Data Min Knowl Disc (2008) 16:165–196
DOI 10.1007/s10618-007-0078-6

A recursive search algorithm for statistical disclosure assessment

Anna M. Manning · David J. Haglin · John A. Keane

Received: 27 March 2006 / Accepted: 6 June 2007 / Published online: 10 July 2007
Springer Science+Business Media, LLC 2007

Abstract A new algorithm, SUDA2, is presented which finds minimally unique itemsets i.e., minimal itemsets of frequency one. These itemsets, referred to as Minimal Sample Uniques (MSUs), are important for statistical agencies who wish to estimate the risk of disclosure of their datasets. SUDA2 is a recursive algorithm which uses new observations about the properties of MSUs to prune and traverse the search space. Experimental comparisons with previous work demonstrate that SUDA2 is several orders of magnitude faster, enabling datasets of significantly more columns to be addressed. The ability of SUDA2 to identify the boundaries of the search space for MSUs is clearly demonstrated.

Keywords Unique itemset · Search space · Algorithm · Recursion · Statistical disclosure

Responsible editor: Hannu Toivonen.

A. M. Manning (✉) · J. A. Keane
School of Computer Science, University of Manchester, Oxford Rd., Manchester, M13 9PL, UK
e-mail: anna@manchester.ac.uk

J. A. Keane
e-mail: john.keane@manchester.ac.uk

D. J. Haglin
Department of Computer and Information Sciences, Minnesota State University, 273 Wissink Hall, Mankato, MN 56001, USA
e-mail: david.haglin@mnsu.edu

1 Introduction

The analysis of personal data is fundamental to government policy-making and academic research. This information is often collected under guarantees of confidentiality, with one of the best known examples being a national census of population. In order for maximum benefit to be gained, the data needs to be made available to as wide a group of researchers as possible. However, this can pose a risk of disclosure has worsened as the potential for linking independent datasets has risen. This increased risk is due to the manufacture of more powerful computers, the development of more sophisticated data exploration tools and the greater access to data, particularly over the Internet. Concerns about the mounting threat to personal privacy have resulted in a call to data collectors and providers to ensure a high level of security.

The problem of preventing statistical disclosure is approached by estimating the risk of a certain individual being identified (*statistical disclosure assessment*) and then by applying *statistical disclosure control*, variously recoding, masking and perturbing the data in order to reduce the statistical disclosure risk (Skinner et al. 1994; Muralidhar and Sarathy 1999; Samarati 2001; Willenborg and de Waal 2001; Singh et al. 2003; Truta et al. 2004). The work reported here focuses on statistical disclosure assessment.

Of particular concern are records whose contents, or attributes, are not only unusual but unique and therefore have the potential to be matched directly with details (including names and addresses) from another dataset. For example, a recent study of the 2000 US census (Golle 2006) revealed that 63% of US citizens can be uniquely identified by just three pieces of information: their gender, zip code, and full date of birth. A record can contain more than one such unique pattern and its classification often depends on the number and size of such attribute sets (referred to as *uniques*) that it contains; the smaller the uniques and the greater number of them there are the more 'risky' the record is considered to be (Elliot et al. 2002; Willenborg and de Waal 1996).

Although it may be possible to anticipate some of the unique patterns in a dataset (for example, women working in non-traditional professions, unusually large households) most of these unique patterns are hidden implicitly within the data and require an exhaustive search if all of them are to be found. The ability to comprehensively locate and grade such records leads to more efficient disclosure assessment of released data but in order to carry out an exhaustive search of this nature all possible attribute sets must be checked (directly or indirectly) for uniqueness. It may be possible to concentrate on the attributes that are most likely to be problematic and therefore avoid considering all possible combinations—for example, studies have been conducted into the most likely 'scenarios of attack' by a data intruder (Elliot and Dale 1999). However, such an approach could mean that some uniques were missed (resulting in a loss of confidence in the algorithm) and could also require substantive knowledge and past experience about the dataset in question, thus restricting the algorithm's application.

A special algorithm, Special Uniques Detection Algorithm (SUDA), has been designed and implemented for this problem (Elliot et al. 2002, 2005) and is currently used by the UK Office for National Statistics (ONS). The ONS releases many data samples from very large confidential datasets, such as the Samples of Anonymized Records (SARs) (SAR1991 1993; SAR2001 2004) from the Census of Great Britain—SUDA played a key role in the production of the 2001 Individual SAR (SAR2001 2004).

SUDA, inspired by techniques used in association rule discovery (Agrawal et al. 1993), employs a two-stage approach: firstly, all uniques (up to a user-specified size) are located at record level; secondly, the size and distribution of uniques within each record are used to make statistical inferences about the records that are likely to be most 'risky' in the entire dataset (Elliot et al. 2002, 2005).

In order to streamline the search SUDA has been structured around the observation that 'Every superset of a unique attribute set is itself unique'. Only unique attribute sets without any unique subsets (i.e., *minimal uniques*) are considered in order to avoid the use of redundant information and to keep the classification process as focused as possible; the smaller the number of attributes contained in a unique pattern the more 'risky' it is considered to be Elliot et al. (2002), Willenborg and de Waal (1996). It is therefore important to know if a unique itemset is minimal. The work reported here focuses on improving the first step—that is, the search for minimal uniques within a sample or Minimal Sample Uniques (MSUs).

The search for MSUs is highly challenging. The MSUs for a given set of attributes are the same as the *minimal keys* for the same set of attributes. The problem of counting the minimal keys of a database has been shown to be #P-Hard (Theorem 9, Gunopulos et al. 2003). (It should be noted that SUDA always conducts an exhaustive search for MSUs over the whole set of attributes, whereas the search for minimal keys often considers only a subset of the attribute set.) The deleterious effect of the massive computational requirement is exacerbated by the fact that statistical disclosure control is often an iterative process, requiring a complete risk assessment as each masking technique is tested for effectiveness.

In this paper a new algorithm, SUDA2, is presented which is specifically designed to find all MSUs (up to a user-specified size) in discrete datasets. SUDA2 addresses a problem for which the current itemset mining algorithms are not easily applicable. New pruning methods have been designed which reflect the nature of MSUs, enabling the boundaries of the search space to be located effectively. The performance results of SUDA2 illustrate that it can outperform SUDA by several orders of magnitude. SUDA2 was designed with statistical assessment in mind and will therefore provide statistical offices with a significantly improved algorithm to work with. However, SUDA2 has wider application, being appropriate for any dataset that possesses the same characteristics as those described in Sect. 2.1 (or any dataset that can be adapted to match this description). The discovery of MSUs has the potential to act as a useful pre-processing tool to the related technique of k-anonymization and a faster version of SUDA2 serves only to improve this impact.

The paper is structured as follows. Section 2 describes the characteristics of the input data for SUDA2, defines notation and provides a discussion of related work; Sect. 3 outlines the original SUDA algorithm; Sect. 4 details the building blocks of the new approach; Sect. 5 presents the SUDA2 algorithm; Sect. 6 gives performance results; Sect. 7 concludes the paper and provides suggestions for further work.

2 Background

2.1 Characteristics of the input datasets for SUDA2

As with any problem in data mining, it is essential to consider the characteristics of the data at an early stage so that an appropriate solution can be reached as quickly as possible. Census-type datasets are generally made up of equal length rows. They consist of numeric attributes (such as *age* and *income*) and numerically coded categorical attributes (such as *marital status* and *occupation*). The main distinction between these types of attributes is that arithmetic operations are possible on numeric attributes whereas these are meaningless for categorical attributes. Categorical attributes are either ordinal (which take values in an ordered range of categories so that $\leq$, *max* and *min* operators can be used) or nominal (which take values in an unordered range of categories and can only be tested for equality). An example of an ordinal attribute is the strength of agreement with a statement and an example of a nominal attribute is *country of birth*. The presence of categorical attributes is an important characteristic of census-type datasets as it restricts the type of algorithms that can be applied—for example, algorithms that use distance metrics cannot be used and this is the main reason why an algorithm such as SUDA was necessary. In the following sections the assumption will be made that the input datasets to SUDA2 are made up of both integers and integer-valued categorical values. Census-type datasets are usually relatively small, often less than a gigabyte in size. The number of columns tends to be determined by the number of questions in the census and is very rarely larger than a few hundred.

2.2 Notation

Given a *dataset*, D, consisting of R rows (records) and C columns (attributes) of integer values, an *item* is a column label juxtaposed with an integer value which is valid for that column. For convenience, we label the columns $A, B, C, \ldots$ Some examples of items include: A_3, B_4 and D_8 (corresponding to $A = 3$, $B = 4$ and $D = 8$). An *itemset* is a set of items with no two items corresponding to the same column. Following this definition, $\{A_4, B_8, C_2\}$ (abbreviated to $A_4B_8C_2$), is an itemset, but $A_4B_2B_8$, is not. The number of rows in which an itemset I appears in D is called the *support* of I and is denoted by $supp(I)$. It should be noted that itemsets in association rule discovery (Agrawal et al. 1993)—where the emphasis is on *frequent itemsets* which have support no less than a given threshold — do not have column constraints, i.e., any item can appear in any itemset. A *unique itemset* is an itemset that appears in exactly one row, and an MSU is a unique itemset with no unique proper subsets.

The *size* of an itemset is the number, k, of items it contains. An MSU of size k is denoted by k-MSU. A *candidate* MSU is a unique itemset which needs to be checked for minimal uniqueness. The *Minimal Uniques Problem* is as follows: given a dataset containing C columns and R rows, find all k-MSUs, $1 \leq k \leq maxk$, where $maxk \leq C$ and is user-specified.

It is assumed that the entire dataset, and all additional data structures used for the algorithm, fit into main memory completely.

2.3 Related work

Techniques that can find *outliers* (unusual records) in datasets often rely on distance metrics and outlier detection algorithms designed for data containing categorical attributes are rare, but they do exist (Ghoting et al. 2004). However, a record classified as an outlier does not automatically contain MSUs, and the detection of outliers does not guarantee that all records containing MSUs are identified. A more focused approach is therefore required.

Some areas of statistical disclosure control research have focused on the location of 'risky' records (Elliot et al. 1998; Skinner and Elliot 2002; Fienberg and Makov 1998; Skinner and Holmes 1998) but this work, although theoretically interesting, does not overcome the exponential search space for MSUs and cannot yet be used to provide a full analysis of the risk associated with any given dataset.

Previous work in data mining has led to the development of algorithms that have been designed to protect the confidentiality of individual rows under certain conditions, for example, techniques for modifying association rule discovery (Fienberg and Slavkovic 2005) and new approaches for conducting clustering (Lin et al. 2005). However, these algorithms have not addressed the Minimal Uniques Problem directly.

The following sections aim to put the Minimal Uniques Problem into context with other areas of data mining. The first explores the relationship between frequent itemset mining in discrete data and the search for MSUs. The second considers the relationship between the Minimal Uniques Problem and k-anonymization of a dataset.

Relationship to frequent itemset mining. Considerable attention has been given to the problem of finding frequent itemsets, where a frequent itemset is one which has support no less than a given threshold. This is due to their application to the mining of association rules (Agrawal et al. 1993, 1996; Bayardo 1998; Brin et al. 1997; Han et al. 2000; Hipp et al. 2000) which is a key area in data mining.

There are important relationships between frequent itemsets and MSUs. Let $\mathcal{I}$ be a set of items and D a dataset consisting of subsets of $\mathcal{I}$ (otherwise known as *rows*). In this description, unlike that in Sect. 2.1, the length of each row is allowed to vary. In this context an *itemset* X is a set of some items of $\mathcal{I}$. The support of X is the number of rows in D that contain all items of X. An itemset is frequent if its support in D exceeds a minimum support threshold value, *minsup*.

The notion of borders of a set (Mannila and Toivonen 1997) can be stated as follows. A set $S \subseteq I$ is closed downwards if, for a $X \in S$, all subsets of X are also in S. S can be represented by its *positive border* $Bd^{+}(S)$ or its *negative border* $Bd^{-}(S)$ defined by

$$Bd^{+}(S) = max_{\subseteq}\{X \in S\}$$
$$Bd^{-}(S) = min_{\subseteq}\{Y \in I - S\}$$

A set of frequent itemsets (FI) in a dataset with respect to a given *minsup* is closed downwards. In this case, $Bd^{+}(FI)$ is often called the set of *maximal frequent itemsets* or MFIs (FIs with no frequent supersets). The negative border, $Bd^{-}(FI)$, is the set of infrequent itemsets whose subsets are all frequent. In the extreme case where $minsup = 2$, the negative border contains the set of MSUs together with infrequent itemsets with support=0 whose subsets are all frequent.

In Gunopulos et al. (2003) an algorithm was designed for finding all minimal keys based on the structure of the *Dualize and Advance* algorithm described in the same paper. (Recall that minimal keys are the same as MSUs if the whole attribute set is used.) This algorithm relies on the ability to find the minimal transversals of a hypergraph—see Berge (1989) for details about hypergraphs. The number of edges of the input hypergraph eventually reflects the number of MSUs or minimal keys. If this input parameter is more than about 100,000, the minimal transversals computation becomes prohibitive as described in Flouvat et al. (2004). For example, we are unable to use the Adaptive Borders Search (ABS) algorithm (Flouvat et al. 2004) (which is based on the concept of dualization between positive and negative borders) to find the 1,371,547 MFIs in the Connect-4 data (Merz and Murphy 1996), with $minsup = 2$. This is particularly pertinent for the search for MSUs in census-type datasets where the number of MSUs tends to be very large, often numbering hundreds of millions. For example, three datasets from the UCI Machine Learning Repository (Merz and Murphy 1996) demonstrate this clearly: the s.ipums.la.97 census dataset contains a total of 239,337,777 MSUs, s.ipums.la.98 contains a total of 395,694,639 MSUs and s.ipums.la.99 contains a total of 653,131,009 MSUs. We are unaware of an implementation for finding minimal transversals that is able to handle such large input hypergraphs.

There are many state-of-the-art algorithms for finding MFIs (Bayardo 1998; Han et al. 2000; Burdick et al. 2001; Gouda and Zaki 2005) and some of these are capable of handling large numbers of MFIs. For example, the MFIs for

Connect-4 above were found using MAFIA (Burdick et al. 2001). It would be useful if these algorithms could be adapted so that it is possible to move from the positive border of FIs with $minsup = 2$ to the MSUs on the negative border. However, algorithms for finding MFIs cannot be adapted easily for finding MSUs.

The following results in this subsection assume that the items in the input dataset are unrelated to each other. SUDA2 can be applied to datasets used in frequent itemset mining (i.e., those with varying length rows as described above): relationships between items can be represented by taxonomies over the items, as described in Srikant et al. (1997) for association rules.

Let D be a dataset consisting of subsets of a set $\mathcal{I}$ (inventory) of items. Let MFI be the maximal frequent itemsets in D with a $minsup = 2$. The first theorem proves that any record-level extension of an MFI with $minsup = 2$ is unique (and is therefore a superset of at least one MSU).

Theorem 1 (Uniqueness of record-level extensions of MFIs with $minsup = 2$) *Given D and MFI as above and an mfi $m \in MFI$ that exists in row $R = \{i_1, i_2, \ldots, i_P\}$ in D. For any $1 \leq j \leq P$ where $i_j \notin m$, $I = m \cup \{i_j\}$ is unique in D and appears in row R.*

Proof By the definition of maximal frequent itemsets, any superset of m is infrequent. If $minsup = 2$, then any superset will have support=1 or support=0. By the construction of I, its support will be 1 since $I \subseteq R$. □

Corollary 1 (Using MFIs to produce candidate sets for MSUs) *Given D and MFI as above and a k-MSU, I, that exists at row R (i.e., $I \subseteq R$) such that at least one of its k $(k-1)$-subsets, S, has support=2 in D, I is contained in the set of record-level extensions of MFI at R.*

Proof If I is an MSU then any $(k-1)$-subset, S, will be frequent in D by the definition of MSUs. S will either be a member of MFI or $S \subset m \in MFI$. If $S \in MFI$ or $m \subseteq R$, we are done. So suppose $S \subset m \in MFI$ and assume $m \nsubseteq R$ (i.e., $m - R \neq \emptyset$). Then S would appear in three rows, the two holding m plus row R, contradicting the assumption that S has support = 2. By Theorem 1, I is contained in the record-level extensions of S at record R. □

However, if I is an MSU where each of its $(k-1)$-subsets has support > 2 it can no longer be guaranteed that I is contained in the record-level extensions of its $(k-1)$-subsets.

Theorem 2 (Incompleteness of using record-level extensions of MFIs to find MSUs) *For any integer $g > 2$, it is possible to generate a dataset D consisting of subsets of a set of items $\mathcal{I}$ and a k-MSU, I, at row R, where $I \subseteq R$, such that every $(k-1)$-subset of I has support at least g and I is not a subset of any record-level extension of some itemset $m \in MFI$, where MFI is the set of all maximal frequent itemsets in D with $minsup = 2$.*

Proof Let the k $(k-1)$-subsets of I be represented by $\{s_1, s_2, \ldots, s_k\}$. As each of the $s_j, 1 \leq j \leq k$, has support $g > 2$, it is possible to construct a dataset D

Table 1 An example of the incompleteness of MFI record-extensions as a method for generating candidate MSUs

	Items	MFIs
Row 1	1,2,3,4,11,12	{11,12}
Row 2	1,2,3,5	{1,2,3,5}
Row 3	1,2,3,5	{1,2,3,5}
Row 4	1,2,4,6	{1,2,4,6}
Row 5	1,2,4,6	{1,2,4,6}
Row 6	1,3,4,7	{1,3,4,7}
Row 7	1,3,4,7	{1,3,4,7}
Row 8	2,3,4,8	{2,3,4,8}
Row 9	2,3,4,8	{2,3,4,8}
Row 10	11,12	{11,12}

such that each s_j is the subset of a frequent itemset I' and $s_j \subset I' \subseteq m \in MFI$ where $m \subseteq R$ and $I' \not\subseteq R$ for $j, 1 \leq j \leq k$. Therefore, I will not be contained in the set of MFI extensions at record R. □

For example, Table 1 shows a dataset with a 4-MSU {1,2,3,4} at Row 1 together with a list of all the MFIs with $minsup = 2$. The only MFI at Row 1 is {11,12}. The row-level extensions of {11,12} at Row 1 are {1,11,12}, {2,11,12}, {3,11,12} and {4,11,12}, none of which is a superset of {1,2,3,4}.

The size of FI can be extremely large when *minsup* is low and when items are highly correlated. As a result, many attempts have been made to compress information about frequent itemsets and their supports in a lossless manner (this can be contrasted to MFIs which do not represent all supports of itemsets in FI). An example of such a representation is *frequent closed itemsets* (Pasquier et al. 1999; Zaki and Hsiao 2002; Burdick et al. 2001; Uno et al. 2004; Lucchese et al. 2004, 2006): an itemset is closed if its support is greater than that of any of its supersets. MSUs are related to closed itemsets with support = 1 in the following way. Given an itemset Q, it is possible to determine an equivalence class $|Q|$, which contains the set of itemsets (also known as *generators*) that have the same support as Q and are included in the same set of rows. Moreover, if $min|Q|$ is defined as the set of minimal itemsets in $|Q|$, an itemset P is a *key itemset* (also known as a *minimal generator*) if $P \in min|Q|$, i.e., no proper subset of P is in $|Q|$. The closed set c of $|P|$ is equal to $max|P|$. The set of MSUs consists of all key itemsets contained in the equivalence classes of closed itemsets of support = 1.

There can be several key itemsets in each equivalence class, but only one is needed to identify the corresponding closed itemset. Therefore, although key itemsets have been used in the mining of closed itemsets (Pasquier et al. 1999; Zaki and Hsiao 2002; Lucchese et al. 2004), such algorithms do not necessarily find all of them. In Sect. 4 it will be shown that the discovery of a closed itemset does not lead easily to the identification of all key itemsets in the related equivalence class; further searching for a subset we call the *special row* is required.

Algorithms for mining all key itemsets (Liu et al. 2006) or generalizations of key itemsets (Dong et al. 2005; Boulicaut et al. 2003; Calders and Goethals 2005) have been developed but have the potential to exhaust main memory when *minsup* is low, even though significant strides have been made to compress the representation of the search space.

In SUDA2, novel properties of MSUs are identified and used to move recursively through the space of key itemsets with $minsup = 1$ using a depth-first search. Only one branch of the recursive tree is required at any given time and, unlike other algorithms, the set of MSUs mined so far is not stored in main memory as no duplicate checks or subset checks are necessary. This design ensures that the use of main memory is kept to a minimum.

Relationship to k-anonymity. k-Anonymity is a popular statistical disclosure technique which has received considerable attention in recent years (Sweeney 2002; Samarati and Sweeney 1998; Machanavajjhala et al. 2006). One of the main reason for this is believed to be the simplicity of the concept and the claim that it avoids the iterative process of statistical disclosure assessment and control where the data is repeatedly masked and then tested for disclosure risk and information loss. It is possible that the work presented here, which assesses disclosure risk rather than controls it, could act as a complement to k-anonymity, focusing attention on the most ‘risky’ attributes.

Attributes in a dataset can be divided into two categories, *sensitive* and *nonsensitive*. A sensitive attribute is an attribute whose value for any particular individual must be kept secret from people who have no direct access to the original data. An example of a sensitive attribute is *Income* or *Medical Condition*. All other attributes are assumed to be nonsensitive.

A set of nonsensitive attributes is called a *quasi-identifier* if these attributes can be linked with external data to uniquely identify at least one individual in the general population. Hence there is an emphasis on unique itemsets in k-anonymity and a relationship with the discovery of MSUs.

A quasi-identifier differs from an *identifier*, such as *National Insurance number* (in the UK) or *Social Security number* (in the US), which unambiguously identifies the individual concerned. An example of a quasi-identifier is the set (Gender, Age, Occupation).

As the quasi-identifiers might uniquely identify tuples in a table T, T is not published. Instead T is subjected to an anonymization procedure and the resulting table, T*, which satisfies k-anonymity for some value of k, is published. A table T* is said to satisfy k-anonymity for $k > 1$ if, for each combination of values of quasi-identifiers, at least k records exist in T* sharing that combination.

The techniques for anonymization can be broadly classified into generalization (recoding), generalization with tuple suppression, data swapping and randomization. These techniques have the potential to be computationally challenging. For example, achieving k-anonymization using generalization and tuple suppression whilst minimizing loss of information has been proven to be NP-hard (Aggarwal et al. 2005). The process of optimally combining generalization and local suppression is itself an open issue (Domingo-Ferrer and Torra 2005). Less computationally expensive approaches based on microaggregation have been proposed (Domingo-Ferrer and Torra 2005).

There are two main reasons why k-anonymity does not guarantee privacy (Machanavajjhala et al. 2006):

1. *Homogeneity Attach:* k-anonymity can create groups that leak information due to a lack of diversity in the sensitive attributes. For example, if the quasi-identifier is (Gender, Age, Occupation) and each person in a given group, say (Gender = female, Age = 24, Occupation = 'Civil Servant'), has an annual income of 100,000 pounds, it is possible to conclude that any person who is known to be in this dataset with this quasi-identifier has an annual income of 100,000 pounds.
2. *Background Knowledge Attack:* k-anonymity may not protect against attacks based on background knowledge. For example, assume there are two levels of annual income in the group with quasi-identifier (Gender = female, Age = 24, Occupation = 'Civil Servant'): one with a high annual income of 100,000 pounds and the other with a very low annual income of 10,000 pounds. Under such circumstances there is a significant possibility of being able to determine the annual income of a person in this dataset with this quasi-identifier given only a small amount of information about her lifestyle.

In addition, unless the number of attributes is small, it is unlikely that all nonsensitive attributes will be included in the set of quasi-identifiers as the anonymization process would impose too high a level of information loss on the data. Also, the data-provider can only guess which attribute values an intruder is most likely to have access to in the wider population. As the work presented here finds all MSUs of each size (up to a user-specified level), it could be used to give information about the contribution of each attribute to the number of MSUs—for example, statistics such as 40% of MSUs of size 2 contain *age*. Such information could be then be employed prior to applying k-anonymity to help select the set of quasi-identifiers by focusing attention on attributes which occur most frequently in the MSUs.

3 SUDA: the original approach

The SUDA algorithm (Elliot et al. 2002, 2005) searches for MSUs by generating all possible column subsets in a depth-first manner and repeatedly sorting the input dataset using a modification of the *bucket sort* (Elliot et al. 2002). Using the observation that no superset of an MSU need be considered, it is possible to prune parts of the search space.

The depth-first approach means that column subsets with the same prefix are selected in succession. (In general, for a column subset $\mathcal{C}$ containing r columns $i_1, \ldots, i_r$, a prefix of size P of $\mathcal{C}$ where $1 \leq P \leq r$ contains the first P items $i_1, \ldots, i_P$ of $\mathcal{C}$.) This means that any extensions of a unique prefix at a given

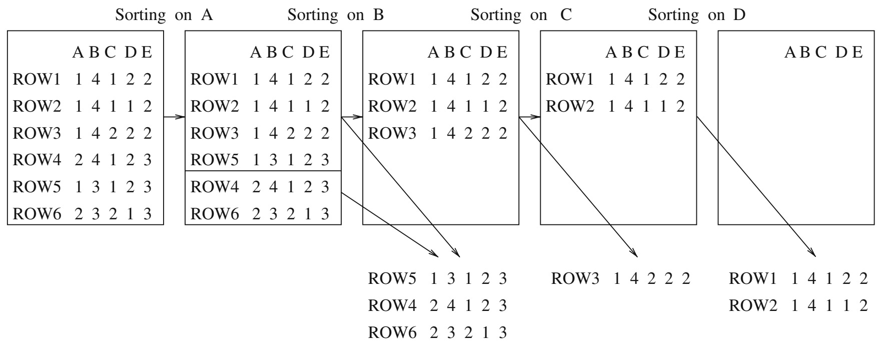


Fig. 1 Sorting process for SUDA

row can be ignored without reverting to stored information. For example, given four columns, labeled A, B, C and D, the column subsets with prefix B are: B, BC, BCD and BD. If the value in column B was found to be unique for row S then column subsets BC, BCD and BD could be ignored for that row. This has the effect of reducing the number of rows that need to be considered for each column subset.

The process of identifying MSUs is conducted by partitioning the dataset according to the value of each of the columns in a given column subset. For example, if the column subset was (*age*, *gender*) the dataset would be divided into groups of rows containing items such as ($age = 20$, $gender = male$), ($age = 40$, $gender = female$). Any group containing only one row represents a unique itemset and a check for minimal uniqueness would then be made to determine MSU status. The minimal uniqueness of a unique itemset X of size n (where $n \geq 2$) is determined by confirming that all subsets of X of size $n - 1$ are non-unique.

Figure 1 shows a dataset with six rows (labeled ROW1 to ROW6) and five columns (labeled A–E). The diagram illustrates the process of finding MSUs with column subsets A, AB, ABC, $ABCD$ and $ABCDE$. Sorting on column A divides the dataset into two partitions but does not yield any unique itemsets. When each of these partitions is further sorted on column B, three unique itemsets are discovered: A_2B_4 at row 4, A_1B_3 at row 5 and A_2B_3 at row 6. These are all MSUs. When the remaining rows (1, 2 and 3) are sorted on column C, row 3 is unique. However, $A_1B_4C_2$ is not an MSU as A_1C_2 is unique. Sorting on D yields unique itemset $A_1B_4C_1D_2$ at row 1 and unique itemset $A_1B_4C_1D_1$ at row 2. However, once the 3-subsets have been checked for uniqueness, only $A_1B_4C_1D_2$ is an MSU. There are no rows left to check for $ABCDE$. The search then proceeds to $ABCE$, ABD, $ABDE$ etc. The algorithm continues until either all possible column subsets have been considered, or where the artificial upper limit on k, *maxk*, is specified, all possible column subsets up to size *maxk* have been checked.

The generation of column subsets according to their prefixes allows the partitioning procedure to be undertaken without redundant sorting. For example, to move from considering MSUs for columns $ABCDE$ to MSUs for columns $ABCE$, the sorting on columns A, B and C does not have to be repeated; only the fourth sort is repeated, sorting on E rather than on D.

The ordering of column subsets also enables the check for non-uniqueness of $(k-1)$-itemsets to be conducted in a straightforward manner. For example, suppose that the column subset under consideration is FGH where F can take values {1,2,3}, G can take values {1,4,5} and H can take values {1,5,6}. Assume that $F_1G_1H_1$ is unique at row R. A check needs to be made for minimal uniqueness. It is already known that F_1G_1 is non-unique (otherwise R would not have been included in the sorting process for column H). However, checks for the non-uniqueness of F_1H_1 and G_1H_1 need to be made. As the rows have been partitioned, the required information can be obtained quite easily. For itemset F_1H_1, the support counts for all partitions containing $F = 1$ and $H = 1$ need to be added together—there are two possibilities: the partitions for rows containing $F_1G_4H_1$ and the partition for rows containing $F_1G_5H_1$, i.e., partitions containing F_1 and H_1 together with any value of G. The same is true for itemset G_1H_1—the support count is determined by summing the support counts for all partitions containing both G_1 and H_1 and any value of F. The algorithm stores the support counts of all partitions for each column subset in a hash table allowing fast lookup for support counts (Elliot et al. 2002).

4 SUDA2: the new approach

Although SUDA has greatly increased the depth of risk assessment possible, it is restricted to datasets with very small numbers of columns. This problem motivated the development of SUDA2.

SUDA2 aims to improve SUDA using a number of methods. First, a set of properties of MSUs have been identified and are used to design pruning strategies. Second, a new approach has been used to represent the search space. Third, a new traversal of the search space is employed.

4.1 Properties of MSUs

As the problem of finding the number of MSUs is known to be #P-Hard (Gunopulos et al. 2003), a reasonable approach is to find effective heuristics that prune large portions of the search space where MSUs are known not to exist.

The SUDA2 algorithm relies upon a set of properties that must hold true for every k-MSU in a dataset of R rows and C columns where $k \leq C$. These properties were briefly introduced in Manning and Haglin (2005) and are presented in detail below.

Consider a k-MSU I. By definition, I must have the following property.

Property 1 (Uniqueness Property) If itemset I is a k-MSU, then $supp(I) = 1$.

The following theorem proves that certain itemsets must exist within the dataset, D, in order for an MSU to exist.

Theorem 3 (Support Row Property) *Given a k-MSU, $I = \{i_1, i_2, \ldots, i_k\}$, in row r, for each $1 \leq j \leq k$ there must exist at least one support row s_j in D containing itemset $I - \{i_j\}$ but not containing i_j.*

Proof Suppose for any j, $1 \leq j \leq k$, there does not exist a row in D (apart from r) containing $I' = I - \{i_j\}$. Then I' is unique at r and therefore I is not minimal. □

Observe that there must exist at least k support rows since the structure of a support row s_j differs for each j. The existence of these k support rows in D is both a necessary and sufficient condition for i to be minimal, as seen by the previous theorem and the following lemma.

Lemma 1 *Given an itemset $I = \{i_1, \ldots, i_k\}$ that is unique in D and has k support rows containing the k subsets of I of size $k - 1$ but not containing I, then I is a k-MSU.*

Proof Suppose I is not minimal. Then there exists $J \subset I$ that is also unique within D. Since D contains the k support rows and J is unique in D, $|J| < k - 1$. Let $X = I - J$. Note that $|X| > 1$. Now pick any item $i \in X$. Observe that J must appear in the support row holding $I - \{i\}$ but not holding i. Since J also appears in the row containing I it appears as least twice in D. □

Observe that for any item i_1 to be part of a k-MSU, I, the item i_1 must appear in at least k different rows: the row holding I and $k - 1$ support rows with i_j, $2 \leq j \leq k$, removed. Without loss of generality, this observation holds for all $i_j \in I$. The Support Row Property can give an efficient way to prune significant areas of the search space if several items have low support counts.

For example, Fig. 2a shows possible support rows for a 4-MSU, $\mathcal{M} = A_1B_2C_3D_4$, and illustrates that some support rows may be repeated ('X's refer to items taking any value that is not contained in $\mathcal{M}$). Row 7 is not a support row as it differs from $\mathcal{M}$ by more than two items.

Corollary 2 (Uniform Support Property) *Given a dataset D, a unique itemset I of size $k \geq 2$, and item $a \in I$ such that item a is contained in every row holding a size $(k - 1)$ subset of I, itemset I is not minimal (i.e., not an MSU).*

Proof If I is a k-MSU in D containing item a, then the Support Row Property ensures the existence of a row in D containing $I - \{a\}$ and not containing item a. Therefore, if I only possesses support rows containing item a it cannot be an MSU in D. □

An example of the Uniform Support Property is given in Fig. 2b. Itemset $A_1B_2C_3D_4$ is unique at row R and has A_1 in each of its support rows. However, it is not minimal as it has a unique subset, $B_2C_3D_4$. The Uniform Support Property can be a useful method for reducing the size of candidate MSUs.

The following result extends the Support Row Property to items that are perfectly correlated—that is, items that occur in exactly the same rows.

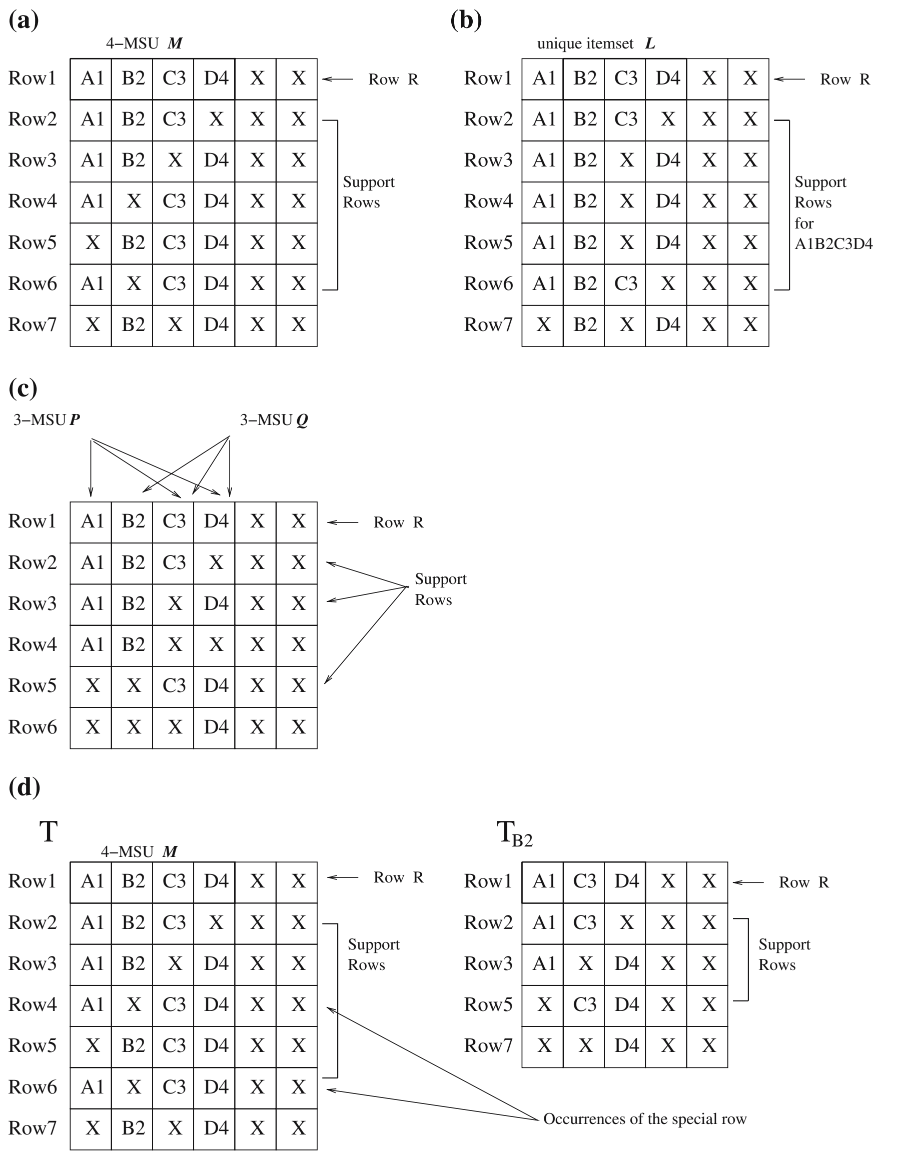


Fig. 2 (**a**) Support Row Property, (**b**) Uniform Support Property, (**c**) Perfect Correlation Property, and (**d**) Recursive Property

Corollary 3 (Perfect Correlation Property) *Given a dataset D, two perfectly correlated items a and b and a k-MSU I containing a and not b, then there exists an MSU J with a replaced by b.*

Proof Itemsets I and J have exactly the same support rows as a and b are perfectly correlated and $I - a = J - b$. Therefore J is an MSU in D. □

For example, Fig. 2c shows two items, A_1 and B_2 that are perfectly correlated. Row R contains two 3-MSUs, $\mathcal{P}$ ($A_1C_3D_4$) and $\mathcal{Q}$ ($B_2C_3D_4$). In terms of the search for MSUs, only one member of a set of perfectly correlated items need be active in the search as long as the corresponding MSUs for the removed items are added at a later stage. However, the procedure required to check for perfect correlation between all items with the same support count is combinatorially explosive and has the potential to become counter-productive in terms of execution time. Section 5.1 describes how SUDA2 balances this conflict.

So far the MSU properties have all been related to the given (input) dataset, D. The following describes properties of MSUs relative to specific subsets of D.

Definition 1 Given dataset D and some item a (called an *anchor* item), the *projection* of D with respect to that anchor, denoted D_a, is defined as those rows of D holding item a.

Since every row in D_a contains item a, corresponding properties of MSUs in D and D_a can be inferred.

Lemma 2 *Given I is a k-MSU in dataset D, for each anchor $a = i_j$, $1 \leq j \leq k$, the itemset $I_a = I - \{i_j\}$ is a $(k-1)$-MSU in dataset D_a.*

Proof Without loss of generality, fix j in the range of 1 to k inclusive. Let $a = i_j$. Since every row in D_a contains item a, the only way I_a could appear more than once in D_a is if I appeared in multiple rows of D. Therefore, I_a is unique in D_a. Similarly, if I_a is not minimally unique in D_a, then there exists $I'_a \subset I_a$ that is also unique in D_a. But $I'_a \cup \{a\}$ would also be unique in D and is a proper subset of I. Hence, I_a must be minimally unique in D_a. □

Unfortunately, it is not the case that all $(k-1)$-MSUs in D_a lead to k-MSUs in the original dataset D. However, the following theorem provides a method for finding k-MSUs.

Theorem 4 *Given a dataset D, an item a, a projection D_a, and I_a as a $(k-1)$-MSU in D_a, the itemset $I = I_a \cup \{a\}$ is a k-MSU in D if and only if there exists a row in D containing I_a but not containing item a.*

Proof If I is a k-MSU in D, then the Support Row Property ensures the existence of a row in D containing I_a but not containing item a. For the other direction, assume a row exists in D containing I_a but not a and that I_a is a $(k-1)$-MSU in D_a. Note that I must be unique in D. All that is required to show is that I has the requisite k support rows. Each of the $k-1$ support rows in D_a, augmented with item a, form a support row in D for the itemset I. As all of these $k-1$ support rows contain item a, the only other support row needed to ensure I is a k-MSU in D is the row stated in the theorem. Since I is unique and since k support rows exist in D, I must be a k-MSU. □

For example, let Table T be the dataset shown in Fig. 2d and $\mathcal{M} = A_1B_2C_3D_4$. Let $\mathcal{N}$ be obtained from $\mathcal{M}$ by removing B_2. Table T_{B2} is obtained from table T by selecting all rows containing B_2, as shown in Fig. 2d. It can be seen that $\mathcal{N}$

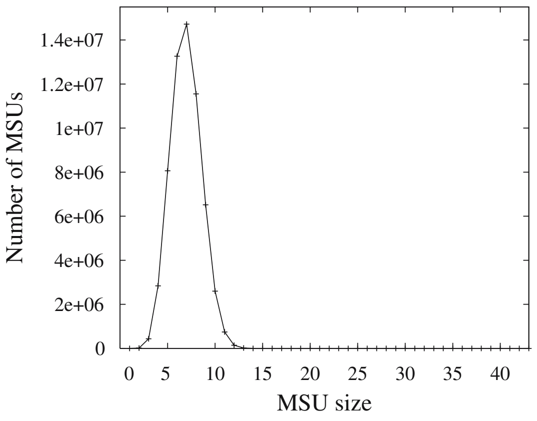


Fig. 3 MSUs for one area of the 1991 Individual SARs data

is an MSU in Table T_{B2}. In the other direction, assume that $\mathcal{N}$ has been found to be an MSU in table T_{B2}. The itemset $\mathcal{M} = \mathcal{N} \cup B_2$ is an MSU in T as the required special row exists at both rows 4 and 6.

The Recursive Property of MSUs helps define the boundaries of the search space by providing a clear indication of the maximum size of candidate MSU.

4.2 Representing the search space for MSUs

The maximum size of an MSU is often far smaller than the number of columns in the input dataset. For example, Fig. 3 shows the number of MSUs of distinct sizes from one geographical area of the 2% 1991 Individual SAR (SAR1991 1993). Even though there are 43 columns in total, there are no MSUs of size greater than 15. The identification of the boundaries of the search space for MSUs is therefore critical.

SUDA repeatedly sorts the dataset according to column subsets in a depth-first manner. This search halts in a given direction if all rows become unique on the given column subset, as illustrated in Sect. 3. Although this method of pruning has the effect of reducing the number of rows in the sorting procedures associated with each column subset, it has less impact on the number of column subsets considered. However, it is important to note that SUDA never considers any itemset that does not exist within a row of the dataset (i.e., an itemset that does not have the potential to be an MSU). If this were not the case redundant searching could ensue. For example, if all possible value pairs of columns *age* and *occupation* were considered, many of these value pairs would not exist within the dataset due to the particular sample and, because people below a certain age do not have an occupation. SUDA2 aims to avoid iterating through all possible column subsets and to use a more flexible representation of the search space while, at the same time, maintaining the data driven approach of SUDA.

Table 2 An example of a small dataset

(a) Original input

	A	B	C	D	E
Row 1	1	4	1	2	2
Row 2	1	4	1	1	2
Row 3	1	4	2	2	2
Row 4	2	4	1	2	3
Row 5	1	3	1	2	3
Row 6	2	3	2	1	3

(b) Initial search space

	Search space
Row 1	$A_1B_4C_1D_2E_2$
Row 2	$A_1B_4C_1D_1E_2$
Row 3	$A_1B_4C_2D_2E_2$
Row 4	$A_2B_4C_1D_2E_3$
Row 5	$A_1B_3C_1D_2E_3$
Row 6	$A_2B_3C_2D_1E_3$

Table 3 The effect of removing one item, A_2, from the search space

	Search space	2-MSUs
Row 1	$A_1B_4C_1D_2E_2$	
Row 2	$A_1B_4C_1D_1E_2$	
Row 3	$A_1B_4C_2D_2E_2$	
Row 4	$B_4C_1D_2E_3$	A_2B_4, A_2C_1, A_2D_2
Row 5	$A_1B_3C_1D_2E_3$	
Row 6	$B_3C_2D_1E_3$	A_2B_3, A_2C_2, A_2D_1

Very few items can usually be removed from a search for MSUs (apart from those that appear either once in the dataset or in every row) so the potential for condensing the search space by removing items whose support falls below the threshold is limited. A search space representation has therefore been designed specifically for SUDA2. A description of the search space representation designed for SUDA2 was published in Manning and Haglin (2005) and a summary is presented here.

Table 2a shows a dataset with six rows and five columns. The search space for MSUs for this table can be thought of as all possible, distinct subsets of each of the itemsets in Table 2b.

This search space can be altered as new information becomes available.

It is possible to consider all MSUs containing a given item in succession. For example, Table 3 shows all MSUs pertaining to A_2 and the corresponding effect on the search space; the search space at rows 4 and 6 is adjusted by removing A_2, a step which is straightforward to implement.

Table 4 The effect of removing four items, A_2, B_3, C_2, D_1, from the search space

	Search space	2-MSUs	3-MSUs	4-MSUs
Row 1	$A_1B_4C_1D_2E_2$			
Row 2	$A_1B_4C_1E_2$	$A_1D_1, B_4D_1, C_1D_1, D_1E_2$		
Row 3	$A_1B_4D_2E_2$	$A_1C_2, B_4C_2, C_2D_2, C_2E_2$		
Row 4	$B_4C_1D_2E_3$	$A_2B_4, A_2C_1, A_2D_2, B_4E_3$		
Row 5	$A_1C_1D_2E_3$	$A_1B_3, A_1E_3, B_3C_1, B_3D_2$		
Row 6	E_3	$A_2B_3, A_2C_2, A_2D_1, B_3C_2$ $B_3D_1, C_2D_1, C_2E_3, D_1E_3$		

Table 5 The completed search for MSUs

	Search space	2-MSUs	3-MSUs	4-MSUs
Row 1			$C_1D_2E_2$	$A_1B_4C_1D_2$
Row 2		$A_1D_1, B_4D_1, C_1D_1, D_1E_2$		
Row 3		$A_1C_2, B_4C_2, C_2D_2, C_2E_2$		
Row 4		$A_2B_4, A_2C_1, A_2D_2, B_4E_3$		
Row 5		$A_1B_3, A_1E_3, B_3C_1, B_3D_2$		
Row 6		$A_2B_3, A_2C_2, A_2D_1, B_3C_2$ $B_3D_1, C_2D_1, C_2E_3, D_1E_3$		

Table 4 shows the search space after items A_2, B_3, C_2 and D_1 have been removed and Table 5 shows the final list of MSUs (i.e., when the search space is empty). It can be seen that as more MSUs are discovered, the remaining search space shrinks. (The actual process for finding MSUs is described in Sect. 5).

The above example clearly shows that the data driven aspect of SUDA is maintained.

5 The SUDA2 algorithm

In this section, the SUDA2 algorithm is described in detail and an example is given of its application to a small dataset.

5.1 Algorithmic details

The first stage of SUDA2 is to scan the input dataset T in order to find support counts for all items that appear at least once. Unique items are recorded and removed as they cannot lead to larger MSUs. Items appearing in all rows of the dataset are redundant by the Uniform Support Property and are also removed.

The list of remaining items is sorted in ascending order of support count with ties broken arbitrarily; the ordered list is referred to below as ITEM-LIST. The relative position within ITEM-LIST is called the *rank* of an item. By giving the items a complete ordering, every itemset—in particular, every MSU—contains an item with lowest rank, called the *reference item*.

Adjacent items with the same support count in ITEM-LIST are checked for perfect correlation—one of each pair of perfectly correlated items is removed so that the corresponding MSUs can be added at a later stage, as described in Sect. 4.1.

The main data structures are then built. First, each item in T is replaced by its rank in ITEM-LIST which means that ranks and relative ranks can readily be determined. Second, each item is associated with a set of rows that contain it so that, for any given item, the rows in which it occurs can be found immediately. This, in turn, enables the co-existing items (i.e., items within the same row) to be located.

The search for MSUs then proceeds as follows:

– For each reference item of rank r, SUDA2 builds a subtable T_r of rows containing r. SUDA2 then searches recursively for all $(k-1)$-MSUs $\mathcal{N}$ in T_r to be used as candidates for finding k-MSUs in T. All properties of MSUs are applied recursively. SUDA2 tries aggressively to limit the potential search space for MSUs by looking for small upper bounds on the size of the candidate MSUs as follows:
  - If the support count of the reference item is k, then, by the Support Row Property, there are not enough support rows for any l-MSU with $l > k$. This is particularly useful near the beginning of ITEM-LIST where the support counts are smallest.
  - If the position of the reference item within ITEM-LIST is near the end, the number of remaining items to the right of the reference item can provide an upper bound on k.
  - As the depth of recursion increases, the upper bound on the size of the MSU decreases just as rapidly. If k is an upper limit on the size of an MSU for a given reference item, an upper bound of $k-1$ can be imposed on the (recursive) subproblem. This is because the reference item will be added to any MSU discovered in the subproblem for candidate MSUs for the next level.

– A candidate $(k-1)$-MSU, $\mathcal{N}$, appearing in row i leads to a k-MSU for row i provided two properties hold:
  1. All of the $k-1$ items in $\mathcal{N}$ have rank higher than r in the original table T.
  2. There is a row in T and not in T_r (the *special row*) holding all of the items in $\mathcal{N}$ but not holding the reference item (see Sect. 4.1). The special row is located by searching the subtable induced by the item of $\mathcal{N}$ (other than the reference item) with the smallest support. (There is no need to check the special row for MSU candidates of size 2 as the second item in the candidate itemset will automatically have support

count greater than or equal to the reference item—as it appears higher in ITEM-LIST—and will therefore appear in at least one other row; if it appeared in exactly the same rows it would have been removed by the Uniform Support Property.)
- When an MSU is discovered the corresponding MSUs of perfectly correlated items are added, as described in Sect. 4.1.

Pseudo code for SUDA2 is given in Algorithm 1.

Algorithm 1 SUDA2(T, $maxk$)

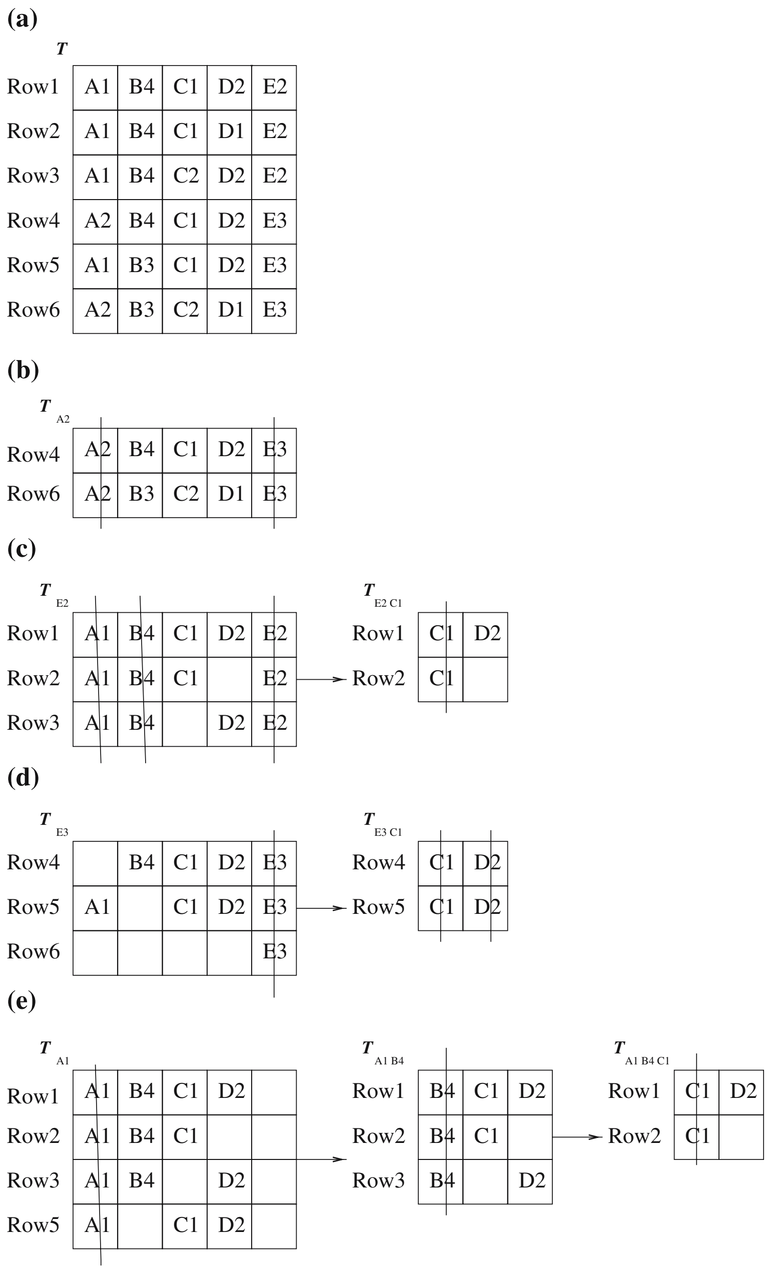
```
1: Input: $T$ = input dataset with $n$ rows and $m$ columns of integers
2: Input: $maxk$ = maximum size of MSU in the search
3: Returns: A set of MSUs for dataset $T$

4: compute $R \leftarrow$ list of all items in $T$
5: if $maxk == 1$ then
6:    Return all items of $R$ that appear only once in $T$
7: else
8:    $M \leftarrow \emptyset$
9:    for each item $i \in R$ do
10:      $T_i \leftarrow$ subset of $T$ consisting of only rows of $T$ holding item $i$
11:      $C_i \leftarrow$ recursive call to SUDA2($T_i$, $maxk$-1)
12:      for each candidate itemset $I \in C_i$ do
13:         if $I \cup \{i\}$ is an MSU in $T$ then
14:            $M \leftarrow M \cup (I \cup \{i\})$
15:         end if
16:      end for
17:   end for
18:   Return $M$
19: end if
```

Theorem 5 *Algorithm* SUDA2 *finds all MSUs up to size maxk.*

Proof Consider an arbitrary MSU I of size no larger than $maxk$. Let k be the number of items in I. Let item a be the item in I whose rank in R is smallest. When a is the anchor in the for loop at line 9, the T_a created at line 10 will contain the row holding I and $k-1$ of the k support rows. The missing support row is the one that does not have item a but all other items from I. Lemma 2 guarantees that $I - \{a\}$ is a $(k-1)$-MSU in T_a and will therefore be returned to C_i as one of the candidate MSUs in line 11. Moreover, Theorem 4 shows that the `if`-check at line 13 will evaluate to `true`. □

5.2 Example

The following example, which appeared with less detail in (Manning and Haglin 2005), applies SUDA2 to the data in Fig. 4a.



(a)

T

Row1	A1	B4	C1	D2	E2
Row2	A1	B4	C1	D1	E2
Row3	A1	B4	C2	D2	E2
Row4	A2	B4	C1	D2	E3
Row5	A1	B3	C1	D2	E3
Row6	A2	B3	C2	D1	E3

(b)

T_{A2}

Row4	A2	B4	C1	D2	E3
Row6	A2	B3	C2	D1	E3

(c)

T_{E2}

Row1	A1	B4	C1	D2	E2
Row2	A1	B4	C1		E2
Row3	A1	B4		D2	E2

$T_{E2\ C1}$

Row1	C1	D2
Row2	C1	

(d)

T_{E3}

Row4		B4	C1	D2	E3
Row5	A1		C1	D2	E3
Row6					E3

$T_{E3\ C1}$

Row4	C1	D2
Row5	C1	D2

(e)

T_{A1}

Row1	A1	B4	C1	D2	
Row2	A1	B4	C1		
Row3	A1	B4		D2	
Row5	A1		C1	D2	

$T_{A1\ B4}$

Row1	B4	C1	D2
Row2	B4	C1	
Row3	B4		D2

$T_{A1\ B4\ C1}$

Row1	C1	D2
Row2	C1	

Fig. 4 An example of the methods used in SUDA2

- The items are sorted in ascending order of support count to form the following ITEM-LIST:

$$A_2, B_3, C_2, D_1, E_2, E_3, A_1, B_4, C_1, D_2$$

  In this example items are not replaced by their ranks in the following tables as it is easier to demonstrate the mechanisms of the algorithm when the items themselves are presented. A subtable of T of rows containing A_1 is denoted T_{A1}, a subtable of T_{A1} of rows containing B_2 is denoted $T_{A_1B_2}$ etc.
- There are no unique items, no items occurring in all rows and no perfectly correlated items in this dataset.
- Subtable T_{A2} is created and is shown in Fig. 4b.
- The column for E_3 is removed from T_{A2} as E_3 occurs in both rows 4 and 6 and is therefore redundant by the Uniform Support Property.
- There are six candidate MSUs containing A_2: these are A_2B_4, A_2C_1 and A_2D_2 at row 4 and A_2B_3, A_2C_2 and A_2D_1 at row 6. No further check for the *special row* is necessary as the candidate MSUs are of size 2.
- The algorithm progresses in a similar manner for items B_3, C_2 and D_1, ignoring items with lower ranks.
- Subtable T_{E2} is shown in Fig. 4c, with blank spaces for items with lower ranks. Both A_1 and B_4 are redundant in this subtable and can therefore be ignored. There are no unique items in T_{E2} so the rows containing the next item in the ITEM-LIST for E_2 , C_1, are extracted to get T_{E2C1}, which is also shown in Fig. 5c. D_2 is unique in T_{E2C1} and C_1D_2 is a candidate MSU for T_{E2}. A check for the *special row* shows that D_2 occurs independently of C_1 in T_{E2} (at row 3) and thus C_1D_2 is an MSU in T_{E2}. The itemset $C_1D_2E_2$ is a candidate MSU for the whole dataset T. MSU status is confirmed by establishing that C_1D_2 occurs independently of E_2 (at row 4 or row 5) in T.
- Subtable T_{E3} is shown in Fig. 4d. Two MSUs, A_1E_3 and B_4E_3, can be seen immediately. Extracting rows containing C_1 leads to an empty subtable and the search proceeds to the next item.
- Subtable T_{A1} is shown in Fig. 4e. There are no unique items so rows containing B_4 (the next item in the ITEM-LIST for T_{A1}) are extracted to get T_{A1B4}. As there are still no unique items subtable T_{A1B4C1} is generated where D_2 is unique and C_1D_2 is a candidate MSU for T_{A1B4}. A check for the *special row* shows that D_2 occurs independently of C_1 at row 3 and therefore C_1D_2 is an MSU in T_{A1B4} and $B_4C_1D_2$ is a candidate MSU for T_{A1}. Further checking shows that C_1D_2 occurs independently of B_4 at row 5 and therefore $B_4C_1D_2$ is an MSU in T_{A1}. Now $A_1B_4C_1D_2$ is a candidate MSU for T, which is confirmed by observing that $B_4C_1D_2$ occurs independently of A_1 at row 4.
- Items B_4, C_1 and D_2 yield no MSUs.

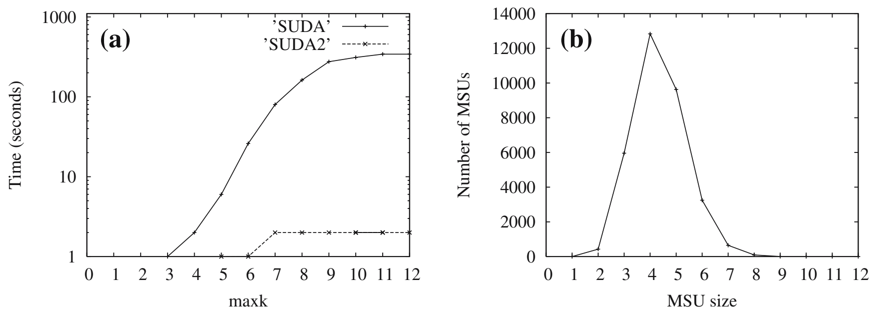


Fig. 5 (**a**) Execution time for the Twelve Attribute Data and (**b**) MSUs for the Twelve Attribute Data

6 Performance results

Performance tests were applied to SUDA and SUDA2 on a Dell Workstation with one 3 GHz Pentium 4 processor with 2 Gbytes main memory. The operating system used was Red Hat Linux 3.3.2-5. SUDA was written in C and SUDA2 in C++ and both were compiled with gcc version 3.2.2 using -03 optimization. Both real-world and synthetic datasets were used in the experiments.

6.1 Datasets in which both SUDA and SUDA2 can find all MSUs

SUDA and SUDA2 were applied to one area of the 1991 Individual SAR (SAR1991 1993) of approximately 9,000 rows. Twelve attributes were selected—AGE, ECONPRIM, ETHGROUP, LTILL, MSTATUS, SEX, BATH, CARS, TENURE, RESIDNTS, DEPCHILD and PENSIONH (SAR1991 1993).

The execution times for SUDA and SUDA2 are shown using a log scale in Fig. 5a and the number of MSUs of each size of k are shown in Fig. 5b. The number of MSUs can be seen to rise to a maximum for $k = 4$ and then gradually tail off, with no MSUs produced after $k = 9$. SUDA2 finds all MSUs in two seconds whereas SUDA takes well over 5 min. The execution time for SUDA2 flattens off after $k = 7$ reflecting the total numbers of MSUs.

6.2 Datasets in which SUDA cannot find all MSUs in a reasonable period of time

The following two examples illustrate the performance of SUDA2 in situations in which SUDA struggles to find MSUs of even a small size in less than 2 days. The 1991 SARs data provides an example of the survey-type data that SUDA2 was specifically designed for and was presented in Manning and Haglin (2005). The Connect-4 dataset provides evidence of SUDA2's performance in a wider context and is included as it is one of the benchmark datasets in the UCI Machine Learning Repository (Merz and Murphy 1996) and possesses

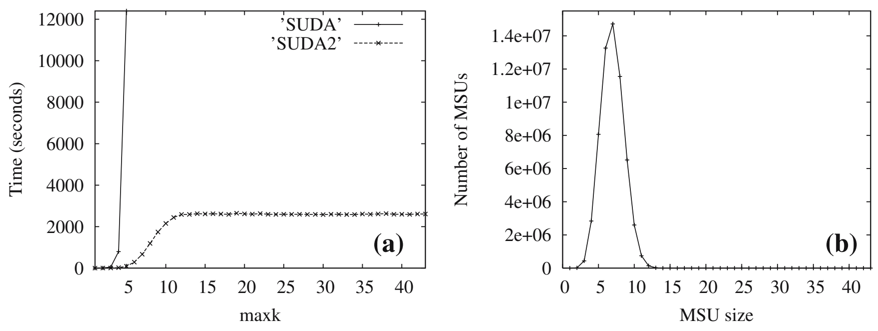


Fig. 6 (**a**) Execution time for the 1991 SARs data and (**b**) MSUs for the 1991 SARs data

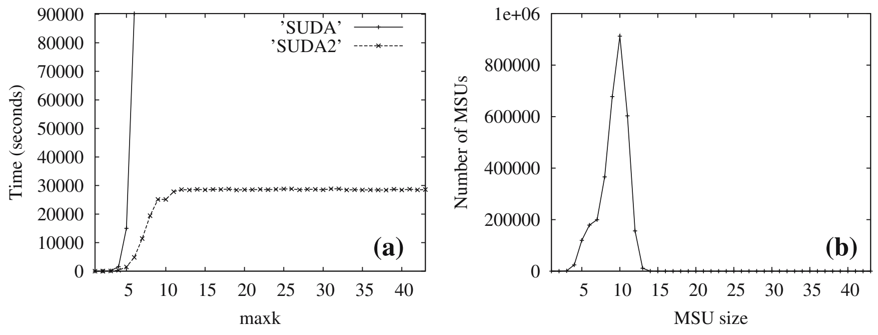


Fig. 7 (**a**) Execution time for the Connect-4 data and (**b**) MSUs for the Connect-4 data

characteristics that reflect those of census-type data discussed in Sect. 2.1 (these being a relatively small dataset of equal-length rows of discrete data).

- *1991 SARs* Both SUDA and SUDA2 were applied to one area of the 1991 SARs (SAR1991 1993) with approximately 9,000 rows and 43 columns. For SUDA, MSUs of no bigger than size six were found over a matter of days. SUDA2 was able to find all MSUs with no limit in size in less than 1 h. Figure 6a shows the execution time in seconds for SUDA and SUDA2 and Fig. 6b shows the numbers of MSUs of each size. As before, the SUDA2 execution time flattens off as the marginal numbers of MSUs fall.
- *The Connect-4 Dataset* The Connect-4 dataset (available from the UCI repository Merz and Murphy 1996) contains all legal 8-ply positions in the game of Connect-4 in which neither player has won yet, and in which the next move is not forced. There are approximately 70,000 rows and 43 columns (one for each of the 42 connect-4 squares together with an outcome column—*win*, *draw* or *loss*). Figure 7a shows that SUDA2 finds all MSUs within 8 h with the execution time flattening off when no further MSUs are produced, as shown in Fig. 7b. SUDA, on the other hand, takes well over a day to find all MSUs up to size 5.

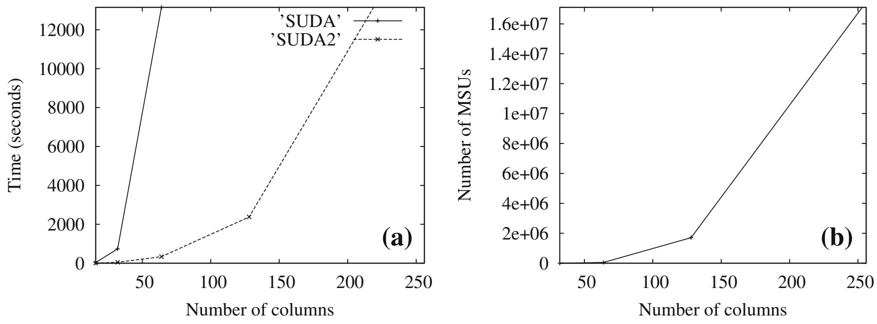


Fig. 8 (**a**) Execution time for the synthetic data over varying numbers of columns and (**b**) MSUs for the synthetic data over varying numbers of columns

6.3 Synthetic datasets

SUDA and SUDA2 were applied to a number of synthetic datasets to illustrate how their performance is affected by increasing numbers of columns and rows. The synthetic data is designed to reflect transactions in the retailing environment (Agrawal and Srikant 1994) by generating a list of transaction numbers together with a set of items bought in each transaction. (The software that generates these synthetic datasets is available at http://www.almaden.ibm.com.) For these experiments, the output was transformed into an input dataset for SUDA/SUDA2 by representing each item by a column in which a purchase is shown by a *1* and no purchase with a *0*. Where a transaction was generated with no purchases a row containing only zeros was placed in the dataset.

Varying the number of columns. Five datasets were generated with 100,000 rows each and numbers of columns 16, 32, 64, 128 and 256. Each of the other parameters was left with the default value of the software. SUDA and SUDA2 were applied to each of these datasets with the maximum MSU size set to 4 (any larger size and it was difficult to compare results). SUDA2 performs significantly better than SUDA, allowing datasets of approximately four times as many columns to be handled in the same time period, as shown in Fig. 8a. The execution time of SUDA2 appears to be closely related to the number of MSUs (shown in Fig. 8b).

Varying the number of rows. Five datasets were generated with 100 columns each and 16,000, 32,000, 64,000, 128,000 and 256,000 rows. The value of *maxk* was again set to 4. The gradients of both the execution times for SUDA and SUDA2 fall as the number of rows rise. However, SUDA has an execution time which is vastly slower, as shown in Fig. 9a. These results need to be considered against the falling numbers of MSUs that were present as the numbers of rows increased, as shown in Fig. 9b.

Figure 10 shows the execution time for SUDA2 over rising numbers of rows for the synthetic data and the hypothetical case where the execution time rises perfectly linearly with the number of rows. It can be seen that the execution time for SUDA2 lies close to this line for this data.

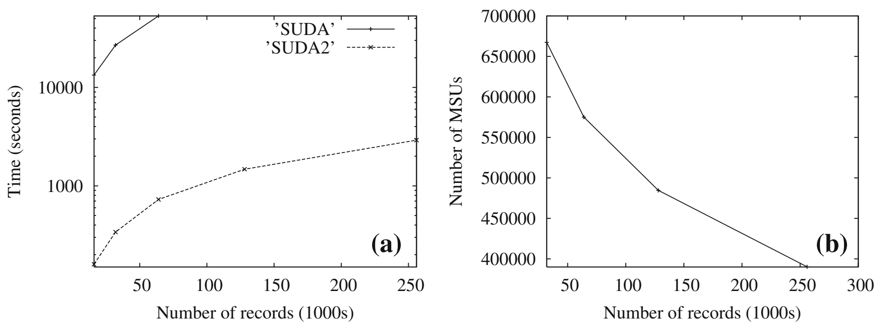


Fig. 9 (**a**) Execution time for the synthetic data over varying numbers of rows and (**b**) MSUs for the synthetic data over varying numbers of rows

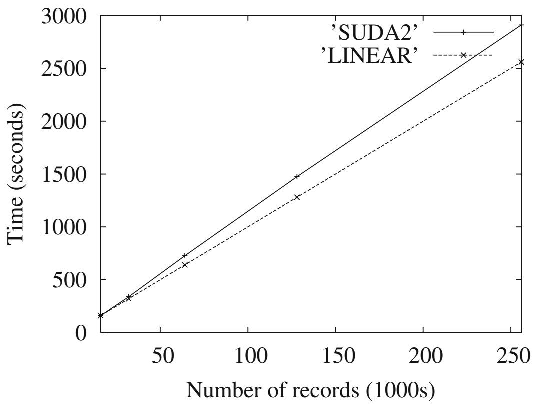


Fig. 10 Execution times for SUDA2 over increasing numbers of rows together with the hypothetical case where the execution time is perfectly linear

6.4 Varying the number of rows and columns for the 1991 individual SAR

The following example first appeared in Manning and Haglin (2005). In the context of this paper it is used in order to give a real world example of how SUDA2 performs as numbers of rows and columns vary, enabling comparisons to be made with the synthetic data. The execution times for SUDA2 were considered according to rising numbers of rows in the 1991 Individual SAR (SAR1991 1993). The execution times are shown in Table 6 ("X"s refer to jobs that were too large to run on the machine described above) and the corresponding numbers of MSUs are shown in Table 7. The rows for each data sample were selected at regular intervals within the SARs file (irrespective of area) and the columns were chosen at random.

Figures 11a, 12a and 13a plot the execution time in seconds for SUDA2 for rising numbers of rows and it can be seen that the performance is not as impressive as for the synthetic data in the previous section. However, Figs. 11b, 12b and 13b show the corresponding number of MSUs for each experiment and it can be seen that these rise with the number of rows. This is in contrast to the numbers of MSUs for the synthetic data which fell as the number of rows

Table 6 Execution times for SUDA2 with time in hours:minutes:seconds and *maxk* = number of columns

Number of rows	8 columns	16 columns	32 columns
1,000	00:00:00	00:00:00	00:00:34
2,000	00:00:00	00:00:01	00:01:47
4,000	00:00:00	00:00:01	00:05:47
8,000	00:00:00	00:00:02	00:19:00
16,000	00:00:01	00:00:07	01:06:09
32,000	00:00:04	00:00:19	03:52:59
64,000	00:00:10	00:00:50	14:05:52
128,000	00:00:10	00:02:29	53:07:55
256,000	00:00:24	00:07:28	X
512,000	00:01:05	X	X
1,024,000	X	X	X

Table 7 Numbers of MSUs for SUDA2, where *maxk* = number of columns

Number of rows	8 columns	16 columns	32 columns
1,000	3500	22053	58554
2,000	6109	42643	101445
4,000	10656	84572	189389
8,000	17414	162429	300506
16,000	29551	317603	900611
32,000	46364	582143	2878971
64,000	73429	1082599	6598705
128,000	110731	1907196	14078636
256,000	168357	3414408	X
512,000	254226	X	X
1,024,000	X	X	X

increased. This gives further evidence that the execution time of SUDA2 is closely associated with the number of MSUs present within the data suggesting that good candidate MSUs are generated.

Execution times rose significantly when the number of columns rose, as can be seen in Table 6. This is reflected in a similar pattern for MSUs in Table 7.

6.5 Datasets from the UCI repository

Ten datasets were selected from the UCI Machine Learning Repository (Merz and Murphy 1996) of varying types and sizes. Six of the datasets consisted of IPUMS US census data. Table 8 shows the execution time of applying SUDA2 to these datasets together with the total number of MSUs, the number of items in

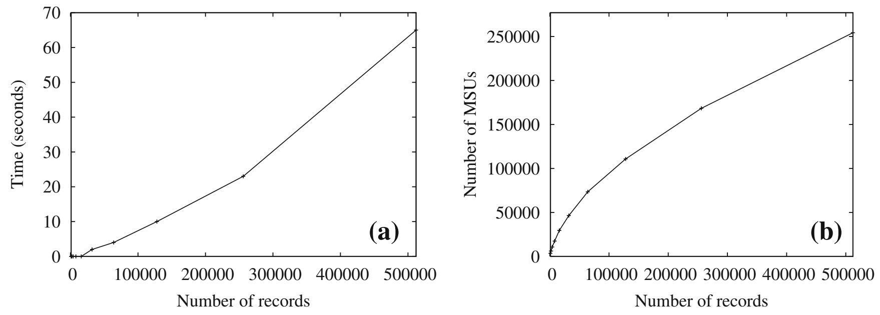


Fig. 11 8 column Individual SARs data (**a**) Execution times and (**b**) Number of MSUs

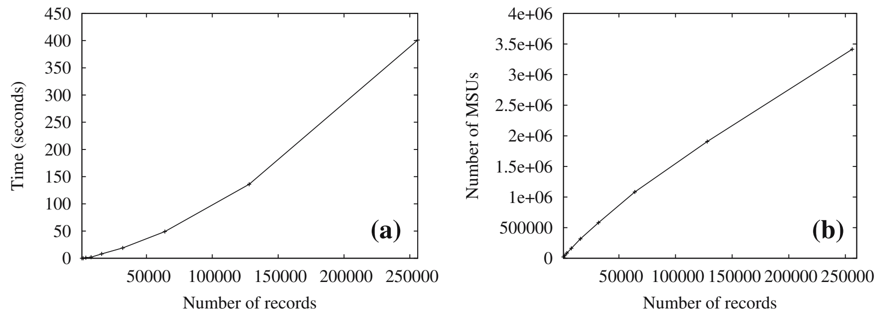


Fig. 12 16 column Individual SARs data (**a**) Execution times and (**b**) Number of MSUs

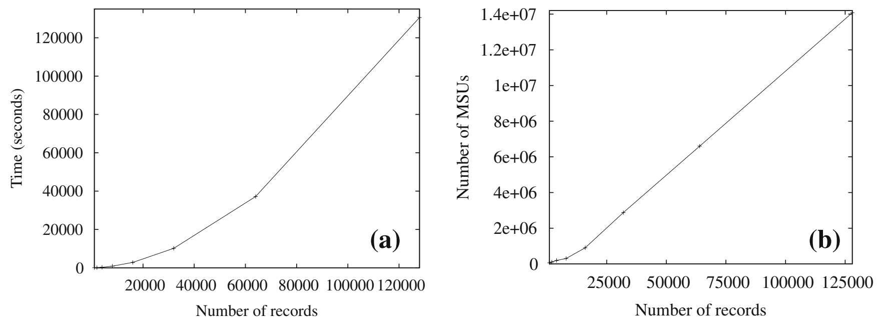


Fig. 13 32 column Individual SARs data (**a**) Execution times and (**b**) Number of MSUs

each dataset, the value of *maxk* and the largest MSU size. Apart from the three large census datasets, *maxk* is set to the number of columns. The three large census datasets had high numbers of items and MSUs and even for $maxk = 5$ the number of MSUs was a few hundred million. For each of the other datasets, it can be seen that the maximum size of MSU is less than the number of columns—and significantly less when the number of columns is large.

Table 8 Execution times for SUDA2 on datasets from UCI

Dataset	Rows	Columns	Anchors	*maxk*	Largest MSU	#MSUs	Execution time
Chess	3,196	37	75	37	16	519,186	01:37:41
Letter	20,000	16	256	16	10	11,392,030	00:04:51
Mushroom	8,124	23	119	23	10	11,507	00:00:03
Poker	1,000,000	11	95	11	9	53,523,046	09:14:46
ipums.la.97	70,187	61	3,985	5	5	459,850,304	01:38:29
ipums.la.98	74,954	61	22,180	5	5	445,156,589	02:16:02
ipums.la.99	88,443	61	40,157	5	5	562,424,468	03:00:30
s.ipums.la.97	7,019	61	3,009	61	16	239,337,777	03:38:00
s.ipums.la.98	7,485	61	9,371	61	16	395,694,639	05:22:10
s.ipums.la.99	8,844	61	10,008	61	17	653,131,009	10:09:55

7 Conclusion and further work

SUDA2 has been shown to outperform SUDA by several orders of magnitude in terms of execution time when applied to a number of datasets. It can also handle significantly more columns than before and find larger MSUs. The ability of SUDA2 to identify the boundaries of the search space for MSUs has also been demonstrated clearly. This has been brought about by the identification of a number of properties of MSUs and a more flexible representation and traversal of the search space. These developments mean that statistical agencies, such as the UK Office for National Statistics, now have a much faster tool to work with and, most importantly, are able to apply statistical disclosure assessment to datasets with more columns than they could previously. SUDA2 has wider application, being appropriate for any dataset that possesses the same characteristics as census-type data (or can be adapted) as described in Sect. 2.1. The next step will be to put SUDA2 onto more powerful machines and architectures. For example, SUDA2 is a good candidate for parallelism as searches can be divided up according to rank; the challenge will be to produce efficient load balancing.

The search strategies used in SUDA2 can be adapted easily for mining rare minimal itemsets and suitable implementations are being explored.

The experimental results suggest a possible *output polynomial time* behavior of SUDA2—that is, it may be that the worst case running time is some polynomial in the length of the *output*. (Recall that the problem of finding the number of MSUs is #P-Hard, so it is not likely to be polynomial in the length of the *input*.) It is not clear whether there are input datasets for which the number of MSUs is exponentially larger than the size of the input. If so, seeking a theoretical analysis of the output running time is an interesting topic for further work.

Acknowledgments The research presented here was funded by the Engineering and Physical Sciences Research Council (Grant Numbers GR/S27790/01 & GR/S22295/01). We would like to thank members of the Centre for Novel Computing at the University of Manchester for supporting the work and for providing invaluable feedback. We would also like to thank the anonymous referees for their insightful comments. The 1991 SARs are provided through the Census Microdata Unit, at the Cathie Marsh Centre for Census and Survey Research (University of Manchester), with the support of the ESRC/JISC/DENI. All tables containing Census data, and the results of analysis, are reproduced with the permission of the Controller of Her Majesty's Stationery Office (Crown Copyright).

References

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: VLDB'94, Proceedings of 20th international conference on very large data bases, September 12–15, 1994, Santiago de Chile, Chile, pp 487–499

Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 international conference on management of data (SIGMOD 93), pp 207–216

Agrawal R, Mannila H, Srikant R, Toivonen H, Verkamo A (1996) Fast discovery of association rules. In: Fayyad U, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds) Advances in knowledge discovery and data mining. The AAAI Press, Menlo Park, pp 307–328

Aggarwal G, Feder T, Kenthapadi K, Motwani R, Panigrahy R, Thomas D, Zhu A (2005) Anonymizing tables. In: Proceedings of the tenth international conference on database theory, Edinburgh. Springer, Berlin, pp 246–258

Bayardo RJ (1998) Efficiently mining long patterns from databases. In: Proceedings of 1998 ACM-SIGMOD int. conf. on management of data, pp 85–93

Berge C (1989) Hypergraphs: combinatorics of finite sets. North-Holland

Boulicaut J-F, Bykowski A, Rigotti C (2003) A condensed representation of boolean data for the approximation of frequency queries. Data Mining Knowl Discov 7(1):5–22

Brin S, Motwani R, Ullman J, Tsur S (1997) Dynamic itemset counting and implication rules for market basket data. In: Proceedings of 1997 ACM-SIGMOD int. conf. on management of data, Tucson, Arizona, pp 255–264

Burdick D, Calimlim M, Gehrke J (2001) Mafia: a maximal frequent itemset algorithm for transactional databases. In: Proceedings ICDE 2001, pp 443–452

Calders T, Goethals B (2005) Depth-first non-derivable itemset mining. In: Proceedings of the 2005 SIAM international conference on data mining

Domingo-Ferrer J, Torra V (2005) Ordinal, continuous and heterogeneous k-Anonymity through microaggregation. Data Mining Knowl Discov 11(2):195–212

Dong G, Jiang C, Pei J, Li J, Wong L (2005) Mining succinct systems of minimal generators of formal concepts. In: Proceedings of the tenth international conference for database systems for advanced applications (DASFAA'05), pp 175–187

Elliot MJ, Dale A (1999) Scenarios of attack: the data intruder's perspective on statistical disclosure risk. Netherlands Official Stat 14:6–10

Elliot MJ, Skinner CJ, Dale A (1998) Special uniques, random uniques and sticky populations: Some counterintuitive effects of geographical detail on disclosure risk. Res Official Stat 1(2):53–67

Elliot MJ, Manning AM, Ford RW (2002) A computational algorithm for handling the special uniques problem. Int J Uncertainty, Fuzziness Knowl Based Syst 10(5):493–509

Elliot MJ, Manning A, Mayes K, Gurd J, Bane M (2005) SUDA: a program for identifying and grading special uniques. In: Proceedings of the Joint United Nations Economic Commission for Europe (UN-ECE) and European Statistics (Eurostat) Worksession on Statistical Confidentiality, Geneva, pp 353–362

Fienberg SE, Makov UE (1998) Confidentiality, uniqueness and disclosure limitation for categorical data. J Official Stat 4:385–397

Fienberg SE, Slavkovic AB (2005) Preserving the confidentiality of categorical statistical databases when releasing information for association rules. Data Mining Knowl Discov 11(2):155–180

Flouvat F, de Marchi F, Petit J-M (2004) ABS: adaptive borders search of frequent itemsets. In: Workshop on frequent itemset mining implementations (FIMI'04), In: conjunction with the IEEE International conference on data mining

Ghoting A, Otey ME, Parthasarathy S (2004) Loaded: link-based outlier and anomaly detection in evolving data sets. In: Proceedings of the fourth IEEE international conference on data mining, Brighton, UK, pp 387–390

Golle P (2006) Revisiting the uniqueness of simple demographics in the US population. In: Proceedings of the 5th ACM workshop on Privacy in electronic society, pp 77–80

Gouda K, Zaki MJ (2005) Genmax: an efficient algorithm for mining maximal frequent itemsets. Data Mining Knowl Discov 11(3):223–242

Gunopulos D, Khardon R, Mannila H, Saluja S, Toivonen H, Sharm RS (2003) Discovering all most specific sentences. ACM Trans Database Syst 28(2):140–174

Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceedings of the ACM SIGMOD conference, Dallas, Texas, pp 1–12

Hipp J, Güntzer U, Nakhaeizadeh G (2000) Algorithms for association rule mining—a general survey and comparison. SIGKDD Explor 2(1):58–64

Lin X, Clifton C, Zhu M (2005) Privacy-preserving clustering with distributed EM mixture modeling. Knowl Inform Syst 8(1):68–81

Liu G, Li J, Wong L, Hsu W (2006) Positive borders or negative borders: How to make lossless generator based representations concise. In: Proceedings of the 2006 SIAM international conference on data mining

Lucchese C, Salvatore O, Perego R (2004) kDCI: on using direct count up to the third iteration. In: Workshop on frequent itemset mining implementations (FIMI'04), in conjunction with the IEEE international conference on data mining

Lucchese C, Salvatore O, Perego R (2006) Fast and memory efficient mining of frequent closed itemsets. IEEE Trans Knowl Data Eng 18(1):21–36

Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M (2006) l-diversity: privacy beyond k-Anonymity. In: Proceedings of the 22nd IEEE international conference on data engineering, Atlanta, Georgia, USA

Manning AM, Haglin DJ (2005) A new algorithm for finding minimal sample uniques for use in statistical disclosure assessment. In: Proceedings of the fifth IEEE international conference on data mining, Houston, Texas, USA, pp 290–297

Mannila H, Toivonen H (1997) Levelwise search and borders of theories in knowledge discovery. Data Mining Knowl Discov 1(3):241–258

Merz G, Murphy P (1996) UCI repository of machine learning databases. Technical Report, University of California, Department of Information and Computer Science: http://www.ics.uci.edu/ mlearn/MLRepository.html

Muralidhar K, Sarathy R (1999) Security of random data perturbation methods. ACM Trans Database Syst 24(4):487–493

Pasquier N, Bastide Y, Taouil R, Lakhal L (1999) Efficient mining of association rules using closed itemset lattices. Inform Syst 24(1):25–46

Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027

Samarati P, Sweeney L (1998) Generalizing data to provide anonymity when disclosing information (Abstract). In: Proceedings of the seventeenth ACM symposium on principles of database systems, p 188

SAR1991 (1993) Office for National Statistics 1991 Great Britain Sample of Anonymised Records, Individual File [computer file] distributed by the Cathie Marsh Centre for Census and Survey Research, University of Manchester, 1993. Available at: http://www.ccsr.ac.uk/sars

SAR2001 (2004) Office for National Statistics 2001 Great Britain Sample of Anonymised Records, Individual File [computer file] distributed by the Cathie Marsh Centre for Census and Survey Research, University of Manchester, 2004. Available at: http://www.ccsr.ac.uk/sars

Singh A, Yu F, Dunteman G (2003) MASSC: A new data mask for limiting statistical information loss and disclosure. In: Joint ECE/EUROSTAT Worksession on Data Confidentiality. Luxembourg

Skinner CJ, Elliot MJ (2002) A measure of disclosure risk for microdata. J Roy Stat Soc Ser B 64:855–867

Skinner CJ, Holmes DJ (1998) Estimating the re-identification risk per record. J Official Stat 14(4):361–372

Skinner C, Marsh C, Openshaw S, Wymer C (1994) Disclosure control for census microdata. J Official Stat 10(1):31–51

Srikant R, Vu Q, Agrawal R (1997) Mining association rules with item constraints. In: Proceedings of the third international conferences on knowledge discovery and data mining (KDD). pp 67–73

Sweeney L (2002) k-anonymity: a model for protecting privacy. Int J Uncertainty, Fuzziness Knowl Based Syst 10(5):557–570

Truta TM, Fotouhi F, Barth-Jones D (2004) Assessing global disclosure risk in masked microdata. In: WPES '04: proceedings of the 2004 ACM workshop on privacy in the electronic society. ACM Press, pp 85–93

Uno T, Asai T, Uchida Y, Arimura H (2004) An efficient algorithm for enumerating closed patterns in transaction databases. In: Proceedings of the 7th international conference on discovery science, pp 16–31

Willenborg L, de Waal T (1996) Statistical disclosure control in practice, Lecture notes in statistics III. Springer, New York

Willenborg L, Waal T (2001) Elements of statistical disclosure control. Springer-Verlag, New York

Zaki MJ, Hsiao C-J (2002) CHARM: an efficient algorithm for closed itemset mining. In: Proceedings of the 2002 SIAM international conference on data mining