

Data privacy with heuristic anonymisation

Sevgi Arca* and Rattikorn Hewett

Department of Computer Science,
Texas Tech University, Texas, USA

Email: sevgi.arca@ttu.edu

Email: rattikorn.hewett@ttu.edu

*Corresponding author

Abstract: Abundance of data makes privacy more vulnerable than ever as it increases the attackers' ability to infer confidential data from multiple data sources. Anonymisation protects data privacy by ensuring that critical data are non-unique to any individual so that we can conceal the individual's identity. Existing techniques aim to minimally alter the original data so that either the anonymised data or its analytical results (e.g., classification) will not disclose certain privacy. Our research aims both. This paper presents *HeuristicMin*, an anonymisation approach that applies generalisations to satisfy user-specified anonymity requirements while maximising data retention (for analysis purposes). Unlike others, by exploiting monotonicity property of generalisation and simple heuristics for pruning, *HeuristicMin* provides an efficient exhaustive search for optimal generalised data. The paper articulates different meanings of optimality in anonymisation and compares *HeuristicMin* with well-known approaches analytically and empirically. *HeuristicMin* produces competitive results on the classification obtained from the anonymised data.

Keywords: anonymisation; data generalisation; bottom-up generalisation; BUG; privacy.

Reference to this paper should be made as follows: Arca, S. and Hewett, R. (2023) 'Data privacy with heuristic anonymisation', *Int. J. Information and Computer Security*, Vol. 20, Nos. 1/2, pp.104–132.

Biographical notes: Sevgi Arca is a PhD candidate in Computer Science at Texas Tech University. She received a Bachelor of Science degree in Computer Science and Engineering from Sabanci University, Turkey. Her current research interests include privacy protection and anonymization, blockchain privacy and security attacks, automating privacy assessment, heuristic search, and intelligent systems.

Rattikorn Hewett is a Whitacre Chair in Computer Science, a Professor and Chair of the Department of Computer Science, Texas Tech University. He was a Postdoctoral Fellow at Stanford University and received a PhD in Computer Science from Iowa State University, an MEngSc from the University of New South Wales, and a BA Honours in Pure Mathematics and Statistics from Flinders University, Australia. Her applied research in artificial intelligence covers four thrust areas: cyber security, data science, automated software and system engineering, and intelligent controls and reasoning. Her research sponsors include NSF, DARPA, DOE, EPRI, Florida State, Texas State, Boeing and IBM. She has published over 100 peer-reviewed technical articles, and has served on several journal editorial boards, and numerous conference program committees.

This paper is a revised and expanded version of a paper entitled ‘Realizing data anonymization as search with optimal guarantee’ presented at Workshop on AI for Privacy (AI4P 2020) in conjunction with European AI Conference, Spain, 4 September 2020.

1 Introduction

The concept of privacy has existed throughout the human history in different degrees, forms and contexts (Clearwater and Hughes, 2013). Privacy is not about having something to hide but having a choice to control what to share. Issues on privacy have recently received a lot of attentions due to the rapid growth of information economy and advanced technologies that allow data to be (automatically) collected, stored, analysed and readily available. As these technologies become more sophisticated, users may end up losing more controls of their data without knowing making privacy protection more difficult.

The problem is compounded by malicious acts. An abundance of data information makes data privacy more vulnerable, as attackers can infer confidential data from different query sources. For example, by linking attribute values of *ZIP code*, *gender* and *birth date* from anonymous data (or de-identified data) to those of other public data, Sweeney (1998) has shown that 87% of US population can be uniquely identified (or re-identified) to a certain degree. To achieve absolute privacy, one can accomplish this easily by simply not sharing or publishing the data. However, this is not possible as long as we need to make use of the data. *Anonymisation* addresses the issue of data privacy by making sure that each set of ‘critical’ data values belongs to more than one individual so that the identity of the individual can be protected.

Much research on anonymisation techniques has been performed (Bayardo and Agrawal, 2005; Fung et al., 2009, 2007, 2005; Hundepool and Willenborg, 1996; Iyengar, 2002; LeFevre et al., 2005, 2006; Li et al., 2007; Machanavajjhala et al., 2006; Samarati, 2001; Sweeney, 1998, 2002a, 2002b; Wang et al., 2004). Basic approaches use generalisation to transform the original data, according to its taxonomies, into a generalised table that complies with anonymity requirements (Samarati, 2001; Sweeney, 2002a, 2002b). Some approaches are theoretically feasible but not practical (Sweeney, 2002b). Many techniques have focused on minimising generalisations to enhance computational efficiency. Several use the same term (e.g., minimal generalisation) with different details. Not all optimal techniques mean the same and that can mislead the selection of appropriate technique. On the other hand, excess manipulation in generalisation to provide anonymity can corrupt important information that the data may convey. As a result, the data become less informative and may give wrong decision-making or inaccurate inferences. It is important for anonymisation to strike these balances.

Because of a trade-off between data privacy and the usefulness of the information that the data provide, recent anonymisation techniques take contents of the generalised table into account. These techniques can be divided roughly into two groups: those that lay formal grounds to minimise changes of the data (or data distortion) (e.g., Li et al., 2007; Sweeney, 2002b) and those that maximise data for a specific use, typically for classification (e.g., Fung et al., 2007, 2005; Wang et al., 2004). These techniques are

useful. However, the first group may not be practical due to its high computational cost while the purpose of the second group can be too limited. There has been no attempt to explore if a solution from one group can be effective for the purpose of the other. Although techniques for anonymisation have been studied extensively, most have been designed to address each of these specific goals (e.g., efficient generalisation, classification) as opposed to an integrated design for a general-purpose solution.

The contribution of this paper includes:

- 1 articulation and comparison of various aspects of privacy objectives for anonymising the identity of the individuals associated with given data
- 2 *HeuristicMin*, a new anonymisation approach that applies generalisations along with optimal artificial intelligence search to securing privacy by satisfying user-specified anonymity requirements while maximising information preservation.

Unlike prior approaches, *HeuristicMin* aims to find an integrated anonymisation system solution for computational efficiency, optimality and general data usage. *HeuristicMin* exploits monotonicity property of generalisation along with heuristics to search for optimal generalised data that comply with anonymity requirements. By using simple heuristics and appropriate grain size of generalisations, *HeuristicMin* prunes and narrows down the search space to enhance efficiency in its optimal search and prefers generalisation that keeps maximum data for general usage (as opposed to for classification). The paper also contributes to the analysis that expresses distinctions of the term ‘optimality’ as well as comparison studies among representative approaches.

The rest of the paper is organised as follows. Section 2 describes related work and Section 3 defines terms, notations and background concepts. Section 4 gives the problem statement and the proposed approach of *HeuristicMin* along with an illustration and complexity analysis. Section 5 presents analytical comparison studies of *HeuristicMin* with other optimal anonymisation techniques as well as empirical comparisons with other anonymisation techniques for classification. Section 6 concludes the paper.

2 Related work

Research in privacy has been studied extensively in various aspects including privacy with rationality and decisions (Acquisti and Grossklags, 2005), protection and preservation of privacy (Bayardo and Agrawal, 2005; Dwork, 2006; Iyengar, 2002; LeFevre et al., 2005, 2006; Samarati, 2001; Sweeney, 1998, 2002a, 2002; Wang et al., 2004). Most current privacy protection techniques are stemmed from the two notorious pioneer concepts of Dwork’s (2006) *differential privacy* and Sweeney’s (2002a) *k*-anonymity. Both aim to protect identity of the individuals associated with the data.

Differential privacy provides objectives to ensure that the probability of statistical query results from the data with or without an individual’s information are (nearly) the same. If the individual’s data does not have a significant effect on the query outcome, then releasing his data does not harm his privacy. To do this, differential privacy injects *random noise* (e.g., noise from Laplace or Gaussian distribution) to the query results to mask the individual’s identity. Consequently, the query results become less accurate. The more accurate the query result is, the less privacy protection we have.

Alternatively, *k-anonymity* (Sweeney, 2002a) aims to guarantee that each group of unique critical attribute (or *quasi identifier*) values must have at least k records in order to prevent them from being personally identifiable (or *de-identified*). Mechanisms for achieving *k-anonymity* are *generalisation* and *suppression*, which involve replacement of a current value by its corresponding more abstract value or general form, respectively. The problem of finding optimal *k-anonymisation* is NP-hard (Meyerson and Williams, 2004). To better protect against *attribute* disclosure, the *k-anonymity* concept has been extended to *l-diversity* (Machanavajjhala et al., 2006) to make sure each sensitive attribute has at least l well-represented values and to *t-closeness* (Li et al., 2007) to ensure the distribution of a sensitive attribute in each group is close to the distribution of the attribute in an overall database. Our work focuses on *k-anonymity* anonymisation; not its extensions nor the differential privacy.

Most approaches dealing with *k-anonymity* focus on minimal *k-anonymisations* (i.e., minimal number of generalisations/suppressions satisfying the *k-anonymity* requirements) (Samarati, 2001; Sweeney, 2002a, 2002b). Sweeney's (2002a, 2002b) *MinGen* algorithm uses exhaustive search to find the minimal *k-anonymisation* with minimal distortion. The algorithm searches for optimal solution on a cell level, which makes it impractical for scalability due to a large search space. However, *MinGen* is an early work that provides a concrete formal model for minimal *k-anonymisation*. Based on binary search, Samarati's (2001) *AllMin* algorithm searches for *k-anonymisations*, where the generalisation is not on a cell but attribute level. Thus, *AllMin* has reduced search space. Nevertheless, since binary search is a blind search, computational cost can still pose a problem since *AllMin* searches for all possible *k-generalisations* (i.e., no pruning). Our proposed approach is similar to the above approach in searching for an optimal *k-anonymisation*. However, we apply heuristics for efficient search as well as maximal data content. We also note that these approaches that yield optimal results have different implications on the term optimality.

Other heuristic techniques include *Datafly* (Sweeney, 1998), the μ -argus system (Hundepool and Willenborg, 1996) and *Mondrian* (LeFevre et al., 2006). The core *Datafly* employs a heuristic based on the number of distinct attribute values to select an attribute for generalisation with additional suppressions to obtain data that satisfy *k-anonymity* requirements. The μ -argus system is similar to *Datafly* in basic structure of the algorithm. However, it uses heuristics based on user-specified ratings of sensitive attributes and unsafe combinations of attributes to test for elimination by generalisation and suppression. The multidimensional algorithm *Mondrian* follows a greedy approach to partition the data space into regions with at least k value. The method considers a top-down approach where it starts with the most general attribute value and partitions recursively as long as the anonymity requirement is satisfied. Since the approach uses a greedy method the optimality is not guaranteed. *Datafly*, μ -argus and *Mondrian* improve efficiency. Unlike our approach, they do not guarantee optimal results in terms of minimal *k-anonymisation* and the μ -argus system may also fail to satisfy *k-anonymity* (Sweeney, 2002b).

It is clear that generalisation and suppression increase the degree of anonymity. However, the more data are anonymised, the less specific information they retain. Explosive applications of machine learning to analyse data have generated a relatively new research direction of anonymisation by not only satisfying *k-anonymity* but also aiming for the data to still be useful for data analysis (Fung et al., 2009, 2007, 2005). All these approaches use the resulting anonymised data for classification except the approach

in Fung et al. (2009) that modifies the algorithm for clustering. Our approach is similar to Wang et al. (2004) in that we use generalisation as a mechanism for k -anonymisations and thus, a bottom up search, whereas approaches in Fung et al. (2009, 2007, 2005) rely on specialisation mechanism and thus, a top-down search for data to comply with k -anonymity. Unlike these approaches that focus on the resulting data for modelling specific task for data analysis, our approach optimises the data content (to be as close as possible to the original data) for general use. Furthermore, none of these data analysis-focused approaches result in optimal solution but ours does.

Work by Iyengar (2002) and Bayardo and Agrawal (2005) address performance of search for minimal k -anonymisations. Using genetic algorithm, the former represents (table) data to be string-like and evolves them toward optimal solution. The latter uses the same data framework and extends to general cost metrics. Similar to these approaches, our approach aims to produce resulting data that accounts for a trade-off between anonymisation and quality data preservation. Unlike these approaches that focus on empirically optimal efficient approaches for anonymisation, our approach focuses on efficiently producing optimal resulting data (will be explained in detail in Section 4).

3 Preliminary concepts

This section describes relevant terms, notations and concepts.

3.1 Data table and anonymity requirements

A table $T(A_1, A_2, \dots, A_n)$ represents a relational database with a schema A of attributes A_1, A_2, \dots, A_n . Each data *record* is an instance of a tuple of the data table of the form (a_1, a_2, \dots, a_n) , where data entry a_i is a domain value of attribute A_i [i.e., $a_i \in \text{dom}(A_i)$] for all i . A table can contain multiple records (or instances) of a tuple. For compactness, each row of the table represents a distinct tuple followed by a column representing its corresponding number of records. For example, the first (distinct) row of Table 1 represents five records of a tuple (*undergrad, female, 3.2*).

Table 1 An illustration of anonymity requirement

<i>Status</i>	<i>Gender</i>	<i>Age</i>	<i>#Records</i>
Undergrad	Female	3.2	5
Undergrad	Male	3.2	6
Undergrad	Female	3.6	2
Graduate	Male	3.6	3
Graduate	Female	3.8	1
Graduate	Male	3.8	2

An *anonymity requirement* specifies an *anonymity degree* required on a certain subset of attributes in the table schema, called *anonymity shield* [or referred in Sweeney (2002a) as *quasi-identifiers*]. More specifically, for a subschema $B \subseteq A$, a k -*anonymity requirement* on shield B , denoted by $\langle B, k \rangle$, states that a table projection on B results in a set of B -projected tuples, each of which has a minimum of k records. We write $[t, r_i]$ to

represent a pair of a tuple t and its corresponding number of records r_t . Thus, $\langle B, k \rangle$ is violated if there is $[b, r_b]$ for some B -projected tuple b such that $r_b < k$.

As an example, consider Table 1 with a given anonymity requirement $\langle \{education, gender\}, 4 \rangle$. Here, $dom(gender) = \{female, male\}$ and $dom(education) = \{undergrad, graduate\}$. Thus, there are four combinations of domain values of the tuple projection on $\{education, gender\}$. First, for a projected tuple $b = (undergrad, female)$, Rows 1 and 3 of Table 1 give a total number of records, $r_b = 7$. Thus, we have $[(undergrad, female), 7]$, which satisfies the required anonymity degree of four. Similarly, for the rest three tuples, we have $[(undergrad, male), 6]$, $[(graduate, male), 5]$ and $[(graduate, female), 1]$. The last case violates the anonymity requirement since a minimum of four records is required. Therefore, the anonymity requirement $\langle \{education, gender\}, 4 \rangle$ is not satisfied.

Intuitively, the anonymity requirement protects the privacy of a record holder by ensuring that for each tuple projected on a shield, the number of records that share the same tuple values must be large enough to help 'hide' the individual identity/information. Thus, the larger the degree of the anonymity is, the harder it is to identify the identity of the record holder. In general, for a given table, one can define more than one anonymity requirement, each of which can have a different anonymity degree and a shield. In practice, the data provider or the data user specifies the anonymity requirement.

Different shield sets are useful when there are many attributes that can be critical to identification of record holders. To determine one single shield may not be practical or easily selected as it depends on different contexts of data usage (e.g., $\{Zipcode\}$ may not be a shield that is as critical for restaurant customers as those for hospital patients because there are likely to have more restaurants than hospitals in a given zipcode). The choice of anonymity degree and a shield (set of projected attributes) in anonymity requirement can impact data inferences that lead to privacy breaches. If the anonymity degree is too low, the shield may or may not be able to protect the individual identity (e.g., when the projected tuple becomes personally identifiable). For example, anonymity requirement $\langle \{education\}, 1 \rangle$ and $\langle \{education, gender\}, 1 \rangle$ would allow publishing every row in Table 1. While the shield $\{education\}$ can protect individual identity of a student based on their educational information [since there are 13 (*undergrad*) and 6 (*graduate*) records], the shield $\{education, gender\}$ results in one record of (*graduate, female*). If we are looking for a female graduate student, because there is only one record of such a person (as shown in Table 1), we can directly infer that she has a GPA of 3.8. Here, the shield $\{education, gender\}$ yields a personally identifiable tuple.

On the other hand, if we set the anonymity degree too high, none of the data can satisfy the requirement and as a result, there is no data release for privacy becomes over protected. For example, applying anonymity requirement $\langle \{education, gender, GPA\}, 7 \rangle$ to Table 1, none of the projected tuples (e.g., (*undergrad, female, 3.2*), (*undergrad, male, 3.2*)) would satisfy the requirement. Even if we lower the anonymity degree to 4, some of the projected tuples, e.g., (*undergrad, female, 3.6*), (*graduate, male, 3.6*), (*graduate, female, 3.8*), and (*graduate, male, 3.8*) would all violate the requirements for data publishing/sharing. Instead of excluding such data completely to protect privacy, a compromising alternative is to publish data in a less informative way. This leads to the concept of generalisation and specialisation that we will describe next.

3.2 Anonymity via generalisation and specialisation

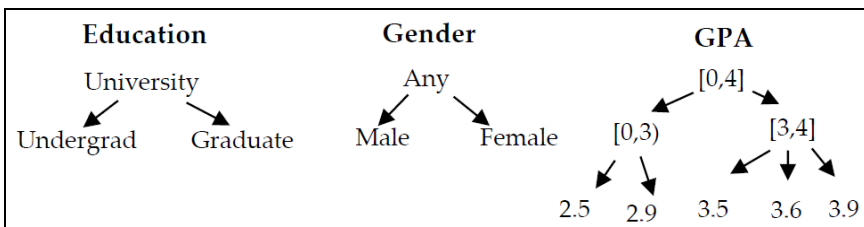
Generalisation replaces an attribute value by a more general but semantically consistent value (e.g., generalise a city by its state). This increases the number of tuples that share the same attribute value, and consequently increases a number of records, and anonymity degree, to hide individual identities. Generalisation provides several advantages to preserve data privacy including consistent interpretation, traceability and minimal content distortion (Sweeney, 2002a). Specialisation is, however, a reverse of generalisation.

By using the taxonomy (or hierarchy) of the corresponding attribute tree, generalisation replaces the attribute value by its parent node to protect privacy. For example, Figure 1 shows three taxonomies of three attributes: *education*, *gender* and *GPA*. As shown in Table 2(a), since there is only one record of a student with *GPA* 3.9 and *GPA* 2.5, we can identify gender and education of each of these students. Since both are females, applying generalisation to *GPA* cells that correspond to female students will give a resulting Table 2(b), where we can no longer identify the education and gender of the student with *GPA* 3.9 (as we have [(*undergrad*, *female*, [3, 4]), 6], i.e., we hide them among six students with *GPA* in [3, 4]). However, we can still infer information about the student with a *GPA* 2.5 (last row).

Table 2 Cell-wise generalisation and effects on anonymity (see online version for colours)

<i>Education</i>	<i>Gender</i>	<i>GPA</i>	#Rec	<i>Education</i>	<i>Gender</i>	<i>GPA</i>	#Rec
(a) Initial table				(b) Generalise on GPA cells			
Undergrad	Female	3.5	5	Undergrad	Female	[3, 4]	5
Undergrad	Male	3.6	6	Undergrad	Male	3.6	6
Undergrad	Female	3.9	1	Undergrad	Female	[3, 4]	1
Graduate	Male	2.9	2	Graduate	Male	2.9	2
Graduate	Female	2.5	1	Graduate	Female	[0, 3]	1
(c) Generalise on gender, GPA cells				(d) Final table			
Undergrad	Female	[3, 4]	5	Undergrad	Female	[3, 4]	6
Undergrad	Male	3.6	6	Undergrad	Male	3.6	6
Undergrad	Female	[3, 4]	1	Graduate	Any	[0, 3]	3
Graduate	Any	[0, 3]	2				
Graduate	Any	[0, 3]	1				

Figure 1 Taxonomy tree of three attributes



To hide information of this student, we can generalise her gender, using taxonomies in Figure 1, to *any* giving [(*graduate*, *any*, [0, 3]), 1]. Next, we generalise the row above it

by replacing the *GPA* 2.9 to [0, 3) as well as replacing *male* to *any* giving [(graduate, any, (0, 3]), 2]. The result is shown in Table 2(c). Table 2(d) shows a final table after merging the rows with the same tuples. Thus, the gender and education of the student with *GPA* 2.5 is protected. If the anonymity degree required on all of the three attributes is two, generalisation transforms Table 2(a) that violates the anonymity requirement into Table 2(d) that complies with the anonymity requirements.

Note that there are many ways one can generalise. Table 2 shows generalisation at a *cell* level (i.e., a *data entry* of a specific row and column of a table) (e.g., Sweeney, 2002b). Some simplify the grain size and perform generalisation by *column*, where generalisation is applied to data entry of *each* row under the same attribute column (e.g., Samarati, 2001). Another type of generalisation is performed at a height level (or by *attribute*). In this case, for a given taxonomy tree of an attribute, if we generalise a child to its parent, we also generalise *all* siblings of the child to the parent as well. This will apply to *all* occurrences of the child and siblings in Table 2 so that all entries of the same level in the table are generalised to the next level at the same time in order to improve efficiency in finding anonymity compliant rows. Thus, when T is generalised on attribute A , the generalisation is applied only to the table rows that contain the child and its siblings. For example, consider Table 2(a). Generalising *female* enforces generalising *male*. Both are replaced by *any* (generalising from height 0 to height 1) on all of their occurrences (rows) in Table 2. This gives Table 2 where every row has value *any* under the column *gender*. On the other hand, if we generalise 2.5 in Table 2(a) on *GPA*, the resulting table would look like Table 2(a) except the data entries under *GPA* of the last two rows of Table 2 are replaced by [0, 3) since 2.9 is the only sibling of 2.5. Generalised by column is more general than that by attribute.

4 Problem and proposed approach

4.1 Problem statement

The paper addresses the problem of how to protect privacy of the owners of the shared/public data. This aims to protect the individual identities and their corresponding data. Although the ultimate privacy can be achieved by generalising the data to the highest level of hierarchy, this results in making every individual's information the same. To the lesser extent of generality, the data will not be specific enough to be useful. Since privacy works against sharing information, a more practical question should be how to protect data privacy while keeping the shared data as informative as possible. Our criterion is to satisfy user-specified anonymity requirements while the amount of non-redundant data released measures the degree of data preservation.

More formally, let $T(A_1, A_2, \dots, A_n)$ (or T when attributes are implicitly defined) be a data table of a schema $A = \{A_1, A_2, \dots, A_n\}$ and taxonomy trees of each attribute A_i . Let R be a set of anonymity requirements $\langle R_i, k_i \rangle$, for shield $R_i \subseteq A$ and anonymity degree $k_i > 0$ for $i = 1, \dots, m$. We say that T satisfies $\langle R_i, k_i \rangle$ if every R_i -projected tuple has a minimum of k_i records and T satisfies R if T satisfies $\langle R_i, k_i \rangle$ for all $i = 1, \dots, m$.

The problem can be stated as follows. *For a given table T , its corresponding taxonomy trees and a set of anonymity requirements R , find an optimal table T' that satisfies R . By optimality, we mean that T' preserves the most information of T . Specifically, T' is a generalised table of T such that T' has a maximum number of rows*

among all T 's generalised tables that satisfy R . Recall each table row represents a distinct tuple that can have one or more instances of records. Thus, the highest number of rows represents the most non-redundant information from data. Thus, we use the number of rows (or the table size) to capture non-redundant information.

4.2 Proposed anonymisation

Our research aims to mask the sensitive information on the table while preserving as much information as possible. We propose a hybrid approach that combines artificial intelligence's heuristic search concepts with a classic anonymisation technique to guarantee optimal informativeness as well as compliance with anonymity requirements. The resulting table is optimal in that it preserves as many rows of the anonymised table as the anonymity requirement allows. Specifically, we use generalisation for anonymisation. Thus, our approach is a bottom-up heuristic approach. Figure 2 provides a general overview of the proposed algorithm.

Figure 2 Proposed algorithm

Procedure Anonymization	
Inputs:	T , a table with a set of attributes A , corresponding taxonomy trees of each attribute in T , and a set of anonymity requirement R with a set of anonymity shield attributes $S \subseteq A$.
Output:	a generalized table T' of T where T' has a maximum number of rows among all generalized tables of T satisfying R .
1	For each violating row and applicable attribute B in S
2	$T' \leftarrow$ generalized table of T on B
3	Add T' in W ;
4	Endfor
5	Repeat
6	Select from W , table T_k that has a maximum number of rows and a non-zero minimum number of rows that violate R
7	For each violating row and applicable attribute B in S
8	$T'_k \leftarrow$ generalized table of T_k on B
9	Add T'_k in W ;
10	Endfor
11	Remove T_k from W
12	Until W is empty or no tables in W has number of rows > number of rows of a table that satisfies R
13	Return T^* that has maximum number of rows over all tables in W that satisfy R

As shown in Figure 2, the proposed algorithm has three inputs: a data table T with a set of attributes A , a taxonomy tree for each attribute, and a set of anonymity requirements R with a set of anonymity shield attributes $S \subseteq A$. Note that, as described in the input specification and Line 1, the algorithm is also applicable to multiple requirements, each of which may have a different shield set as well as a different anonymity degree. We assume that T does not satisfy R , otherwise there is nothing to be done and T is our optimal table.

The algorithm iteratively generalises a table on an appropriate attribute using its corresponding taxonomy tree to increase anonymity degree. In Lines 1–4, a generalised table of T on each attribute in S is generated and maintained in set W . Each generalised table keeps track of two key heuristics: the number of rows that violate R and the total number of rows on the table. The former tells how close we are to finding the table that satisfies the anonymity requirements R while the latter measures how much non-redundant data is preserved. As shown in Figure 2, Line 6, we select, among generalised tables in W , a table that has the highest number of rows with the lowest violation number of rows to be further generalised (Lines 7–10). Each selected table will be removed from W (Line 11). As shown in Lines 5–12, this generalisation process repeats until there is no more tables left in W or no tables in W has the number of rows $>$ the number of rows of a table that satisfies R . In the latter case, all the tables in W either have:

- 1 zero number of rows that violate R
- 2 non-zero number of rows that violate R out of the total number of rows that is smaller than those tables in Case 1.

In other words, we stop expanding the search when we find a table that satisfies R or a table that is smaller than the biggest table that satisfies R found so far (even though it violates R). The principle rationale behinds this is the monotonicity behaviour of generalisation. As we increase the generalisation steps, the table can never grow in size. Because our goal is to maximise the size of the table that satisfies R , the algorithm only further generalises the table that is larger than those found to satisfy R so far. If a table violates R but is already smaller than the biggest table found so far to satisfy R , further generalising it would not result in a larger table that satisfies R .

As shown in Line 13 of Figure 2, the algorithm selects the largest table among the tables in W with no anonymity requirements violation. Note that it is possible to have more than one of such table of the same size. In such a case, the algorithm selects the first one found as it represents the table that has the least number of generalised steps. In other words, it retains most specific data that are closest to the given data table.

Since generalisation procedure monotonically decreases the number of rows, our approach uses this property to prune the fruitless path of an exhaustive search. Thus, it finds an optimal solution of a table that maximises the information preserved (i.e., the table size) from the original table while hiding sufficient privacy by satisfying anonymity requirements (i.e., zero violation rows).

4.3 Illustration

Figure 3 shows the three taxonomy trees of attributes corresponding to a given table T on top of Figure 4. Given the anonymity requirements set R as $\{<\{education, sex, hours\}, 4>\}$. Each of rows 3, 6 and 7 of T has two records and thus, violates R that requires a minimum of four records. (Note that if R contains multiple requirements, Line 1 would collect all violating rows. For example, if R includes an additional requirement of $<sex, 20>$ then Line 1 would have included rows 4–7 as violating rows.) We now apply the proposed approach. Let $T(n, m)$ represent a generalised table T generated with n number of rows that violate R out of m total number of rows in T . The rows that violate R in each

table are indicated by colour highlight. The column *class* will be used for comparison later and should be ignored for now, as it is not directly a part of our approach.

Figure 3 Taxonomy trees

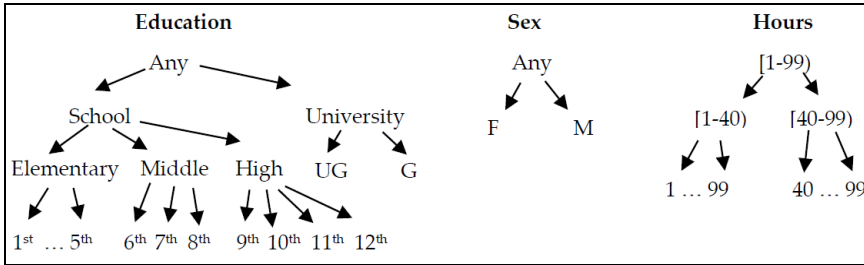
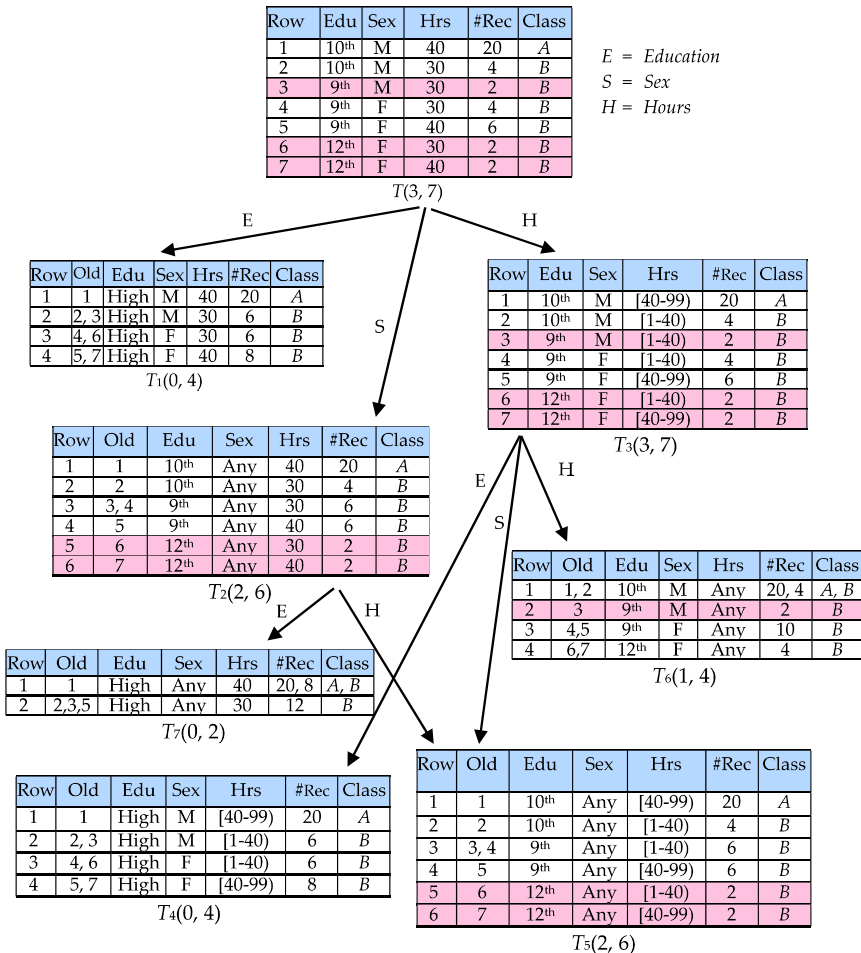


Figure 4 Example of our proposed anonymisation approach (see online version for colours)



As shown in Figure 4, starting from $T(3, 7)$, by applying Lines 1–3 of the algorithm, we generalise T on education, sex, and hours and obtain $T_1(0, 4)$, $T_2(2, 6)$ and $T_3(3, 7)$,

respectively. Looking more closely, for example for T_1 , by generalising T on *education* to T_1 , all rows in T_1 share the same value of *high* under column *education*.

Thus, a total number of rows in T_1 is 4, due to four combinations of two values of *sex* and two of *hours* in T . For example, rows 2 and 3 both represent a tuple (*high*, *M*, 30) with 4 and 2 records, respectively. Both rows are merged to row 2 of T_1 with a combined number of records as 6. The rest is similar, and we obtain $T_1(0, 4)$ with four rows, none of which violate R . On the other hand, by generalising T on *sex* results in T_2 where now rows 3 and 4 are the same and the number of records can be combined giving $T_2(2, 6)$ with a reduced number of violation rows to 2 and size of the table to 6. For T_3 , by generalising T on *hours*, a data entry of each row under column *hours* in T is replaced by either [1, 40) or [40, 99) depending on its value and appropriate generalisation governed by the corresponding taxonomy [e.g., 30 is replaced by [1, 40)]. As a result, T_3 still maintains the same number of rows (i.e., 7) and violation rows (i.e., 3).

Applying Line 6 of the algorithm to $W = \{T_1, T_2, T_3\}$, we stop generalising T_1 (as it satisfies R and further generalisation will only shorten the table). As among tables that do not satisfy R , namely T_2 and T_3 , T_3 has a larger size. Therefore, T_3 is next selected for further generalisation. Generalising T_3 on *education*, *sex*, and *hours* and obtain $T_4(0, 4)$, $T_5(2, 6)$ and $T_6(1, 4)$, respectively. Update $W = \{T_1, T_2, T_4, T_5, T_6\}$.

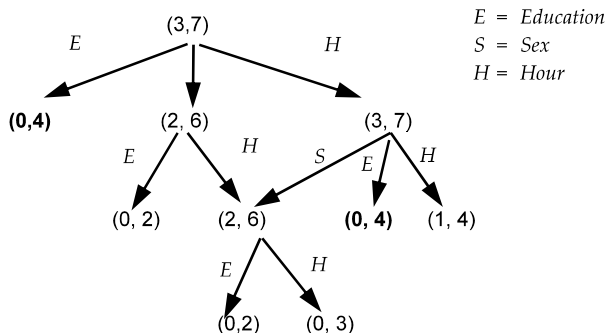
We stop generalising $T_1(0, 4)$, $T_4(0, 4)$ and $T_6(1, 4)$ (as generalising them would obtain smaller tables). Thus, T_2 is selected for generalisation. Since we cannot generalise T_2 further on *sex*, only generalisation on *education* and *hours* remain.

Here, generalising T_2 on *education* and *hours* results in T_7 and T_5 , respectively. The rest of the process continues in a similar fashion. Figure 4 only shows a partial detailed generalisation of our approach. It should be noted that tables of the same two key heuristic values may not be the same. For example, T_2 and T_5 both have two violation rows out of six rows. T_1 and T_4 are similar. Thus, the solutions may not be unique (see later on how we select a unique optimal solution).

We can represent the approach as a search tree where each node represents an ordered pair (n, m) where n and m are previously defined as in $T(n, m)$. Figure 5 shows a complete search tree resulting from our approach.

As shown in Figure 5, the search starts from the root node $(3, 7)$ of a given table T and follows the same step as illustrated in Figure 4. By generalising T on each of the three attributes, $(3, 7)$ branches to three children, namely $(0, 4)$, $(2, 6)$ and $(3, 7)$ corresponding to tables T_1 , T_2 and T_3 in Figure 4.

Figure 5 A complete search tree for optimal generalised table



As mentioned above, we stop generalising node (0, 4) as it would only shorten the table. Among (2, 6) and (3, 7), node (3, 7) is selected to be generalised on *education*, *sex* and *hours* to (0, 4), (2, 6) and (1, 4), respectively. At this point among frontier nodes (0, 4), (2, 6), (0, 4), (2, 6) and (1, 4), the first node (2, 6) is selected for further generalisation producing nodes (0, 2) and (2, 6). So far, all the leave nodes (corresponding to tables in W) include two (0, 4) nodes, (0, 2), (2, 6) and (1, 4). Because node (1, 4) represents a table of size 4, which is not larger than those of nodes (0, 4). Thus, as shown in Figure 5, both (1, 4) and (0, 4) will have no further generalisation leaving (2, 6) to be generalised to (0, 2) and (0, 3). Furthermore, tables with $n = 0$ (i.e., satisfy anonymity requirements R), has maximum $m = 4$ (i.e., non-redundant data preserved). The search stops when there is no node for further generalisation that will yield a bigger table. Thus, all the leave nodes of the tree either have $n = 0$, or non-zero n with $m < 4$ [e.g., (1, 4) as shown in Figure 5]. At this point, there are nodes with zero violations and there are no other nodes where number of rows is greater than the ones with zero violations. Because there are two optimal (0, 4) nodes (as shown in bold in Figure 5), the algorithm returns the first one generated, i.e., the one that is of level 1 [i.e., $T_1(0, 4)$ as opposed to $T_4(0, 4)$ of level 2 because its data information is closer to the original table (less generalisation steps)]. This can be easily verified that T_1 is more specific than T_4 . Thus, the algorithm returns an optimal table T_1 .

4.4 Complexity analysis

The algorithm searches for an optimal table in that it minimally generalises the given table to obtain a table with maximum size among all the tables that do not violate the anonymity requirements. It uses heuristics based on the monotonicity of generalisation to prune away some fruitless paths to improve efficiency. However, in theory in the worst case, the algorithm exhaustively finds all generalised tables in order to determine the optimal table. Our analysis gives an upper bound that may not be as tight as possible to simplify the analysis and to give an idea of the time complexity of our approach.

Let n be the number of records of a table tuple of k attributes, each of which has at most m domain values. The time it takes the algorithm to search for a solution depends on the number of generalisations, each of which depends on the structure of the corresponding taxonomy tree. For each generalisation on each attribute, the number of ways to generalise m values, from the bottom-up, is one in the best case (i.e., when all values have the same parent) and at most $m / 2$ in the worst case. This is because each generalisation has a branching factor ranging from 2 (binary tree) to m . The maximum branching factor m gives a minimum of one possible generalisation whereas the minimum branching factor two gives a maximum of $m / 2$ possible generalisations. Since there are k attributes, there are at most $km / 2$ possible generalisations. For each attribute of m values, the number of generalisation steps is $m - 1$ in the worst case (e.g., from the top of the binary tree with m leaf nodes). Since there are k attributes, the longest generalised path is at most $k(m - 1)$. Thus, each data entry value in each record requires $O(bd)$ time for all possible generalisation, where $b = km / 2$ and $d = k(m - 1)$. Since there are n records, the algorithm takes $O(nb^d)$ (i.e., $O(n(km / 2)^{k(m-1)})$).

5 Analytical and experimental comparisons

In this section, we illustrate different approaches with examples and compare the results on anonymity, data preservation and data utility for classification modelling. We consider two comparison sections. On the first section, we do the comparison with techniques that aims for optimality with different meanings, *MinGen* (Sweeney, 2002b) and *AllMin* (Samarati, 2001). On the second one, we compare our method with two anonymisation techniques that are for classification, *top-down specialisation (TDS)* (Fung et al., 2005) and *bottom-up generalisation (BUG)* (Wang et al., 2004).

5.1 Anonymisation for optimality

This section analyses the three ‘optimal’ approaches including Sweeney’s (2002b) *MinGen*, Samarati’s (2001) *AllMin* and ours, *HeuristicMin*. All achieve anonymity by complying with given anonymity requirements. All find the ‘optimal’ solution by searching for a generalised table with minimal generalisations. However, it is not clear how each of these solutions and approaches differ. By observing details of these approaches on a specific example, we aim to gain understanding of their differences and the real meaning of optimality. In Section 4, we show how *HeuristicMin* works and obtain its optimal generalised table. We next apply *MinGen* and *AllMin* to the same example in details before we provide comparison analysis.

5.1.1 *MinGen* approach

MinGen uses generalisation to produce a table that satisfy k -anonymity with minimal distortion. The generalisation is performed at a *cell* level. *MinGen* exhaustively searches on each cell for generalised tables with minimum number of generalisations that comply with k -anonymity degree requirement (or k -minimal generalisations). In our example, $k = 4$. We now apply *MinGen*, to table T of Figure 4. For convenience, we display T again in Figure 6(a). Rows 3, 6 and 7 violate anonymity requirements R . There are numerous ways to generalise T cell-wise. For example, for each record in each row, there are 6 ($3 \times 1 \times 2$) generalisations based on a taxonomy height of each attribute. Thus, table T has a total of 240 possible generalisations.

To produce a generalised table that complies with R , one option is to generalise T on *education* cell of rows 2 and 3 allowing them to be merged to one row of six records. Thus, this eliminates row 3 violation. Similarly, generalising T on *education* cell of rows 4 and 6 to be merged as well as those of rows 5 and 7 produces table T_8 as shown in Figure 6(b). Overall, a total number of generalisations at cell level required to generalise T_8 is 20 (i.e., $1 \times 6 + 1 \times 6 + 1 \times 8$ for generalising on *education* of rows 2 and 3, 4 and 6, and 5 and 7, respectively). Another option is to generalise *education* cell of rows 2 and 3 to be merged as in T_8 but also generalising two levels on *hours* cell of rows 6 and 7. This produces a generalised table T_9 , as shown in Figure 6(c), which requires 14 cell level generalisations. Similarly, we can obtain T_{10} , as shown in Figure 6(c), which also requires 14 cell level generalisations. Both T_9 and T_{10} require minimal generalisations at cell level of 14 in order to satisfy the four-anonymity requirement R .

Figure 6 Sweeney’s *MinGen* approach, (a) an initial table T (b) generalised T to T_8 (c) minimal generalised T to T_9 (d) minimal generalised T to T_{10} (see online version for colours)

Row	Edu	Sex	Hrs	#Rec
1	10 th	M	40	20
2	10 th	M	30	4
3	9 th	M	30	2
4	9 th	F	30	4
5	9 th	F	40	6
6	12 th	F	30	2
7	12 th	F	40	2

(a)

Row	Old	Edu	Sex	Hrs	#Rec
1	1	10 th	M	40	20
2	2, 3	High	M	30	6
3	4, 6	High	F	30	6
4	5, 7	High	F	40	8

(b)

Row	Old	Edu	Sex	Hrs	#Rec
1	1	10 th	M	40	20
2	2, 3	High	M	30	6
3	4	9 th	F	30	4
4	5	9 th	F	40	6
5	6, 7	12 th	F	Any	4

(c)

Row	Old	Edu	Sex	Hrs	#Rec
1	1	10 th	M	40	20
2	2	10 th	M	30	4
3	3, 4	9 th	Any	30	6
4	5	9 th	F	40	6
5	6, 7	12 th	F	Any	4

(d)

As we can see, non-unique minimal generalised tables that satisfy R may exist (e.g., T_9 and T_{10}). Moreover, such tables do not necessary contain the same in-formation and thus, have different degree of distortion. Intuitively, the cell’s distortion depends on a proportion of generalised data. Comparing row 2 of T_9 and row 3 of T_{10} , the proportion of generalised data on *education* of the former is about 1/3 (1 out of 3 possible generalisation steps) whereas that on *sex* of the latter is about 1/1. Thus, T_9 is less distorted and should be preferred. *MinGen* uses the preference metric *Prec*, to measure a *precision* of a table, where a ratio of each cell’s generalisation level over the total number of levels (height) in a corresponding taxonomy tree is used to represent the cell’s distortion. The less distortion the table is, the more ‘precise’ the table becomes in terms of its closeness of the generalised data to its original. *Prec* of a generalised table is defined by subtracting all normalised cell distortions from one. Specifically, let T be a generalised table with number of attributes n , and number of records m . For each cell t_{ij} of attribute i and instance j , let $l(t_{ij})$ and $h(t_{ij})$ be its corresponding taxonomy level and height (i.e., taxonomy tree of attribute i). We then define *Prec* (or P) as follows:

$$P = 1 - \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m \frac{l(t_{ij})}{h(t_{ij})}$$

Applying the formula to T_9 , the normalised sum of cell distortions is 0.05 [i.e., $1/20$ ($6 \times 1/3 + 4 \times 2/2$), where the first amount in the sum indicates distortion on *education* and the second indicates distortion on *hours* for all relevant cells]. This gives a precision of T_9 to be 0.95. Similarly, we obtain T_{10} ’s precision to be 0.92. Thus, by the preference criterion, *MinGen* returns T_9 as an optimal generalised table.

5.1.2 AllMin approach

Samarati’s *AllMin* uses a binary search indexing by a total number of generalisations Gen, which is a sum of a taxonomy level of each generalised data on each shield attributes. In our example, since there are three attributes, search state $[i, j, k]$ represents a taxonomy level i, j and k on attribute *education*, *sex* and *hours*, respectively. Recall each of the given three taxonomy trees in Figure 3 has height 3, 1, and 2 on *education*, *sex* and

hours, respectively. Thus, a state with maximum generalisations is [3, 1, 2], where *Gen* is 6, and an initial table has an initial state [0, 0, 0]. *AllMin* generalises by column.

Based on binary search, *AllMin* starts looking for anonymity compliance generalised tables with minimal *Gen* at level [6/2], since *Gen* of [3, 1, 2] is 6. There are six generalised tables at level 3, namely [2, 1, 0], [1, 1, 1], [1, 0, 2], [2, 0, 1], [3, 0, 0] and [0, 1, 2]. As shown in Figure 7, all of them satisfy *R*. *AllMin* moves to a more specific level (i.e., level 2) and repeats checking all generalised tables in this level, namely [1, 1, 0], [1, 0, 1], [0, 1, 1], [2, 0, 0] and [0, 0, 2] in a similar manner.

Figure 7 *AllMin*'s generalised tables corresponding to states in level 3 (see online version for colours)

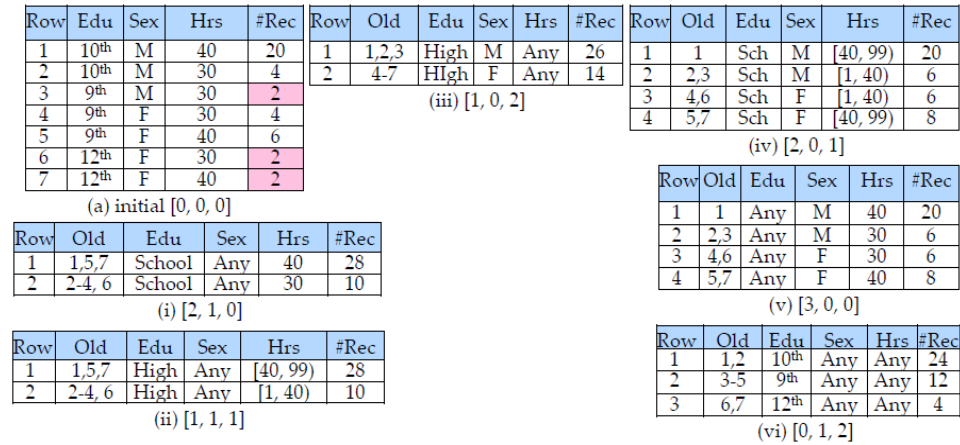


Figure 8 *AllMin*'s generalisation states at all levels (see online version for colours)

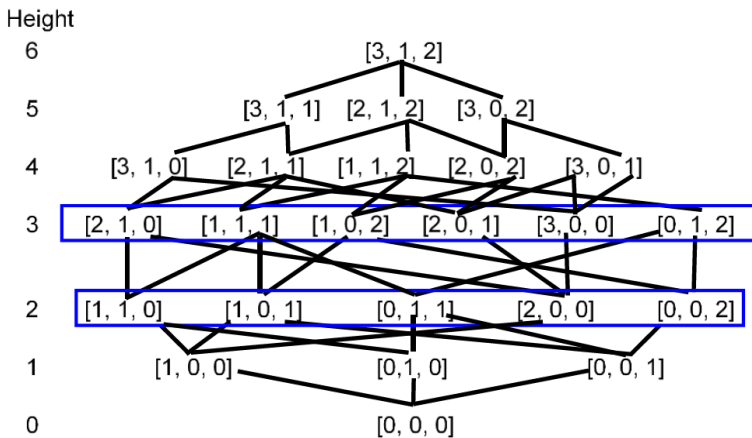


Figure 8 shows the two levels of generalised tables that *AllMin* has evaluated so far. As a result, [0, 1, 1] and [0, 0, 2] do not satisfy *R* but the rest do. Thus, candidates for generalised tables with minimal *Gen* at level 2 are [1, 1, 0], [1, 0, 1] and [2, 0, 0]. If none of the specialised (i.e., a reverse of generalised) tables (in level 1) of these three candidates satisfies *R* then they are indeed the optimal solutions. As seen from

Figure 8, these relevant generalised tables correspond to states in level 1. Thus, *AllMin* checks if:

- 1 [1, 0, 0] (specialised table of [1, 1, 0], [1, 0, 1] and [2, 0, 0])
- 2 [0, 1, 0] (specialised table of [1, 1, 0])
- 3 [0, 0, 1] (specialised table of [1, 0, 1]) satisfies *R*.

The results shown in Figures 9(a)–9(c) indicate that only [1, 0, 0] satisfies *R*. Since a lower level state [0, 0, 0] corresponds to an initial table, which does not satisfy *R*, therefore the generalised table corresponding to [1, 0, 0] as shown in Figure 9(a) is an optimal generalised table produced by *AllMin*.

Figure 9 *AllMin*'s generalisations in level 1, (a) [1, 0, 0] (b) [0, 1, 0] (c) [0, 0, 1] (see online version for colours)

Row	Edu	Sex	Hrs	#Rec
1	High	M	40	20
2	High	M	30	6
3	High	F	30	6
4	High	F	40	8

(a)

Row	Edu	Sex	Hrs	#Rec
1	10 th	Any	40	20
2	10 th	Any	30	4
3	9 th	Any	30	6
5	9 th	Any	40	6
6	12 th	Any	30	2
7	12 th	Any	40	2

(b)

Row	Edu	Sex	Hrs	#Rec
1	10 th	M	(40, 99)	20
2	10 th	M	[1, 40)	4
3	9 th	M	[1, 40)	2
4	9 th	F	[1, 40)	4
5	9 th	F	(40, 99)	6
6	12 th	F	[1, 40)	2
7	12 th	F	(40, 99)	2

(c)

5.1.3 Comparative analysis

Figure 10 summarises our findings from this example. As shown in Figure 10(a), *MinGen*'s resulting table [i.e., T_9 of Figure 6(c)] clearly retains the data closer to the original than that of *AllMin*'s and ours, which, as shown in Figure 10(b), happens to be the same [i.e., $T_1(0, 4)$ in Figure 4 and the table of state [1, 0, 0] in Figure 9(a)]. The observation is verified by the precisions 0.95 and 0.89 of the two tables as shown in Figure 10(c). This is to be expected since *MinGen* aims to find a table with minimal generalisations to satisfy anonymity requirements *R* with minimal distortion. On the size of the resulting tables, *MinGen* produces a slightly larger table than that of the rest. This is reasonable since both *HeuristicMin* and *AllMin* apply generalisation on a larger grain size than *MinGen* (i.e., rows on a certain attribute column as opposed to record cells). Thus, the minimum number of generalisations obtained by *MinGen* (i.e., 14) is double of that of *AllMin* and *HeuristicMin* (i.e., 7 from generalising seven rows of the initial table *T* on education). Because of this generalisation gran size, the solution space of 240 possible generalisations of *MinGen* is significantly higher than 24 [i.e., sum of all states in Figure 8] of the other two approaches. The main difference between *HeuristicMin* and *AllMin* is the optimal criteria and the effort in searching for optimal solution (due to heuristics that help pruning in *HeuristicMin*).

Figure 10 Summary of results on generalising table T by all three methods, (a) *MinGen* (b) ours and *AllMin* (c) comparisons on this specific example (see online version for colours)

Edu	Sex	Hrs	#Rec
10 th	M	40	20
High	M	30	6
9 th	F	30	4
9 th	F	40	6
12 th	F	Any	4

(a)

Edu	Sex	Hrs	#Rec
High	M	40	20
High	M	30	6
High	F	30	6
High	F	40	8

(b)

Characteristics	<i>MinGen</i>	<i>AllMin</i>	<i>Heuristic Min (Ours)</i>
Precision of the resulting table	0.95	0.89	0.89
No. of rows in the resulting table	5	4	4
No. of generalisations in the resulting table	14	7	7
Generalisation by	Cell	Column	Attribute
No. of generalisations to find solutions	Trial/Error	14	9
Size of solution space (all generalisations)	240	24	24

(c)

As a result, in this example, the number of generalisations to find solutions is 9 for *HeuristicMin* (see all states generated in Figure 5) and 14 for *AllMin* (see a total number of states in levels 1–3 in Figure 8).

To summarise, *MinGen*'s optimality means a minimum number of *cell-wise* generalisations that satisfy anonymity requirement and minimise data distortion (i.e., changes of original data).

AllMin's optimality means a minimum number of *column-wise* generalisations that satisfy anonymity requirement. *HeuristicMin*'s optimality means a minimum number of *attribute-wise* generalisations that satisfy anonymity requirement and maximise non-redundant data information (i.e., number of rows).

On solution criteria, *HeuristicMin* is similar to *MinGen* in that their optimal solutions must meet two criteria:

- 1 satisfying anonymity requirements with minimal generalisations
- 2 maximise data preservation of the given table.

On the other hand, *AllMin*'s optimal solution only needs to meet only the first criterion. On generalisation, *MinGen* generalises by cells, *AllMin* by column and *HeuristicMin* by attribute. Thus, *MinGen*'s generalisation is the finest grain size, while *AllMin*'s is the coarsest and *HeuristicMin*'s compromises between the two. Because of this, *MinGen* produces the optimal table with best precision but at the same time it has a large search space. Therefore, *MinGen* serves as a good formal approach but not feasibly practical (Sweeney, 2002b). To remedy this, Sweeney (1998) has developed *Datafly*, a heuristic approach that is computationally feasible. However, because it does not guarantee optimal generalisations, we exclude it from our comparison.

Although the results obtained from *AllMin* and *HeuristicMin* are the same in this example, it is not necessary the same in general. Both of their generalisations are based on the attribute column but *HeuristicMin*'s generalisation applies to certain table rows (i.e., those contain attribute values violating anonymity requirements and its sibling), which, in the worst case, could be all rows as in *AllMin*. In searching for optimal solutions, *AllMin* exploits a binary search, which is a blind search, whereas *HeuristicMin* uses heuristics on the number of rows violating the anonymity requirements and the table size. By using the monotonicity property of generalisation, *HeuristicMin* is an admissible A^* search, giving an optimal solution (Russell and Norvig, 2010). Note that the search

may not find a unique optimal solution. In such a case, *MinGen* and *HeuristicMin* apply a preference metric to select a unique solution, while *AllMin* does not but gives possible criteria. Finally, both *MinGen* and *AllMin* include *suppression* (not releasing or masking) in generalisation. For example, generalising the *zip* string 12345 to 1234* has the same effect as masking the last *zip* digit value. *HeuristicMin* could do the same.

These comparisons show that our *HeuristicMin* approach is simple and effective in that it produces a practical and useful optimal result comparable to those of other methods that likely have higher computational cost or not realistically realisable.

5.2 Anonymisations for classification

Several existing data privacy approaches aim to find a generalised data table that not only satisfies a given anonymity requirements to mask sensitive information but also preserves data useful for modelling classification. *TDS* (Fung et al., 2005) and *BUG* (Wang et al., 2004) are designed for this purpose. Specifically, the resulting data tables aim to maximise accuracy of the classification models. This section compares *TDS*, *BUG* and our *HeuristicMin* (even though *HeuristicMin* is not designed for classification purpose).

Starting from the most general table, *TDS* searches for its table solutions by repeatedly performing appropriate specialisations until no further applicable specialisation exists. That is, it either violates the anonymity requirements or does not increase useful information for classification [or *information gain* (MacKay, 2003; Mitchell, 1997; Russell and Norvig, 2010)]. *TDS* selects an attribute with a maximum metric score to be specialised using its corresponding taxonomy tree. The scoring metric is defined as a ratio of *information gain* to *anonymity loss*.

In our context, information gain from a given attribute tells us how well specialisation of the attribute helps us gain better classification. The classification quality depends on the ‘mix’ of classes, which is measured by *entropy* (MacKay, 2003; Russell and Norvig, 2010). The less mix the classes are (e.g., frequency of one class dominates the rest), the easier it is to classify because the separation of classes are more distinct. Thus, information gain uses entropy to measure the mix of classes before and after specialisation. Good specialisation increases information gain and reduces the mix of class. Note that specialisation on an attribute that involves records of one single class will not increase information gain and thus, such specialisation is inapplicable.

TDS’s anonymity loss on a given attribute *A* is defined as an average of anonymity loss before and after specialisation on *A* overall requirements containing *A*. The anonymity loss on *A* for each requirement is measured from the largest difference (i.e., the worst case) of the number of records before and after specialisation. In other words, we compare the record number of the parent node with that of the child node with minimum number of records. If the difference is high, then the anonymity loss is also high, but the score will be low. In sum, the metric score is high when information gain is high (to increase classification quality) and/or anonymity loss is low (i.e., low distance to satisfy anonymity requirements) (see more details in Fung et al., 2005).

BUG addresses the same problem as *TDS* but searches for the solution in a bottom-up manner. Being developed by the same researchers as those of *TDS*, *BUG* uses a similar scoring metric. Since *BUG* applies generalisation rather than specialisation, *BUG*’s score is a reverse of *TDS*’s score, which is a ratio of information loss to anonymity gain. Here, we apply generalisation on the attribute with a minimum score (i.e., either low information loss or high anonymity gain) (see more details in Wang et al., 2004).

Next we compare our approach, *HeuristicMin*, on four examples with *TDS* and for the sake of simplicity, compare all three approaches only on one of those four examples. To evaluate the results on classification modelling we use naive Bayesian classifier with ten-fold cross-validation on the resultant tables of the four examples. On the comparison tables, we consider the number of distinct rows and classification accuracy.

Figure 11 Comparison of *TDS* and *HeuristicMin* on classification Case 1, (a) initial table (b) *HeuristicMin*'s result (c) *TDS*'s result (d) comparisons (see online version for colours)

Row	Edu	Sex	WorkHrs	Class	#Rec
1	10 th	M	40	A	20
2	10 th	M	30	B	4
3	9 th	M	30	B	2
4	9 th	F	30	B	4
5	9 th	F	40	B	6
6	8 th	F	30	B	2
7	8 th	F	40	B	2

(a)

Row	Edu	Sex	WorkHrs	Class	#Rec
1	Any	M	[40, 99]	A	20
2	Any	M	[1, 40]	B	6
3	Any	F	[1, 40]	B	6
4	Any	F	[40, 99]	B	8

(c)

Row	Edu	Sex	WorkHrs	Class	#Rec
1	High	M	40	A	20
2	High	M	30	B	6
3	High	F	30	B	6
4	High	F	40	B	8

(b)

	# Rows Input	# Rows Result	% Acc
<i>HM</i>	7	4	100%
<i>TDS</i>	7	4	100%

(d)

5.2.1 Case 1: *HeuristicMin* vs. *TDS* on single requirements

Consider an illustration in Section 4.3, where an initial table is shown again here in Figure 11(a) with the anonymity requirements set R as $\{<\{education, sex, hours\}, 4>\}$. The results obtained from our *HeuristicMin* (*HM*) and *TDS* is shown in Figures 11(b) and 11(c), respectively.

Figure 11(d) gives a summary of comparisons between a specialised table obtained from *TDS* and a generalised table obtained from *HM*. Both algorithms preserved the same number of rows and have the same result for classification. However, *HeuristicMin* produces a table with data closer to the original than the data of *TDS* [e.g., original *work hours* of 40 is generalised to [40, 99] and *education* attribute value high is closer to the original than any]. Thus, on the criterion of informativeness, *HeuristicMin* performs better than *TDS* in this case.

5.2.2 Case 2: *HeuristicMin* vs. *TDS* on two requirements

Consider an example given in Fung et al. (2005), where the taxonomy trees are shown in Figure 12 and an initial table is shown in Figure 13(a). Some rows have multiple classes. For example, for row 3, a tuple (11th, M, 35) has five records, 2 of which are of class *A* and 3 are of class *B*. In this example, a set of anonymity requirements $R = \{R_1, R_2\}$, where $R_1 = <\{education, sex\}, 4>$, and $R_2 = <\{sex, WorkHrs\}, 11>$. The data projected on the shield attributes of R_1 and R_2 are surrounded by the two boxes shown separately in Figure 13(a) for R_1 and Figure 13(b) for R_2 . Recall that we write $[t, r_t]$ for a pair of a tuple t and its corresponding number of records r_t . Thus, we can represent R_1 -projected tuple in row 1 as $[(9th, M), 3]$.

Figure 12 Taxonomy trees of Case 2

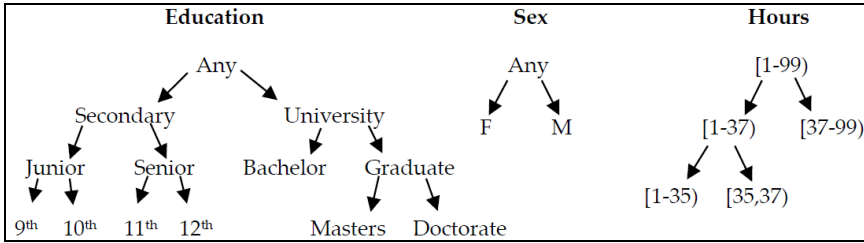


Figure 13 Comparison of *TDS* and *HeuristicMin* on classification Case 2, (a) initial table showing R_1 violation (b) initial table showing R_2 violation (see online version for colours)

Row	Edu	Sex	WorkHrs	Class	#Rec
1	9 th	M	30	B	3
2	10 th	M	32	B	4
3	11 th	M	35	A, B	2, 3
4	12 th	F	37	A, B	3, 1
5	Bac	F	42	A, B	4, 2
6	Bac	F	44	A	4
7	Mas	M	44	A	4
8	Mas	F	44	A	3
9	Doc	F	44	A	1

(a)

Row	Edu	Sex	WorkHrs	Class	#Rec
1	9 th	M	30	B	3
2	10 th	M	32	B	4
3	11 th	M	35	A, B	2, 3
4	12 th	F	37	A, B	3, 1
5	Bac	F	42	A, B	4, 2
6	Bac	F	44	A	4
7	Mas	M	44	A	4
8	Mas	F	44	A	3
9	Doc	F	44	A	1

(b)

On anonymity requirements R_1 with shield $\{education, sex\}$, rows 1, 8 and 9 violate R_1 . On the other hand, for R_2 , we have $[(F, 44), 8]$ from rows 6, 8 and 9. Each of the rest of the rows represents a unique tuple that violates R_2 's anonymity requirements of 11. Thus, none of the rows satisfy R_2 .

Figure 14 shows an execution of specialisations in *TDS*. Starting from the most general table shown in Figure 14(a), *TDS* computes scores of each attribute and selects the attribute with a maximum score (i.e., *WorkHrs*, in this case) to specialise. Using the taxonomy tree of *work* in Figure 12, *TDS* replaces [1-99] by its children: [1-37] and [37-99] and uses the initial table to count the corresponding records resulting in the table shown in Figure 14(b). The same process repeats. Each specialisation generates a more detailed table that does not violate the anonymity requirements R . As shown in Figure 14, *TDS* next continues specialising on *education*, *secondary*, *senior* and *university* and obtains tables shown in Figures 14(c)-14(d), respectively.

Figure 14(f) shows the specialisations of the tables produced by *TDS*. Further specialising *junior* in Figure 14(f), will split its first row into rows 1 and 2 of Figure 14(g), where $[(9^{th}, any), 3]$, in row 1, violates R_1 . Further specialising *sex* in Figure 14(g), will split its last row into row 7 and rows 8-9 of Figure 14(h), where we have $[(M, [1-37]), 12]$ and $[(F, [37-99]), 18]$ satisfying R_2 but $[(M, [37-99]), 4]$, in row 7, violate R_2 . Further specialising [1-37] in Figure 14(f), will result in Figure 14(i), where $[(any, [1-35]), 7]$ and $[(any, [35-37]), 5]$, in its first two rows, violate R_2 . Similarly, further specialising *graduate* will result in R_1 violation.

Figure 14 TDS’s specialisations to its solution, (a) initial TDS table (b) specialise on *WorkHrs* (c) specialise on *education* (d) specialise on *secondary* (e) specialise on *senior* (f) specialise on *university* (g) specialise on *junior* (h) specialise on *sex* (i) specialise on [1–37] (see online version for colours)

Rows	Edu	Sex	Hrs	#Rec
1-9	Any	Any	[1-99]	34

(a)

Rows	Edu	Sex	Hrs	#Rec
1-3	Any	Any	[1-37]	12
4-9	Any	Any	[37-99]	22

(b)

Rows	Edu	Sex	Hrs	#Rec
1-3	Sec	Any	[1-37]	12
4	Sec	Any	[37-99]	4
5-9	Uni	Any	[37-99]	18

(c)

Rows	Edu	Sex	Hrs	#Rec
1-2	Jr	Any	[1-37]	7
3	11 th	Any	[1-37]	5
4	12 th	Any	[37-99]	4
5-6	Bac	Any	[37-99]	10
7-9	G	Any	[37-99]	8

(d)

Rows	Edu	Sex	Hrs	#Rec
1-2	Jr	Any	[1-37]	7
3	11 th	Any	[1-37]	5
4	12 th	Any	[37-99]	4
5-6	Bac	Any	[37-99]	10
7-9	G	Any	[37-99]	8

(e)

Rows	Edu	Sex	Hrs	#Rec
1-2	Jr	M	[1-37]	7
3	11 th	M	[1-37]	5
4	12 th	F	[37-99]	4
5-6	Bac	F	[37-99]	10
7	G	M	[37-99]	4
8-9	G	F	[37-99]	4

(f)

Rows	Edu	Sex	Hrs	#Rec
1-2	Jr	Any	[1-35]	7
3	11 th	Any	[35-37]	5
4	12 th	Any	[37-99]	4
5-6	Bac	Any	[37-99]	10
7-9	G	Any	[37-99]	8

(g)

Rows	Edu	Sex	Hrs	#Rec
1	9 th	Any	[1-37]	3
2	10 th	Any	[1-37]	4
3	11 th	Any	[1-37]	5
4	12 th	Any	[37-99]	4
5-6	Bac	Any	[37-99]	10
7-9	G	Any	[37-99]	8

(h)

Rows	Edu	Sex	Hrs	#Rec
1-2	Jr	Any	[1-37]	7
3	11 th	Any	[37-99]	5
4	12 th	Any	[37-99]	4
5-6	Bac	Any	[37-99]	10
7-9	G	Any	[37-99]	8

(i)

Figure 15 HeuristicMin’s generalisations toward its optimal solution, (a) initial table (b) generalise on education, R_1 is satisfied but not R_2 (c) generalise on *WorkHrs*, R_1 is satisfied but not R_2 (d) generalise on *Sex*, R_1 is satisfied but not R_2 (see online version for colours)

Row	Edu	Sex	Hrs	Class	#Rec
1	9 th	M	30	B	3
2	10 th	M	32	B	4
3	11 th	M	35	A, B	2, 3
4	12 th	F	37	A, B	3, 1
5	Bac	F	42	A, B	4, 2
6	Bac	F	44	A	4
7	Mas	M	44	A	4
8	Mas	F	44	A	3
9	Doc	F	44	A	1

(a)

Row	Edu	Sex	Hrs	Class	#Rec
1	Jr	M	30	B	3
2	Jr	M	32	B	4
3	11 th	M	35	A, B	2, 3
4	12 th	F	37	A, B	3, 1
5	Bac	F	42	A, B	4, 2
6	Bac	F	44	A	4
7	G	M	44	A	4
8	G	F	44	A	3
9	G	F	44	A	1

(b)

Row	Edu	Sex	Hrs	Class	#Rec
1	Jr	M	[1, 35]	B	3
2	Jr	M	[1, 35]	B	4
3	11 th	M	[35, 37]	A, B	2, 3
4	12 th	F	[37, 99]	A, B	3, 1
5	Bac	F	[37, 99]	A, B	4, 2
6	Bac	F	[37, 99]	A	4
7	G	M	[37, 99]	A	4
8	G	F	[37, 99]	A	3
9	G	F	[37, 99]	A	1

(c)

Row	Edu	Sex	Hrs	Class	#Rec
1	Jr	Any	[1, 35]	B	3
2	Jr	Any	[1, 35]	B	4
3	11 th	Any	[35, 37]	A, B	2, 3
4	12 th	Any	[37, 99]	A, B	3, 1
5	Bac	Any	[37, 99]	A, B	4, 2
6	Bac	Any	[37, 99]	A	4
7	G	Any	[37, 99]	A	4
8	G	Any	[37, 99]	A	3
9	G	Any	[37, 99]	A	1

(d)

Thus, considering all possible specialisations (i.e., specialisation that has not reached the leaves of the taxonomy trees) of Figure 14(f), none results in a table that satisfies anonymity requirements R . TDS terminates with Figure 14(f) as a solution.

Applying $HeuristicMin$ to the initial table, Figure 15 shows resulting tables and violations obtained by each generalisation. For example, generalising on *education* of violating rows 1, 8, 9 results in replacement of *education* data values of rows 1–2 by *junior (Jr)* and those of rows 7–9 by *graduate (G)* as shown in Figure 15(b). The generalisation continues as shown in Figures 15(c) and 15(d), where generalising table in Figure 15(d) on rows 1–3 on *WorkHrs* results in an optimal table that complies with anonymity requirements as shown in Figure 16(a).

Using the resulting table for classification, of course, gives the same result as shown in Figure 16(b). In Case 2, the accuracy obtained is 80% [Figure 16(b)].

Figure 16 Resulting solution of (a) TDS and $HeuristicMin$ and (b) comparison (see online version for colours)

Row	Edu	Sex	WorkHrs	Class	#Rec
1	Jr	Any	[1-37)	B	7
2	11 th	Any	[1-37)	A, B	2, 3
3	12 th	Any	[37-99)	A, B	3, 1
4	Bac	Any	[37-99)	A, B	8, 2
5	Grad	Any	[37-99)	A	8

(a)

	# Rows Input	# Rows Result	% Acc
HM	9	5	80%
TDS	9	5	80%

(b)

So far, the classification results obtained from solution tables from TDS and $HeuristicMin$ in both Cases 1 and 2 have the same accuracy even though the data tables are different (in Case 1) or the same (in Case 2). In all, even $HeuristicMin$ is not designed for classification; it produces the same classification results.

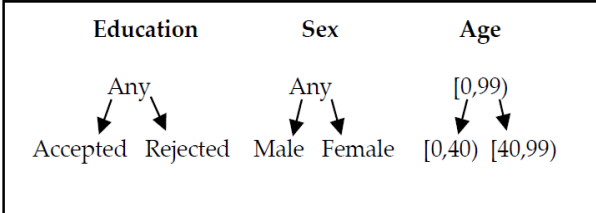
5.2.3 Case 3: $HeuristicMin$ vs. TDS and BUG

Consider an initial table as given in Figure 18(a) where its corresponding taxonomy trees are shown in Figure 18(b). The data table represents the Walmart cashier job applications, where class A refers to applicants with references from previous jobs and class B , otherwise. In this scenario, the anonymity requirements $R = \langle \{application, sex\}, 25 \rangle$.

Figure 17 Given inputs of Case 3, (a) initial table (b) taxonomy trees of the three attributes (see online version for colours)

Row	App	Sex	Age	Class	#Rec
1	Acc	F	30	A	5
2	Acc	F	20	A	16
3	Acc	F	40	A	3
4	Acc	M	25	B	4
5	Acc	M	30	B	5
6	Rej	M	30	B	10
7	Rej	F	30	A	15
8	Rej	F	50	A	5
9	Rej	F	20	A	5
10	Rej	M	70	B	15

(a)



(b)

As shown in the initial table of Figure 17(a), we use *Acc* (for *accepted*) and *Rej* (for *rejected*) for attribute values of *application*. Here, we have [(*Rej*, *M*), 25] (rows 6, 10), and [(*Rej*, *F*), 25] (rows 7–9) that satisfy anonymity *R*. However, [(*Acc*, *F*), 24], (rows 1–3) and [(*Acc*, *M*), 9] (rows 4–5) violate *R*. Thus, the original state in *HeuristicMin* is (5, 10) (i.e., 5 out of 10 rows violate *R*).

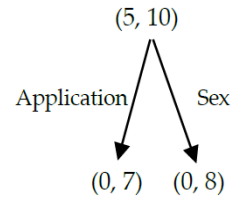
Figure 18 Results obtained from *HeuristicMin* on Case 3, (a) generalise on *application*: (0, 7) (b) generalise on *sex*: (0, 8) (c) *HeuristicMin*'s search tree (see online version for colours)

Rows	App	Sex	Age	Class	#Rec
1,7	Any	F	30	A	20
2,9	Any	F	20	A	21
3	Any	F	40	A	3
4	Any	M	25	B	4
5,6	Any	M	30	B	15
8	Any	F	50	A	5
10	Any	M	70	B	15

(a)

Rows	App	Sex	Age	Class	#Rec
1, 5	Acc	Any	30	A, B	5, 5
2	Acc	Any	20	A	16
3	Acc	Any	40	A	3
4	Acc	Any	25	B	4
6, 7	Rej	Any	30	A, B	15, 10
8	Rej	Any	50	A	5
9	Rej	Any	20	A	5
10	Rej	Any	70	B	15

(b)



(c)

Based on the violating rows, *HeuristicMin* applies generalisation to *Acc* and its sibling *Rej* on *application*. The resulting table is shown in Figure 18(a), where we have [(*any*, *F*), 49], [(*any*, *M*), 34]. Thus, it satisfies *R* and is represented by state (0, 7). Similarly, *HeuristicMin* also applies generalisation to *M* and *F* on *sex* of Figure 18(a) giving a resulting table with state (0, 8) as shown in Figure 18(b). Since there are no more violations, the algorithm stops searching [as shown in Figure 18(c)] and returns the table corresponding to node (0, 8) because its number of rows is higher than that of the table corresponding to node (0, 7). In other words, by our heuristic to select a bigger table to maximise data information, *HeuristicMin* selects Figure 18(b) as an optimal solution.

Applying *TDS* to the most general form, *TDS* calculates the score of each attribute and chooses to specialise *application* and continues specialising on *age* until specialisation yields anonymity violation. *TDS* then stops and returns the resulting table as shown in Figure 19(a). Applying *BUG* to the initial table, *BUG* uses the same scoring metric as that used by *TDS* to select the attribute to generalise. Consequently, *BUG* generalises ‘by attribute’ on *application* and the resulting table of Figure 19(b) is obtained. Figure 19(c) shows the result obtained from *HeuristicMin*, where generalisation on *sex* was applied as shown earlier in Figures 19(b) and 19(c). Figure 19(d) shows comparison results of the three approaches. Our *HeuristicMin* gives the highest output/input (i.e., O/I) ratio between the table size after and before anonymisation of 0.8, while *TDS*'s ratio is 0.4, the lowest. *HeuristicMin* preserves twice as many rows as those of *TDS*. This is to be expected because *HeuristicMin* aims to maintain the size of the table to maximise data preservation while obtaining a table that satisfies anonymity requirements. When we use the precision measure to indicate how much the anonymisation process via generalisation/specialisation manages to keep the data undistorted, both *HeuristicMin* and *BUG* have the same precision of 0.67 whereas *TDS*'s precision of 0.5 is considerably less. This is because precision measures are based on ratios of taxonomy levels. However, comparing the attribute values of resulting tables of

all the three techniques, our *HeuristicMin* keeps the table contents closest to the original table.

Figure 19 Comparison results obtained from all methods on Case 3, (a) result from *TDS* (b) result from *BUG* (c) result from *HeuristicMin* (d) comparisons (see online version for colours)

Rows	App	Sex	Age	Class	#Rec
1,2,7,9	Any	F	[0, 40)	A	41
3,8	Any	F	[40,99)	A	8
4-6	Any	M	[0, 40)	B	19
10	Any	M	[40,99)	B	15

(a)

Rows	App	Sex	Age	Class	#Rec
1, 5	Acc	Any	30	A, B	5, 5
2	Acc	Any	20	A	16
3	Acc	Any	40	A	3
4	Acc	Any	25	B	4
6, 7	Rej	Any	30	A, B	15,10
8	Rej	Any	50	A	5
9	Rej	Any	20	A	5
10	Rej	Any	70	B	15

(c)

Rows	App	Sex	Age	Class	#Rec
1, 7	Any	F	30	A	20
2, 9	Any	F	20	A	21
3	Any	F	40	A	3
4	Any	M	25	B	4
5, 6	Any	M	30	B	15
8	Any	F	50	A	5
10	Any	M	70	B	15

(b)

	O/I Row Ratio	Prec.	% Acc
<i>HM</i>	8/10	0.67	63%
<i>TDS</i>	4/10	0.5	100%
<i>BUG</i>	7/10	0.67	100%

(d)

On classification, *TDS* and *BUG* perform equally well and outperform *HeuristicMin* as to be expected as they both are designed for classification use. Looking more closely, both *TDS* and *BUG* use the scoring metric that depends on a ratio of the information loss over anonymity gain. The information loss refers to the effects of the class mixing (which effects the classification accuracy), while the anonymity gain refers to the effects of record aggregation (which increases the degree of anonymity). As shown on Figures 19(a) and 19(b), both resulting tables from *TDS* and *BUG* do not have class mixing, whereas heuristic’s solution shown in Figure 19(c) does, particularly the first and fifth rows. When choosing which generalisation to apply, *HeuristicMin* chooses to generalise on the attribute that creates a big table size, *BUG* on the other hand chooses the attribute that creates a table that does not mixes the classification labels. As a result, ours choose the attribute *sex* and *BUG* chooses *application* attributes. This explains our results that are guided by the heuristic measured applied.

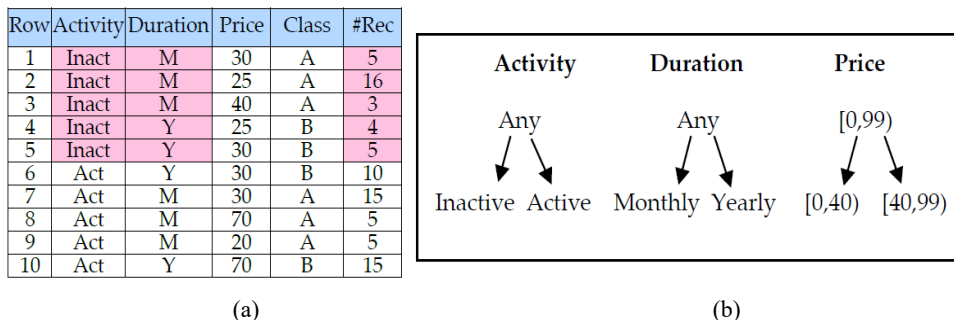
In summary of Case 3, *HeuristicMin* preserves twice as many rows as those of *TDS*. However, the result of the classification is not as good as *TDS* and *BUG* that aim to preserve data for classification. *BUG* performs as well as *TDS* in classification as expected and preserves more information than *TDS* but not as good as *HeuristicMin*. The performances on this case reflect the design and heuristics used in each of these approaches.

5.2.4 Case 4: *HeuristicMin* vs. *TDS*

Given a table representing gym data records with three attributes: *activity*, *duration* and *price*. The account of a gym member becomes inactive if he/she does not attend more

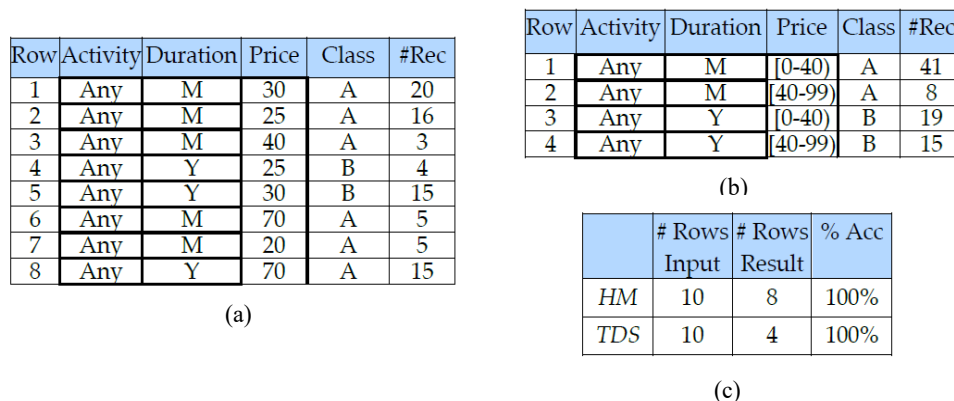
than two weeks. Gym members pay extra for special classes such as spinning or yoga. Classes include those with (class *A*) or without (class *B*) individual gym instructors. The data table and corresponding taxonomy of the attributes are shown in Figure 20. We write *Inact* and *Act* for *inactive* and *active* attribute values, respectively. Similarly, *M* is for monthly and *Y* for yearly.

Figure 20 Given inputs of Case 4, (a) initial table (b) taxonomy trees of the three attributes (see online version for colours)



Given the anonymity requirements R to be $\langle \{activity, duration\}, 25 \rangle$. It is clear that rows 1–3 give $[(Inact, M), 24]$ and rows 4–5 give $[(Inact, Y), 9]$ both of which violate R . The results obtained from *HeuristicMin*, and *TDS* are shown in Figures 21(a) and 21(b), respectively.

Figure 21 Comparison results obtained on Case 4, (a) *HeuristicMin*'s result (b) *TDS*'s result (c) comparisons (see online version for colours)



Applying *HeuristicMin* (*HM*) to the initial table, the optimal table is obtained [Figure 21(a)] by generalising on *activity* on rows 1–5 that cause the violation and rows 6–10 that contain associated sibling attribute values. Applying *TDS* to the initial table, the resulting table obtained through top-down specialising is shown in Figure 21(b). Figure 21(c) compares characteristics and accuracy obtained from the resulting tables. The result shows that our *HeuristicMin* preserves more information than *TDS* with number of rows 80% of the original while *TDS* preserves only 40% of the original. Both achieve the same classification accuracy of 100%. This comparison of

Case 4 indicates that even though *HeuristicMin*'s information preserving metric is not specifically designed for classification like *TDS*, it can still generate reasonable results to be used for classification.

In summary, our empirical experiments in all four cases in this section show that each approach performs accordingly to its design criteria and the metrics or heuristic employed. Both *TDS* and *BUG* aim to produce tables that satisfy anonymity requirements while preserving information for the best quality of the classifiers produced by the anonymised data. *HeuristicMin* aims to anonymise data while keeping data as informative and close to the original data as much as possible. The interesting observation from these experiments is that even though *HeuristicMin* is not designed for classification use, it performs just as well as *TDS* and *BUG* in 3 out of 4 cases including cases with a single anonymity requirement (Case 1 and Case 4) and two anonymity requirements (Case 2). The study of Case 3, where *HeuristicMin*'s classification accuracy is worse than those of *TDS* and *BUG*, helps us gain insight that explains reasons for distinct classification accuracy between these two groups of techniques. *TDS* (or *BUG*) favours specialisation (or generalisation) that produces a row of non-mixed classes (as measured by *information gain* in their scoring metric). This enhances the quality of data for classification. On the other hand, *HeuristicMin* does not target for classification use. As a result, its optimal table may have mixed classes that result in poor classification accuracy as in Case 3.

6 Conclusions

This paper studies privacy objectives for anonymising the identity of the individuals associated with given data. In particular, we present *HeuristicMin*, an anonymisation approach that applies generalisations to securing privacy by satisfying user-specified anonymity requirements while maximising information preservation. Unlike prior approaches, *HeuristicMin* exploits monotonicity property of generalisation along with two simple heuristics:

- 1 the number of rows violating the anonymity requirements
- 2 the table size (or number of non-redundant rows of records) to efficiently obtain optimal generalised table.

The first heuristic measures how far the generalised table being considered is from meeting the anonymity requirements whereas the second heuristic represents the amount of data retention from the original table.

HeuristicMin has several advantages. First, it accommodates anonymity compliance and maximum data information for 'general' purposes (as opposed to special purpose such as classification). Second, it uses 'simple' heuristics (compared to *MinGen*'s precision or complex *TDS*'s or *BUG*'s information gain over anonymity loss) to find the generalised table solution. Third, its optimal solution is theoretically grounded by a well-known efficient optimal A^* search. Finally, its use of generalisation by 'attribute' provides a compromising generalisation grain size between the cell level (that can be too fine to be computationally feasible in practice) and the column level (that can be too coarse to maintain close content to the original).

The paper also presents a comparison study between *HeuristicMin* and popular techniques that produce optimal solutions to give semantic interpretation of ‘optimality’ in each technique and how they differ. In addition, we compare *HeuristicMin* with techniques that aim to anonymise data for classification. The experimental results show that even though *HeuristicMin* is not designed for classification, it can still produce competitive results. We further note that generalisations or specialisations that do not produce rows with mixing classes are likely to produce high classification accuracy. To the best of our knowledge, there has been no attempt to compare approaches to anonymisation with and without classification.

References

- Acquisti, A. and Grossklags, J. (2005) ‘Privacy and rationality in individual decision making’, *IEEE Security & Privacy*, Vol. 3, No. 1, pp.26–33.
- Arca, S., Kijisanayothin, P. and Hewett, R. (2020) ‘Realizing data anonymization as search with optimal guarantee’, *Proceedings of Workshop on AI for Privacy (AI4P 2020) in conjunction with European AI Conference*, Santiago De Compostela, Spain.
- Bayardo, R.J. and Agrawal, R. (2005) ‘Data privacy through optimal k-anonymization’, in *Proc. of the 21st International Conference on Data Engineering (ICDE)*, Tokyo, Japan, pp.217–228.
- Clearwater, A. and Hughes, J. (2013) ‘In the beginning ... an early history of the privacy profession’, *Ohio State Law Journal*, Vol. 74, No. 6, p.897.
- Dwork, C. (2006) ‘Differential privacy’, in *Proc. ICALP’06 33rd International Conference on Automata, Languages and Programming*, Vol. Part 2, pp.1–12.
- Fung, B.C.M., Wang, K. and Yu, P.S. (2005) ‘Top-down specialization for information and privacy preservation’, in *Proceedings of the 21st International Conference on Data Engineering*, IEEE Computer Society, pp.205–216.
- Fung, B.C.M., Wang, K. and Yu, P.S. (2007) ‘Anonymizing classification data for privacy preservation’, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 5, pp.711–725.
- Fung, B.C.M., Wang, K., Wang, Y. and Hung, K. (2009) ‘Privacy-preserving data publishing for cluster analysis’, *Data & Knowledge Engineering*, Vol. 68, No. 6, pp.552–575.
- Hundepool, A. and Willenborg, L. (1996) ‘ μ - and τ -argus software for statistical disclosure control’, *Third International Seminar on Statistical Confidentiality*.
- Iyengar, V.S. (2002) ‘Transforming data to satisfy privacy constraints’, in *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, AB, Canada, pp.279–288.
- LeFevre, K., DeWitt, D.J. and Ramakrishnan, R. (2006) ‘Mondrian multidimensional k-anonymity’, *ICDE*, Vol. 6.
- LeFevre, K., DeWitt, D.K. and Ramakrishnan, R. (2005) ‘Incognito: efficient fulldomain k-anonymity’, *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ACM.
- Li, N., Li, T. and Venkatasubramanian, S. (2007) ‘T-closeness: privacy beyond k-anonymity and l-diversity’, in *Proc. 23rd International Conference on Data Engineering*.
- Machanavajjhala, A., Gehrke, J., Kifer, D. and Venkita-Subramaniam, M. (2006) ‘l-diversity: privacy beyond k-anonymity’, in *Proc. 22nd International Conference on Data Engineering (ICDE)*, pp.24–75.
- MacKay, D.J.C. (2003) *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, Cambridge.

- Meyerson, A. and Williams, R. (2004) 'On the complexity of optimal k-anonymity', in *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ACM, pp.223–228.
- Mitchell, T.M. (1997) *Machine Learning*, McGraw Hill, New York.
- Russell, S. and Norvig, P. (2010) *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey.
- Samarati, P. (2001) 'Protecting respondents' identities in microdata release', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13, No. 6, pp.1010–1027.
- Sweeney, L. (1998) 'Datafly: a system for providing anonymity in medical data', in *Database Security XI: Status and Prospects, IFIP TC11 WG11.3 11th Int'l. Conf. on Database Security*, pp.356–381.
- Sweeney, L. (2002a) 'K-anonymity: a model for protecting privacy', *Int. J. Uncertain. Fuzz.*, Vol. 10, No. 5, pp.557–570.
- Sweeney, L. (2002b) 'Achieving k-anonymity privacy protection using generalization and suppression', *Int'l. Journal on Uncertainty, Fuzziness, and Knowledge-base Systems*, Vol. 10, No. 5, pp.571–588.
- Wang, K., Yu, P.S. and Chakraborty, S. (2004) 'Bottom-up generalization: a datamining solution to privacy protection', in *Proc. 4th 22nd International Conference on Data Mining*.