $Part^*$ be the anonymized partition holding t, i.e., the indistinguishable group of records after suppression. The set of records in $Part^*$ whose class label has the biggest occurrences is the majority class of $Part^*$. And let $minority(Part^*)$ be the set of records in $Part^*$ belonging to minority classes, i.e., not belonging to the majority class.

If $t \in minority(Part^*)$,

$$cost\_g(t, Cut) = 1,$$

else

$$cost\_g(t, Cut) = 0.$$

Notice that the cost does not depend on the specific level of generalization. The rationale is just to penalize the impurity of the anonymized data.

3.5.3.2 Suppression Cost with CM

CM only considers record suppression, so we only use the *vioRec* and *allRec* schemes with CM. Suppose that t holds a SA value s_i, if t is suppressed,

$$cost\_s(t, Cut) = c_i = 1.$$

Notice that suppression of a record t may change the majority class and the minor classes of an indistinguishalbe group of records, therefore, may affect *cost_g*.

3.5.3.3 Properties of CM

CM satisfies the metric monotonicity (a) because $|minority(P_1 \cup P_2)| \geq |minority(P_1)| + |minority(P_2)|$ (proven by [17]), which implies that the generalization cost never increases by splitting a partition. CM satisfies the metric monotonicity (b) because $cost\_g(t, Cut) \leq 1 = cost\_s(t, Cut)$. CM with the *vioRec* scheme does not observe the independence property because suppressing a record from a majority class could cause the minority class to become a majority class. So, Corollary 3.7 is applicable to CM if the *allRec* scheme is used.

3.6 Experimental Evaluation

Our goal is to study the utility gain of optimal solutions and the feasibility of finding optimal solutions for a very large search space. We used the widely adopted benchmark Adult dataset [12]. This dataset consists of 45,222 records after removing records with missing values. Table 3.5 describes the QI attributes (the first 7), the sensitive attribute, and the class attribute. The taxonomy tree for each QI attribute is from [52]. The full search space consists of 356 million (356,440,500) cuts, which makes a complete search infeasible. We further amplified the size of the search space and the size of the dataset to assess the efficiency and scalability of our algorithm. The dataset, the QI taxonomy trees, and the l^+-Optimize executable can be downloaded from the author's personal website.

We implemented three algorithms in C++, including l^+-Optimize proposed in this chapter, the *l*-diverse variant [67] of Incognito [59], and the simulated annealing algorithm [56]. As multi-dimensional generalization models [40][60] do not ensure the domain exclusiveness of the anonymized data, so we did not compare with them since it is less meaningful. All experiments were run on an HP tablet PC with a 2GHz Intel Pentium M CPU and 1GB RAM running Windows XP.

Table 3.5 Description of the Adults dataset

	Attribute	|Domain|	Generalization Type	#Level
1	Age	74	Ranges-5,10,20,40	5
2	Education	16	Taxonomy tree	5
3	Native Country	41	Taxonomy tree	4
4	Work Class	7	Taxonomy tree	3
5	Marital Status	7	Taxonomy tree	3
6	Gender	2	Taxonomy tree	2
7	Race	5	Taxonomy tree	3
8	Occupation	14	Sensitive attribute	
9	Income Class	2	Class column	

We used the three cost metrics LM, DM, and CM. As explained in Section 3.5, the suppression schemes *NoSupp*, *allRec*, and *vioRec* are used with all cost metrics, and the suppression schemes *allSA* and *vioSA* are used with LM. For all sets of experiments but one, we used a uniform threshold θ for all SA values. In section 3.6.1.2, we explored the utility gain by allowing a different θ_i for a different SA value s_i.

3.6.1 Utility Evaluation by Comparing with Incognito

Incognito finds the optimal solution under the restrictive full domain generalization model. Incognito does not incorporate suppression as an integral part rather as an external constraint because the amount of suppression is not monotone and no cost based pruning is employed. So, we only consider Incognito without suppression.

We use *LO* as the abbreviation of l^+-Optimize. So, *LO-NoSupp* denotes l^+-Optimize without suppression, *LO-all Rec* denotes l^+-Optimize with the *allRec* suppression scheme, and so on.

3.6.1.1 Information Loss with a Uniform Threshold for All SA Values

Figure 3.4(a)-(c) show the cost, i.e., the information loss of the anonymized data produced by Incognito and l^+-Optimize with different cost metrics. A uniform l ($1 / \theta$) is used for all SA values, which ranges from 2 to 6 (θ = 50%, 33.33%, 25%, 20%, 16.67%).

The cost of Incognito is always equal to or greater than the cost of l^+-Optimize. With LM as in Figure 3.4(a), the gap between Incognito and l^+-Optimize without suppression (i.e., *NoSupp*) is not significant. The gap increases when l^+-Optimize incorporates suppression. With DM as in Figure 3.4(b), the gap is quite significant even without suppression: for most cases, the cost of Incognito is 44% to 176% more than the cost of l^+-Optimize. The gap is further increased when l^+-Optimize integrates various suppression schemes. Figure 3.4(c) shows that l^+-Optimize also outperforms Incognito with CM. In summary, integrating suppression with generalization helps reduce information loss; the finer the granularity of suppression, the more the reduction, where the order of granularity from coarse to fine is *NoSupp*, *allRec*, *allSA*, *vioSA*, and *vioRec*.

We also studied the usefulness of the anonymized data for classification on Income Class. We anonymized the data by running algorithms with CM, constructed the C4.5 classifier from the training set (the first 30,162 anonymized records), and reported the classification error on the test set (the last 15,060 anonymized records). The baseline of the classification error is 17.1% which is the error with the original data. In Figure 3.4(d), the classification error by Incognito is similar to l^+-Optimize with the *NoSupp*

scheme: for a small $l \leq 4$, the classification error is pretty close to the baseline; when l becomes larger, the error is high. However, the error by l^+-Optimize with *vioRec* is always very close to the baseline (with a gap $\leq 1\%$).

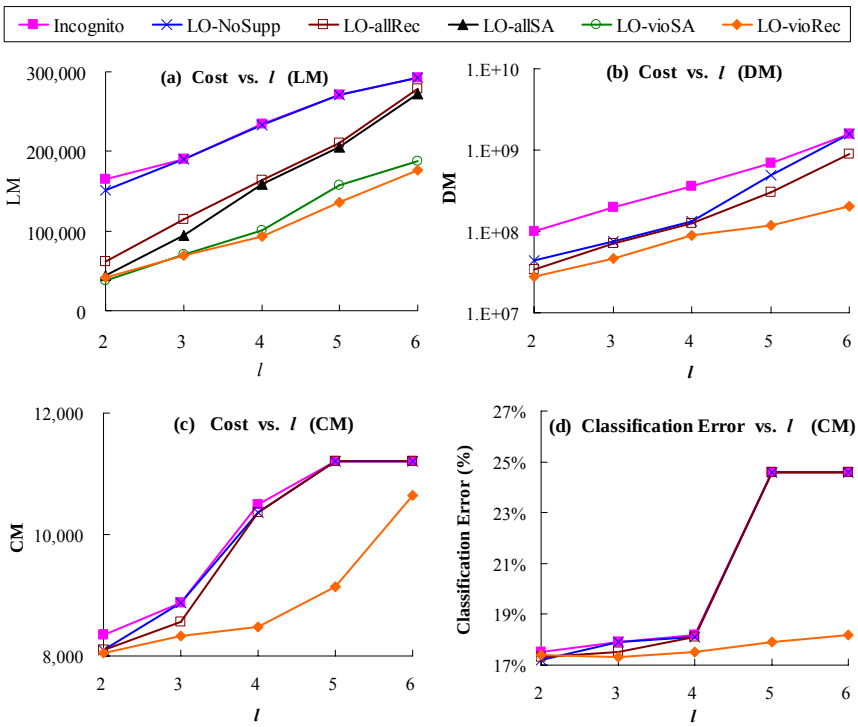


Figure 3.4 Comparing with Incognito on optimal cost and classification error

3.6.1.2 Information Loss with a Distinct Threshold per SA Value

We explore the flexibility of our l^+-diversity principle, i.e., setting a different θ_i for a different SA value s_i as follows. Givin θ be a baseline threshold, θ_i for s_i is determined by θ and the sensitivity of s_i. If the frequency of s_i in the original table, $fr(s_i)$, is below the average frequency fr_{avg}, s_i is deemed very sensitive and $\theta_i = \theta$, otherwise s_i is deemed less sensitive and $\theta_i = \theta \cdot fr(s_i) / fr_{avg}$. In case $\theta_i > 1$, we should reset θ_i to 1. Let θ_{avg} be the average threshold, then $\theta_i \leq \theta_{avg}$ if $fr(s_i) \leq fr_{avg}$. In words, the more sensitive is s_i, the more restrictive is θ_i and the better the protection is for s_i.

Figure 3.5 shows that all cost curves of l^+-Optimize shift downwards compared to Figure 3.4. That is, by making use of the flexibility of our l^+-diversity principle, l^+-Optimize greatly reduced the information loss and classification error with all cost metrics and all suppression schemes. E.g., comparing Figure 3.5 with Figure 3.4 shows that the information loss by l^+-Optimize is reduced by 13% to 22% with the LM cost metric and the *NoSupp* scheme, 9% to 600% with DM and *NoSupp*, and 8% to 26% with CM and *NoSupp*. The cost curves of Incognito remain the same as it only uses a uniform l, thus l^+-Optimize preserves even more utility than Incognito.

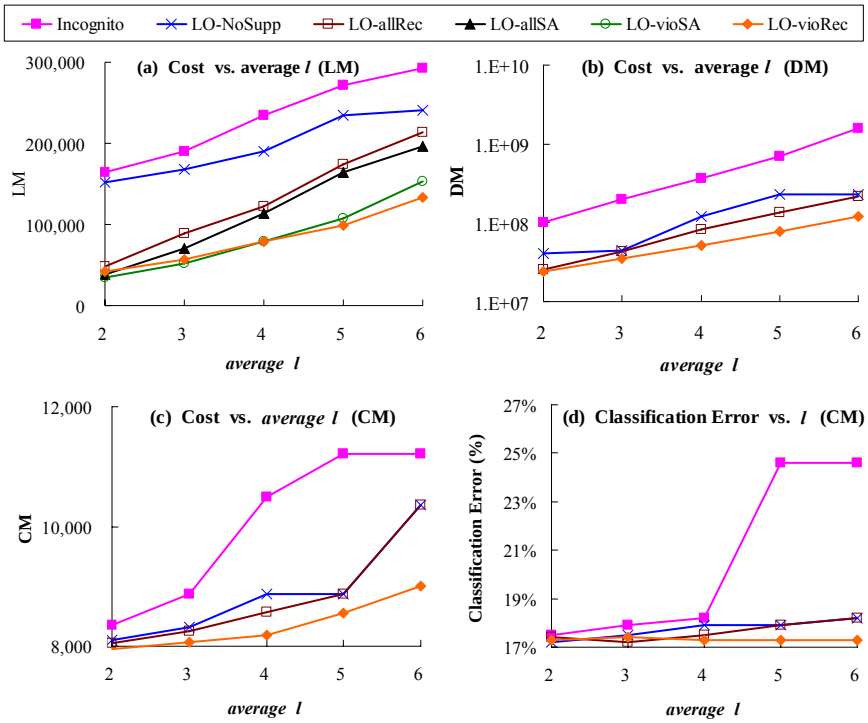


Figure 3.5 Comparing with Incognito on optimal cost and classification error by exploring the flexibility of l^+-Optimize: a different θ_i per s_i (*average* $l = 1 / \theta_{avg}$)

In this set of experiments, we are quite *conservative* in that Incognito is compared with l^+-Optimize using the *same* average privacy threshold θ_{avg}. In fact, Incognito has to

enforce the most restrictive threshold to avoid breaching the privacy, and hence the utility gain of l^+-Optimize over Incognito will be more striking.

In summary, the utility gain by adopting the flexible full subtree generalization model is significant. l^+-Optimize further reduces information loss by incorporating suppression as an integral part of anonymization, and provides more protection for more sensitive values by allowing setting privacy thresholds based on sensitivities.

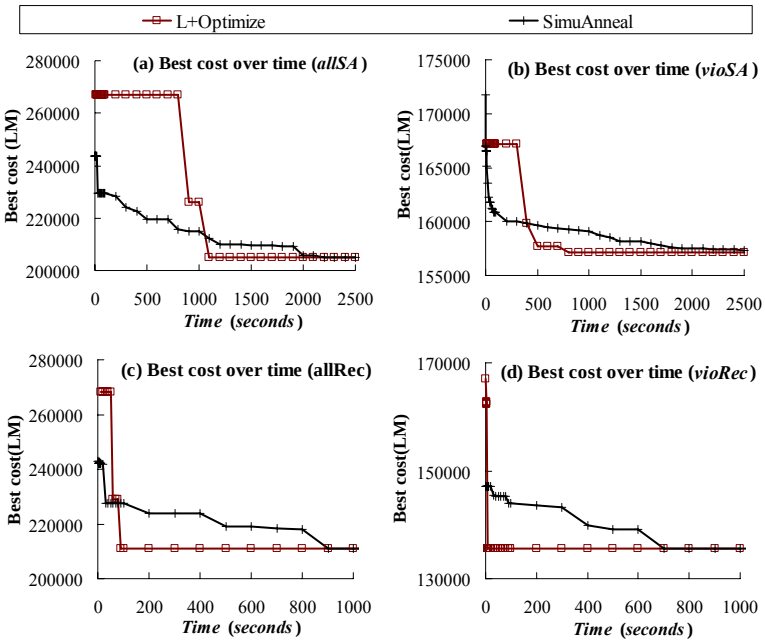


Figure 3.6 Comparing with simulated annealing on best cost over time

3.6.2 Utility Evaluation by Comparing with Simulated Annealing

Simulated annealing is a stochastic algorithm that employs a greedy heuristic to search for a good solution, while avoiding being trapped at local optima [56]. Simulated annealing, abbreviated as SimuAnneal, is used as a representative of greedy algorithms to compare with l^+-Optimize. We collected the average best cost at any given time point based on 6 random runs.

Figure 3.6 shows the results with LM, all suppression schemes, and $l = 5$ ($\theta =$ 20%). SimuAnneal starts with some solutions not terribly bad and improves the quality *gradually*, and given a long enough time, it can produce a high quality solution. In contrast, l^+-Optimize starts with the most generalized solution and improves the quality *quickly* within a short time. l^+-Optimize finds the best solution by several orders of magnitude faster than SimuAnneal.

In short, the gain from the optimal solution relative to heuristic solutions is also significant. This experiment also suggests another usage of l^+-Optimize as an *anytime* algorithm giving time is a constraint.

3.6.3 Efficiency and Scalability Evaluation

In this set of experiments, we evaluate the performance of l^+-Optimize by examining two key indicators of the efficiency, the runtime and the percentage of cuts examined. Figure 3.7 shows the runtime (left column) and the percentage of cuts examined (right column) by l^+-Optimize, for the three cost metrics. For most of the cases, 99.99% to 99.97% of cuts are pruned. In other words, typically only 0.01% to 0.03% of cuts are examined, out of which about 2/3 to 9/10 are examined before the optimal solution was found (according to statistics collected in addition). This indicates that our lower bound pruning is highly effective. We observed a strong correlation between the runtime and the percentage of cuts examined. The runtime usually is under a few hundred to a little bit more than a thousand seconds, which is highly efficient giving 356 million cuts in the search space.

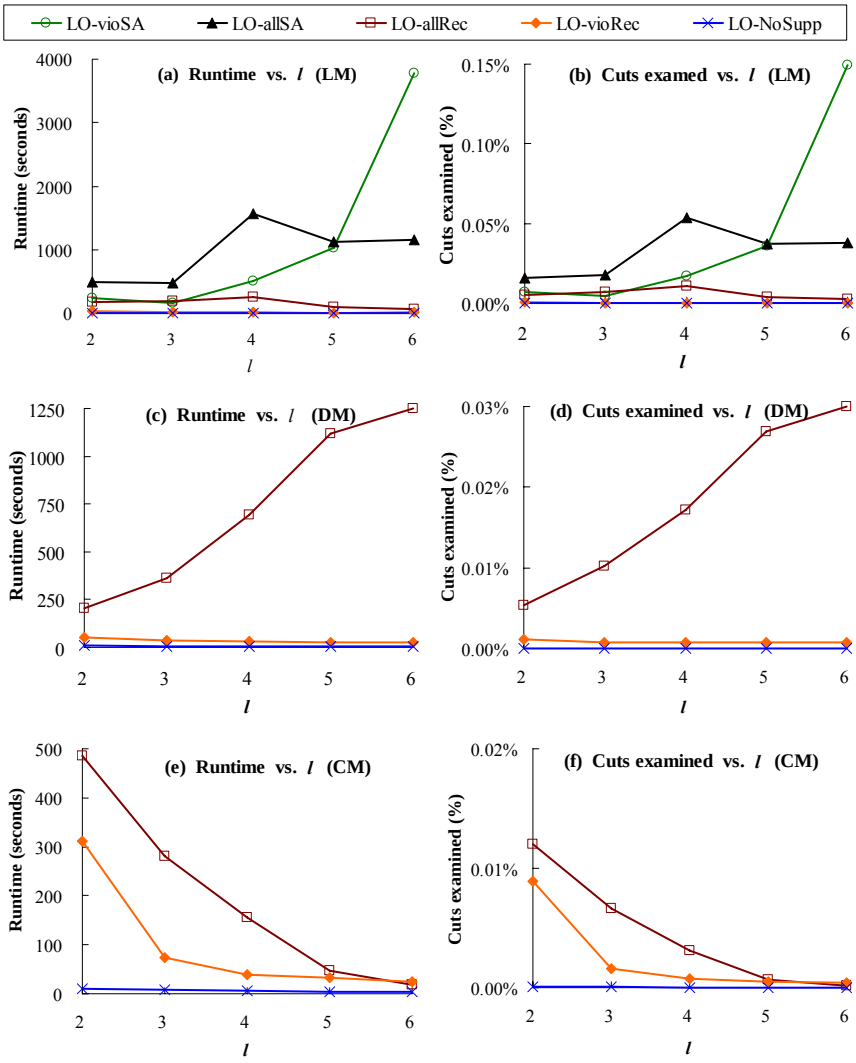


Figure 3.7 Performance of l^+-Optimize

It is observed that different metrics have different pruning behaviours. With LM and DM, the percentage of cuts examined increase with the increase of l, whereas the trend is opposite with CM. The reason is as follows. With LM and DM, the top-down search starts with a very large initial cost (*cost_g*=100%, *cost_s*=0). When l is small, specialization always reduces *cost_g* a lot while augmenting *cost_s* a little. Hence, we can get a very small running best cost at the early stage of the search, so the pruning is

very strong. When l becomes larger, we need to examine more cuts, before finding the optimal cut, so the pruning is less effective.

With CM, the top-down search starts with a quite small initial cost (*cost_g*=24.6%, i.e., the frequency of the minority class, *cost_s*=0). Therefore, the decrement of *cost_g* when specializing a cut is always small, and *cost_s* is a dominant factor. When l is large, the increment of *cost_s* is larger than the decrement of *cost_g* when specializing a cut, and so the search tends to prune most of cuts at lower levels, i.e., the pruning is very strong. When l is small, the dominance of *cost_s* is relatively weak, so is the pruning.

In the following, we did a series of experiments to explore the effects of individual pruning techniques, and the impacts of the QI taxonomy selection, the size of QI, and the size of the dataset on the performance. The default setting is as follows: cost metric DM, $l = 4$ ($\theta = 25\%$), and the suppression schemes *NoSupp*, *allRec*, and *vioRec*.

3.6.3.1 Individual Pruning Techniques

We evaluated the effectiveness of the each dynamic technique in Section 3.4 *individually*. "full l^+-Optimize" denotes the algorithm using both techniques, i.e., arranging values in the descending order of occurrences, and updating lower bounds when backtracking. "partial (Asc)" denotes the partial algorithm with values being arranged in the reverse of the proposed order, and "partial (noUpdate)" denotes the partial algorithm without dynamically updating lower bounds on backtracking. It turned out that none of the two partial algorithms terminated within a reasonable time limit with the default QI taxonomy trees. To obtain results for partial algorithms, we reduced the

taxonomy trees of the age and native-country attributes to the top two levels.

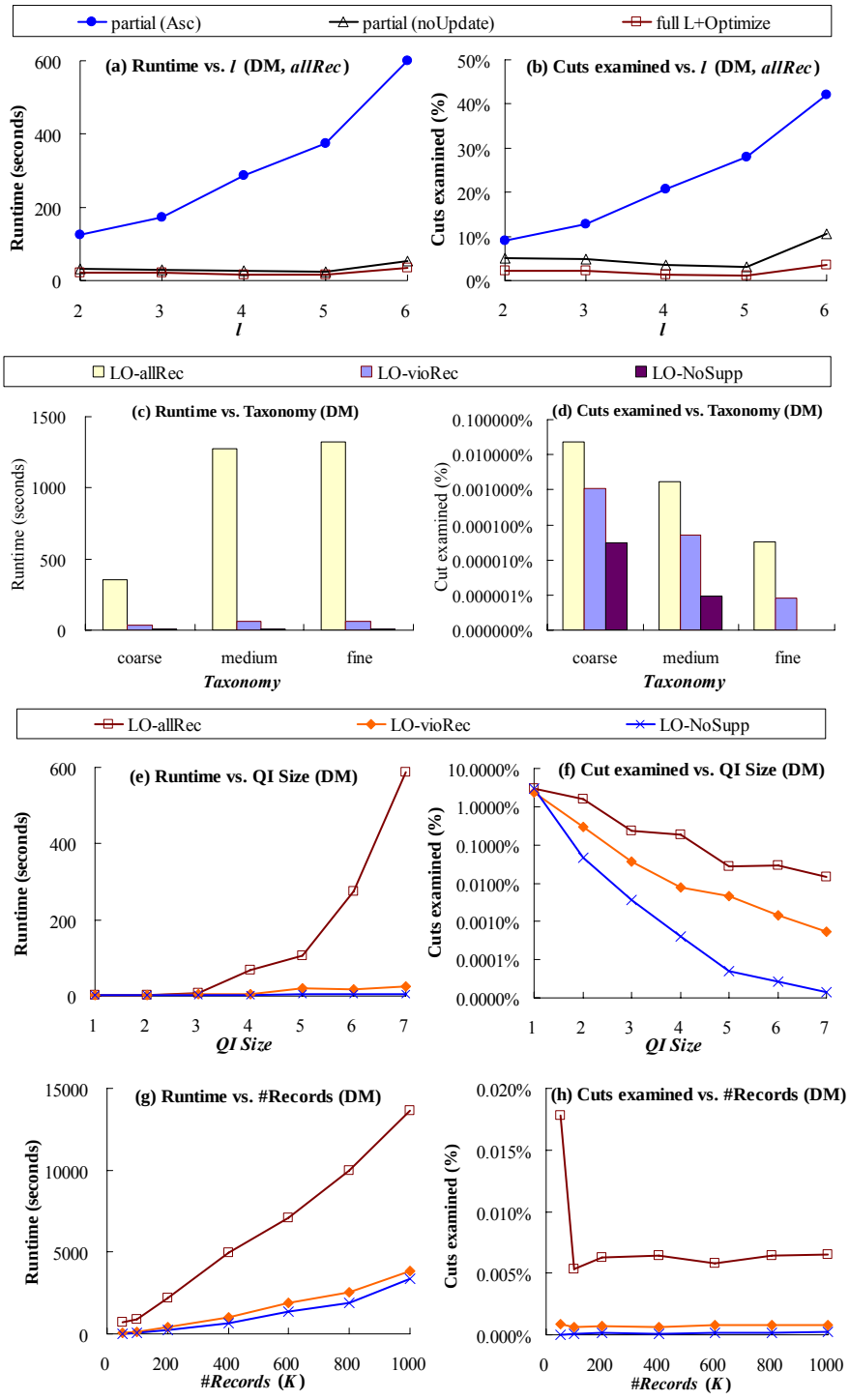


Figure 3.8 Performance by (a)(b) different dynamic techniques, (c)(d) taxonomies of different granularities, (e)(f) different QI sizes, (g)(h)different dataset sizes

Figure 3.8(a)-(b) shows the runtime and the percentage of cuts examined with DM and *allRec* (the results with the other cost metrics and suppression schemes are similar). The efficiency of "partial (noUpdate)" drops by a factor of 2 to 3 from the full algorithm, and the efficiency of "partial (Asc)" is several orders of magnitude worse than the full algorithm. That is, the order of values in a cut has much larger impact on the efficiency. Nevertheless, both techniques are critical for achieving a strong pruning.

3.6.3.2 Amplifying the Search Space

We investigated the impact of the selection of QI taxonomies on the efficiency of l^+-Optimize. We created 3 taxonomies with the coarse, medium, and fine granularity respectively, based on the taxonomies in [37] and [52]. Three taxonomies have vastly different search space sizes: the coarse taxonomy yields 152 million (152,685,000) cuts, the medium one yields 7 billion (6,943,460,940) cuts, and the fine one yields 431 billion (431,934,597,900) cuts.

Figure 3.8(c)-(d) show the runtime and the percentage of cuts searched. For a finer taxonomy, the algorithm took a longer time because the search space is larger. However, the increase in the runtime is several orders less than the increase in the size of the search space, which means that the pruning is stronger with a larger search space. For example, with the *allRec* scheme, the search space of the medium taxonomy is 40 times of the coarse taxonomy, but only 0.0017% of cuts for the medium taxonomy were examined compared to 0.02% for the coarse taxonomy. In fact, a finer taxonomy provides more flexibility, which helps us get a smaller running best cost in the early stage of search, so we have a stronger pruning.

3.6.3.3 Varying the Size of QI

We studied the impact of the QI size on pruning. We selected the first m attributes listed in Table 3.5 to comprise the QI, for m = 1, 2, .., 7. Figure 3.8(e) shows that the runtime increases with the increase of the QI size since the search space size increases quickly with the QI size. Figure 3.8(f) shows that the pruning becomes stronger with a larger QI size, which mitigates the impact of the QI size on performance.

3.6.3.4 Amplifying the Size of the Dataset

To study the scalability of l^+-Optimize, we amplified the size of the dataset by inserting "variants" of each original record into the Adult dataset. A variant of an original record t was created by randomly selecting q attributes from QI, with q being uniformly distributed in the range [1, |QI|], and replacing t's value on each selected attribute with a value randomly drawn from the attribute domain.

Figure 3.8(g) depicts the runtime of l^+-Optimize for 50K to 1000K data records. The runtime is linear to the data size. In fact, the search space depends on the taxonomies of the QI attributes rather than the size of the dataset. Figure 3.8(h) shows that there is little change in the number of cuts to be examined. Notice that the overhead pertaining to each examined cut mainly comes from the partition split / merge operations, and is proportional to the size of the dataset. In short, l^+-Optimize is quite scalable.

3.7 Discussions and Extensions

3.7.1 Discussions

3.7.1.1 Choosing the Full Subtree Generalization Model

The full subtree generalization model [52] is a single-dimensional global recoding model [59]. As discussed in Section 2.1, this model ensures the *domain exclusiveness*,

which implies that the anonymized data can be analysed by any existing algorithm without modification.

For example, given T in Table 3.1(a) and $\theta = 50\%$ ($l = 2$), the full subtree generalization model may produce 3 partitions, <Junior, Europe>, <Senior, Europe>, and <University, America> in the anonymized T^*. The values in T^* are *exclusive* of each other. We can analyse T^* by any existing algorithm, e.g., mine association rules in T^*.

In comparison, the latest multi-dimensional generalization model [40] may produce 5 multi-dimensional regions (partitions), <Junior-or-Senior, UK>, <Senior-or-Bachelor, France-or-Canada>, <Junior-or-Graduate, France-or-Canada>, <Bachelor-or-Graduate, USA>, and <Bachelor-or-Graduate, UK-or-Canada>. Notice that [40] first maps the m-dimension data to 1-dimension, and then maps the *anonymized* 1-dimension data back to m-dimension. Now, the values in the anonymized T^* are not exclusive of each other, i.e., they overlap on the domains, which is a flexibility that helps reduce the information loss. However, this comes with a price: the domain exclusiveness is destroyed.

For example, we can not directly run existing algorithms on T^* to mine association rules as we can not count the exact occurrences of values in T^*. We have to either develop new customized algorithms to get approximation of the data mining result, or further generalize all descendent values to their ancestor values in the anonymized data to such an extent that the domain exclusiveness is observed. The latter means that the further generalization ultimately follows the full subtree generalization model. Therefore, the utility of the further generalized data will be no better than our approach as we find the optimal full subtree generalization solution.

3.7.1.2 Handling Minimality Attacks

Minimality attacks, identified by [103], arise from the fact that knowledge of the anonymization algorithm, in particular the *minimality principle* employed by most algorithms, can yield clues about how to infer detailed information from the anonymized data. Notice that both single dimensional and multi-dimensional generalization models may suffer from minimality attacks.

To thwart minimality attacks, a simple strategy is to apply an additional precaution step to the optimal solution, such as randomly introducing a little bit more generalization or suppression. Such a precaution step breaks the minimality principle and can be easily incorporated into our algorithm while the key elements of our approach still work properly.

Another strategy is to return a solution randomly selected from the top-h optimal solutions with h being a small integer, which also breaks the minimality principle. At the same time, the returned solution is guaranteed to be among the top-h optimal ones. To adopt this strategy, we maintain the current top-h best solutions and use the cost of the h-th best solution in the pruning condition. All techniques employed in our algorithm remain unchanged.

3.7.2 Extensions

The l^+-*Optimize* algorithm can be extended in at least two ways.

First, it can be modified into an approximate algorithm that finds a solution within $1+\alpha$ of the optimal by relaxing the pruning condition as follows, i.e., revising line 2 and line 4 in Algorithm 3.1 accordingly.

$$(1+\alpha)\cdot LB_{cost}(T, Cut) \geq cost(T, Cut_{best}).$$

Second, it can be changed into an anytime algorithm by adding a stopping condition that limits the maximum amount of time or steps for the algorithms to run. Such an anytime algorithm can always yield a valid anonymization solution because of the top-down search strategy employed by l^+-*Optimize*.

3.8 Summary

This chapter proposed a new privacy notion, the l^+-diversity notion, which extends the most used form of the *l*-diversity notion. The main contribution is an efficient algorithm, l^+-Optimize, for finding an optimal anonymization satisfying l^+-diversity under the full subtree generalization model with various suppression schemes.

The algorithm is generic in the sense that it can be instantiated with any reasonable cost metric. The novelty lies in several strong techniques: a novel structure for enumerating all solutions, methods for estimating cost lower bounds, strategies for dynamically arranging the enumeration order and updating lower bounds.

Extensive experiments on real world datasets confirm the following new findings. By considering a large search space, the optimal solution has a significant utility gain compared to heuristic solutions. By setting each sensitive value's privacy threshold according to its sensitivity, we can provide better protection for more sensitive values and incur less information loss. It is possible to find the optimal solution efficiently on real world datasets by employing strong pruning strategies.

4: Optimal KL(m, n)-Privacy for Publishing Set-valued Data

Set-valued data is a collection of transactions, and each transaction is a set of an arbitrary number of items. An example of set-valued data is shown in Table 4.1. Set-valued data has a wide range of applications in data mining research. For example, customer behaviour analysis is made possible by mining the shopping transactions [9][16][91]. Movie rental companies have been successful in recommending movies to subscribers based on subscriber preference analysis obtained from the movie rating data [71]. However, set-valued data may contain significant amount of sensitive information. Releasing such data in its raw form to a third party could breach privacy [15][49][71] as explained by the following motivating example.

Example 4.1 (**A motivating example**): A supermarket released one week's transactions shown in Table 4.1 to a data mining company for marketing strategy analysis to promote the sale of AdultToy, Viagra, and PregnancyTest. These items are profitable, but are also sensitive in that customers are unwilling to let their friends know that their shopping transactions contain such items, while other items, such as Beer, Wine, and Milk, are non-sensitive. Bob working in the data mining company is analyzing the data, and he happened to know his colleague, Alice, bought Wine and Yogurt from this supermarket in the last week. Given the data, Bob is 100% sure that Alice's transaction is t_1 (*identity disclosure*), and Alice also bought AdultToy (*sensitive item disclosure*). Bob also knew that his friend Darth bought Beer but no Wine. Thus, Bob is sure that Darth's transaction is t_2, and Darth also bought AdultToy and Viagra.

Table 4.1 Shopping basket database *D*

TID	Transactions
t_1	Wine, Milk, Yogurt, AdultToy
t_2	Beer, Jacket, Pants, AdultToy, Viagra
t_3	Yogurt, Jacket, Hose, Shoe, Viagra
t_4	Milk, Yogurt, Jacket, Geta, PregancyTest
t_5	Beer, Wine, Milk, Jacket, Pants

Alice's privacy is breached because of Bob's knowledge about the presence of Wine and Yogurt in Alice's transaction. Darth's privacy is breached because of Bob's knowledge about the absence of Wine and knowledge about the presence of Beer. ■

The challenges with this problem, in general, are the high dimensionality [41][108] and lack of a quasi-identifier [95][108]. That is, if we treat an item as an attribute then set-valued data has hundreds of thousands of attributes [41]; it is not practical to define a fixed subset of attributes (items) as a quasi-identifier [95] with set-valued data, while all approaches for relational data assume that the quasi-identifier is given.

Surprisingly, such challenges have received little attention. In general, only a handful of works tackle the privacy protection problem in publishing set-valued data [48][95][108][41]. In particular, when handling the privacy attacks explained by Example 4.1, that is, both the identity disclosure and the sensitive item disclosure based on both the knowledge about the *presence* and the knowledge about the *absence* of certain items, the prior works [48][95][108][41] have two limitations.

The first limitation with the prior works is that they either provide over-protection that causes excessive distortion or do not prevent certain privacy attacks as follows.

[48][41] take an approach that treats the set-valued data as a high dimensional relational table with binary attributes so that they can apply techniques developed for

relational data to set-valued data. In particular, [48] only deals with the identity disclosure and takes all items together as the quasi-identifier in that it requires that each transaction is *identical* to at least k-1 other transactions. [41] differentiates non-sensitive items and sensitive items, considers only the sensitive item disclosure, and takes all non-sensitive items as the quasi-identifier in that it requires that *all* inferences about a sensitive item have a probability no more than a privacy degree 1 / p.

On one hand, [48][41] do not prevent certain privacy attacks. On the other hand, [48][41] provide over-protection that inevitably causes excessive distortion because of the curse of dimensionality [3] as explained as follows. [48] employs the multi-dimensional partitioning technique [60]. But, it stops partitioning the data at a high level of the item taxonomy because specializing an item with n children will generate 2^n-1 possible sub-partitions with set-valued data (while only n possible sub-partitions with relational data). Such an exponential branching quickly diminishes the size of a sub-partition and stops the partitioning at a high level. Moreover, it destroys the domain exclusiveness as "Beer" and "Liquor" may co-occur in the anonymized data. [41] modifies *all* inference about a sensitive item to a low certainty, which does not serve the purpose of the publication. Notice that keeping sensitive items instead of removing them is because the published data is mainly used for analyzing trends and patterns regarding sensitive items. In Example 4.1, the supermarket wants to study the marketing strategy for promoting sensitive items, such as Viagra. Any inference that is beyond the knowledge of adversaries should be kept *intact*.

[95][108] take an approach that is specific to set-valued data, and consider the privacy attacks based on the knowledge about presence of a combination of no more than

m items. The issue with these works is that they do not tackle the privacy attacks based on the knowledge about absence of items. E.g., Darth's privacy breach can not be prevented by enforcing the privacy notions in [95][108].

The second limitation with the prior works is that they only provide heuristic based except that [95] presented an optimal solution OA. The authors of [95] acknowledged that OA is *inapplicable* to large, realistic problems, which to our understanding is resulted from a few reasons. First, the bottom-up, breadth-first approach by OA only employs validity-based pruning and cannot facilitate cost-based pruning. Second, OA needs to expand transactions by inserting generalized items, which is not scalable. Third, the count-tree used by OA for computing the supports of all the combinations of *m* (generalized) items does not facilitate effective pruning. Therefore, the authors of [95] emphasizes that their contribution focuses on a heuristic algorithm AA. Nevertheless, even such a heuristic algorithm [95] as well as the heuristic algorithm in [108] also suffer from the efficiency and scalability bottlenecks.

To address these limitations, this chapter proposes a new privacy notion, KL(*m*, *n*)-privacy, for protecting privacy in publishing set-valued data, with an optimal algorithm. The contributions are detailed as follows.

First, this chapter proposed the new privacy notion, KL(*m*, *n*)-privacy. This notion prevents both the identity disclosure and the sensitive item disclosure, and explicitly models the knowledge of an adversary by a logical expression, that is, a conjunction of *m* + *n* clauses with *m* clauses expressing the presence of items and *n* clauses expressing the absence of items. This notion is more flexible yet more general than [41][48][95][108]. That is, *transaction k-anonymity* [48] is the special case only considering the identity

disclosure ($l = 1$) with $m + n = |I_N|$ (the size of the domain of non-sensitive items). Privacy degree [41] is the special case only considering the sensitive item disclosure ($k = 1$) with $m + n = |I_N|$. k^m-*anonymity* [95] is a special case only considering the identity disclosure ($l = 1$) with $n = 0$; and *coherence* [108] is the special case with $n = 0$.

Second, this chapter presented the first optimal solution that is applicable for anonymizing large, real world databases. This solution is also the first generalization solution that handles both the identity disclosure and the sensitive item disclosure in set-valued data. We employ the full subtree generalization. A generalization solution is represented by a *cut* on the taxonomy tree of non-sensitive items. An optimal solution is a valid solution (i.e., a cut) with minimum information loss. This chapter showed that *materializing the most general threats* is the best way to check if a cut is valid, and proposed a novel data structure, the *inverse itemset enumeration tree* for mining and materializing the *most general threats*. The inverse itemset enumeration tree has a natural structure that facilitates powerful pruning of privacy threats to be materialized. Validity checking in set-valued data is much harder than in relational data and must be highly efficient because it is performed at each step in searching an optimal solution. Such techniques are the key components of our algorithm.

Third, this chapter demonstrated that an optimal solution has a significant gain in data utility relative to heuristic solutions, and it is possible to find an optimal solution efficiently for a large search space. We evaluate the data utility in terms of a general purpose metric and usefulness in frequent pattern mining. Extensive experiments showed that our optimal algorithm outperforms the heuristic generalization algorithm AA [95] and the heuristic suppression algorithm MM [108] in terms of both information loss and

efficiency on anonymizing real world databases. Our optimal algorithm also outperforms the local generalization algorithm LG [48] in terms of data utility. This chapter also proposed a multi-round OG approach that further improves the efficiency and finds a solution very close to the optimal.

The rest of this chapter is organized as follows. Section 4.1 describes the privacy attack model, Section 4.2 proposes the KL(m, n)-privacy notion and a two phase optimal generalization algorithm, Section 4.3 discusses the first phase, i.e., how to mine the most general privacy threats, Section 4.4 describes the second phase, i.e., a systematic approach for searching an optimal solution among all valid ones, Section 4.5 evaluates algorithms, and Section 4.6 considers the variations and extensions of our algorithm, and Section 4.7 summarizes this chapter.

4.1 Privacy Attack Model

A data publisher releases an anonymized version D' of a set-valued database $D = \{t_1, t_2, \ldots, t_{|D|}\}$ that is a collection of transactions. Each transaction t_i is a set of items from the universe $I = \{i_1, i_2, \ldots, i_{|I|}\}$. Among items in I, some are sensitive, such as Viagra, and some are non-sensitive, such as Beer. The differentiation of sensitive and non-sensitive items is application-dependent and is defined by the data publisher who will employ his/her domain expertise or some automatic tools to accomplish this task. We denote the set of all sensitive items as I_S, and the set of all non-sensitive items as I_N.

An adversary has access to the published (anonymized) data. The adversary acquired the knowledge about presence of some non-sensitive items in and the knowledge about absence of certain non-sensitive items from his/her target transaction.

However, the adversary has no knowledge about sensitive items. The adversary tries to infer the target individual's transaction and sensitive items with high probabilities.

The data publisher employs generalization technique [52][59] to generalize non-sensitive items in producing the anonymized data *D′*. The suppression technique is not suitable because suppressing the presence (absence) of items could introduce *false* absence (presence) of items and hence create *artificial* privacy threats, which is unacceptable for data analysis especially with the false presence of items.

The taxonomy H_N of non-sensitive items used by the data publisher for generalization is also available to the adversary.

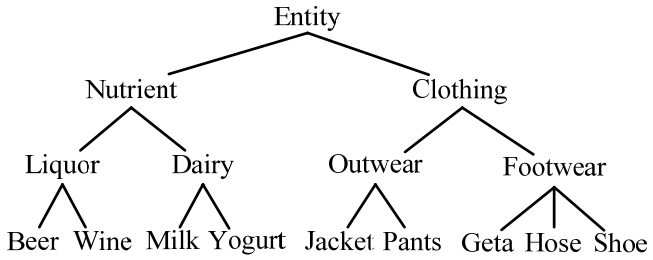


Figure 4.1 Hierarchical taxonomy of non-sensitive items H_N

Table 4.2 A generalized version *D′* of *D*

TID	Generalized Transactions
t'_1	*Liquor, Dairy* , AdultToy
t'_2	*Liquor* , Jacket, Pants, AdultToy, Viagra
t'_3	*Dairy* , Jacket, *Footwear,* Viagra
t'_4	*Dairy* , Jacket, *Footwear,* PregancyTest
t'_5	*Liquor* , *Dairy* , Jacket, Pants

Example 4.2: Figure 4.1 shows such a taxonomy tree where Entity is the *taxonomic parent* of Nutrient and the *taxonomic ancestor* of Liquor and Beer, and so on. Non-sensitive items in the original database *D* in Table 4.1 are all at the leaf level of the

taxonomy. Thus, we call Beer a leaf item, and so forth. Table 4.2 is a generalized version D' where all Beer and Wine are generalized to Liquor, all Milk and Yogurt are generalized to Dairy, and all Geta and Hose and Shoe are generalized to Footwear. ■

4.1.1 Adversary's Knowledge-Expression

We model an adversary's knowledge about the presence and absence of a *single*, non-sensitive item by decorating the item with + and – respectively.

Definition 4.1 (Positive item and negative item): For an item $i \in I_N$, $i+$ and $i-$ represent the presence and absence of i respectively. We call $i+$ a positive item, and $i-$ a negative item. We also call $i+$ or $i-$ a *decorated* item, and i itself an *undecorated* item. ■

As any privacy adversary has no knowledge about sensitive items, we need no *decoration* for sensitive items. We further model the adversary's knowledge about his/her target individual by a logical expression.

Definition 4.2 (Knowledge-expression): The knowledge of an adversary about a target individual is a logical expression of decorated, non-sensitive items at the *leaf level* of the generalization taxonomy of the following form:

$$KE = (j_{11}^{+} \vee ... \vee j_{1v}^{+}) \wedge ... \wedge (k_{m1}^{+} \vee ... \vee k_{mw}^{+}) \wedge i_1^{-} \wedge ... \wedge i_n^{-}$$

which is a conjunction of m disjunctions of positive, leaf items with n negative, leaf items. For a knowledge-expression KE, we call the pair of the number m of disjunctions and the number n of negative items the power of KE, and denote it as $power(KE) = (m, n)$. ■

Notice that each disjunction of positive items as a whole states a basic fact that must hold, while a single positive item in the disjunction may be true or false. Therefore,

each disjunction as a whole is counted toward the power of the adversary by one unit in Definition 4.2.

Example 4.3: A simple form of knowledge-expression could be just a conjunction of decorated, non-sensitive items. For instance, Bob's knowledge is "Wine+ $\wedge$ Yogurt+" regarding Alice with power being (2, 0), and "Beer+ $\wedge$ Wine-" regarding Darth with power being (1, 1). A general form of knowledge-expression could be "(Hose+ $\vee$ Geta+) $\wedge$ Milk-" with power being (1, 1). ■

We only consider a realistic adversary whose prior knowledge *KE* is bounded, i.e., $power(KE) \leq (m, n)$. Such an assumption is reasonable because obtaining background knowledge often requires effort on the adversary side.

4.1.2 Adversary's Basic Protocol in Making Attacks

For each (generalized) transaction, we only publish items present in the transaction. The adversary launches attacks by finding out the subset of published transactions that satisfy his/her knowledge-expression. The protocol that the adversary follows to make attacks falls into three cases.

In case the published transactions are not generalized, i.e., all non-sensitive items in the publication are at the leaf level of the taxonomy tree, the adversary simply follows Definition 4.1 to check if his/her knowledge-expression is satisfied by a transaction.

Definition 4.3 (Satisfaction of a knowledge-expression): Given an adversary's knowledge-expression *KE* and a (published) transaction *t*, *KE* is *satisfied* by *t* if for every disjunction *p* of positive items in *KE*, there is a positive item *i*+ in *p* whose undecorated

item *i* is present in *t*, and for every negative item *i*- in *KE*, its undecorated item *i* is absent from *t*. ■

Example 4.4: If Table 4.1 is published, the first transaction, {Wine, Milk, yougurt, AdultToy} satisfies Bob's knowledge regarding Alice, "Wine+ ∧ Yogurt+", and the second transaction, {Beer, Jacket, Pants, AdultToy, Viagra}, satisfies Bob's knowledge regarding Darth, "Beer+ ∧ Wine-". ■

4.1.3 Protocol for Generalizing Knowledge-Expression

In case the published transactions are generalized by the data publisher, the adversary has to rely on the same semantics of generalization to derive the most specific information available for determining possible satisfactions.

Example 4.5: Suppose that *D′* in Table 4.2 is published. The generalized transactions t'_1 and t'_5 in *D′* contain {Liquor, Dairy} that may be generalized from a transaction containing {Wine, Yogurt}. Thus, t'_1 and t'_5 satisfy "Wine+ ∧ Yogurt+", i.e., Bob's knowledge about Alice. Similarly, t'_1 and t'_2 and t'_5 contain {Liquor}, each of which may be generalized from a transaction containing Beer but no Wine. Thus, t'_1 and t'_2 and t'_5 satisfy "Beer+ ∧ Wine-", i.e., Bob's knowledge about Darth. Moreover, if Bob's knowledge about a third person is "Milk-", then the target transaction may be generalized into a transaction t'_a containing {Dairy} or t'_b not containing {Dairy} as Bob did not know whether Yogurt is present in the target transaction. In other words, both t'_a and t'_b possibly satisfy "Milk-", which is the most specific information available to Bob.■

Therefore, in such a case, the adversary can be thought of making attacks in two steps. The adversary first generalizes his/her knowledge-expression to the same level as

the published data, and then find out those generalized transactions in the published data that satisfy the generalized knowledge-expression by Definition 4.3.

Example 4.6: The last example can be interpreted as follows. Bob first generalizes his knowledge about Alice, i.e., "Wine+ $\wedge$ Yogurt+", into "Liquor+ $\wedge$ Dairy+", and then finds out the generalized transactions that satisfy "Liquor+ $\wedge$ Dairy+". Similarly, Bob generalizes his knowledge about Darth, i.e., "Beer+ $\wedge$ Wine-" into "Liquor+". However, Bob cannot generalize his knowledge about the third person, i.e., "Milk-" cannot be generalized into "Dairy-". ■

From the foregoing discussion, we have an important observation regarding the protocol how the adversary generalizes his/her knowledge-expression.

Observation 4.1 (Generalization of knowledge-expression and its power). A generalized form *gKE* of an original knowledge expression *KE* is in the same form as *KE* but may contain decorated, generalized items. Each disjunction p_g of positive items in *gKE* is derived from a disjunction *p* in *KE* such that for every item *i* whose positive decoration is in p_g, the positive decoration of a taxonomic descendant of *i* is in *p*. For an item *i* whose negative decoration is in *gKE*, the negative decoration of every taxonomic descendant of *i* is in *KE*. Similarly, we can define the power of *gKE* as follows:

$$power(gKE) = \left(\#dp_g, \sum_{i^- \in gKE} \#leaves(i) \right)$$

where $\#dp_g$ is the number of disjunctions of positive items in *gKE*, and $\#leaves(i)$ is the number of leaves in the taoxonomic subtree rooted at *i*. ■

It follows that if *gKE* is derived by generalizing *KE*, $power(gKE) \leq power(KE)$.

4.1.4 Protocol with Knowledge-Expression of General Nature

Although the knowledge-expression is defined at the leaf level of the generalization taxonomy, there is a case that the adversary's knowledge-expression *KE* is of general nature, i.e., quite vague and not specific. In such a case, some generalized form *gKE* derived from *KE* could have the same identifying ability, i.e., the set of transactions satisfying *gKE* is the same as *KE*. For example, "(Beer+ $\vee$ Wine+)" is vaguer, less specific, and hence less powerful than "Beer+". As Beer and Wine are the only children of Liquor, "(Beer+ $\vee$ Wine+)" means any children of Liquor may be present in the target transaction. Therefore, "(Beer+ $\vee$ Wine+)" has the same identifying ability as "Liquor+".

Observation 4.2 (Generalized, conjunctive form of knowledge-expression): A generalized, conjunctive form *gKE*, i.e., a conjunction of decorated, generalized, non-sensitive items, can be derived from an original knowledge-expression defined at the leaf level with exactly the same identifying ability. ■

In such a case, the adversary can be thought of directly employing *gKE*. If the published data is at a level lower than *gKE*, the adversary will first generalize the published data, and then find out the set of transactions satisfy *gKE* by Definition 4.3.

We can also use a set to denote a conjunction for simplicity. We call a set of decorated, generalized, non-sensitive items an *itemset*. For example, "Beer+ $\wedge$ Wine-" can be denoted by itemset {Beer+, Wine-}.

In the rest of this chapter, we will use the set-notation.

4.2 KL(m, n)-Privacy Notion and Optimal Generalization

4.2.1 KL(m, n)-Privacy

We prevent both identity disclosure and sensitive item disclosure to an adversary whose knowledge KE is bounded by $power(KE) \leq (m, n)$. KE is defined at the leaf level. By Observation 4.2, KE may be translated into an equivalent form, an itemset X (i.e., a conjunction) of decorated, generalized, non-sensitive items with $power(X) \leq (m, n)$.

To safe-guard the privacy, we take a precautious step, that is, to prevent such an itemset X from becoming a privacy threat. The rationale is that if we thwart attacks based on X, we definitely prevent attacks based on any knowledge-expression given by Definition 4.2. In particular, we assure that the probability to identify any individual's transaction by X is no more than $1 / k$, and the probability to guess any individual's sensitive items by X is no more than $1 / l$.

Definition 4.4 (KL(m, n)-privacy): A set-valued database D observes KL(m, n)-privacy if for any itemset X of decorated, generalized, non-sensitive items with $power(X) \leq (m, n)$ by Observation 4.1, (a) either no transaction or more than k transactions in D *support* X, and (b) for each sensitive item s at most $1 / l$ of transactions in D that support X also contain s. We say that a transaction t supports X if t or the generalization of t satisfies X by Definition 4.3. We use $sup(X)$ to denote the number of transactions supporting X. Thus, we can rewrite (a) $sup(X) = 0$ or $sup(X) \geq k$, and (b) $conf(X \rightarrow \{s\}) = sup(X \cup \{s\}) / sup(X) \leq 1 / l$. If X violates (a) or (b), X is called a *privacy threat* (or simply a threat). ■

This privacy notion prevents identity disclosure and sensitive item disclosure by two independent requirements because of two reasons. First, in case $k > l$ the satisfaction

of requirement (b) does not necessarily means the satisfaction of requirement (a). Second, in case not all transactions contain a sensitive item, the prevention of the sensitive item disclosure does not necessarily result in the prevention of the identity disclosure.

Example 4.7: Bob's power (m, n) in acquiring knowledge is no less than (2, 1). Suppose that Alice and Darth require that both the probability associating them to a particular transaction and that guessing their sensitive items are no more than 50%. Then, publishing D in Table 4.1 violates KL(m, n)-privacy with (k, l, m, n) = (2, 2, 2, 1). ■

4.2.2 Full Subtree Generalization Technique

To enforce the KL(m, n)-privacy, we employ the full subtree generalization technique [52][59] to generalize non-sensitive items as we want to keep the domain exclusiveness of generalized data, that is, "Beer" and "Liquor" will not co-exist in the anonymized data. The domain exclusiveness is an important property of data required by many data mining tools. Sensitive items will not be generalized since we want to analyze the trends and patterns regarding the sensitive items.

With this technique, a generalization solution is represented by a cut on the taxonomy tree H_N. A cut contains *exactly one* item on every root-to-leaf path on H_N, and is denoted by the set of such items. Given *Cut* be such a cut, the generalized version D' of D, denoted as $D' = gen(D, Cut)$, is generated by replacing each non-sensitive item in D with its taxonomic ancestor in *Cut*.

Example 4.8: D' in Table 4.2 is the generalized version of D in Table 4.1 given *Cut* = {Liquor, Dairy, Jacket, Pants, Footwear} on the taxonomy tree H_N in Figure 4.1. ■

4.2.3 Measuring Information Loss

The generalization process will cause information loss on the data, which can be measured by various metrics. Our approach applies to any *well-behaved* metric. A well-behaved metric has the following properties: (1) it is *item-based*, i.e., charges information loss per item rather than per transaction, since transactions are of different sizes; (2) it is *monotonic*, i.e., charges more on a general item than a specific item; (3) it does not charge leaf items. E.g., *loss metric* LM [52] is such a metric as discussed in Section 3.5.1.

We use $cost(Cut)$ to denote the information loss incurred by the generalization solution defined by *Cut*. Property (1) implies that *Cost(Cut)* is decomposed into the information loss of constituent items of *Cut*, that is,

$$cost(Cut) = \Sigma_{x^* \in Cut}\, cost(x^*).$$

A metric can be employed to guide algorithms in finding an anonymization solution and in evaluating the utility of the anonymized data. We will also evaluate the utility in terms of usefulness in frequent pattern mining.

4.2.4 Optimal Generalization Problem and High-Level Algorithm Description

Our goal is to find a generalization solution defined by a cut that achieves our KL(m, n)-privacy notion and incurs the least information loss, i.e., a valid solution with the least information loss.

Definition 4.5 (Optimal generalization problem): Given a transaction database D, a taxonomy H_N for non-sensitive items I_N, the set I_S of sensitive items, the privacy parameters (k, l, m, n), find a generalization solution defined by Cut_{best} such that Cut_{best} is a valid cut by Definition 4.4, and $cost(Cut_{best})$ is the minimum among all valid cuts. ■

The KL(m, n)-privacy notion requires that no privacy threat is supported by the published data. Such a requirement is monotone in that if a privacy threat is supported by a solution defined by *Cut*, then the privacy threat is also supported by any solution defined by a specialization of *Cut*, i.e., the specialization of an invalid cut is also invalid. Therefore, we can employ the top-down search approach proposed in Chapter 3 for finding an optimal solution, in particular, we can adapt the l^+-Optimize algorithm (Algorithm 3.1 in Section 3.2.3) into our new algorithm OG, namely Optimal Generalization. The difference is that the l^+-Optimize algorithm only employs cost based pruning while OG will employ both cost based pruning and validity based pruning.

It turns out that the additional component of OG, i.e., checking the validity of possible solutions (cuts), is the most challenging part of the OG algorithm, in terms of computational overhead. There are two approaches to handle such a challenge.

On-the-fly validity checking approach. For each *Cut* to be verified, we generalize D on-the-fly to generate $D' = gen(D, Cut)$, and then mine D' to see if there is any privacy threat *contained* in D'.

Materialized validity checking approach. We first mine and materialize the privacy threats *supported* by the original data D, i.e., the privacy threats contained in all possible generalized version of D. Then, to check the validity of any *Cut*, we just retrieve the set of *materialized privacy threats* to see if there is any threat supported by $D' = gen(D, Cut)$ without actually generating and mining D'.

We choose the materialized validity checking approach. The on-the-fly validity checking approach is inefficient since a lot of itemsets, i.e., *privacy threat candidates*, need to be repeatedly enumerated when checking different cuts. For example in Figure

4.1, {Liquor+} will be enumerated 12 times when checking 12 different cuts. Moreover, the overhead for creating generalized versions of D is also huge. Therefore, our optimal generalizatio algorithm OG works in 2 phases. The high-level description of OG is as follows.

Phase 1: OG mines and materializes privacy threats. We will show an efficient and scalable way to do so in Section 4.3.

Phase 2: OG finds an optimal cut (solution) by adapting the l^+-Optimize algorithm proposed in Chapter 3, which utilizes the materialized privacy threats found in Phase 1 for checking validities of cuts.

4.3 Finding the Privacy Threats

This section focuses on the most challenging part of the OG algorithm, i.e., the first phase of OG, which materializes privacy threats. We first show that only a small portion of privacy threats need to be materialized, and then present an efficient and scalable way to do so.

4.3.1 The Most General Undecorated Privacy Threats

Materializing all the privacy threats contained in all possible generalized versions of D is not scalable because of the large number of threats. Our observations are as follows.

First, a threat is contained in $D' = gen(D, Cut)$ if and only if after removing the decoration of each constituent item of the threat, the resulting itemset is a subset of Cut. For example, {Liquor+} is contained in $gen(D,$ {Liquor, Dairy, Outwear,Footwear}) since after striping the decoration the resulting {Liquor} is a subset of the cut.

Second, by Definition 4.4, a valid solution D' defined *Cut* should not *support* any privacy threat. It follows that after striping the decoration of any threat, the resulting itemset should not be a generalization of *Cut*.

Therefore, for checking the validities of cuts, we only need to find the most general undecorated privacy threats as in Definition 4.6.

Definition 4.6 (The most general *undecorated* privacy threats): We call the itemset derived by striping the decoration of a threat (by simply removing + and -) an *undecorated* threat. An undecorated threat X is a *most general undecorated threat* if there is no other undecorated threat that is a generalization of X. ■

We will further show that only a subset of the most general undecorated threats needs to be materialized. For the brevity of presentation, in the next two subsections, we will first consider the case without negative items ($n = 0$). In such a case, all items are positive, so we simply omit the + sign in the presentation. We will discuss the general case ($n > 0$) in Section 4.3.4.

4.3.2 Inverse Itemset Enumeration Tree

We propose a novel structure, the *inverse itemset enumeration tree* for finding the most general undecorated privacy threats. Such a tree *conceptually* organizes all sets of non-sensitive items, i.e., possible candidates of threats. The process of finding the most general privacy threats can be thought of as depth-first searching such a tree, which enables strong pruning and hence helps accomplishing the job efficiently.

On the inverse itemset enumeration tree, each node u holds a non-sensitive item, which is called the *item* of u; the path from the null root to u represents an itemset, which

is simply called the *itemset* of u, denoted by $IS(u)$; and the set of all transactions that support $IS(u)$ is called the *transaction set* of u, denoted by $TS(u)$. So, the size of $TS(u)$ is the support of $IS(u)$, i.e., $sup(IS(u)) = |TS(u)|$, which is attached to the node as a count. We also attach to the node $sup(IS(u) \cup \{s\})$ for every sensitive item s, to facilitate computing $conf(IS(u) \rightarrow s)$, defined as $sup(IS(u) \cup \{s\}) / sup(IS(u))$.

We construct the inverse itemset enumeration tree as follows. First, we impose an ordering of items in which *any general item is listed before its taxonomic descendants*. This ordering corresponds to the preorder traversal of the item taxonomy H_N. Given this imposed ordering, we create the null root to represent the empty itemset, which can be thought of containing the null item that is the last one in the imposed ordering. For any node u, we grow the child nodes to hold items that are listed *before* the item of node u in the imposed ordering, and arrange child nodes from left to right in the imposed ordering of items. It follows that along any path starting from the null root, items are listed in the inverse order of the imposed one.

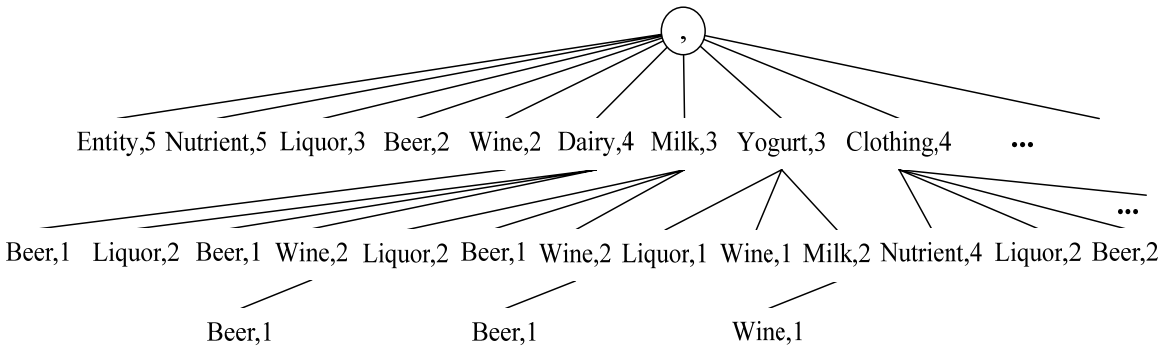


Figure 4.2 Inverse itemset enumeration tree

Example 4.9: Figure 4.2 shows part of the inverse itemset enumeration tree organizing all generalized itemsets *supported* by D in Table 4.1. Given H_N in Figure 4.1,

items by the imposed ordering are Entity, Nutrient, Liquor, Beer, Wine, Dairy, Milk, Yogurt, Clothing, Outwear, Jacket, Pants, Footwear, Geta, Hose, and Shoe, which is the preorder traversal of H_N in Figure 4.1.

Non-sensitive itemsets enumerated by a depth-first traversal are {Entity}, {Nutrient}, {Liquor}, {Beer}, {Wine}, {Beer, Wine}, {Dairy}, {Liquor, Dairy}, and so on. {Entity} is the most general non-sensitive itemset, and is the first to be enumerated. {Liquor} is a generalization of {Beer}, {Wine}, {Beer, Wine}, and {Dairy, Liquor}, and is enumerated before the latter. The path from the null root to node (Liquor, 3) represents the itemset {Liquor} with a support of 3. When enumerating this node, we can get the support of each sensitive item in the transaction set of this node, which of AdultToy is 2. So, $sup(\{\text{Liquor, AdultToy }\}) = 2$ and $conf(\{\text{Liquor}\} \rightarrow \{\text{AdultToy}\}) = 2 / 3$. ■

A key property observed from the foregoing example is that the depth-first traversal of the inverse itemset enumeration tree always visits a subset of items before its superset, and visits a general itemset before its specialization. Such a construction differentiates it from the *ordinary* set-enumeration tree [16]. Theorem 4.1 formalizes this key property, which enables the pruning of such a tree as elaborated in Section 4.3.3.

Theorem 4.1: For any itemsets X and Y such that X is a generalization of Y, X is enumerated before Y in a depth-first search of the inverse itemset enumeration tree.

Proof: Let us compare item by item the paths representing X and Y on the inverse itemset enumeration tree*, namely* $path_X$ and $path_Y$, in the top-down direction. Let x be the first item on $path_X$ that differs from the corresponding item y on $path_Y$. Since X is a generalization of Y, there is an item y' in Y such that $x = y'$ or x is a taxonomic ancestor of y'. For the both cases, y' cannot appear above y along $path_Y$ in the top-down direction

since otherwise y' also appears on $path_X$, i.e., y' is also in X, which contradicts to the fact that there is no duplicate item and no ancestor-descendant relationship among items in an itemset. So, by the construction of the inverse itemset enumeration tree, y' is listed no later than y in the imposed ordering (y and y' may be the same item).

Notice that $x = y'$ or x is a taxonomic ancestor of y', means that x is listed no later than y' in the imposed ordering (which lists a taxonomic ancestor item before its descendant items). Therefore, x is also listed no later than y in the imposed ordering. Notice that $x \neq y$ and the nodes holding x and y are siblings, thus the node holding x is at the left of the node holding y by the construction of the tree. It follows that X (i.e., $path_X$) is enumerated before Y (i.e., $path_Y$) by a depth-first search. ■

4.3.3 Prune the Inverse Itemset Enumeration Tree

We search the nodes in the inverse itemset enumeration tree in a depth-first traversal to identify the most general undecorated threats. If the itemset $IS(c)$ of the node c currently being visited is a threat, which can be told by counting $TS(c)$, and is not a specialization of any previously found threat, $IS(c)$ is a most general threat and will be materialized. We prune the subtree rooted at c if we can tell that there is no most general threat in the subtree, since we are only interested in finding the most general threats. Pruning is critical as such a tree is huge. In the following, we present a series of pruning techniques.

Baseline pruning: We can prune the subtree rooted at c if IS(c) is beyond the power of any adversary's knowledge by Observation 4.1, or if there is no sensitive item in $TS(c)$ and we only consider the sensitive item disclosure ($k = 1$). Under these

conditions, there will be no privacy threat in the subtree. We call it the *baseline* pruning as it is far from enough.

Pruning I (Global pruning): If *IS*(*c*) is a specialization of a previously materialized threat, we can prune the subtree rooted at *c* and the subtrees rooted at the *c*'s siblings (in the inverse itemset enumeration tree) whose items are taxonomic descendants of *c*'s item since it is impossible for any node in such subtrees to represent a most general threat.

Example 4.10: Node (Liquor, 3) in Figure 4.2 represents {Liquor}. As {Liquor} is the first threat found by the depth-first search, node (Liquor, 3) is materialized. Afterwards, any node holding Liquor, Beer, or Wine will be pruned by Pruning I. ■

Note that Pruning I comes with the computational overhead of *globally* checking if *IS*(*c*) is a specialization of any threats materialized so far, which is the reason we call it the global pruning. The second and third pruning techniques below try to avoid as many such global checks as possible, and hence to improve the efficiency.

Pruning II (Local pruning): If *IS*(*c*) is a threat and *c* is not pruned by Pruning I then *IS*(*c*) is a most general threat, we keep *c* on the tree and stop growing its children since they represent the specializations of *IS*(*c*), and prune *c*'s siblings (in the inverse itemset enumeration tree) whose items are taxonomic descendants of *c*'s item.

Example 4.11: Continue with the example for Pruning I. What Pruning II does is to *only* prune the siblings of node (Liquor, 3), i.e., node (Beer, 2) and node (Wine, 2), which comes with little computational overhead. Any other node holding Liquor, Beer, or Wine will be left to the global pruning. This is why it is called the local pruning. ■

Theorem 4.2: For any node c and its parent node p, if $IS(c)$ and $IS(p)$ have the same support, there is no most general threat in the subtree rooted at c.

Proof: We show that for any node v in the subtree rooted at c, there is a node w ($\neq v$) in the subtree rooted at p such that (1) $IS(v)$ is a specialization of $IS(w)$, and (2) if $IS(v)$ is a threat, so is $IS(w)$. This means that v will never represent a most general threat. Below, we construct the node w.

First, $TS(c)$ is the subset of transactions in $TS(p)$ that support node c's item. Since $IS(c)$ and $IS(p)$ have the same support, i.e., $|TS(c)| = |TS(p)|$, we have $TS(c) = TS(p)$. Second, let $Z = IS(v) - IS(c)$. Every item in Z is listed before the item of p in the imposed ordering by the construction of the tree, so there is a node w in the subtree rooted at p such that $IS(w) = IS(p) \cup Z$. Clearly, $IS(w)$ is derived by removing node c's item from $IS(v)$. Therefore, $w \neq v$ and point (1) holds.

Notice that $TS(v)$ is the subset of transactions in $TS(c)$ that *support* Z, and $TS(w)$ is the subset of transactions in $TS(p)$ that *support* Z. Since $TS(c) = TS(p)$, we have $TS(v) = TS(w)$, which implies that if $IS(v)$ is a threat, so is $IS(w)$. Therefore, point (2) holds. ■

Pruning III (Generic pruning): For a child node c and its parent node p, if $IS(c)$ and $IS(p)$ have the same support, the subtree rooted at c can be pruned by Theorem 4.2. We call it the generic pruning as it depends only on supports of itemsets and is independent of the privacy requirement (i.e., regardless of the values of k, l, and m).

Example 4.12: As a trivial example in Figure 4.2, on the path from the null root through node (Clothing, 4) to node (Nutrient, 4), the last two nodes have the same

support of 4. By pruning III, node (Nutrient, 4) can be pruned without missing any most general threat. ■

The fourth pruning can drastically reduce the number of privacy threats to be materialized. We will show the effectiveness of all the pruning technique in the experimental evaluation.

Pruning IV (Permanently locked item pruning): If a threat only consists of the taxonomic children of an item x, specializing x will always introduce the threat because it results in a specialization of the threat. Therefore, x will not be specialized by any valid cut. For this reason, we call such x a *permanently* locked *item* and we can remove all the taxonomic descendants of such a permanently locked x from the inverse itemset enumeration tree.

As a consequence of Pruning IV, only a subset of the most general threats will be materialized. If we exclude any cuts containing the descendants of permanently locked items (which are invalid cuts) in the process of searching an optimal cut, this subset of the most general threats is sufficient for the validity check of any other cut.

Example 4.13: In the motivating example, {Liquor} is a threat, so Liquor's taxonomic parent Nutrient is a permanently locked item. By pruning IV, all taxonomic descendants of Nutrient, i.e., Liquor, Beer, Wine, Dairy, Milk, and Yogurt can be excluded from further consideration. Therefore, {Liquor}, one of the most general threats, will not be included in the subset of the most general threats materialized, and so forth. ■

4.3.4 Extending to the General Case ($n > 0$)

In the previous two subsections, we only consider the case with $n = 0$ for the brevity of presentation. For the general case with $n > 0$, the inverse itemset enumeration tree holds both i+ and i- for each non-sensitive item i. Basically, we can treat i+ and i- as two different items except the following.

First, the imposed ordering of decorated items ensures that after striping decoration (removing + and -), any general item is listed before its taxonomic descendents. In our example, all decorated items by such an ordering are Entity+, Entity-, Nutrient+, Nutrient-, Liquor+, Liquor-, Beer+, Beer-, Wine+, Wine-, Dairy+, Dairy-, Milk+, Milk-, Yogurt+, Yogurt-, Clothing+, Clothing-, and etc.

Second, the pruning is based on the generalization relationship without considering decoration. For example, {Liquor+} as a threat will prune {Beer-} since the former is a generalization of the latter when ignoring the decoration.

4.4 Searching an Optimal Solution

Our OG algorithm consists of two phases. The second phase of OG is an adaptation of the l^+-Optimize algorithm in Chapter 3, which is originally proposed to find an optimal anonymization solution for a relational table to observe the l^+-diversity notion.

In this section, we first show how to adapt l^+-Optimize to OG, then show how to use the materialized set of the most general undecorated threats presented in the last section to develop the new component, validity checking.

4.4.1 The OG Algorithm

One difference between OG and l^+-Optimize is that OG only employs generalization while l^+-Optimize employs both generalization and suppression. Therefore, the second phase of OG employs both cost based pruning and validity based pruning while l^+-Optimize only employ cost based pruning.

4.4.1.1 Adaptation of l^+-Optimize

We also employ the cut enumeration tree proposed in Section 3.2.1 to organize all cuts (all possible solutions) conceptually. For example, Figure 4.3 shows such a tree that enumerates all possible cuts on the taxonomy in Figure 4.1. Hereafter, *a cut also refers to the node* representing it on the cut enumeration tree. A child cut is derived from a parent cut by specializing exactly one constituent item of the parent cut. To avoid duplicate enumeration, we differentiate the constituent items of a cut between open items and locked items. An open item in a cut will be further specialized to derive a child of the cut, while a locked item will not. On the cut enumeration tree, e.g. in Figure 4.3, open items are in italic and locked items are underlined.

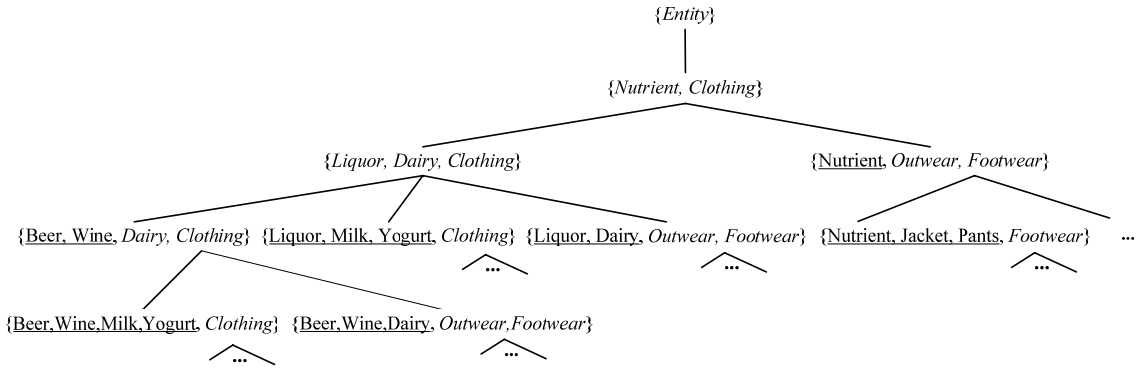


Figure 4.3 Cut enumeration tree

Example 4.14: In Figure 4.3, there is an edge from cut {Liquor, Dairy, Clothing} to cut {Liquor, Dairy, Outwear, Footwear} as the latter is derived by specializing Clothing. Although we could also derive the latter by specializing Nutrient in cut {Nutrient, Outwear, Footwear}, there is no edge between the two as otherwise {Liquor, Dairy, Outwear, Footwear} will be enumerated twice. This explains that Clothing is open in cut {Liquor, Dairy, Clothing} while Nutrient in cut {Nutrient, Outwear, Footwear} is locked. ■

OG depth-first searches the cut enumeration tree to find the optimal solution as in Section 3.2.2. Since the number of cuts is huge with any non-trivial taxonomy, it is critical to prune the cut enumeration tree. We also employ the cost based pruning, and techniques that improve the chance of pruning.

Cost based pruning. We are interested in finding a cut with the minimum information loss, if a lower bound, denoted by $LB(Cut)$, on the costs of cuts in the subtree rooted at *Cut*, exceeds the running best cost, the subtree rooted at *Cut* can be pruned. This cost based pruning condition can be expressed as $LB(Cut) > cost(Cut_{best})$.

Notice that the locked constituent items of *Cut* will not be specialized in the subtree rooted at *Cut* while the open constituent items can be specialized to leaf items. Therefore, we have

$$LB(Cut) = \sum_{x \in Cut \wedge x \text{ is locked}} Cost(x)$$

for any well-behaved metric because leaf items do not incur information loss.

Techniques that improve the chance of pruning. We also employ the two dynamic techniques presented in Section 3.4 to improve the cost based pruning. One is to

list the open items of *Cut* in the descending order of costs, i.e., to specialize items with large cost early by the construction of the cut enumeration tree. The other is to update cost lower bound dynamically when backtracking in the depth-first searching of the cut enumeration tree.

4.4.1.2 Two Additional Ingredients of OG

Taxonomy reduction. This is related to Pruning IV for the inverse itemset enumeration tree discussed in Section 4.3.3 where we pointed out that any cuts containing the descendants of permanently locked items are invalid. Therefore, we can simply exclude the descendants of permanently locked items from the item taxonomy H_N so that the taxonomy and hence the cut enumeration tree is reduced significantly. Moreover, before enumerating cuts, we can keep on pushing an initial cut down from the root of the taxonomy tree as long as it does not contain a constituent item of any materialized threat, and then start top-down searching of cuts from such an initial cut.

Example 4.15: As in Example 4.13, Nutrient is a permanently locked item, so its descendants Liquor, Beer, Wine, Dairy, Milk, and Yogurt can be removed from the taxonomy in Figure 4.1. And cut {Nutrient, Jacket, Pants, Footwear} is such an initial cut, in the other words, the upper portion of the taxonomy containing items Entity, Clothing, and Outwear can also be removed. So, the cut enumeration tree in Figure 4.3 will also be reduced. ■

Validity based pruning. If a threat is supported by $D' = gen(D, Cut)$, then this threat is also supported by $D'' = gen(D, Cut')$ if Cut' is a specialization of Cut. In other

words, if *Cut* is invalid, all its specializations are invalid. So, the subtree rooted at *Cut* can be pruned.

4.4.1.3 Check the Validity of a Cut

We discuss how to check the validity of a cut to facilitate the validity based pruning, an additional ingredient of OG just presented in the foregoing subsection.

Clearly, it is inefficient to compare the cut with every materialized threat. Notice that when top-down searching the cut enumeration tree, we stop searching the substree rooted at a cut if the cut is invalid. As a consequence of this strategy, actually only a small portion of such materialized threats need to be check as suggested by Theorem 4.3.

Theorem 4.3: Suppose *Cut* is derived from a *valid* cut, Cut_y, by specializing item *y*. If *Cut* is invalid, i.e., if it is a specialization of a certain threat *X*, then *X* must have a constituent item that is a taxonomic child of *y*.

Proof: Let *children*(*y*) be the set of the taxonomic children of *y*. So, $Cut = Cut_y - \{y\} \cup children(y)$. Since Cut_y is valid, *X* cannot be a generalization of Cut_y. But *X* is generalization of *Cut*. According to the difference between *Cut* and Cut_y, there must be an item *x* in *X* such that *x* is not a generalization of *y* but is a generalization of a certain item in *children*(*y*). This holds only if *x* itself is an item in *children*(*y*), i.e., $x \in children(y)$. This shows that an item (i.e., *x*) in *X* is a taxonomic child of *y*. ■

OG only evaluates each child Cut_c of a valid parent Cut_p in a top-down fashion. Therefore, the first threat that invalidates Cut_c must be the most general one. To determine the validity of Cut_c, we only need to retrieve the materialized, most general undecorated threats that contain an item from $Cut_c - Cut_p$ by Theorem 4.3.

Example 4.16: Suppose we first verified cut {Nutrient, Outwear, Footwear} and found it is valid. Then, when we verify cut {Nutrient, Jacket, Pants, Footwear}, derived by specializing Outwear into Jacket and Pants, we only need to compare it with the most general threats that contain Jacket or Pants. ■

4.4.1.4 The Pseudo Code of OG

Now, we present the full version of our optimal generalization algorithm OG. The *MaterializeThreats*(*D*) procedure (line 1) produces the materialized set of the most general undecorated threats, *mmgts*, by the approach presented in Section 4.3. The reduced H_N (line 2) is related to Pruning IV for the inverse itemset enumeration tree in Section 4.3.3. The *validity*(*Cut*, *mmgts*) procedure checks the validity of *Cut* by retrieving relevant threats in *mmgts* as discussed in Section 4.4.1.3.

The *OptimalSearch* procedure is an adaptation of l^+-Optimize as discussed in Section 4.4.1.1. Given Cut_{best} be the current best cut among the cuts searched so far and *Cut* be the cut to be visited, the procedure searches the subtree rooted at *Cut* and returns the updated current best cut as follows.

If the cost lower bound of the subtree rooted at *Cut* is greater than the cost of the current best cut (line 4) or *Cut* is invalid (line 5), the subtree is pruned, and the search ends. If the cost of *Cut* is less than the cost of Cut_{best}, *Cut* becomes the new current best cut (line 6). For each `open` constituent item of *Cut*, a child cut is derived by specializing the item, and the subtree rooted at this child cut is recursively searched (line 7-9). At the end of the search, the current best cut is returned.

Algorithm 4.1 The optimal generalization algorithm OG

OG(*D*)

1: $mmgts \leftarrow MaterializeThreats(D)$
2: $Cut_{best} \leftarrow Cut \leftarrow$ {roots of the reduced H_N }
3: **return** *OptimalSearch* (Cut, Cut_{best}, $mmgts$)

OptimalSearch(Cut, Cut_{best}, $mmgts$)
4: **if** $LB(Cut) > cost(Cut_{best})$ **then return** Cut_{best}
5: **if** $validity(Cut, mmgts) = false$ **then return** Cut_{best}
6: **if** $cost(Cut) < cost(Cut_{best})$ **then** $Cut_{best} \leftarrow Cut$
7: **for each** item x **in** the list of the open items of Cut **do**
8: $Cut_{child} \leftarrow Cut - \{x\} \cup children(x)$
9: $Cut_{best} \leftarrow OptimalSearch(Cut_{child}, Cut_{best}, mmgts)$
10: **return** Cut_{best}

A note on implementation. There are some implementation details that also contribute to the efficiency of OG. One detail is how to compute $Sup(IS(u))$ for a node u in the inverse itemset enumeration tree. Another detail is how to represent transactions with hierarchical items. We address these issues by extending the FP-tree [44] with taxonomic labels and adapting the frequent itemset mining algorithm [64] to implement the $MaterializeThreats(D)$.

4.4.2 A Multi-Round OG Approach

OG is more efficient than AA as the extensive experiments in Section 4.5.2 will show. However, when m and n are extremely large, the number of threats materialized by the first phase is still quite large, which affects the efficiency. To further improve the efficiency, we propose a multi-round OG approach, namely mOG, which can find a solution very close to the optimal.

The idea is to do anonymization progressively. That is, to observe KL(m'', n'')-privacy, we first anonymize D to produce D' that observes KL(m',n')-privacy with $m' < m''$

and $n' < n''$, and then we anonymize D' to produce D'' observing KL(m'',n'')-privacy. This idea can be applied multiple times.

In particular, we can run OG in $m + n$ rounds to produce $D^{m,n}$ observing KL(m, n)-privacy. That is, we can first produce $D^{i,0}$ for $i = 1, .., m$, and then produce $D^{m,j}$ for $j = 1, .., n$. mOG is efficient in the early rounds as it only eliminates threats of small sizes, whose number is limited even with a large number of distinct items. mOG is also efficient in the later rounds as it works with a decreased number of distinct items, which results in a limited number of threats although it deals with threats of larger sizes.

Notice that the information loss with $D^{i,j}$ is measured relative to the original database D. The runtime for mOG to produce $D^{m,n}$ is the total runtime of the $m+n$ rounds.

4.5 Experimental Evaluation

In this section, we investigate two key points by comparative experiments. The first key point is that assuming the adversary has unbounded knowledge as in [41][48] will introduce excessive distortion, which is investigated by comparing with the local generalization algorithm LG [48]. The conclusion also applies to [41] in general.

The second key point is that KL(m, n)-privacy provides sufficient privacy protection, and our optimal algorithm and the multi-round approach help retain more data utility than heuristic algorithms and is efficient and scalable. This point is investigated by comparing with the global generalization algorithm AA [95], and the total suppression algorithm MM [108]. We obtained the code of AA from the authors of [95] and obtained the code of MM from the authors of [108]. We implemented LG [48] as its executable is not available.

Therefore, we conduct experiments on 3 real world databases which were introduced by [115] and also used by [41][48][95]. The BMS-POS (POS) database is a transaction log from several years of sales of an electronics retailer. BMS-WebView-1 (WV1) and BMS-WebView-2 (WV2) contain several months of click-stream data of two e-commerce web sites. Table 4.3 summarizes the characteristics of the 3 databases, where $|D|$ is the size of a database, $|I|$ is the number of items, *max* $|t|$ is the maximum length of transactions, and *avg* $|t|$ is the average transaction length. Taxonomy trees were created by the procedure in [95], which randomly assigns domain items to the leaves of the tree. We report the average result based on 3 such random assignments.

Table 4.3 Characteristics of the databases

Database	$\|D\|$	$\|I\|$	*max* $\|t\|$	*Avg* $\|t\|$
POS	515,597	1,657	164	6.5
WV1	59,602	497	267	2.5
WV2	77,512	3,340	161	5.0

We first evaluate the data utility in the terms of a general information loss metric, NCP [109] that is a variant of LM [52], and usefulness in frequent pattern mining, then assess the efficiency and scalability of algorithms, and analyze the effectiveness of each individual technique at last. All experiments were conducted on a 2.6GHz Intel Pentium IV PC with 1GB RAM.

4.5.1 Data Utility Evaluation

4.5.1.1 Information Loss Comparison with MM, LG, AA

We investigate if our optimal algorithm OG and our multi-round approach mOG have a significant gain in data utility relative to MM [108], LG [48], and AA [95]. We set

l to 1 and n to 0, which is the setting applicable to all the algorithms except LG [48]. However, LG [48] is still included in the comparison as we want to show that assuming $m+n = |I_N|$ as in LG [48] will incur unnecessary distortion. The default setting of other parameters is $k = 5$, and $m = 3$ with the databases POS and WV1, and $m = 6$ with WV2.

Figure 4.4(a) shows the information loss on POS with varying k. The information loss incurred by MM is the highest (over 29%), that by LG is the second highest, that by AA is the third highest. OG has 1/4 to 1/2 less information loss than AA. The information loss by our multi-round approach, mOG, is almost the same as OG.

Figure 4.4(b) shows the information loss on POS with varying m. The information loss by MM increases drastically with the increase of m and is the highest when $m \geq 2$. OG (mOG) incurs the least information loss. When m is small the gap in information loss between AA and OG (mOG) is not significant, but when m is large (≥3), AA incurs 1/2 more information loss than OG.

Although the maximum length of the original transaction is very large (i.e. 164), the information loss does not increase when $m \geq 10$ since the generalized transactions shrink. We observed the same results on WV2 and WV1 as shown in Figure 4.4(c)-(f).

In short, OG incurs the least information loss. Our multi-round approach, mOG, though not guaranteeing optimality, finds a solution very close to the optimal. The comparison with AA [95] shows that an optimal solution has a significant gain in data utility relative to a heuristic solution. The information loss by LG [48] is the second highest, which is because LG exerts over-protection that causes excessive distortion. The

information loss by MM is the highest, as [108] acknowledged because suppression is not suitable for sparse data.

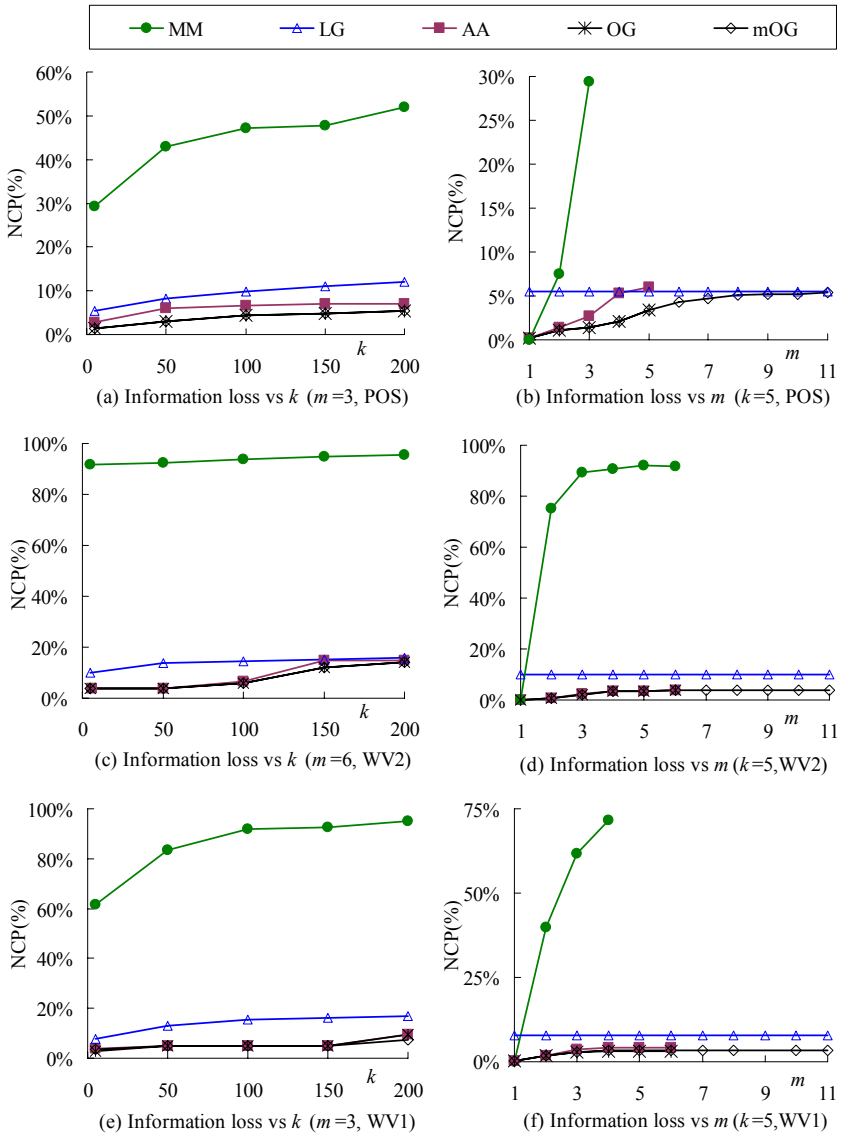


Figure 4.4 Information loss comparison

4.5.1.2 Data Utility in Frequent Pattern Mining

We further evaluate the data utility by comparing our mOG with LG in frequent pattern mining. Notice that LG treats all items as non-sensitive. So, in this set of

experiements, we enforce KL(m, n)-privacy with l = 1, i.e. do not consider the sensitive item disclosure. For simplicity, we set the same value for m and n, i.e. $m = n$.

We measure the data utility in terms of two indicators as defined as follows. Given a support threshold, let *OFP* be the set of generalized frequent patterns [91] in the original data, and *AFP* be that in the anonymized data. First, we define $\#Patterns = |AFP|$, which is an indicator of the amount of information that is retained. Second, we define an indicator of the accuracy of the retained information, where $sup(r, OFP)$ and $sup(r, AFP)$ are the supports of pattern r in *OFP* and *AFP* respectively.

$$Error = \frac{\sum_{r \in OFP \cap AFP} |sup(r, OFP) - sup(r, AFP)|}{\sum_{r \in OFP \cap AFP} sup(r, OFP)}$$

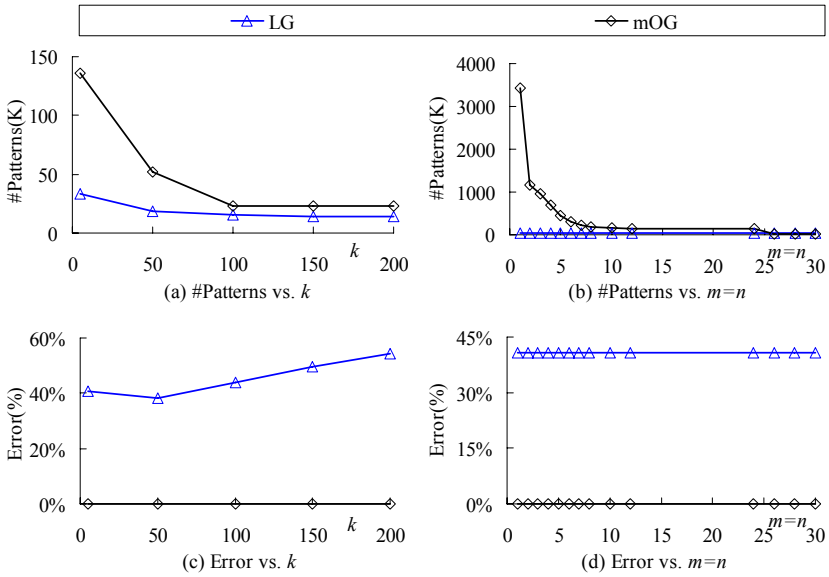


Figure 4.5 Data utility in frequent pattern mining

Figure 4.5(a) and (b) show #*Patterns*, and (c) and (d) show Error with varying k and with varying m respectively, given 1% be the frequency threshold. (a) and (b) depicted that mOG retains much more information than LG. (c) and (d) depicted that mOG keeps the supports of patterns accurately, i.e. the same as in the original data, while LG has an error rate around 40%. In short, our algorithm retains more data utility than LG in terms of frequent pattern mining.

4.5.1.3 Information Loss in the General Case

We investigate if we can provide sufficient protection and preserve enough data utility in enforcing the full requirement of our KL(m, n)-privacy notion. We run mOG instead of OG since mOG is more efficient than OG for large m and n, and mOG can find a solution very close to that of OG. There is no comparison with other algorithms as none of them enforces KL(m, n)-privacy with $n > 0$ and $l > 1$.

As in [41][108], we randomly selected a fixed number β (beta) of sensitive items from items with a support less than 50%, and consider the remaining items as non-sensitive items. For simplicity, we set $m = n$. The default setting is (m, n, k, θ, β) = (3, 3, 5, 20%, 60), where θ (theta) stands for 1 / l.

Figure 4.6(a)-(c) show the information loss with m and n varying from 1 to 30, with β varying from 20 to 100, and with θ (1 / l) varying from 5% to 30% respectively. Under all setting, the information loss is no more than 7%. In short, we provide sufficient protection (with m and n up to 30) while retaining enough data utility (with NCP ≤ 7%).

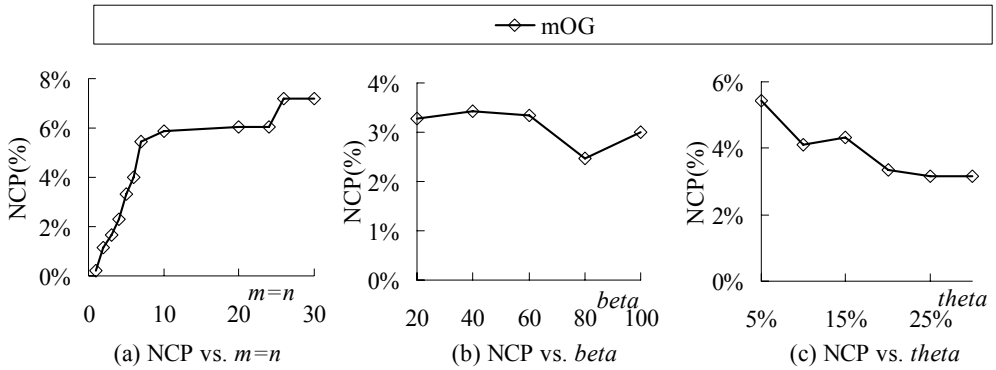


Figure 4.6 Information loss in the general case (n>0, *l* >1)

4.5.2 Evaluation of Efficiency and Scalability

We evaluate the efficiency of our optimal algorithm OG and our multi-round approach mOG by comparing with MM, LG, and AA in enforcing KL(m, n)-privacy with $l = 1$ and $n = 0$. Figure 4.7(a) shows the runtime on POS with varying k. LG is the most efficient, MM is the second, and OG is the third. OG is 2 to 4 times faster than AA. The runtime of mOG is slightly longer than OG with $m = 3$.

Figure 4.7(b) shows the runtime on POS with varying m. LG is the most efficient; MM is the second when m is small, but MM cannot run when $m \geq 4$ which probably is because of scalability issue. OG is 65% to 4 times faster than AA. When $m \geq 6$, and mOG is the second most efficient. We observed the same results on WV2 and WV1 as shown in Figure 4.7(c)-(f).

In short, LG is the most efficient as it employs a divide-and-conquer approach. But, it comes with an expense, i.e. the anonymized data does not observe the domain exclusiveness. Although OG is less efficient than LG, the significant gain in data utility is worth the longer runtime. OG is more efficient than AA and MM while guaranteeing optimality. AA and MM are less efficient because the breadth-first search approaches

they employ are not efficient. mOG further reduces runtime for large m and n. Overall, we recommend OG. If m and n are large with a very large database, mOG could be more efficient and provide a solution with a near-optimal utility.

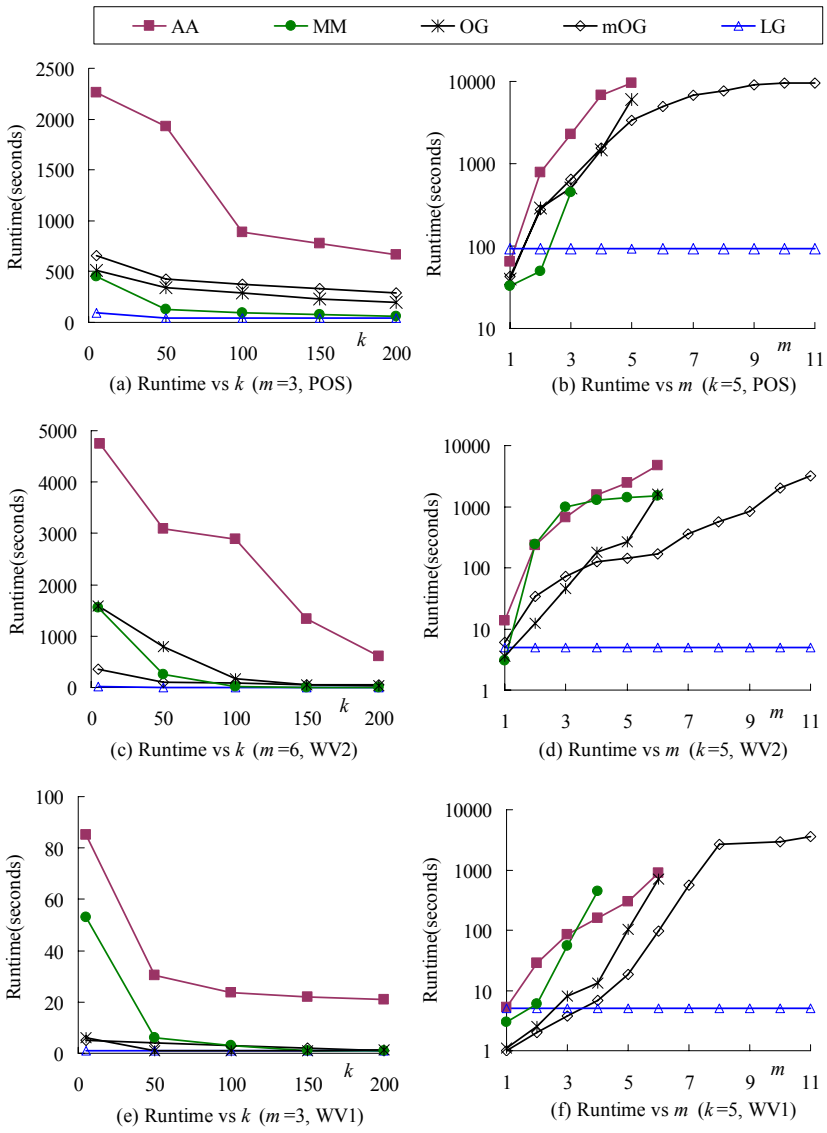


Figure 4.7 Runtime comparison

We briefly discuss the scalability of OG and mOG. To do so, we created variants of the POS database by randomly selecting transactions. Therefore, we have a suite of datasets made of 100K up to 1,000K transactions on which the runtime of OG and mOG

are depicted in Figure 4.8. As the search space depends much more on the item taxonomy than the size of the database. The runtime is nearly linear to the number of transactions. In short, OG and mOG is quite scalable.

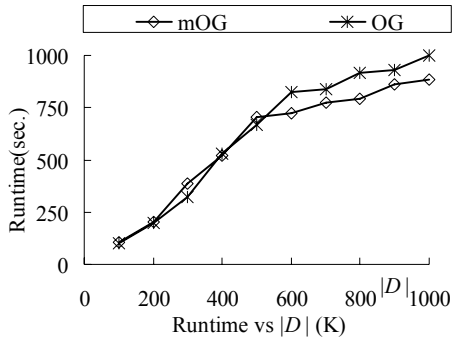


Figure 4.8 Scalability test

4.5.3 Analysis of the Effectiveness

Furhtermore, we evaluated the effectiveness of pruning techniques employed in finding the most general threats in Phase 1 of our algorithm, and effectiveness of cost based pruning methods employed in finding an optimal solution in Phase 2. The analytical experiments were performed on the WV1 database and the POS database with the default setting (m, n, k, l) = (3, 0, 5, 1).

4.5.3.1 Effectiveness of Techniques in Phase 1

The effectiveness of the techniques in Phase 1 (mining the most general privacy threats) can be measured by 2 indicators. One is the number of the inverse itemset enumeration tree nodes that are examined, abbreviated by *#nodes-examined*, and the other is the number of the privacy threats that are materialized, abbreviated by *#threats-materialized*. A small *#nodes-examined* suggests a strong pruning and a small *#threats-materialized* suggests less memory usage. Both contribute to the efficiency of OG.

Table 4.4 Effectiveness of the pruning techniques in Phase 1

DB	#	Techniques	#threats-materialized	(%)	#nodes-examined	(%)	Time (seconds)
WV1	1	Baseline: mining threats by Defintion 1	4,445,319	100%	5,521,578	100%	200
	2	Pruning I (global pruning)	1,115,701	25%	3,453,701	63%	164
	3	Pruning II (local pruning)	3,367,660	76%	4,852,293	88%	145
	4	Pruning III (generic pruning)	4,302,056	97%	5,165,683	94%	153
	5	Pruning I and II and III	1,115,701	25%	3,275,754	59%	113
	6	Pruning IV (permanently locked items)	65,891	1.5%	788,806	14%	9
	7	All prunings (OG)	39,265	0.9%	388,966	7%	5
	8	Multi-rounds (mOG)	13,590	0.3%	140,206	2.5%	3
POS	9	Baseline: mining threats by Defintion 1	47,978,473	100%	53,464,069	100%	24,361
	10	All prunings (OG)	130,685	0.3%	1,075,845	2%	570
	11	Multi-rounds (mOG)	48,029	0.1%	358,916	0.7%	578

We start our analysis by disabling all pruning techniques except the baseline pruning, and use *#nodes-examined* and *#threats-materialized* at this setting as the *baseline* of analysis. Then we turn on one pruning strategy at a time and compare these indicators with the baseline. The results are shown in Table 4.4.

With the WV1 database, the baseline is *#threats-materialized* = 4.4M and *#nodes-examined* = 5.5M, shown at line 1 in Table 4.4. That is, after examining 5**.**5 million inverse itemset enumeration tree nodes, 4**.**4 million threats are materialized.

Line 2 in Table 4.4 shows the results of turning on the global pruning (Pruning I). This strategy is fundamental in that it prunes all threats that are not the most general ones. As a result, it decreases *#threats-materialized* to 25% and decreases *#nodes-examined* to 63%, relative to the baseline. Note that this pruning comes with a computational overhead associated with comparing the generalization relationship between threats materialized so far and the itemset being currently examined.

Line 3 shows the results of turning on the local pruning (Pruning II). This technique removes some threats that are not most general and decreases *#threats-materialized* to 76% and decreases *#nodes-examined* to 88%. Although the local pruning

is not as thorough as the global pruning, it makes Phase 1 more efficient since the local pruning requires little extra overhead.

Line 4 shows that the generic pruning (Pruning III) did less than the local pruning and the global pruning in decreasing *#threats-materialized* and *#nodes-examined*; however, it also contributes to the overall improved efficiency.

When Pruning I, II, III are all turned on (line 5), *#threats-materialized* is the same as only employing Pruning I, and *#nodes-examined* decreases moderately from 63% to 59%. However, the local pruning and the generic pruning contribute a lot in improving the efficiency in terms of runtime, compared to Pruning I alone, as shown in the last column in Table 4.4.

By the permanently locked item pruning (Pruning IV), only a subset of the most general privacy threats needs to be materialized for checking the validities of cuts. What surprised us is the drastic effect of Pruning IV. As shown at line 6, by turning on Pruning IV, *#threats-materialized* is 1.5% and *#nodes-examined* is 14%, and the runtime is 1 / 22, relative to the baseline.

Putting all the 4 pruning techniques together, as shown at line 7, *#threats-materialize* decreases to 0.9% and *#nodes-examined* to 7% relative to the baseline. Even doing so, Phase 1 may still face efficiency issue when the problem is of a very large scale. This is why we further introduced a multi-round OG approach, mOG.

At line 8, relative to the baseline, *the total number* of threats materialized *by all rounds* in mOG is 0.3%, and *the total number* of the inverse itemset enumeration tree nodes examined *by all rounds* in mOG is 2.5%. And *the total runtime for Phase 1 of all*

rounds in mOG is 3 seconds. This means mOG is more efficient than OG (in Phase 1). Although with the default setting (m, n, k, l) = (3, 0, 5, 1) the efficiency improvement of mOG over OG looks not that significant in terms of runtime, the efficiency improvement by mOG will be drastic when m and n increase.

With the POS database, the baseline is *#threats-materialized* = 50M and *#nodes-examined* = 53M, and the runtime is nearly 7 hours without employing pruning techniques (line 9). With all pruning techniques, *#threats-materialized* is 131K and *#nodes-examined* is 1M, i.e., 0.3% and 2% relative to the baseline respectively (line 10). *The total number* of threats materialized *by all rounds* in mOG is 48K, and *the total number* of the inverse itemset enumeration tree nodes examined *by all rounds* in mOG is 358K, i.e., 0.1% and 0.7% relative to the baseline (line 11). T*he total runtime for Phase 1 of all rounds* in mOG is 578 seconds. Again with the default setting (m, n, k, l) = (3, 0, 5, 1), the runtimes of OG and mOG have no much difference. As m and n increase, the benefit of mOG will be significant.

4.5.3.2 Effectives of Methods in Phase 2

The effectiveness of the methods in Phase 2 (searching the optimal generalization solution) can be measured by the number of cuts that are examined, abbreviated by *#cuts-examined*. The baseline for analysis is the number of valid cuts, i.e., the number of cuts examined when only employing *validity pruning*, which is 35K with the WV1 database shown at line 1 in Table 4.5. Note that the total number of cuts is 23G with WV1. So, the validity pruning of cuts is crucial. The discussion below refers to Table 4.5.

Table 4.5 Effectiveness of the methods in Phase 2

DB	#	Techniques	#cuts-examined	(%)	Time
WV1	1	**Baseline: pruning cuts by validity**	35,004	100%	47
	2	**Method I (cost based pruning of cuts)**	19,744	56%	11
	3	**Method I and II (ordering items)**	4,596	13%	4
	4	**I and II and III (tighten lower bounds)**	2,266	6%	3
	5	**Method IV (taxonomy reduction)**	12,088	34.5%	9
	6	**All methods (OG)**	132	0.4%	2
	7	**Multi-rounds (mOG)**	509		2
POS	8	**All methods (OG)**	52	n/a	9
	9	**Multi-rounds (mOG)**	24,039	n/a	2

The cost based pruning of cuts (Method I) without *item ordering* helps reduce *#cuts-examined* to 56% (line 2) relative to the baseline. By sorting items in the descending order of costs (Method II), *#cuts-examined* decreases to 13% (line 3). Further integrating the method that tightens the lower bounds (Method III), *#cuts-examined* is reduced to 6% (line 4). The taxonomy reduction method (Method IV) *alone* can reduce *#cuts-examined* to 34.5% (line 5) relative to the baseline.

The full strength OG that employs all methods examines only 0.4% of valid cuts (line 7). A little surprise is that the total number of cuts examined by all rounds in mOG is more than OG. This is because in the early rounds of mOG, there are much more valid cuts than in the solo round of OG.

4.6 Discussions and Extensions

4.6.1 Discussions

This chapter assumes that an adversary has the knowledge about the absence of certain items from his / her target individual's transactions. Therefore, we do not allow suppression as suppressing presence (absence) of items will introduce false absence (presence) of items and hence artificial privacy threats. However, if no adversary has

such knowledge about absence in a particular application, suppression can be integrated with generalization in anonymizing the data for that particular application [66].

This chapter models a privacy threat as a set X of decorated, generalized, non-sensitive items with $power(X) \leq (m, n)$. By Definition 4.2 and Observation 4.1, $power(X)$ is the number of disjunctions of positive items, and the total number of taxonomic leaf descendants of items whose negative decorations are in X. However, our algorithm is not limited to such definitions. Specifically, any measurement of the adversary's power can be plugged into our algorithm by modifying the condition that decides whether an itemset is beyond the power of the adversary's knowledge.

4.6.2 Extensions

As the l^+-Optimize algorithm in Chapter 3, the OG algorithm has two extensions.

First, OG can be run as an "*anytime*" algorithm in a time constrained setting. In particular, OG can be terminated in a pre-set amount of time. This anytime version does not guarantee optimality. However, as the top-down, depth-first search employed by OG tends to enumerate cuts with small costs first, we expect that the anytime version would find a cut very close to the optimal in a short time.

Another extension is that if a small compromise of optimality is tolerable, we can further strengthen the cost based pruning of cuts by replacing the pruning condition $LB(Cut) > cost(Cut_{best})$ with the condition $(1+\alpha) \cdot LB(Cut) > cost(Cut_{best})$, where α is a small positive value. This condition increases the chance to prune cuts while guaranteeing that the solution is a $(1+\alpha)$-approximation of the optimal.

4.7 Summary

Privacy protection in publishing set-valued data is an important problem. However, privacy notions proposed in prior works either do not prevent certain privacy attacks or provide over-protection that causes excessive distortion. Moreover, there is a lack of scalable optimal solutions.

This chapter proposed a new privacy notion, KL(m, n)-privacy, which prevents both the identity disclosure and the sensitive item disclosure to the privacy adversary who has both the knowledge about the presence of some items in and the knowledge about the absence of some items from his / her target individual's transaction. This notion is more flexible yet more general than the prior works.

In addition to the new privacy notion, the contribution is an optimal algorithm OG that is applicable for anonymizing large, real world databases. OG employs several novel techniques, the inverse itemset enumeration tree for materializing the most general privacy threats, the cut enumeration tree for representing solutions, and various pruning techniques.

Extensive experiments on real world databases showed that our algorithm outperforms the state of the art algorithms. OG outperforms the heuristic generalization algorithm AA [95] and the heuristic suppression algorithm MM [108] both in data utility and in efficiency. OG also outperforms the local generalization algorithm in terms of data utility. We also presented a multi-round approach to further improve efficiency, which finds a solution very close to the optimal.

5: Vocabulary *k*-Anonymity for Releasing Web Search Data

Web search has become the most essential tool for people to find information in their daily lives. Web query logs retained by search engines provide a rich wealth of information extremely useful as a research or marketing tool [27][76][107], but they also present serious privacy risks [2][15]. For example, Table 5.1 shows a (disguished) tiny portion of the web query log published by AOL in 2006 [76], which resulted in an incident ranked as one of the "101 Dumbest Moments in Business" [49].

Indeed, a query log published in its entirety is extremely vulnerable to privacy attacks even after heavily disguised. For example, a query log anonymized by replacing every query-term with a hash value can be compromised by statistical analysis to figure out the mapping between terms and hashes [58]. A query log anonymized by removing names and numbers can be used to build classifiers that map a sequence of queries to the gender, age, and location of the user issuing the queries, which facilitates trace attacks and person attacks by integrating with session information [54]. The implication is that there is no one-size-fits-all solution for privacy preserving publishing of query logs. Publication scenarios and sound privacy guarantees are application dependent.

This chapter considers publishing the vocabularies extracted from a query log instead of the query log itself, and protecting privacy in such a scenario. A vocabulary is the bag of query-terms derived by merging queries issued by a user. Such data has a variety of applications, including web search personalization, advertisement, query suggestion, and query spelling correction [27][107]. We allow a spectrum of granularities

in merging queries into vocabularies since a different application may require a different granularity level [107]. For example, for the web search personalization application the granularity may be the user level, i.e., all the queries by each user may be merged into one vocabulary, while for the query suggestion application one possible granularity could be the session level, i.e., all the queries of each user at one session comprise an independent vocabulary.

Table 5.1 Part of the original AOL query log

AnonID	Query	QueryTime	ItemRank	ClickURL
0001	care packages	1:00 Jan 1	3	a.com
0001	movies for dogs	2:00 Jan 1	8	b.com
0001	big cuddly dog	2:20 Jan 1	3	c.com
0001	movies on bipolar	1:00 Jan 2	5	d.com
0001	rescue of older dogs	1:00 Jan 1	8	e.com
0001	blue book	1:15 Jan 1	4	f.com
0001	school supply for children	1:15 Jan 1	1	g.com
0001	blue fingers	2:05 Jan 1	7	h.com
…	…	…	…	…

We treat a vocabulary as a bag rather than a set since the number of occurrences of a query-term is very important. For example, consider two users who both issued 100 queries. For the first user, 99 queries included *apple* and 1 query included *orange*, while for the second user the reverse is true, that is, 1 query included *apple* and 99 queries included *orange*. The two users have quite different interests on *apple* and *orange*. Such information can only be documented by bags. A bag is a generalization of a set. While a term in a set has only one occurrence, a term in a bag can have multiple occurrences.

While a single query contains little information to reveal the identity of a user, the vocabulary can breach the privacy of the user. For example, the user with ID 4417749 in the released AOL query log was identified by a newspaper journalist who has some background knowledge about certain combination of query-terms that uniquely links to

the user [15]. Therefore, we extend the *k*-anonymity principle [85] to our scenario, that is, to ensure every vocabulary for a given granularity, i.e., query-terms together with the occurrence counts, indistinguishable from at least *k*-1 other vocabularies. Let us call such a requirement the *vocabulary k-anonymity* principle. Such a principle aims at thwarting linking attacks from an adversary who has knowledge about how often his/her target individual using any combination of certain query-terms.

Conceptually, the foregoing problem is different from those in prior works. The work [116] is the most similar one, which however considers a different privacy notion, i.e., it only prevents privacy attacks linking a user to an individual term while we thwarts linking attacks based on any combination of terms. [2][57][78][46] consider publication scenarios different from ours. The work in this chapter is seemingly similar to the work on text document sanitization [25][53][86]. However, while we consider making multiple bags of terms indistinguishable of each other, the latter considers sanitizing a set or a bag of terms, i.e., a single document, and focuses on detecting sensitive named identities and removing each named entity from the document by distortion, suppression, or generalization individually.

Technically, because of the key feature of query logs, i.e., the extreme sparsity, the conventional techniques, such as the generalization techniques [48][95] and the total suppression technique [108], not working well in achieving our vocabulary *k*-anonymity requirement in the sense of retaining data utility as explained as follows.

Example 5.1 (A motivating example): As an anecdotal ananlysis, consider publishing vocabularies extracted from the AOL query log [76] as shown in Table 5.1 with the granularity at the user level. For instance, for the user with original queries

shown in part in Table 5.1, the original vocabulary of this user consists of 181 distinct query-terms as shown in part in the first row in Table 5.2. We applied the multi-dimensional generalization algorithm LG [48] on the extracted vocabularies to enforce the vocabulary k-anonymity with $k = 5$. Unfortunately, the resulting content of the anonymized vocabularies by LG [48] as shown in the second row in Table 5.2 is of little use as the anonymized vocabulary of the same user shrinks into 21 general terms, such as activity, artifact, attribute, etc., which are too general to convey useful information. ■

Table 5.2 One user's vocabulary extracted from AOL query log

Original vocabulary	**care:1, package:1, movie:2, dog:3, blue:2, book:1, school:1, child:1, supply:1, finger:1, rescue:1, …**
Generalized one by LG[48]	**activity, artifact, attribute, organism, social-event, …**
Our approach	**care:1, package:1, movie:2, dog:3, book:1, school:1, child:1, …**

The existing works on generalization techniques [48][95] cannot retain enough data utility in anonymizing such data since two issues have yet to be addressed properly.

The first issue is that existing works treat bags as sets and underestimate the information loss in such a treatment [48][95]. Vocabularies are bags, if treated as sets, they shrink drastically with generalization [48][95]. For example, when an *apple* and an *orange* in a set are both generalized to *fruit*, only one occurrence of *fruit* is kept in the generalized set by [48][95]. However, the information loss is computed in such a way as if the two occurrences of *fruit* were all kept by [48][95]. Therefore, while the information loss computed in such a way is seemingly not high, e.g., 12% in LM [52] in the foregoing

anecdotal example, the actual content of the anynymized data is of little use. We will have more discussion in Section 5.5.1.

The second issue is that generalization techniques in general are vulnerable to outliers in that data needs to be generalized to very high levels to hide outliers. Unfortunately, vocabularies extracted from query logs are extremely sparse and are full of outliers. Since people hardly use the same query-terms even for the same concepts, most of query-terms are infrequent and hence are outliers. In this chapter, we will address these two issues.

The similar issues are also observed with another common approach, i.e., suppression. As the authors of [108] pointed out that suppression techniques are not good at handling sparse data. In the anecdotal analysis, we also applied the total suppression approach [108] on the vocabularies extracted from the AOL query log. The result is that over 90% query-terms have to be suppressed. Suppression is also employed in [25], which sanitizes a single document by suppressing certain terms to ensure the resulting document cannot be linked to less than k records in the given context database. The goodness of the solution depends on the setting of the context database. Our problem is different from [25], and we have no such an external context database.

To address the issues with the conventional techniques, this chapter propose a new approach, i.e., semantic similarity based clustering, which is different from the clustering approaches in [46][116]. The third row in Table 5.2 shows the result by the proposed approach, which is more useful than that by LG [48] in the context of web related applications. The contributions of this chapter can be summarized as follows.

First, this chapter proposes the vocabulary *k*-anonymity principle tailored for publishing vocabularies extracted from web query logs. The subtlety is that vocabularies are a collection of bags rather than a collection of sets. Such a notion is tailored in the context of web query logs, and we allow a different granularity in merging queries into vocabularies for a different application. To address the first issue with generalization techniques in anonymizing such data, we present a bag-valued version of the general loss metric [52], and show how to extend the generalization technique [48] for bag-valued data and how to gauge the information loss properly in such a context as detailed in Section 5.5.1.

Second, this chapter proposes the semantic similarity based clustering approach for addressing the second issue with the generalization techniques, i.e., to retain more data utility in enforcing the vocabulary *k*-anonymity. The intuition behind our approach is that although people hardly use the same query-terms even for the same concepts, those terms should be semantically similar to each other. Our approach is non-trivial since how to measure the semantic similarities between query-terms and that between vocabularies is an open problem. We measure the semantic similarity between two vocabularies by a bipartite matching with the minimum semantic distance. The basis of the matching is the measurement of the semantic distance between any two query-terms. In many cases, the semantic relationships between query-terms comprise a network. Our approach works with any semantic distance measurement, e.g., the shortest distances between vertices (query-terms) in a network. As a tree is just a special network, our approach also works with a taxonomy tree. However, we do not require a taxonomy tree, and in that sense, our approach is *taxonomy free*.

In short, the vocabulary *k*-anonymity principle is different from the *k*-anonymity principles with set-valued data [48][95][108][41] and relational data [85], and this work also differs from query log anonymization [2][57][78][46] and text document sanitization [25][53][86]. The semantic similarity based clustering approach retains more data utility than the state of the art approach as shown by extensive experimental evaluations from multiple perspectives including information loss metric and frequent pattern mining.

The rest of this chapter is organized as follows. Section 5.1 defines the privacy principle and describes our anonymization model, Section 5.2 discusses the measurement of semantic similarity, Section 5.3 analyzes the hardness of our problem, Section 5.4 presents our greedy algorithm, Section 5.5 discusses data utility metrics and revisits the generalization techniques, Section 5.6 evaluates the our approach, Section 5.7 considers the variations and extension of the proposed approach, and Section 5.8 summarizes this chapter.

5.1 Privacy Notion and Anonymization Model

5.1.1 Vocabulary *k*-Anonymity

A search service provider wants to release the collection T of vocabularies extracted from a query log Q. As the number of occurrences of a query-term is important information, the provider treats the vocabularies as bags instead of sets. The provider considers a spectrum of granularities in merging queries into vocabularies since there is no one-size-fits-all setting of the granularity, i.e., a different application may require a different granularity level. E.g., web search personalization may require a granularity at the user level, while query suggestion probably needs a granularity at the session level.

One way to specify the granularity, i.e., to delimit the logical vocabularies of users, is by the gap between the times that queries were posted, which is an approach also documented in web mining literature. Given a query log Q in the format as shown in Figure 5.1(a) where for q in Q, $q.AnonID$ is the anonymous ID of the user issuing q, $q.Query$ is the set of query-terms in q, and $q.QueryTime$ is the time that q was posted, vocabularies can be extracted based on an application dependent logical session gap, *sGap*, as follows.

AnonID	Query	QueryTime	ItemRank	ClickURL
01	Wine	1:00Jan1	3	a.c
01	Wine	1:00Jan1	8	b.c
01	Jackets,Boots	1:15Jan1	3	c.c
01	Jackets,Boots	1:15Jan1	5	d.c
01	Wine	2:05Jan1	4	x.c
02	…	…	…	…

(a) *Q*: original web query log

#	Vocabulary of each user
t_1	Wine, Wine, Jackets, Boots
t_2	Vino, Vino, Jackets, Shoes
t_3	Wine, Vino, Raw-milk, Jackets, Shoes
t_4	Vino, Raw-milk, Raw-milk, Homo-milk
t_5	Raw-milk,Homo-milk,Homo-milk,Jackets,Pants

(b) *T*: vocabularies extracted from *Q*

Figure 5.1 Vocabularies extracted from a query log

Definition 5.1 (***Collection T of vocabularies***)**:** Given a logical session gap *sGap* and $Q = \{q_1, q_2, \ldots q_n\}$ with queries in ascending order of *AnonID* and *QueryTime*, queries in Q are distributed into groups G_j. Suppose that a query q_i is distributed to G_j. The next query q_{i+1} is distributed to G_{j+1} if $q_i.AnonID < q_{i+1}.AnonID$ or $q_i.QueryTime +$

$sGap < q_{i+1}.QueryTime$, otherwise q_{i+1} is also distributed to G_j. The bag containing all the terms in $q.Query$ for $q \in G_j$ is the jth vocabulary in $T = Vocabulary(Q, sGap)$. ■

One additional option in extracting vocabularies is to leave out the queries in which the search users are not interested. That is, for q in Q if $q.ItemRank$ and $q.ClickURL$ are empty, i.e., no search result of q is clicked, we can leave out q.

Example 5.2: Figure 5.1(a) shows 3 queries, the first query about {Wine} has two clicked search results, i.e., the first 2 lines with *ClickURL* "a.c" and "b.c", and so on. Figure 5.1(b) shows the vocabularies extracted with $sGap=\infty$ (one vocabulary per user). ■

The search service provider wants to protect the identities of search service users by observing the following privacy principle.

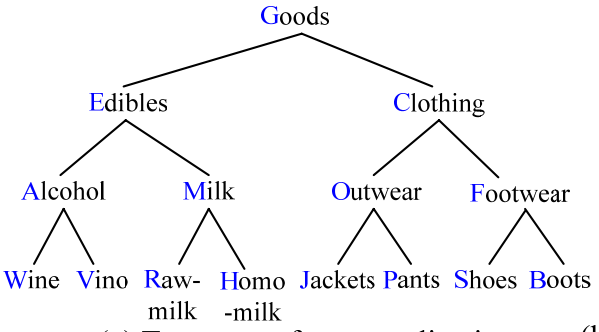
Definition 5.2 (*Vocabulary k-anonymity*): Given a query log Q and a logical session gap $sGap$, the vocabulary k-anonymity principle is observed if for any vocabulary $t \in T = Vocabulary(Q, sGap)$, there are at least k-1 other vocabularies in T that are indistinguishable from t. ■

The assumption is that an adversary has knowledge about how often his/her target individual using any combination of certain query-terms. This privacy principle aims at preventing such an adversary from linking any vocabulary to his/her target individual with a probability greater than 1 / k.

5.1.2 Anonymizing Vocabularies by Semantic Similarity Based Clustering

This chapter proposes the semantic similarity based clustering to achieve the vocabulary k-anonymity as follows. $T = Vocabulary(Q, sGap)$ is partitioned into a set of clusters by the semantic similarities such that each cluster contains at least k vocabularies.

The centre of each cluster is selected as the typical vocabulary to represent all the vocabularies in the cluster. In clustering vocabularies and selecting the typical vocabularies, we allow setting a semantic distance threshold ε to control the similarity degree that is required for vocabularies to comprise a cluster and for one typical query-term to represent others.



(a) Taxonomy for generalization

#	Vocabulary	Our approach
t_1	W, W, J, B	W, V, J, S
t_2	V, V, J, S	W, V, J, S
t_3	W, V, R, J, S	W, V, J, S
t_4	V, R, R, H	R, R, H
t_5	R, H, H, J, P	R, R, H

(b) original T and anonymized T^*

Figure 5.2 Anonymizing vocabularies by semantic similarity based clustering

Example 5.3 (A running example): Consider enforcing vocabulary 2-anonymity on the collection T of vocabularies in Figure 5.1(b). To comparing with generalization techniques, the taxonomy tree in Figure 5.2(a) is always used to measure the semantic relationship between query-terms. We partition the five vocabularies into two clusters. The first cluster consists of the first three vocabularies, and the second cluster is made of the last two vocabularies. The cluster centre, i.e., the typical vocabulary, of each cluster is elected on a *majority vote* basis, that is {Wine:1, Vino:1, Jackets:1, Shoes:1} for the first cluster, and {Raw-Milk:2, Homo-Milk:1} for the second cluster. The last column in Figure 5.2(b) shows the anonymized vocabularies by our approach. ■

The proposed approach can retain more data utility than conventional approaches for the following reasons.

First, instead of generalizing query-terms, we substitute query-terms by semantically similar, *original* query-terms. One may argue that generalizing Wine and Vino into Alcohol can keep more *truth* than distorting Wine into Vino. However, because of the extreme sparsity, Wine and Vino will be generalized into Edibles or even Goods, instead of Alcohol, which conveys little information. More discussion is presented in Section 5.5.1.

Second, the query-terms that do not match with the others in a cluster will be left out from the cluster, i.e., local suppression is an inherent component of our approach. As an alternative, we can also determine the typical vocabulary of a cluster by locally generalizing semantically matched terms in the cluster. With this alternative, our approach becomes the integration of local suppression and local generalization, which implies that our approach takes advantage of both the generalization and suppression techniques.

5.2 Measure Semantic Similarity

Basically, any clustering algorithm can be used to cluster vocabularies. The key technique is how to measure the semantic similarities between vocabularies, which is the focus of this section. We will present a greedy clustering algorithm in Section 5.4.

5.2.1 Semantic Distance between Two Terms

Our ultimate goal is to minimize the distortion in anonymizing vocabularies. Subsequently, we want to maximize the semantic similarities between vocabularies in a cluster based on the similarities between query-terms. The semantic similarity between two query-terms can be depicted by the semantic distance, which is given externally, e.g.,

derived from a semantic network. The smaller the semantic distance is, the more similar the two query-terms are. One extreme case is that the distance is 0, meaning that two terms are identical. If the distance is quite large, the two should be deemed as irrelevant.

Definition 5.3 (*Semantic distance between two terms*): For two query-terms *i* and *j*, the semantic distance between *i* and *j*, denoted by *distance*(*i*, *j*), is computed by a metric. Given a user-defined semantic distance threshold ε, *i* and *j* are semantically *relevant* if *distance*(*i*, *j*) ≤ ε, otherwise *i* and *j* are semantically *irrelevant*. ■

The followings are two sources of information that can be used to derive the metric for measuring the semantic distance between two query-terms.

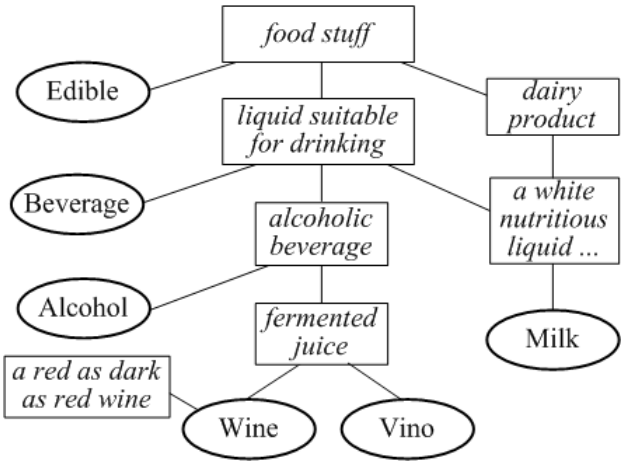


Figure 5.3 Network of words (e.g., WordNet [35])

Source 1: Online lexical dictionaries. For example, WordNet [35] is such a dictionary with which nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. The specific meaning of one word under one type of POS (Parts of Speech, either noun, verb, adjective, or adverb) is called a sense. Each sense of a word is in a different synset, which contains the set of words with synonymous meanings. Synsets are connected to one another through

explicit semantic relations. The resulting network of meaningfully related words and synsets can be used to measure the semantic distance between words [22]. There can be multiple paths between any two words. One way is to use the shortest distance between the two words as their semantic distance. Figure 5.3 shows a simple network consisting of a few words (depicted by ellipses) and their synsets as nouns (depicted by boxes).

Source 2: A given taxonomy tree. For some cases, a taxonomy tree for generalization is available. We can derive semantic distances directly from such a simplified network.

Example 5.4: The taxonomy tree in Figure 5.2(a) is a simplification of the network in Figure 5.3. In our running examples, we always assume that the tree in Figure 5.2(a) is used to derive the semantic distances. Therefore, *distance*(Wine, Vino) = 2, *distance*(Wine, Jackets) = 6, and so on. If the semantic distance threshold ε is 2, Wine and Vino are semantically relevant, and Wine and Jackets are irrelevant. ■

5.2.2 Distance between Two Vocabularies

It is an open problem to measure the semantic similarity / distance between two vocabularies as vocabularies have no fixed structure.

There is a whole range of similarity / distance metrics in the literature some of which could be a possible solution. For example, we could employ the Jaccard coefficient if we treated vocabularies as sets, we could use the Hamming distance if we handled vocabularies as strings, or we could apply the Cosine similarity if we normalized vocabularies into vectors, and so on. However, those metrics treat each query-term independently while we want to exploit the semantic relationships between query-terms.

Therefore, we propose to employ a weighted bipartite matching model that takes into account the semantic relationships between query-terms. The idea is to find such a matching between the two vocabularies that the sum of semantic distances between matched query-terms is minimized.

Definition 5.4 (*Semantic distance between two vocabularies*): For any two vocabularies $t_1, t_2 \in T$, the semantic distance between t_1 and t_2, denoted by *distance*(t_1, t_2), is the minimum weight sum of the weighted bipartite matching between t_1 and t_2. ■

To be flexible in matching two vocabularies, we allow two occurrences of a same term in the vocabulary t_1 match to two different terms in the other vocabulary t_2. For example, if t_1 contains 100 *apples* and t_2 contains 99 *apples* and 1 *pear*, we can match 99 *apples* in t_1 with 99 *apples* in t_2, and either match the remaining 1 *apple* in t_1 with 1 *pear* in t_2 or simply suppress 1 *apple* from t_1 and 1 *pear* from t_2.

Therefore, the duplicate occurrences of a term in a vocabulary t are handled as different terms hereafter. For the convenience of discussion, $i \in t$ merely denotes an occurrence of the term i in the vocabulary t, and $|t|$ denotes the sum of all occurrences.

There are two slightly different bipartite models. The first model is the minimum weight *maximum* matching model with the bipartite constructed as follows: t_1 and t_2 make the two parts; for a term i in t_1 and a term j in t_2, if i and j are semantically relevant, there is an edge connecting i and j with a weight equal to *distance*(i, j).

Example 5.5: The left half of Figure 5.4 shows the bipartite made of the first two vocabularies in Figure 5.1(b). Suppose the user-given semantic distance threshold $\varepsilon = 4$. Since *distance*(Wine, Vino) = $2 \leq \varepsilon$, there is an edge connecting Wine and Vino with a

weight 2. As *distance*(Wine, Jackets) = 6 > ε, there is no edge connecting Wine and Jackets, and so on. Clearly, the maximum matching is {{Wine, Vino}, {Wine, Vino}, {Jackets, Jackets}, {Boots, Shoes}} with the minimum weight sum being 6. Thus, the semantic distance between the two vocabularies is 6. ■

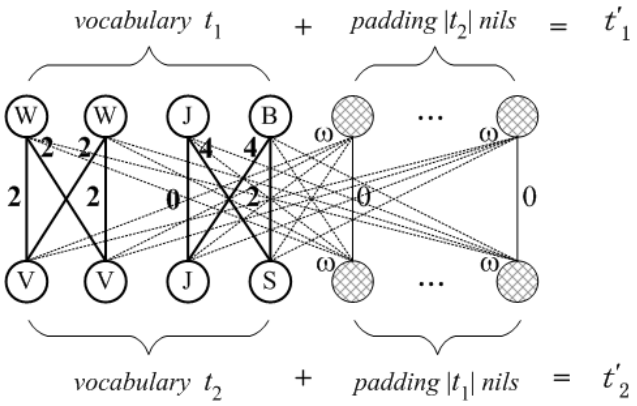


Figure 5.4 *distance*(t_1, t_2) by a weighted bipartite matching

In general, not all terms in the two vocabularies will be matched. The unmatched terms contribute to the *dissimilarity* between the two vocabularies. It is reasonable to charge each unmatched term with a uniform cost ω. We can think of ω as the distance between an unmatched term and a special *nil* term. Clearly, ω should be no less than ε.

The second model is the minimum weight *perfect* matching model with an *extended* bipartite derived by padding some special *nil* terms into the basic bipartite. As shown in Figure 5.4, the first part t_1' is t_1 padded with $|t_2|$ special *nil* terms, and the second part t_2' is t_2 padded with $|t_1|$ special *nil* terms. A special *nil* term in t_1' is connected to every term in t_2 with a weight ω, and to a special *nil* term in t_2' with a weight 0. So does a special *nil* term in t_2'. Clearly, any matching between t_1 and t_2 in the first model corresponds to a perfect matching between t_1' and t_2' in the second model with the same weight sum. While the first model tends to match as many query-terms as possible, the

second model tends to get the weight sum as small as possible. When $\omega >> \varepsilon$, the values of *distance*(t_1, t_2) computed by the two models are the same.

5.3 Hardness of the Problem

5.3.1 The Formal Description of the Problem

Our semantic similarity based clustering problem is formalized as follows. Given $T = Vocabulary(Q, sGap)$, i.e., the collection of vocabularies of a query log Q with a logical session gap *sGap*, partition T into a set CL of clusters such that each cluster in CL has a size no less than k and the sum of the semantic distances over every vocabulary and its cluster centre is minimized.

$$\min_{CL} \sum_{C \in CL} \sum_{t \in C} distance(t, centre(C))$$

$$s.t. \begin{cases} \bigcup_{C \in CL} C = T \\ |C| \geq k \ \ for\ C \in CL \\ C_1 \cap C_2 = \varnothing \ \ for\ C_1, C_2 \in CL \end{cases}$$

The centre of a cluster C, *centre*(C), is published as the typical vocabulary of the cluster. And *centre*(C) should be elected by a majority vote. Ideally, such an election should be based on a weighted m-partite matching among all the vocabularies in C given the size of C is m.

Example 5.6: Figure 5.5 shows the 3-partitie matching result for the first three vocabularies in Figure 5.1(b), from which we can select a typical term for each bag of matched terms to comprise the centre of the cluster of the three vocabularies, which is {Wine, Vino, Jackets, Shoes}. ■

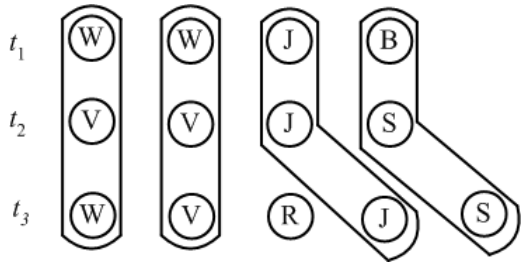


Figure 5.5 *m*-partite matching for electing the cluster centre

5.3.2 The Hardness of the Problem

According to the foregoing description, our problem is a constrained clustering problem. In general, a constrained clustering problem is NP-hard [4] even if the atomic operation, i.e., computing the semantic distance between a vocabulary and its cluster centre in our case, was in polynomial time.

Unfortunately, our problem is even harder: the atomic operation is based on the computation of the centre of each cluster, as discussed in Section 5.3.1, which is a weighted *m*-partite matching problem and is NP-hard for $m \geq k \geq 3$.

5.4 A Greedy Algorithm

Our problem is NP-hard as discussed in Section 5.3. Therefore, we propose a greedy algorithm to solve the problem. We present the overall picture in Section 5.4.1, and then describe the core component of the algorithm in Section 5.4.2.

5.4.1 Algorithm SSG

We propose a greedy algorithm in producing clusters of vocabularies with a size no less than *k*. Our algorithm creates one cluster at a time and tries to make the sum of semantic distances between vocabularies in the cluster as small as possible.

Roughly speaking, it is relatively easy for a small size vocabulary to match another vocabulary with a small semantic distance. In other words, a cluster made of small size vocabularies is more likely to have a smaller sum of distances than a cluster made of large size vocabularies. This suggests that we should start grouping with vocabularies of small sizes by sorting vocabularies in the ascending order of sizes.

Our greedy algorithm, SSG, namely semantic similarity based grouping, is shown in Algorithm 5.1 where T is the set of vocabularies, C is the cluster currently being created, and CL is the collection of final clusters produced by SSG. We will discuss how to compute *centre*(C) in Section 5.4.2.

Algorithm 5.1 The semantic similarity based group algorithm SSG

SSG (*T*, *k*)

Input: the set T of vocabularies, the privacy requirement k
Output: the collection CL of clusters of vocabularies

```
 1:while |T| ≥ k do
 2:    a new cluster C ← the first t in T
 3:    for l = 2 to k do
 4:        find r in T nearest to centre(C)
 5:        move r from T to C
 6:        update centre(C) with r
 7:    add C into CL
 8:for each remaining t in T do
 9:    find C in CL with centre(C) nearest to t
10:      move t from T to C and update centre(C)
11:    return CL
```

SSG works in 2 phases. In the first phase (line 1 to 7), SSG creates one cluster at a time when there are at least k vocabularies in T. SSG picks the first vocabulary t in T to initialize a new cluster C by moving t from T to C (line 2). SSG adds k-1 nearest vocabularies into C, and places C in CL (line 3 to 7). In the second phase (line 8 to 11), SSG assigns each remaining vocabulary t in T to an existing cluster nearest to t.

Example 5.7: For the running example ($k = 2$), the first phase, SSG creates the first cluster by initializing it with the first vocabulary, {Wine, Wine, Jackets, Boots}, in Figure 5.1(b). Then, SSG finds the nearest vocabulary, i.e., the second in Figure 5.1(b), {Vino, Vino, Jackets, Shoes}. SSG continues to build the second cluster, which groups the last two vocabularies in Figure 5.1(b) together. At the end of the first phase, the vocabulary remaining in T is the third, {Wine, Vino, Raw-milk, Jackets, Shoes}, which is assigned to the first cluster in the second phase. ■

5.4.2 Computing the Centre of a Cluster

Computing *centre*(C) for a cluster C of vocabularies, is a core component of our greedy algorithm. As discussed in Section 5.3.1, *centre*(C) is derived in two steps: an m-partite matching for all vocabularies in C, and then electing the centre by a majority vote.

A heuristic solution to the *m*-partite matching incorporated in SSG. The idea is to run a bipartite matching in m iterations, where m is the size of the cluster. In each iteration, we match one vocabulary with the result obtained in the previous iteration. Let gt_l be the result obtained in the l-th iteration. We treat gt_l as a *hyper* vocabulary that is a collection of size-l bags. Each bag contains l matched query-terms from l different vocabularies, which we call a *hyper* query-term.

In the first iteration, we pick one vocabulary from the cluster and make the current hyper vocabulary gt_1. In iteration l+1, we do a bipartite matching between the hyper vocabulary gt_l and a vocabulary t in the cluster that is not picked yet.

A query-term i in t is semantically relevant to a hyper query-term, i.e., a bag in gt_l, if i is semantically relevant to every query-term in the bag. If this is the case, the distance

between i and the bag is the sum of the semantic distance between i and every term in the bag. If a query-term in t is matched with a hyper query-term in gt_l, the former will be put into the latter to make a new hyper query-term in gt_{l+1}. The bags in gt_l and the query-terms in t that are not matched will not be represented by the typical vocabulary of the cluster, i.e., will be suppressed.

Example 5.8: The 3-partite matching in Figure 5.5 can be solved in 3 steps. First, we select t_1 to make gt_1 = {{Wine}, {Wine}, {Jackets}, {Boots}}. Second, we match gt_1 with t_2, which results in gt_2 = {{Wine, Vino}, {Wine, Vino}, {Jackets, Jackets}, {Boots, Shoes}}. Third, we match gt_2 with t_3, which yields gt_3 = {{Wine, Vino, Wine}, {Wine, Vino, Vino}, {Jackets, Jackets, Jackets}, {Boots, Shoes, Shoes}}. ■

Electing the centre of a cluster. Given the hyper vocabulary of a cluster derived by the foregoing matching solution, we have much flexibility in electing the centre of the cluster (i.e., the typical vocabulary). The followings are a few methods.

The first method is to compute on a majority vote basis. We first select the typical term for each bag of matched query-terms, i.e., select the term that are most close to other terms in the bag in the sense of weighted semantic distances. We then piece together all the typical terms to get the centre of the cluster.

Example 5.9: The hyper vocabulary of the first cluster is gt_3 = {{Wine, Vino, Wine}, {Wine, Vino, Vino}, {Jackets, Jackets, Jackets}, {Boots, Shoes, Shoes}}. By the first method, Wine is selected as the typical term of the first bag in gt_3, Vino the second bag, Jackets the third bag, and Shoes the fourth bag. Thus, {Wine, Vino, Jackets, Shoes} is elected as the typical vocabulary. ■

The second method is to directly use the hyper vocabulary, i.e., the matching result, as the centre. The third method is to locally generalize each hyper query-term in a cluster, which is the integration of local generalization with local suppression, supposing the taxonomy for generalization is available.

5.4.3 Complexity and Implementation

We briefly analyze the time complexity as follows. First, SSG sorts vocabularies in T once, which has a time complexity of $O(|T|\cdot\log(|T|))$. Second, SSG creates $n = |T| / k$ clusters. For the ith cluster C_i, SSG selects k vocabularies, one at a time. Suppose that each selection checks h_i vocabularies to find the one nearest to C_i, and the complexity of each check is M that is the time for a bipartite matching to compute the distance. Then, the time complexity for creating clusters is $\sum_{i=1}^{n} k \cdot M \cdot h_i$.

In the worst case, $h_i = (n\text{-}i)\cdot k$, i.e., we need to check all vocabularies remaining in T for selecting each vocabulary in C_i. Thus, the worse case complexity for creating clusters is $O(M\cdot|T|^2)$ with M being polynomial in the average vocabulary size, and so is the overall complexity, which is unfavorable. Therefore, we reduce the complexity by the following greedy implementation.

First, we implement an approximation for the weighted matching problem [88]. Our implementation is of linear time in s, which simply keeps on selecting an edge connecting unmatched vertices (query-terms) with the minimal weight. Second, we choose $\omega >> \varepsilon$. Remember that ω is the distance between an unmatched query-term and a special *nil* term, a.k.a., the unit cost for penalizing an unmatched query-term. How to select a concrete value for ω is subjective. Choosing $\omega >> \varepsilon$ means that we do not favor

suppression, i.e., we try to keep as many query-terms as possible. Such a simplification lessens the burden of selecting ω. This greedy approach makes M linear in s, and limits the search to a small portion of T in finding the vocabulary nearest to a cluster (line 4 in Algorithm 1). Such a limited search is of the same order as a binary search, i.e., we have $h_i < h \sim \log(|T|)$.

With this greedy approach, the overall complexity is reduced to $O(s \cdot |T| \cdot \log(|T|))$ where s is the average vocabulary size, and $|T|$ is the number of vocabularies.

5.5 Data Utility Metrics

The data utility of our approach is evaluated by comparing with the multiple dimensional generalization LG [48]. As LG [48] employs the set model, to do a fair comparison, this section extends LG [48] to the bag model, and denote such a bag-valued variant of LG [48] as LG(bag).

This section adapts the loss metric [52] originally proposed for relational data to bag-valued data in order to measure the information loss in the vocabularies anonymized by LG(bag) and our approach. Two indicators are also presented for measuring utility of anonymized data in frequent pattern mining.

5.5.1 Information Loss Metric and Bag-valued Variant of LG

5.5.1.1 A Bag-valued Variant of LG [48]

Simply put, LG [48] can be extended to its bag-valued variant, LG(bag), by keeping duplicate occurrences of a (generalized) query-term in a vocabulary to the extent that is consistent with the original vocabulary and does not destroy the indistinguishable group.

Example 5.10: With the the original vocabularies T in the second column in Figure 5.6 (copying from Figure 5.1(b)) and the taxonomy tree in Figure 5.2(a), all the duplicate occurrences of Goods in the first and fourth vocabularies are kept, 2 out of 3 occurrences of Edibles are kept in the third and fifth vocabularies as the other vocabulary in the same indistinguishable group has only 2 occurrences of Edibles. The vocabularies T^* anonymized by LG(bag) is shown in the fourth column in Figure 5.6 where the first and fourth generalized vocabularies are {Goods:4} and the other three are {Edible:2, Clothing:2}, while T^* anonymized by LG [48] is shown in the third column. ■

An observation is that by LG(bag), some occurrences of generalized query-terms, e.g., a total of 2 occurrences of Edibles in the above, have to be suppressed from publication. If such suppression is ignored in computing the information loss [48], the result is an underestimation as the number of occurrences of a query-term is important information to convey.

#	Vocabulary	LG [47]	LG (bag)	Our approach
t_1	W, W, J, B	G	G, G, G, G	W, V, J, S
t_2	V, V, J, S	E, C	E, E, C, C	W, V, J, S
t_3	W, V, R, J, S	E, C	E, E, C, C	W, V, J, S
t_4	V, R, R, H	G	G, G, G, G	R, R, H
t_5	R, H, H, J, P	E, C	E, E, C, C	R, R, H

Figure 5.6 original T and T^* anonymized by LG [48], LG(bag), and our approach respectively

5.5.1.2 Loss Metric for the Bag-valued Vocabulary

General-purpose metrics, such as LM [52], were originally proposed to quantify information loss in generalizing relational data. In particular, LM [52] measures the information loss of generalizing an original value v on an attribute to a general value v^*

by the ambiguity of v^*, which depends on the number of values represented by v*. Specifically,

$$LM(v^*) = \frac{\#CoveredBy(v^*) - 1}{\#Domain - 1}$$

where $\#CoveredBy(v^*)$ is the number of values represented by v^*, and *#Domain* is the total number of values in the domain. The overall information loss with a dataset is the weighed average over all values.

A proper adaptation of LM [52] for measuring the information loss of the bag-valued vocabularies anonymized by LG(bag) and our approach should be as follows.

First, our approach does not generalize query-terms while LG(bag) does. Thus, the published results of the two approaches cannot be *directly* compared in LM [52]. However, if our approach publishes the bag of matched query-terms, i.e., the hyper query-term, represented by each term in the typical vocabulary of each cluster, and LG(bag) publishes that represented by each generalized term in an indistinguishable group, then the two approaches can be compared in LM [52].

Second, each occurrence of a query-term that is not published in the anonymized vocabulary should be considered as suppressed, whose information loss should be no less than generalizing it in terms of LM [52], which we choose the minimum, i.e., 100%. Let us call such an adaptation the bag-valued version of LM [52], abbreviated as bLM.

Example 5.11: In Figure 5.6, the first cluster in the last column by our approach consists of the first three vocabularies, and Wine in the typical vocabulary represents the hyper query-term, {Wine:2, Vino:1}, whose information loss is computed as 3·bLM({Wine:2,Vino:1}), and the occurrence of Raw-milk in the third vocabulary is not

published by our approach, whose information loss is 100%, and so on. The overall information loss by our approach is 25.3% in bLM.

With the LG(bag) approach, the first and fourth vocabularies form a group, and the general query-term Goods represents the hyper query-term, {Wine:2, Vino:1, Raw-milk:2, Homo-milk:1, Jackets:1, Boots:1}, whose information loss is 8·bLM({Wine:2, Vino:1, Raw-milk:2, Homo-milk:1, Jackets:1, Boots:1}). Both in the third and fifth vocabularies, one occurrence of Edibles is not published, each of which should be charged with 100%, and so forth. The information loss by LG(bag) is 54.5% in bLM. ■

5.5.2 Data Utility in Frequent Pattern Mining

Frequent pattern mining is a core component of a wide range of applications, e.g., keyword recommendation and query suggestion. The data utility in frequent pattern mining is measured by *Recall* and *sDist* (semantic distance) of the top-*n* frequent closed patterns [45] in the original data with regard to those in the anonymized data. Let *OFCPs* and *AFCPs* be the sets of the top-*n* frequent closed patterns of query-terms in the original data and in the anonymized data respectively. We define

$$Recall = \frac{\sum_{O \in OFCPs} \max_{A \in AFCPs} |O \cap A|}{\sum_{O \in OFCPs} |O|}.$$

For a frequent pattern O in *OFCPs*, $\max |O \cap A|$ is its largest subset that is a frequent pattern in the anonymized data. *Recall* reflects the degree to which the frequent patterns in the original data are retained in the anonymized data. Similarly, we define

$$sDist = \frac{\sum_{O \in OFCPs} \min_{A \in AFCPs} distance(O, A)}{\sum_{O \in OFCPs} |O|}$$

For a frequent pattern O in *OFCPs*, min *distance*(O, A) is the shortest semantic distance of O from *AFCPs* with $\varepsilon = \infty$ (so, the two bipartite models in Section 5.2.2 yield the same distance). The *sDist* is an indicator of precision in that it reflects the semantic closeness between the patterns in the original data and those in the anonymized data.

Example 5.12: To explain how to compute *Recall* and *sDist* by a simplified example, consider the top-1 frequent pattern, i.e., *OFCPs* only contains the most frequent pattern O_1 in T, and *AFCPs* contains the most frequent pattern A_1 in T^*. Suppose that O_1 = {Wine, Jackets} and A_1 = {Wine, Outwear}. Then,

$$Recall = | O_1 \cap A_1 | / | O_1 | = | \{\text{Wine}\}| / | O_1 | = 50\%.$$

$$sDist = distance(O_1, A_1) / | O_1 | = (0 + 1) / 2 = 0.5. \quad \blacksquare$$

We consider *Recall* with regard to the original frequent patterns. We do not extend the concept to *generalized* patterns as otherwise *Recall* is always 100% even when every query-term is generalized to the most general term, which is not meaningful. Nevertheless, *sDist* captures the utility of generalized patterns in terms of their semantic distances from the original patterns.

5.6 Experimental Evaluation

We evaluated our algorithm SSG by comparing with the bag-valued variant, LG(bag), of the multi-dimensional generalization algorithm LG [48] in anonymizing vocabularies extracted from the AOL query log [76]. Other techniques [41][95][108] cannot be adapted to achieve our privacy notion, so are not comparable. The experiments were performed on a PC with 3GHz CPU and 3G RAM running Windows 7.

The AOL query log [76] consists of web queries collected over three months. Each query is of the format {AnonID, Query, QueryTime, ItemRank, ClickURL}. Queries are sorted by anonymous user ID (AnonID) and are sequentially arranged. The AOL query log is released in 10 subsets, each of which has similar characteristics and size. We will use the first subset to evaluate the two algorithms. LG requires a taxonomy tree for generalization that is created by using WordNet [35].

In the pre-processing, queries are grouped into different logical sessions based on the gap among QueryTime, *sGap*. All the queries with non-empty ClickURL of each user at one session comprise an independent vocabulary. We interpreted each noun by its most frequently used sense, and left out query-terms that are not in WordNet or not nouns since they do not form a tree while LG [48] requires such a taxonomy tree. Therefore, nouns together with the *is-a-kind-of* links among them comprise a tree.

On the taxonomy tree, there are 19,029 leaves (i.e., distinct query-terms in the preprocessed dataset), 7,144 internals, and 19 levels with the average root-leaf path length being 10. Figure 5.7(a) and (b) show the number of vocabularies (#vocabularies) and the average vocabulary size (avgVsize) respectively with the session gap (*sGap*) varying from 1, 5, 10, 20, 30 minutes, 1 and 8 hour, through 1 day and 1 week and 1 month, to all the time. We select 1 hour as the default session gap, i.e., *sGap* = 1 hour, where #vocabularies = 383K, and *avgVsize* = 3.36.

Figure 5.7(c) shows the average generalization height (gHeight) by LG(bag), the bag-valued variant of LG [48] with varying *sGap*. We computed the semantic distance between two query-terms by the shortest distance among them on the same taxonomy tree although SSG is not restricted to a taxonomy tree.

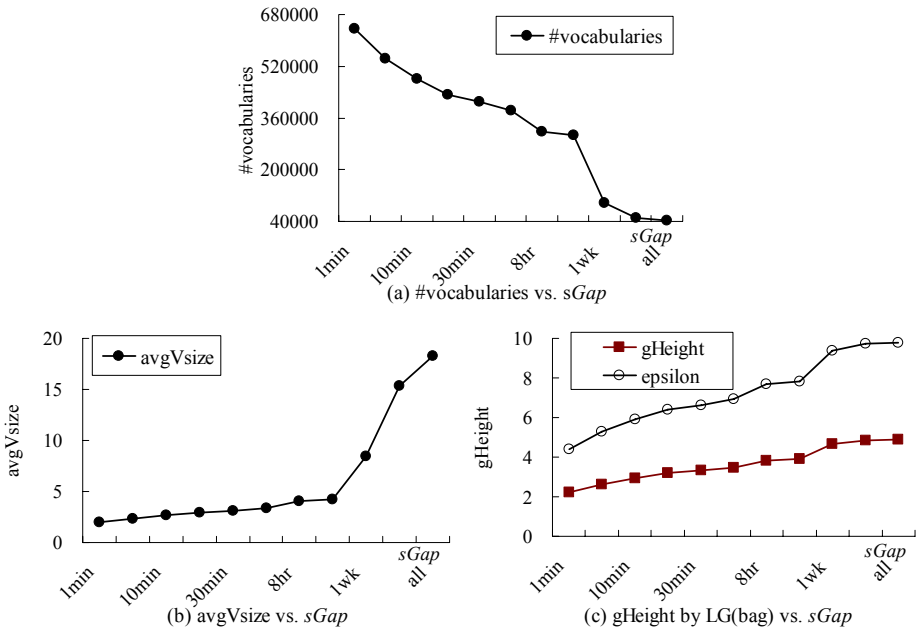


Figure 5.7 Data characteristics: number of vocabularies, average vocabulary size, average generalization height by LG(bag)

For a fair comparison, the semantic distance threshold ε should be set to 2·gHeight as in average it is the maximum distance between the two query-terms that are generalized to a same general term by LG(bag). Figure 5.7(c) also shows such a recommended ε with a median over 6. Therefore, we set the default ε to 6. In the experiments, ε varies from 0 to 10. The settings of other parameters are as follows: For the vocabulary k-anonymity requirement, k varies from 2 to 30 with 5 being the default. For the number of top-n frequent closed patterns, *#patterns* varies from 1000 to 5000 with 1000 being the default.

5.6.1 The Basic Features of Anonymized Vocabularies

We measured the data utility from multiple perspectives. The first basic feature is %DistinctTerms, the percentage of distinct query-terms retained in the anonymized

vocabularies, which is the ratio of the number of distinct (generalized) query-terms in the anonymized vocabularies to that in the original vocabularies.

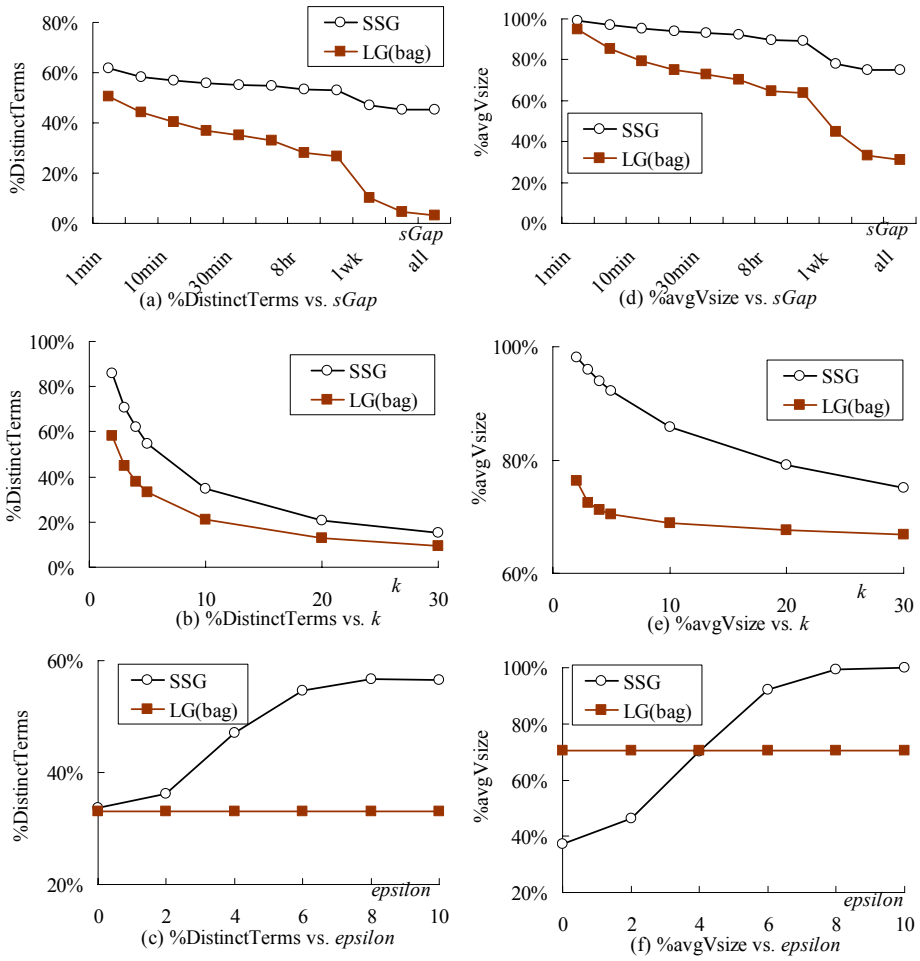


Figure 5.8 Relative number of distinct terms and relative average size of anonymized vocabularies with varying *sGap*, *k*, ε

Figure 5.8(a) shows %DistinctTerms with *sGap* varying from 1 minute through 1 hour and 1 day to all the time. The %DistinctTerms by SSG ranges from 62% through 53% to 46% while that by LG(bag) ranges from 49% through 26% to 3%. Figure 5.8(b) shows the %DistinctTerms with *k* varying from 2 to 30. The %DistinctTerms by SSG ranges from 86% to 15% while that by LG(bag) ranges from 58% to 9%. Figure 5.8(c) shows the %DistinctTerms with ε varying from 0 to 10. The %DistinctTerms by SSG

ranges from 34% to 57% while that by LG(bag) does not change with ε since it does not utilizes the semantic distance threshold ε.

For most cases, SSG retains 2 to 10 times more distinct terms than LG(bag). Notice that when $\varepsilon = 0$, any two distinct query-terms are deemed as semantically irrelevant, and hence our approach reduces to the local suppression approach. Even in this special case, SSG still retains more distinct terms than LG(bag).

The second basic feature is %avgVsize, the relative average vocabulary size in the anonymized query log, which is the ratio of the average vocabulary size in the anonymized query log to that in the original query log.

Figure 5.8(d) shows the %avgVsize with the varying *sGap*. The %avgVsize by SSG ranges from 99% through 89% to 75% while that by LG(bag) ranges from 94% through 63% to 31%. Figure 5.8(e) shows the %avgVsize with the varying *k*. The %avgVsize by SSG ranges from 98% to 75% while that by LG(bag) ranges from 76% to 66%. Figure 5.8(f) shows the %avgVsize with the varying ε. With a flexible setting of ε (≥ 4), SSG keeps more query-terms in the vocabulary of an average user. In short, %avgVsize by SSG is up to 2 times of that by LG(bag).

By comparing the basic features of vocabularies anonymized by SSG and by LG(bag), we can see that SSG retains much more information than LG(bag). An observation is that the information gain by SSG over LG(bag) increases with the increase of the session gap and the vocabulary size.

5.6.2 Information Loss in the Bag-valued LM

We propose the bag-valued adaptation of LM [52], bLM, to quantify information loss in anonymizing vocabularies in Section 5.5.1. The following reports information loss in terms of bLM.

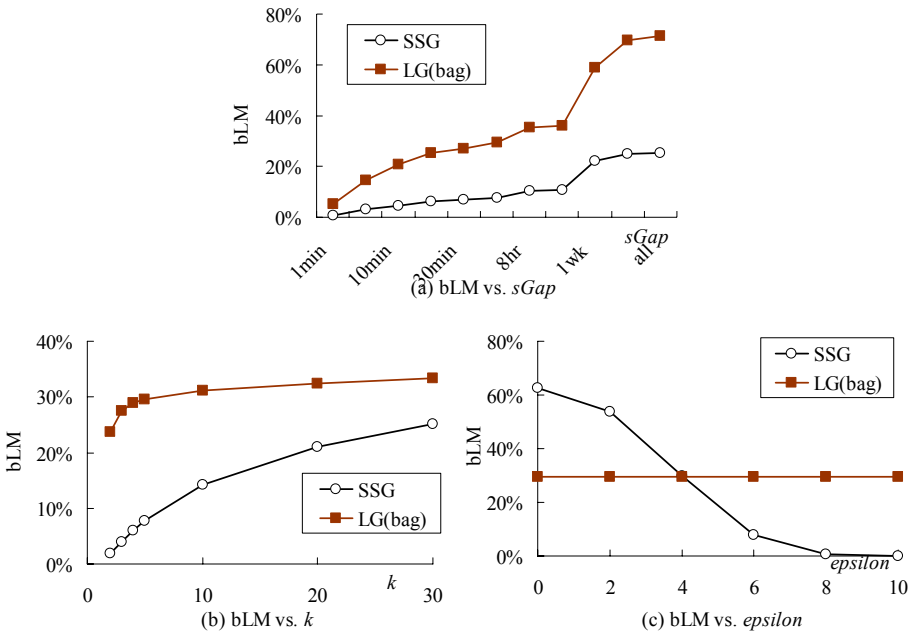


Figure 5.9 Information loss in bLM, the bag-valued version of LM [52], with varying *sGap*, *k*, ε

Figure 5.9(a) shows the information loss with the varying *sGap*. The information loss by LG(bag) ranges from 5.4% through 36% to 71% while that by SSG ranges from 0.7% through 10% to 25%. The former is 3 to 8 times of the latter. Figure 5.9(b) shows the information loss with the varying *k*. The information loss by LG(bag) ranges from 23% to 34% while that by SSG ranges from 2% to 25%. The former is 2 to 10 times of the latter. Figure 5.9(c) shows the information loss with the varying ε. For $\varepsilon \geq 4$, LG(bag) has much more information loss than SSG.

An observation is that with a session gap used in extracting vocabularies no less than 5 minutes, i.e., $sGap \geq 5$ minutes, the information loss by LG(bag) is always greater than 15%. With the increase of *sGap* and the vocabulary size, the information loss by LG(bag) increases much faster than SSG. In short, LG(bag) incurs huge information loss that is why it is inappropriate in anonymizing vocabularies as shown in the motivating example and the running example.

5.6.3 Data Utility in Frequent Pattern Mining

Now, let us evaluate the data utility of the anonymized vocabularies in frequent pattern mining. As most frequent pattern mining algorithms only mine sets and developing an efficient algorithm mining bags is beyond the scope of this work, we ignored the duplicate occurrences of a query-term in each vocabulary in this set of experiments. Nevertheless, we believe that the results under this simplification are good approximations at least with a small session gap (*sGap*), since most of bags are reducing to sets when the granularity of vocabularies is reducing from the user level through the session level down to the query level with the decreasing *sGap*.

We mine the top-*n* frequent closed patterns [45] both in the original vocabularies T extracted from the original query log Q and in the anonymized vocabularies T^*, and collect *Recall* and *sDist* for the top-*n* frequent closed patterns as defined in Section 5.5.2. These two indicators depict the degrees to which the original patterns are retained.

Figure 5.10 (a)-(d) show *Recall* with *sGap* varying from 1 minute through 1 hour and 1day to all the time, with k varying from 2 to 30, with ε varying from 0 to 10, and

with *#patterns* varying from 1000 to 5000 respectively. For most cases, *Recall* by SSG is around 80%, while that by LG(bag) ranges from between 9% and 40%.

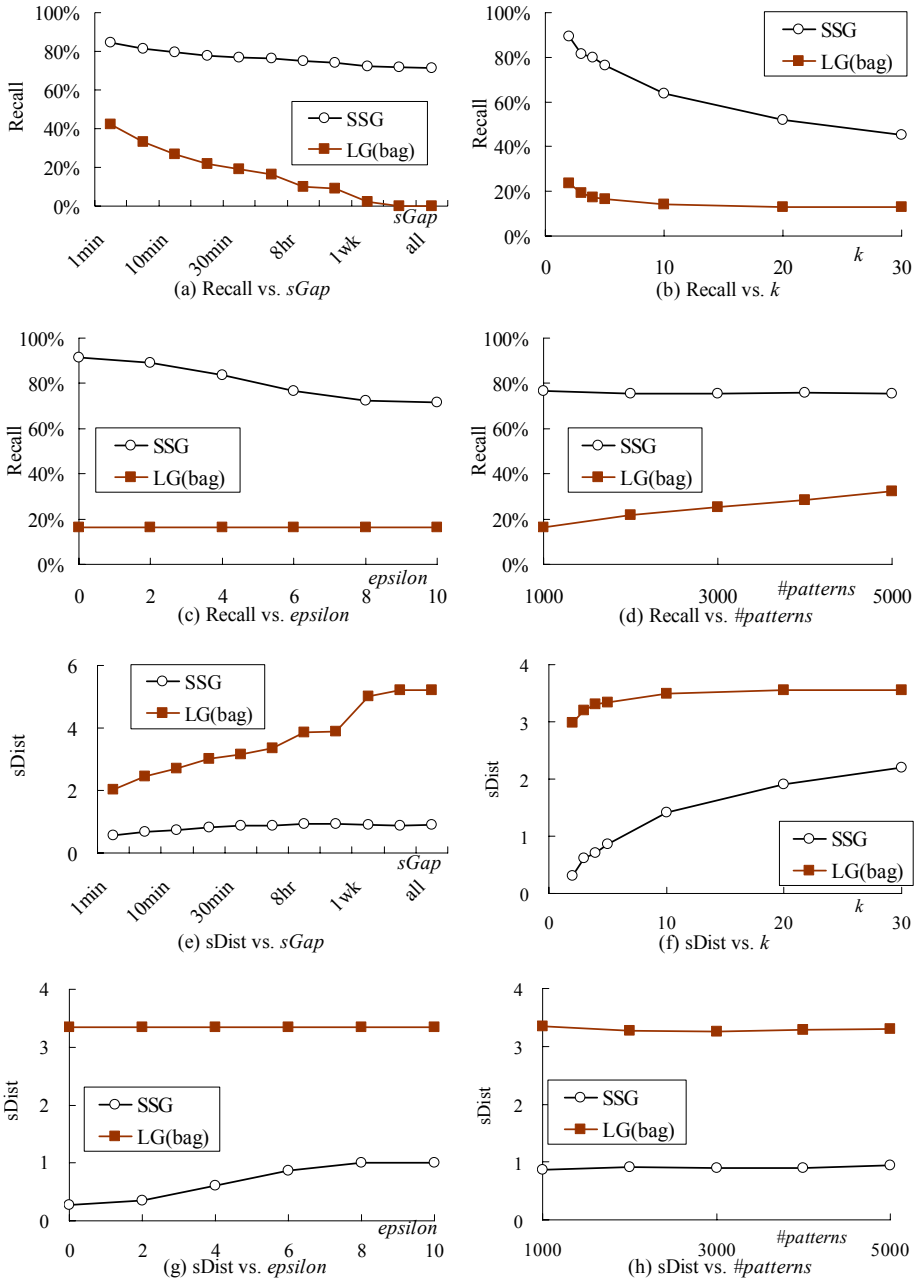


Figure 5.10 *Recall* and *sDist* in frequent pattern mining with varying *sGap*, *k*, ε, #patterns

Figure 5.10(e)-(h) show *sDist* with the varying *sGap*, *k*, *ε*, and *#patterns*. For most cases, *sDist* by SSG is less than 1, while *sDist* by LG(bag) is between 3 to 4.

The results show that SSG retains much more *original* frequent patterns than LG(bag) (indicated by *Recall*), and the patterns retained by SSG is semantically much closer to those in the original data than LG(bag) (indicated by *sDist*).

The reason is as follows: SSG clusters similar vocabularies together and selects the typical vocabulary of a cluster on a majority vote basis, which helps retain the frequent patterns of query-terms. LG(bag) retains less data utility since most of query-terms are excessively generalized as also observed with the motivating example.

5.6.4 Runtime Comparison between Algorithms

Finally, we briefly discuss the efficiency. Figure 5.11(a)-(c) show the runtime with the varying *sGap*, *k*, and *ε* respectively. For all the cases, LG(bag) runs in less than 100 seconds while the runtime of SSG ranges from 200 to 3,635 seconds. LG(bag) is more efficient than SSG.

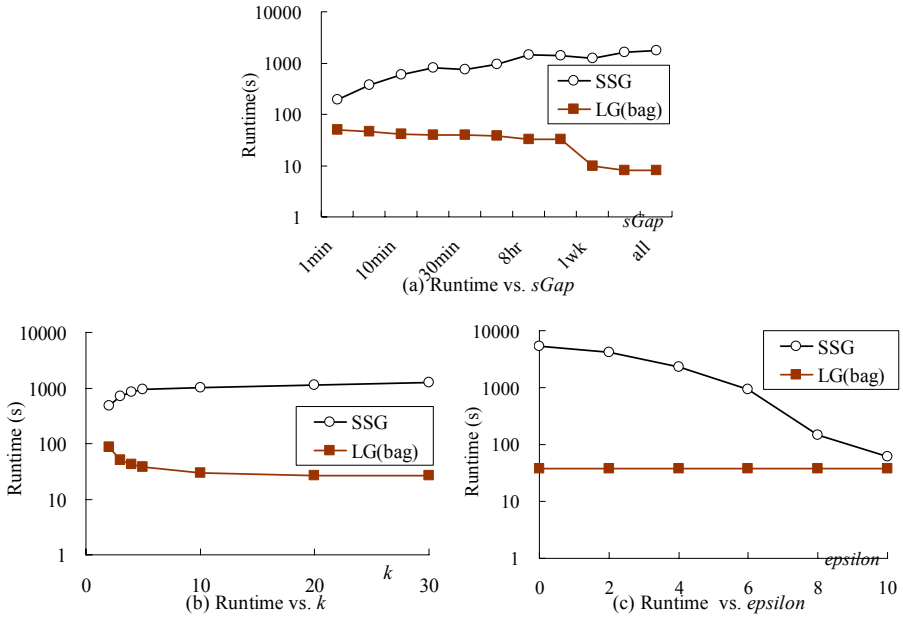


Figure 5.11 Runtime comparison with varying *sGap*, *k*, ε

Part of the reason is as follows: LG(bag) searches the taxonomy tree in a top-down manner. The overly generalized data suggests that the search process stops at a very high level on the taxonomy. Thus, LG(bag) takes less time. This result is within our expectation as our major concern is the quality of the anonymized data. To present an algorithm that has a significant gain in data utility and runs in a reasonable time is our objective.

5.7 Discussions and Extension

This chapter considers the scenario of publishing bag-valued data. Since set-valued data is just a specific case of bag-valued data, the proposed approach applies to set-valued data in general.

However, the proposed approach does not apply to the scenario in Chapter 4 since this chapter treats suppression as an inherent component while suppression is not allowed in Chapter 4 due to the assumption with Chapter 4 that is the adversary has both the knowledge about the presence of items and the knowledge about the *absence* of items. Concretely, Chapter 4 does not employ suppression since suppressing presence (absence) of items can introduce false absence (presence) of items and hence artificial privacy threats, which is unacceptable especially for data analysis with the false presence of items.

This chapter does not differentiate non-sensitive and sensitive query-terms as we doubt in this particular application domain (web search services) there is an agreement on the explicit classification of sensitive and non-sensitive query-terms although there are quite a few works on how to detect sensitive terms in a text document [86].

This chapter focuses on publishing vocabularies. One future work is to extend the semantic similarity based clustering approach to the context of publishing sequences of queries, which is more challenging. Another future work is to construct a more sophiscated semantic network of query-terms, and to improve the implementation of the weighted bipartite matching.

5.8 Summary

Web query logs provide a rich wealth of information, but also present serious privacy risks. This chapter considers publishing vocabularies, bags of query-terms extracted from web query logs with a given granularity, which has a variety of applications. This chapter aims at preventing linking attacks on such bag-valued data. The key feature of such data is the extreme sparsity, which renders conventional anonymization techniques not working well in retaining enough utility. This chapter proposes a semantic similarity based clustering approach to address the issue. This chapter measures the semantic similarity between two vocabularies by a weighted bipartite matching and presents a greedy algorithm to cluster vocabularies by the semantic similarities. Extensive experiments on the AOL query log show that the proposed approach retains more data utility than existing approaches.

6: Conclusions

Privacy preserving data publishing is an important research problem motivated by the privacy issues emerged from real world applications. Many promising approaches to this problem have been proposed. Nevertheless, much more challenges remain to be addressed.

6.1 Summary of Dissertation

This dissertation addresses one big challenge with privacy preserving data publishing, that is, how to balance privacy and data utility. The emphasis is on data utility, which is achieved by three basic strategies. The first strategy is to propose new privacy notions that allow strong tradeoff between privacy and utility at the conceptual level. The second strategy is to introduce flexible anonymization models that consider a large solution space to help find a solution with high utility. The third strategy is to present optimal algorithms finding solutions that are optimal in utility. The three strategies are applied to three privacy preserving data publishing scenarios with relational data, set-valued data, and web search data respectively. By doing so, this dissertation makes three contributions.

The first contribution is the optimal l^+-diverse anonymization for publishing relational data. We proposed an extended notion of the *l*-diversity principle, and presented a pruning based algorithm l^+-Optimize for finding an optimal solution under the full subtree generalization model with various suppression schemes. The novelty lies

in several strong techniques: a novel structure for enumerating all solutions, methods for estimating cost lower bounds, strategies for dynamically arranging the enumeration order and updating lower bounds. This approach is generic in that it can be instantiated with any reasonable cost metric observing the monotonicity property. Experiments showed that the flexible privacy notion, the flexible anonymization model, and the optimal algorithm help retain more data utility.

The second contribution is the optimal KL(m, n)-privacy solution for publishing set-valued data. We proposed the KL(m, n)-privacy principle, which tackles privacy threats both from the adversary's knowledge about the presence of some items and from the adversary's knowledge about the absence of certain items, and handles both the identity disclosure problem and the attribute disclosure problem. This notion is realistic and nevertheless embraces all the notions proposed so far. Moreover, we presented the first optimal algorithm by proposing an inverse itemset enumeration tree with strong pruning for mining privacy threats and a top-down search strategy integrated with cost based pruning to find an optimal solution. We also presented a multi-round approach to improve efficiency, which finds a solution very close to the optimal. Our approach outperforms the state of the art approaches both in data utility and in efficiency.

The third contribution is the vocabulary k-anonymity notion with the semantic similarity based clustering approach for publishing vocabularies, bags of query-terms, extracted from web query logs with a given granularity. Such bag-valued data have a wide range of applications. The key feature of such data is the extreme sparsity, which renders conventional anonymization techniques not working well in retaining enough data utility. We presented a semantic similarity based clustering approach together with a

greedy algorithm to address the issue. Extensive experiments on the AOL query log showed that our approach preserves more data utility than the state of the art approach.

6.2 Future Research Directions

In the light of the dissertation, we identify the following future research directions.

Privacy preserving approaches for new data sharing forms and new data types. New privacy protection problems are ever emerging with new data sharing forms and services, such as cloud computing and location-based services, and new data types, such as social network data and web query logs. Gellman discussed the issue of cloud computing and its implications for the privacy of personal information as well as its implications for the confidentiality of business and governmental information [39]. Bettini, et al. discussed the privacy issues in location-based services [18]. Backstrom, et al. discussed the privacy protection problem in social networks [14]. Xiong and Agichtein discussed query log ananlyis applications and possible privacy risks in this context, and pointed out research directions [107]. These works are good strarting points for research into privacy protection problem in the respective contexts.

Integration of Privacy preserving technology and information security technology. The ever-lasting challenge with privacy preserving data publishing is that privacy and utility are two contradicting goals. In many cases, it is hard to find a solution that provides sufficient privacy protection and retains enough data utility. Integrating privacy protection techniques with information security techniques could be a promising approach. The mix network protocol proposed by the information security research community have been adapted for privacy protection in data collection [11][20][110].

The secure multi-party computation protocol [111] in information security has been adapted for privacy protection in data sharing among multiple data collectors [8][114]. Recently, Shang, et al. proposed a privacy protection scheme that integrates access control policies and group key management for selective distribution of information [89]. The open problem is that: can we adapt information security techniques for privacy protection in data publishing?

Social and legal issues with privacy preserving technologies. Privacy protection is a complicated social and psychorogical issue, which cannot be fully addressed by technical solutions. We need to study the problem from multiple perspectives. For instance, Cooper examined the privacy protection techniques for publishing web query logs from a policy perspective [27]. Inter-disciplianry collaborative research is critical for providing systematic approaches to the real world privacy protection scenarios.

BIBLIOGRAPHY

[1] N.R. Adam, J.C. Worthmann. *Security-control methods for statistical databases: a comparative study*. ACM Computing Surveys, 21(4):515-556, December 1989.

[2] E. Adar. *User 4XXXXX9: Anonymizing query logs*. Query Log Analysis Workshop, In WWW, 2007.

[3] C.C. Aggarwal. *On k-anonymity and the curse of dimensionality*. In VLDB, 2005.

[4] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. *Achieving anonymity via clustering*. In PODS, 2006.

[5] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, A. Zhu. *Anonymizing tables*. In ICDT, 2005.

[6] C.C. Aggarwal and P.S. Yu. *A condensation approach to privacy preserving data mining*. In 9th International Conf. on Extending Database Technology, 2004.

[7] C.C. Aggarwal and Philips S. Yu. *Privacy-preserving data mining: Models and algorithms*. Springer 2008.

[8] R. Agrawal, A. Evfimievski, R. Srikant. *Information sharing across private databases*. In SIGMOD, 2003.

[9] R. Agrawal, T. Imielinski, and A. Swami. *Mining association rules between sets of items in large databases*. In SIGMOD, pages 207-216, 1993.

[10] R. Agrawal and R. Srikant. *Privacy-preserving data mining*. In SIGMOD, pages 439-450, 2000.

[11] M.Z. Ashrafi, S.K. Ng. *Collusion resistant anonymous data collection method*. In SIGKDD, 2009.

[12] A. Asuncion and D.J. Newman. *UCI Machine learning repository, 2007*. Irvine, CA: Uni of California, School of Information and Computer Science.

[13] M. Atzori, F. Bonchi, F. Giannotti, D. Pedreschi. *Anonymity preserving pattern discovery*. VLDB Journal, 2008.

[14] L. Backstrom, C. Dwork, J. Kleinberg. *Wherefore art thou R3579x?: Anonymized social networks, hidden patterns, and structural steganography*. In WWW, 2007.

[15] M. Barbaro and T. Zeller. *A face is exposed for AOL searcher No. 4417749*. New York Times, Aug 2006.

[16] R. J. Bayardo. *Efficiently mining long patterns from databases*. In SIGMOD, pages 85-93, 1998.

[17] R. J. Bayardo Jr. and R. Agrawal. *Data privacy through optimal k-anonymization*. In ICDE, pages 217-228, 2005.

[18] C. Bettini, X. SeanWang and S. Jajodia. *Protecting privacy against location-based personal identification*. In VLDB(SDM), 2005.

[19] A. Blum, C. Dwork, F. McSherry, and K. Nissim. *Practical privacy: The SuLQ framework*. In PODS, 2005.

[20] J. Brickell and V. Shmatikov. *Efficient anonymity preserving data collection*. In Proc. of the ACM SIGKDD, 2006, 76-85.

[21] Y. Bu, A. Fu, R. Wong, L. Chen. *Privacy preserving serial data publishing by role composition*. In VLDB, 2008.

[22] A. Budanitsky and G. Hirst. *Semantic distance in WordNet: an experimental, application-oriented evaluation of five measures*. In NAACL 2001 Workshop on WordNet and Other Lexical Resources, Pittsburgh, June 2001.

[23] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. *Secure anonymization for incremental datasets*. In VLDB (SDM), 2006.

[24] D. M. Carlisle, M. L. Rodrian, and C. L. Diamond. *California inpatient data reporting manual, medical information reporting for California, 5th edition.* Technical report, Office of Statewide Health Planning and Development. July, 2007.

[25] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. Mohania. *Efficient techniques for document sanitization*, In CIKM, 2008.

[26] B. Chen, K. LeFevre, and R. Ramakrishnan. *Privacy skyline: privacy with multidimensional adversarial knowledge*. In VLDB, 2007.

[27] A. Cooper. *A survey of query log privacy-enhancing techniques from a policy perspective*. ACM Trans. on the Web, Vol. 2 , Issue 4 (2008).

[28] I. Dinur and K. Nissim. *Revealing information while preserving privacy*. In Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pages 202–210, 2003.

[29] W. Du and Z. Zhan. *Using randomized response techniques for privacy-preserving data mining*. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2003.

[30] C. Dwork. *Differential privacy*. The 33rd International Colloquium on Automata, Languages and Programming—ICALP 2006, Part II, pp. 1–12, 2006.

[31] C. Dwork, F. McSherry, K. Nissim, and A. Smith. *Calibrating noise to sensitivity in private data analysis*. The 3rd Theory of Cryptography Conference—TCC 2006, pp. 265–284, 2006.

[32] C. Dwork and K. Nissim. *Privacy-preserving datamining on vertically partitioned databases*. In Matthew K. Franklin, editor, CRYPTO, volume 3152 of Lecture Notes in Computer Science, pages 528–544. Springer, 2004.

[33] A. Evfimievski, J. Gehrke, and R. Srikant. *Limiting privacy breaches in privacy preserving data mining*. In PODS, 2003.

[34] A. Evfimievski, R. Srikant, R. Agrawal, J. Gehrke. *Privacy preserving association rule mining*. In SIGKDD, 2002.

[35] C. Fellbaum. *WordNet: An electronic lexical database*. MIT Press, Cambridge MA, 1998.

[36] B. Fung, K. Wang, R. Chen, and P. S.Yu. *Privacy-preserving data publishing: a survey on recent developments*. ACM Computing Surveys, ACM Press 2010.

[37] B. Fung, K. Wang, and P.S. Yu. *Top-down specialization for information and privacy preservation*. In ICDE, 2005.

[38] J. Gehrke. *Models and methods for privacy-preserving data publishing and analysis*. In Tutorial at the 12th ACM SIGKDD. Philadelphia, PA, 2006.

[39] R. Gellman. *Privacy in the clouds: Risks to privacy and confidentiality from cloud computing*. World Privacy Forum, February 23, 2009.

[40] G. Ghinita, P. Karras, P. Kalnis, N. Mamoulis. *Fast data anonymization with low information loss*. In VLDB, 2007.

[41] G. Ghinita, Y. Tao, and P. Kalnis. *On the anonymization of sparse high-dimensional data*. In ICDE, 2008.

[42] J. Ginsberg, M. H. Mohebbi, R.S. Patel, L. Brammer, M.S. Smolinski, L. Brilliant. *Detecting influenza epidemics using search engine query data*. Nature 457, 1012-1014 (19 February 2009).

[43] K. Hafner. *And if you liked the movie, a Netflix contest may reward you handsomely*. New York Times, October 2, 2006.

[44] J. Han, J. Pei, and Y. Yin. *Mining frequent patterns without candidate generation*. In SIGMOD, 2000.

[45] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. *Mining top-k frequent closed patterns without minimum support*. In ICDM, 2002.

[46] Y. Hong, X. He, J. Vaidya, N. Adam, and V. Atluri. *Effective anonymization of query logs*. In CIKM, 2009.

[47] Z. Huang, W. Du. *OptRR: Optimizing randomized response schemes for privacy-preserving data mining*. In ICDE, 2008.

[48] Y. He and J. Naughton. *Anonymization of set-valued data via top-down, local generalization*. In VLDB, 2009.

[49] A. Horowitz, D. Jacobson, T. McNichol, and O. Thomas. *101 dumbest moments in business, the year's biggest boors, buffoons, and blunderers*. In CNN Money, 2007.

[50] A. Hundpool and L. Willenborg. *Mu-argus and Tau-argus: Software for statistical disclosure control*. The Third Int'l Seminar on Statistical Confidentiality, 1996.

[51] T. Iwuchukwu and J. Naughton. *k-Anonymization as spatial indexing: Toward scalable and incremental anonymization*. In VLDB, 2007.

[52] V. Iyengar. *Transforming data to satisfy privacy constraints*. In SIGKDD, pages 279-288, 2002.

[53] W. Jiang, M. Murugesan, C. Clifton, and L. Si. *t-Plausibility: Semantic preserving text sanitization*. In PASSAT, 2009.

[54] R. Jones, R. Kumar, B. Pang, A. Tomkins. *I know what you did last summer*. Query Logs and User Privacy, CIKM 2007.

[55] D. Kifer and J. Gehrke. *Injecting utility into anonymized datasets*. In Proc. of ACM SIGMOD, pp217–228, 2006.

[56] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi. *Optimization by simulated annealing*. Science, New Series 220 (4598): 671-680, 1983.

[57] A. Korolova, K. Kenthapadi, N. Mishra, A. Ntoulas. *Releasing search queries and clicks privately*. In WWW, 2009.

[58] R. Kumar, J. Novak, B. Pang, A. Tomkins. *On anonymizing query logs via token-based hashing*. In WWW, 2007.

[59] K. LeFevre, D. DeWitt, and R. Ramakrishnan. *Incognito: Efficient full-domain k-anonymity*. In SIGMOD, 2005.

[60] K. LeFevre, D. DeWitt, and R. Ramakrishnan. *Mondrian multidimensional k-anonymity*. In ICDE, 2006.

[61] N.Li, T. Li, S. Venkatasubramanian. *t-Closeness: Privacy beyond k-anonymity and l-diversity*. In ICDE, 2007.

[62] J. Li, Y. Tao, and X. Xiao. *Preservation of proximity privacy in publishing numerical sensitive data*. In SIGMOD, 2008.

[63] K.P. Lin, M.S. Chen. *Releasing the SVM classifier with privacy-preservation*. In ICDM, 2008.

[64] J. Liu, Y. Pan, K. Wang, J. Han. *Mining frequent item sets by opportunistic projection*. In SIGKDD, 2002.

[65] J. Liu and K. Wang. *On optimal anonymization for l^+-diversity*. In ICDE, 2010.

[66] J. Liu and K. Wang. *Anonymizing transaction data by integrating suppression and generalization*. In PAKDD, 2010.

[67] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. *l-Diversity: Privacy beyond k-anonymity*. In ICDE, 2006.

[68] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. *Privacy: From theory to practice on the map*. In ICDE, 2008.

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

[70] A. Meyerson and R. Williams. *On the complexity of optimal k-anonymity*. In PODS, pages 223-228, 2004.

[71] A. Narayanan and V. Shmatikov. *How to break anonymity of the Netflix prize dataset*. ArXiv Computer Science e-prints, October 2006.

[72] M. Nergiz, M. Atzori, and C. W. Clifton. *Hiding the presence of individuals from shared databases*. In SIGMOD, 2007.

[73] Ontario Ministry of Health and Long Term Care. *Personal health information protection act*, 2004.

[74] Office of the Privacy Commissioner of Canada. *The personal information protection and electronic documents Act*, 2000.

[75] S. Papadimitriou, F. Li, G. Kollios, and P. Yu. *Time series compressibility and privacy*. In VLDB, 2007.

[76] G. Pass, A. Chowdhury, C. Torgeson. *A picture of search*. The 1st Intl. Conf. on Scalable Information Systems, Hong Kong, June, 2006.

[77] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang. *Maintaining k-anonymity against incremental updates*. In SSDBM, 2007.

[78] B. Poblete, M. Spiliopoulou, R. Baeza-Yates. *Website privacy preservation for query log publishing*. In PinKDD 2007.

[79] President information technology advisory committee. *Revolutionizing health care through information technology*. Technical report, Executive Office of the President of the United States. June, 2004.

[80] M. Rasch. *Google's data minefield*. In The Register, 2006. http://www.theregister.co.uk/2006/01/31/google_subpoena_us_government/.

[81] V. Rastogi, M. Hay, G. Miklau, D. Suciu. *Relationship privacy: Output perturbation for queries with joins*. In PODS, 2009.

[82] V. Rastogi, S. Hong, and D. Suciu. *The boundary between privacy and utility in data publishing*. In VLDB, 2007.

[83] S. Rizvi and J. R. Haritsa. *Maintaining data privacy in association rule mining*. In VLDB, 2002.

[84] P.Samarati. *Protecting respondents' identities in microdata release*. TKDE, 13(6): 1010-1027, 2001.

[85] P. Samarati and L. Sweeney. *Generalizing data to provide anonymity when disclosing information*. In PODS, 1998.

[86] Y. Saygin, D. Hakkani-Tur, and G. Tur. *Sanitization and anonymization of document repositories*. In Web and Information Security, 2005.

[87] Y. Saygin, V. S. Verykios, C. Clifton. *Using unknowns to prevent discovery of association rules*. Conference on Research Issues in Data Engineering, 2002.

[88] J. Schwartz, A. Steger, A. Weißl. *Fast algorithms for weighted bipartite matching*. Experimental and Efficient Algorithms (2005), pp. 476-487.

[89] N. Shang, M. Nabeel, F. Paci, E. Bertino. *A privacy-preserving approach to policy-based content dissemination*. In ICDE, 2010.

[90] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz. *Analysis of a very large web search engine query log*. SIGIR Forum, 33(1):6-12, 1999.

[91] R. Srikant and R. Agrawal. *Mining generalized association rules*. In VLDB, 1995.

[92] W. Stallings. *Network security essentials: applications and standards* (*3rd edition*). Upper Saddle River, NJ : Pearson/Prentice Hall, 2007.

[93] L. Sweeney. *Achieving k-anonymity privacy protection using generalization and suppression*. Int'l Journal on Uncertainty, Fuzziness, and Knowledge-Base Systems 10(5): 571-588, 2002.

[94] Y. Tao, X. Xiao, J. Li, D. Zhang. *On anti-corruption privacy preserving publication*. In ICDE, 2008.

[95] M. Terrovitis, N. Mamoulis, P. Kalnis. *Privacy preserving anonymization of set-valued data*. In VLDB, 2008.

[96] U.S. Department of Health and Human Services. *Summary of the HIPAA privacy rule*. 2003.

[97] V.S. Verykios, E. Bertino, I.N. Fovino, L.P. Provenza, Y. Saygin, Y. Theodoridis. *State-of-the-art in privacy preserving data mining*. SIGMOD Record, Vol. 33, No. 1, March 2004.

[98] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, E. Dasseni. *Association rule hiding*. TKDE, 16(4), 2004.

[99] K. Wang and B.C.M. Fung. *Anonymizing sequential releases*. In SIGKDD, 2006.

[100] K. Wang, B.C.M. Fung, and P.S. Yu. *Template-based privacy preservation in classification problems*. In ICDM, 2005.

[101] K. Wang, P.S. Yu, and S. Chakraborty. *Bottom-up generalization: a data mining solution to privacy protection*. In ICDM 2004.

[102] Y. Wang, X. Wu. *Approximate inverse frequent itemset mining: Privacy, complexity, approximation*. In ICDM, 2005.

[103] R.C. Wong, A.W. Fu, K. Wang, J. Pei. *Minimality attack in privacy preserving data publishing*. In VLDB, 2007.

[104] X. Xiao and Y. Tao. *Personalized privacy preservation*. In SIGMOD, 2006.

[105] X. Xiao and Y. Tao. *Anatomy: Simple and effective privacy preservation*. In VLDB, pages 139-150, 2006.

[106] X. Xiao and Y. Tao. *m-Invariance: Towards privacy preserving re-publication of dynamic datasets*. In SIGMOD, June 11–14, 2007, Beijing, China.

[107] L. Xiong and E. Agichtein. *Towards privacy-preserving query log publishing*. In Query Logs Workshop at WWW, 2007.

[108] Y. Xu, K. Wang, A. Fu, and P.S. Yu. *Anonymizing transaction databases for publication*. In SIGKDD, 2008.

[109] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. Fu. *Utility-based anonymization using local recoding*. In SIGKDD, 2006.

[110] Z. Yang, S. Zhong, and R.N. Wright. *Anonymity-preserving data collection*. In Proc. of the 11th ACM SIGKDD, 2005, 334–343.

[111] A.C. Yao. *How to generate and exchange secrets*. Proceedings of Twenty-seventh IEEE Symposium on Foundations of Computer Science, Toronto, Canada, October 1986, 162-167.

[112] J. Yi, F. Maghoul, and J. Pedersen. *Deciphering mobile search patterns: A study of Yahoo! Mobile search queries*. In WWW 2008.

[113] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. *Aggregate query answering on anonymized tables*. In ICDE, 2007.

[114] N. Zhang and W. Zhao. *Distributed privacy preserving information sharing*. In VLDB, 2005.

[115] Z. Zheng, R. Kohavi, L. Mason. *Real world performance of association rule algorithms*. In SIGKDD, pages 401-406, 2001.

[116] Y. Zhu, L. Xiong, C. Verdery. *Anonymizing user profiles for personalized web search*. In WWW, 2010.

[117] http://blog.netflix.com/2010/03/this-is-neil-hunt-chief-product-officer.html.