Exploring efficient privacy preserving technique for association rule data mining

Carolyn LaMacchia
Bloomsburg University of Pennsylvania

ABSTRACT

Databases of operational and customer activity are a critical resource for an organization not only to support business processes but also for use in strategic planning. The databases are analyzed within an organization and shared with trading partners to improve efficiency and direct marketing efforts. The frequent item set hiding problem is an area of active research to study approaches for hiding the sensitive knowledge patterns before disclosing the data for mining outside the organization. Several methods address hiding sensitive item sets including an exact approach that generates an extension to the original database that, when combined with the original database, limits the discovery of sensitive association rules without impacting the non-sensitive information. To generate the database extension, this method formulates a constraint optimization problem (COP). Solving the COP formulation is the dominant factor in the computational resource requirements of the exact approach. This research developed a heuristic that improves the performance of an exact hiding method by reducing the size of the COP formulation without significantly affecting the quality of the solutions generated. Results of the heuristic processing were compared with an existing exact approach in terms of size of database extension, ability to hide sensitive data, and impact on non-sensitive data.

Keywords: Association Rules, Big Data, Data mining, Data sharing, Frequent item set hiding, Privacy preserving,

INTRODUCTION

Typically the production and distribution of a product is a result of the cumulative value-adding effort of multiple organizations (Tsai, Rahgu, & Shao, 2013). This practice has changed management perspective from focusing on activities within an individual organization to focusing on activities of all organizations participating in a value chain. Effective management of product development, sourcing, production, and distribution tasks to achieve a sustainable competitive advantage for the value chain. Management of the value chain depends on shared information so that value chain partners may control the day-to-day movement of goods and material and coordinate long-term plans for sustainable profitability.

Since the sharing of information is critical to its success and operation, many value chain agreements include data sharing provisions. Despite the realization that the management of the value chain requires information, participating organizations are reluctant to share all of their data. As an example, organizations analyze their transaction purchasing data to determine customer buying behavior. Data mining of transaction data is used to discover association rules which identify products that are purchased together. The identification of association rules can be useful in decisions concerning product pricing, promotion activity, retail layout, and website design (Adamo, 2012; Bertino, Fovino, & Povenza, 2005). Data mining techniques that reveal this critical information to an organization may also be used by an outside organization who examine the shared data. As a result, the competitive efforts of an individual organization may be compromised. Organizations are looking for solutions to protect their sensitive association rules while participating in value chains and fulfilling the requirements of data sharing agreements (Bertino, et al., 2005; Giannotti, Lakshmanan, Monreale, Pedreschi, & Wang, 2013; Menon & Sarkar, 2007; Oliveira & Zaiana, 2002; Verkios et al., 2004).

The problem limiting the discovery of sensitive item sets in shared databases without impacting other non-sensitive information has been the focus of much research (Atallah, Elmagarmid, Ibrahim, Bertino, & Verykios, 1999; Atallah, et al., 2009; Evfimievski, Srikant, Agrawal, & Gehrke, 2004; Gkoulalas-Divanis, & Verykios, 2009a, 2009b; Menon, et al., 2005; Oliveira & Zaïane, 2002; Verykios, Elmagarmid, Bertino, Saygin, & Dasseni, 2004). This is described as the frequent item set and association rule hiding problem or frequent item set hiding (FIH) problem, for brevity (Aggarwal & Philip, 2008; Gkoulalas-Divanis, A. & Verykios, V. S,. 2009a). Frequent item sets are sets of items that reoccur in a database and identifying them is usually the first step toward association rule, correlation rule and sequential pattern mining (Gkoulalas-Divanis, & Verykios, 2009b; Menon, et. al., 2005).The frequent item set hiding problem refers to limiting the disclosure of sensitive rules that may be discovered in a shared database through data mining techniques. To avoid disclosing strategic association rules with external parties it is important to hide sensitive item sets before sharing data. The underlying NP-hard problem of hiding sensitive relationships before sharing databases is well established (Atallah, et al., 1999; Domadiya & Udai, 2013; Evfimievski, et al., 2004; Lin (2014); Oliveira & Zaïane, 2003; Saygin, Verykios, & Clifton, 2001; Stavropoulos, Verykios & Kagklis (2015); Verykios et al., 2004). Since its introduction, research in the frequent item set hiding problem has included many new approaches and improvements in existing approaches (Atallah, et al., 1999; Domadiya & Udai, 2013; Evfimievski, et al., 2004; Lin (2014); Oliveira & Zaïane, 2003; Saygin, Verykios, & Clifton, 2001; Stavropoulos, Verykios & Kagklis (2015); Verykios et al., 2004).

A number of approaches have been suggested to address the frequent item set hiding problem by limiting the disclosure of sensitive rules that may be discovered in a shared database through data mining techniques. There is active research in improving both the scalability and the quality of the techniques. Many of the proposed methodologies for association rule hiding are of a heuristic nature in order to effectively tackle the combinatorial nature of the problem. Heuristic approaches are efficient, fast algorithms that selectively sanitize a set of transactions from the original database to hide the sensitive association rules. Heuristic algorithms are attractive is their computational and memory efficiency which allows them to scale well to very large datasets. However, in general heuristic methodologies make locally optimal decisions that are usually not globally optimal.

Another category of approaches, known as border-based processes, are heuristic approaches that select item sets frequently occurring (statistically significant, positive border) and item sets infrequently occurring (statistically insignificant, negative border) for sanitation. The borders are used in an attempt to compress the representation of many item sets into one set (Gkoulalas-Divanis & Verykios, 2009b). In general, the quality of the borders directly impacts the quality of the sanitized database (Gkoulalas-Divanis & Verykios, 2009b).

The final category, exact hiding approaches comprise the most recent direction of research, Theses approaches hide item set associations through a constraint optimization problem that guarantees an optimal hiding solution, if it exists (Gkoulalas-Divanis & Verykios (2009b). Exact hiding approaches operate by transforming the hiding problem into an equivalent optimization problem, where the objective is to minimize the distortion that is cased to the original database to facilitate the hiding of all the sensitive knowledge with the least side-effects. Exact algorithms operate on the original database by considering all possible solutions for the problem in order to find the one that optimizes the criterion function. An exact approach typically provides a superior solution in terms of hiding sensitive item sets but is usually much slower than a heuristic approach because of the processing time required by the integer programming solver for a problem with many constraints (Gkoulalas-Divanis & Verykios, 2009b).

Gkoulalas-Divanis and Verykios (2009b) introduce the first exact methodology that strategically hides sensitive item sets by generating an extension to the original database that includes nonsensitive data. This approach applies an extension to the original database instead of modifying existing transactions. The extended potion of the database contains transactions that lower the importance of the sensitive patterns to where they become uninteresting from the perspective of the data mining algorithm while minimally affecting the importance of the nonsensitive ones. The hiding process maximizes the data utility of the sanitized database by introducing the least possible amount of side-effects. Extending the original database for sensitive item set hiding provides optimal solutions to an extended set of hiding problems, compared to previous approaches, as well as to lead to hiding solutions of typically superior quality (Gkoulalas-Divanis & Verykios, 2009b, 2010). To address scalability, Gkoulalas-Divanis and Verykios (2009b) suggest a few techniques including a database partitioning methodology and the application of a distributed processing approach to the solution of the COP.

This research developed a heuristic that addresses the scalability of the exact hiding method presented in Gkoulalas-Divanis and Verykios (2009b) without significantly affecting the quality of the solutions generated. In the Gkoulalas-Divanis and Verykios (2009b) method, the number of constraints in the COP formulation depends upon the number of item sets in the border of frequent and infrequent item sets. This research developed the substitution heuristic to reduce the computational cost of the COP process. The substitution heuristic replaces multiple items in the border of frequent and infrequent item sets with composite variables so that there are fewer constrains in the formulation of the COP when compared to the Gkoulalas-Divanis and Verykios (2009b) method. The heuristic was developed with the intention that there would be little impact to the quality of the result when compared to the Gkoulalas-Divanis and Verykios (2009b) approach.

As in the evaluation of Gkoulalas-Divanis and Verykios (2009b) exact hiding algorithm, the quality of the exact solutions generated by this research was assessed using the distance metric and quality standards for the item sets. Distance quantifies the number of items included in the COP generated database extension. A smaller value for distance is preferred because it indicates less "harm" to the original database by the sanitization process. Quality standards also consider the status of the item sets before and after the sanitation process. This means that all sensitive item sets are infrequent, frequent item sets remain frequent, infrequent item sets remain infrequent, and no new frequent item sets have been introduced to the sanitized database. In addition, experiments were analyzed to build a rationale for appropriate application of the heuristic.

EXPERIMENTAL DESIGN

This research developed the substitution heuristic to address the computational cost of an exact hiding algorithm. To assess the heuristic, the exact hiding algorithm of Gkoulalas-Divanis and Verykios (2009b) was recreated. A second version of this algorithm was implemented then modified to include the substitution heuristic processing. Problem instances were defined based on publically available datasets. Both algorithms processed using a set of problem instances. The processing results were compared and analyzed for performance and quality. In the discussion that follows, the recreation of the exact hiding algorithm of Gkoulalas-Divanis and Verykios (2009b) is referred to as the Exact Algorithm. The recreation of the exact hiding algorithm of Gkoulalas-Divanis and Verykios (2009b) that incorporates the substitution heuristic processing is referred to as the Substitution Algorithm.

Problem Instances

A problem instance is specified by a dataset ($\mathcal{D}_{\mathcal{O}}$), minimum frequency (mfreq), and set of sensitive item sets (S). Dataset ($\mathcal{D}_{\mathcal{O}}$) represents the original database of transactions that contains sensitive and non-sensitive relationships that may be discovered through data mining techniques. Support, discovered through an examination of the transactions, represents the measure of how frequent a relationship appears in the database. The minimum frequency mfreq refers to the threshold of interest in relationships; relationships in the data may be considered interesting if their support is greater than the mfreq. For example, mfreq = .3 means that item sets with support greater than or equal to .3 are considered frequent; item sets with support less than .3 are consider infrequent.

This research used the same real-world datasets publicly availability through the Frequent Item set Mining Implementations (FIMI) Repository and the Data Mining Forum that were used in the evaluation of Gkoulalas-Divanis & Verykios (2009a) including the Chess, Mushroom, and BMS-Webview-1 datasets (FIMF, 2014). The BMS-Webview-2 dataset was also used in the

Gkoulalas-Divanis and Verykios (2009b) research but was not available at the time of this study (FIMI, 2014).

Each dataset exhibits a unique distribution of item sets so investigation was required to select an appropriate minimum frequency (mfreq) for mining the dataset. The minimum frequency used to mine a dataset influences the number and support of item sets considered frequent and infrequent. The minimum frequency used in the problem instances for this study yielded a sufficient number of item sets that could be identified as sensitive and a small enough (in many cases) number of infrequent item sets so that the solver could process the COP formulation. A relationship exists between the mfreq and the number of items sets considered frequent and infrequent. When mining a dataset, the higher the mfreq, the smaller the number of frequent item sets and the larger the number of infrequent item sets. The longer the size of the sensitive items sets in terms of the number of items within the item set, the larger the size of the infrequent item sets.

The Exact Algorithm and the Substitution Algorithm were processed for each problem instance. As a result, there were two processes for each of the 18 problem instances for a total of 36 experiments. Table 1 (Appendix) describes the 18 problem instances used in experiments to compare The Exact Algorithm and the Substitution Algorithm. Support Category Low identifies sensitive item sets with support close to the minimum threshold while High identifies sensitive item sets with higher support values. The result of each experiment was evaluated in terms of quality and performance. Performance was evaluated both in terms of the number of constraints generated in the COP formulation and solver processing time where smaller numbers for these measurements is desirable. The quality evaluation considered the distance measurement and quality standards for frequent, infrequent, and sensitive item sets. Table 2 (Appendix) illustrates the data collected during each experiment.

Implementation of the Exact Algorithm is accomplished through a four-step process:

(1) Generation of item sets
(2) Identification of the sensitive item sets
(3) Identification of the frequent item sets and the infrequent items on the border between frequent and infrequent item sets,
(4) Formulation of the COP based on the items sets on the border between frequent and infrequent item sets.
(5) Execution of the solver.

Constraints required in the formulation of the COP are determined by exploiting the borders and the “cover” relationships among the frequent and infrequent item sets. Supersets of item sets are said to “cover” subsets of item sets. Constraints generated for the COP need only consider supersets. As a result, the number of constraints in the COP is greatly reduced yet the process provides the same solution as solving for the entire set of item sets (Gkoulalas-Divanis & Verykios, 2009b; Sun & Yu, 2005).

Implementation of the Substitution Algorithm is very similar process; two additional steps are require:

(1) Generation of item sets
(2) Identification of the sensitive item sets
(3) Identification of the frequent item sets and the infrequent items on the border between frequent and infrequent item sets
(3b) Generation of composite item sets based on the frequent and infrequent item sets identified in step 3.

(4) Formulation of the COP based on the items sets on the border between frequent and infrequent item sets.

(5) Execution of the solver.

(5b) Mapping of the composite item sets back to their original values.

Details of the logic of the Substitution Algorithm and the generation of the composite item sets follows.

Substitution Heuristic

Solver resource requirements increase with the number of item sets included in the COP formulation process. This study evaluates a second category of heuristics that finds instances for substituting a subset of items with a composite variable, so there are fewer item sets in the COP formulation process. The items are selected in a way that minimally affects the quality of the solution when compared to the solution derived by the Exact Algorithm. Presented with a fewer number of item sets, the Substitution Algorithm generates a COP formulation that includes fewer variables and constraints than the Exact Algorithm COP formulation. Based on a smaller COP, the solver processes the Substitution Algorithm formulation with less resources and generates a transaction set that includes the composite items. A mapping process reviews the solver solution and replaces each composite item in the transaction set with the original items to create the database extension, $\mathcal{D}_x$, that is close to the database extension created by the Exact Algorithm.

Candidate item sets for the substitution process are selected from the border of frequent and infrequent item sets in a way that minimally affects the quality of the solution. Candidate item sets satisfy the following conditions: the item set is not a sensitive item set and all occurrences of the item set follow a consistent pattern within the border of item sets. A consistent pattern is described as a subset of items that appear in the same item set of the positive border, and, appear with the same item(s) in all instances of the negative border. For example, given a positive border that includes item set {a, b, c}, and a negative border that includes item sets {a, d} and {b, d}. The subset {a, b} follows a consistent pattern since it appears in the same positive border item set, {a, b, c}, and all instances in the negative border are with the same item, d. Applying the Substitution Heuristic, candidate subset {a, b} is replaced by composite variable C_1. The positive border item set {a, b, c} is replaced with {C_1, c}, and the negative border item sets {a, d} and {b, d} are replaced with {C_1, d}. Based on a positive border that includes item set {a, b, c}, and a negative border that includes item sets {a, d} and {b, d}, the Exact Algorithm COP formulation generates 13 constraints for every transaction required to hide the sensitive item sets. By comparison, the Substitution Algorithm positive border that includes item set {C_1, c }, and the Substitution Algorithm negative border that includes item set {C_1, d }, the Substitution Algorithm COP formulation generates 8 constraints, about 40% fewer, for every transaction required to hide the sensitive item sets.

All item sets in the revised negative border that include only one item are candidates for the substitution heuristic. These items do not appear with any other item in either the revised positive or negative borders because, by definition, all supersets of negative border item sets are frequent item sets. For example, given a negative border that includes item sets {e} and {f}. Applying the Substitution Algorithm, candidates {e} and {f} are replaced by composite variable C_2 so the Substitution Algorithm negative border includes item set {C_2}. Based on this example, the Exact Algorithm COP formulation generates six constraints for every transaction required to hide the sensitive item sets. By comparison, the Substitution Heuristic COP formulation generates three constraints, 50% fewer, for every transaction required to hide the sensitive item sets.

Like the other COP formulation processes, the Substitution Algorithm COP formulation process calculates a value for threshold when generating constraints for the item sets of the positive and negative border. The value of threshold is determined for each item set in the positive and negative border through the same equation used in the Exact Algorithm (Gkoulalas-Divanis & Verykios, 2009b):

$$\text{Threshold} = mfreq \times (\mathcal{N} + \mathcal{Q} + \mathcal{SM}) - sup(I, \mathcal{D}_o)$$

In this equation, mfreq is the minimum frequency which is used to determine frequent and infrequent item sets; $\mathcal{N}$ is the number of transactions in $\mathcal{D}_o$; $\mathcal{Q}$ represents the minimum number of transactions required to hide the sensitive item sets; $\mathcal{SM}$ is the safety margin, and sup(I, $\mathcal{D}_o$) represents an item set's support which denotes the number of transactions in $\mathcal{D}_o$ that include the item set.

In some cases, it may be necessary to include a *safety margin*, SM, which is added to the calculated minimum value in determining an appropriate size of the database extension. Safety margins can be either predefined or dynamically computed based on properties of database D_0. Full analysis of the safety margins is outside the scope of this research. Preliminary experiments revealed that the solver could not reach an exact solution in some cases where the support of the sensitive item set was close to the border of the frequent and infrequent item sets without including a *safety margin*. Therefore, all experiments used a *safety margin* of 10 matching the *safety margin* of the Gkoulalas-Divanis and Verykios (2009b) experiments.

Applying the Substitution Algorithm, positive border item sets that include composite variables retain the same value for support, sup(I, $\mathcal{D}_o$), that the item set had prior to the substitution. As a result, the threshold calculated in the Substitution Algorithm COP formulation is the same as the Exact Algorithm COP formulation. Item sets in the negative border with composite variables may be created from more than one item set each with a different value for support, sup(I, $\mathcal{D}_o$). In this case, the item set with the composite variable adopts the support of the item set that has the lowest support of the candidate items. For example, given a positive border that includes item set {a, b, c} with sup(I, $\mathcal{D}_o$) = 32, and a negative border that includes item sets {a, d} where sup(I, $\mathcal{D}_o$) is 15 and {b, d} where sup(I, $\mathcal{D}_o$) is 20. Applying the Substitution Heuristic the positive border includes item set {C_1, c } with sup(I, $\mathcal{D}_o$) = 32, and the negative border item set{ C_1, d } with sup(I, $\mathcal{D}_o$) = 15.

Preliminary experiments with the Substitution Algorithm considered three options for selecting the appropriate support for the composite variable formed from the substituted candidate item sets in the negative border: the highest support, the lowest support, and the average support. There is an inverse relationship between the value of the support and the value of the threshold calculated by the COP formulation process. The COP formulation process calculates lower thresholds for item sets with higher support. Preliminary experiments where the Substitution Algorithm formulation used the highest support and the average support generated solutions that were larger than the solutions generated using the lowest support among the item sets. Since smaller values for solutions are preferred (representing the transactions in the database extension), the Substitution Algorithm COP formulation selects the lowest support among the negative border candidate item sets to calculate the threshold for the item set with the composite variable.

In the case of item sets in the revised negative border that include only one item, there are typically many candidates for substitution. Candidate item sets are combined until the sum of their individual support, sup(I, $\mathcal{D}_o$), is less than, but very close to, the minimum support threshold, msup. The minimum support threshold, msup = mfreq × $\mathcal{N}$, represents the minimum number of occurrences for an item set to be considered frequent. Since the Substitution Algorithm COP formulation is calculating a threshold for an (infrequent) item set in the negative border, the support should not exceed the minimum support threshold. Once this value is reached, a composite variable, representing the combined candidate item sets, is included in the formulation of the COP. The support for the composite variable for the COP formulation process becomes the sum of the support among the candidates. In the typical case, several composite variables are required for the candidate item sets in the revised negative border that include only one item.

Preliminary experiments with the Substitution Algorithm COP formulation considered several scenarios to generate composite variables and calculate support for single item negative border item sets. Less desirable results were reached in cases where single item negative border item sets and their support, sup(I, $\mathcal{D}_o$), were combined in a COP without considering the minimum support threshold, msup, of the original database, $\mathcal{D}_o$, which separates frequent and infrequent item sets. The solver failed to generate a solution in cases where a composite variable replaced multiple candidate item sets where the combined support was greater than the minimum support threshold. The solver generates larger solutions in cases where the smallest support of the candidate item sets or the average support was used in the threshold calculation. The solver generated the smallest solution (considered favorable) when a composite variable was substituted for candidate items until the sum of their individual support, sup(I, $\mathcal{D}_o$), was less than, but very close to, the minimum support threshold, msup.

The solver processes the problem formulated by the Substitution Algorithm formulation and generates a solution which includes a set of transactions representing the database extension, $\mathcal{D}_x$. The set of transactions includes instances of the composite variables which must be mapped back to each original item sets. The Substitution Algorithm Mapping process replaces composite variables in the solver solution with the original item sets based on the support, sup(I, $\mathcal{D}_o$), of each item set in the original database, $\mathcal{D}_o$. Replacements are based on the following equation:

Given: Set of candidate items substituted by composite variable C*1*: {i_1}, {i_2}, … { i_c} where *c* is the number of items and the support of each item is support($Im, \mathcal{D}o$)

The number of occurrences of composite variable mapped to an item becomes support($Im, \mathcal{D}o$) / $\sum_{m=1}^{c}$ support($Im, \mathcal{D}o$)

Note: Since partial transactions are not possible, numbers are rounded to the nearest whole number.

After the mapping process completes, the database extension, $\mathcal{D}x$, is combined with the original database, $\mathcal{D}o$, to form a new database, $\mathcal{D}$, which is now ready for sharing.

Quality standards of the Exact Algorithm and the Substitution Algorithm evaluate solutions in terms of the measurement of distance, the securing of sensitive item sets and the frequency of the item sets. Distance measurements using the Substitution Algorithm may be larger than solutions using the Exact Algorithm. The Exact Algorithm COP formulation is based on each item sets support, sup(I, $\mathcal{D}_o$); the Substitution Algorithm COP formulation is based on the support value calculated for each composite variable. The Substitution Algorithm COP formulation, by design, should secure sensitive item sets and maintain the frequency of the item sets. This means that when mining Database $\mathcal{D}$ at mfreq, all sensitive item sets are infrequent, nonsensitive frequent item sets remain frequent, infrequent item sets remain infrequent, and no new frequent item sets have been introduced to $\mathcal{D}$ that were not in $\mathcal{D}_o$. The Substitution Algorithm is illustrated in Table 3 (Appendix).

Resources

This study used real-world datasets publicly availability through the Frequent Item set Mining Implementations (FIMI) Repository and the Data Mining Forum (Data Mining Forum, 2014; FIMI, 2014). The datasets used for the evaluation of the study were the same as those used in Gkoulalas-Divanis and Verykios (2009b) and include BMS-WebView-1, Mushroom, and Chess. The BMS-Webview-2 dataset was also used in Gkoulalas-Divanis and Verykios (2009b) but was not available for this study.

The process was developed and experiments were conducted with a personal computer running with Windows 7 Professional 64-bit operating system and an Intel® Core™ i7 CPU, Q820, 1.73 GHz with 4 Gbytes of RAM. No other applications were processing during solver execution. Program code included the executable code of Boden (2005) Apriori Implementation, version 2.4.9, to assist in the identification of sensitive item sets. Program code was developed based on the C++ source code of a special implementation of the iZi algorithm provided by Flouvat (2013) to identify the border of frequent and infrequent item sets. All additional programs required for algorithm development and experiments were implemented in C++ programming code. Eclipse Juno Software Development Kit, version 4.2.1, was installed with JRE System Library [JavaSE-1.7] and IBM's ILOG CPLEX, version 12.4 to read the COP formulation, execute the solver, and provide the results of the COP.

In some cases, the COP formulation was too large for the processing by the solver as indicated by the error message: *java.lang.OutOfMemoryError: Java heap space* (Stack Overflow (2014). The limit of the solver's processing capability is directly related to the size of the COP formulation. The solver failed due to insufficient physical memory when attempting to process the COP formulations for the Chess dataset at higher support threshold and for all problem instances based on the BMS1-Webview dataset. Examining the operation system's performance statistics when the solver was processing large COP formulations revealed that the CPU usage was at 99% and the Physical Memory was at 99% just prior to the failure of the Java process. The solver processed the formulations based on the Chess dataset at the lower support threshold and for all problem instances based on the Mushroom dataset.

RESULTS

The performance and quality of the Exact Algorithm and the Substitution Algorithm were compared through experiments based on the same problem instance. Table 4 (Appendix) compares the results of the experiments. The table reveals the number of substitution candidates, the change in the number of constraints, solver runtime and the size of the database extension. The Substitution Algorithm formulates a COP that is very close to the size of the COP formulated by the Exact Algorithm. The solver run time is very close to the solver run time for the Exact Algorithm. The size of the database extension for solutions generated by the Substitution Algorithm are about 1% greater than solutions generated by the Hiding Algorithm.

The insignificant differences between the performances of the two algorithms can be explained by the fact that there were very few candidates for substitution in the Chess and Mushroom datasets. The COP formulation files for the Exact Algorithm and the Substitution Algorithm were very similar. However, the Substitution Algorithm COP formulations based on the BMS-WebView1 dataset were about 90% smaller than the Exact Algorithm. Unfortunately, the smallest Substitution Heuristic formulation of about 7 million constraints was too large for processing with available resources. A trend may be observed in the data. There are more candidate variables found with an increase in the number of sensitive item sets and when the support level increases from low to high.

The processing of the solver is the most resource intensive portion of both versions of the algorithms. The most restrictive resource is the memory available on the computer system during the solver process. For problem instances four, five, and six based on the Chess dataset and all problem instances based on the BMS-WebView1 dataset, the solver had insufficient physical memory to process any of the COP formulations.

Dataset Mining Implications

Each dataset exhibits a unique distribution of item sets so investigation is required to select an appropriate minimum frequency (mfreq) for mining the dataset. The minimum frequency used to mine a dataset influences the number and support of item sets in the border of item set. The minimum frequency used in the problem instances for this study yielded a positive border with sufficient item sets that could be identified as sensitive and a negative border that was small enough (in many cases) so that the solver could process the COP formulation. A relationship exists between the mfreq and the number of items sets in the border. When mining a dataset, the higher the mfreq, the smaller the number of item sets in the positive border and the larger the number of item sets in the negative border. The longer the size of the sensitive items sets in terms of the number of items within the item set, the larger the size of the item sets in the negative border.

Relationships are observed in data for all problem instances regardless of which algorithm was used to formulate the COP. The greater the number of sensitive item sets in a problem instance, the larger the COP formulation. The greater the length of the sensitive item sets in terms of the number of items within each item set, the larger the COP formulation. The higher the support of the sensitive item sets, the larger the COP formulation. The number of transactions required to hide the sensitive item sets (the value of$Q$) has a substantial impact on the number of constraints in the COP formulation. In addition, sensitive item sets with very high support values required the generation of more transactions to become hidden (higher values for$Q$). The higher the support of sensitive item sets, the higher the number of negative item sets leading to an increase in the number of constraints in the COP formulation.

Quality

As in the evaluation of the Exact Algorithm based on the Gkoulalas-Divanis and Verykios (2009b) algorithm, the quality of the exact solutions were assessed using the distance metric and quality standards for the item sets. The solver produced exact solutions for both algorithms but the Exact Algorithm consistently produced the ideal (smallest) solution in all experiments with the Chess and Mushroom datasets. The Substitution Algorithm results were consistently very close in size to Exact Algorithm's ideal solution. This is explained by that fact that the COP formulations of both algorithms were similar due; there were only a few candidates for substitution in the Chess and Mushroom datasets. The solver failed to process any problem instances for the BMS1-WebView dataset due to insufficient resources. Gkoulalas-Divanis and Verykios (2009b) do not report any solver results for this dataset.

The solver adjusts the variables representing the item sets of the COP in a way that all inequalities pertaining to an exact solution are satisfied, while the size of the database extension is minimized. In all cases where the solver produced a solution, all sensitive item sets were hidden (infrequent). All frequent item sets remained frequent; none were lost or hidden. All infrequent item sets remained infrequent, none were lost or became frequent. No new frequent item sets were added. In the case of the Substitution Algorithm processing, the occurrences of composite variables in the solver solution are mapped back to the original item sets to form the database extension. The mapping process confirmed that sensitive item sets were hidden and item sets maintained their appropriate status after the database extension is combined with the original database.

Performance

All versions of the hiding algorithms are resource intensive in two areas: the identification of the border of frequent and infrequent item sets and the processing of the solver. Preliminary and final experiments consistently revealed, for all datasets, that the lower the level of mfreq, the longer the processing time of the apriori-based algorithms. In addition, the longer the average record length of the dataset or the larger item set lattices, the greater the processing time of the apriori-based algorithms. Random samples of different number of records revealed that larger record counts leads to longer processing times. Random samples based on differences in the length of the item set lattice revealed that the more items in the lattice, the longer the processing times for determining the border of frequent and infrequent item sets.

The processing of the solver is the most resource intensive portion of all versions of the hiding algorithms. Solver processing times are longer for the larger COP formulations. The most restrictive resource for all of the hiding algorithms is the memory available on the computer system during the solver process. The limit of the solver's processing capability is directly related to the size of the COP formulation. The solver failed due to insufficient physical memory when attempting to process the COP formulations for the Chess dataset at higher support threshold (problem instances four, five, and six) and all of the BMS-WebView-1 COP formulations (problem instances thirteen through eighteen).

CONCLUSION AND FUTURE RESEARCH

The Substitution Algorithm produced promising results. The Algorithm produced results that were close to the Exact Algorithm. Future research could examine additional methods to evaluate datasets as ones that contain potential candidates for substitution items. Future research could also focus on discovery patterns or sections of a dataset that contain items that are isolated from the remaining items. These isolated item sets could lead to more comprehensive definition for item sets that are candidates for substitution.

All versions of the hiding algorithms are resource intensive in two areas: the identification of the border of frequent and infrequent item sets and the processing of the solver.

Early in the processing of the hiding algorithms, the border of frequent and infrequent item sets is identified, through an apriori-based algorithm, based on mfreq and a collection of sensitive item sets. This is the longest running portion of the entire hiding process in terms of elapsed real time. Identifying frequent item sets and the positive and negative border of item sets continues to be an active area of research. The performance of hiding algorithm is also dependent on the composition of the datasets. Evaluating datasets continues to be an active area of research.

The processing of the solver is the most resource intensive portion of all versions of the hiding algorithms. Solver processing times are longer for the larger COP formulations. The most restrictive resources for all of the hiding algorithms is the memory available on the computer system during the solver process. In problem instances where the solver did not have sufficient physical memory to process the larger COP formulations the process failed due to insufficient physical memory. Technology improvements in the solver and increase capability of hardware will improve solver performance.

REFERENCES

Adamo, J. M. (2012). Data mining for association rules and sequential patterns: sequential and parallel algorithms. Springer Science & Business Media.

Aggarwal, C. & Philip, S. (2008). *Privacy-preserving data mining: models and algorithms*. New York: Spring Science + Business Media.

Atallah, M., Elmagarmid, M., Ibrahim, M., Bertino, E., & Verykios, V. (1999) Disclosure Limitation of Sensitive Rule. *Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange, (KDEX '99).* (pp. 45-52). Washington, DC: IEEE Computer Society.

Bertino, E., Fovino, I.N., Povenza, L.P. (2005). A Framework for Evaluating Privacy Preserving Data Mining Algorithms. *Data Mining and Knowledge Discovery*, 11(2), 121–154.

Boden, F. (2005). APRIORI implementation of Ferenc Boden. Retrieved January 14, 2014, from http://www.cs.bme.hu/~bodon/en/apriori/.

Data Mining Forum. Retrieved June 4, 2014 from: http://forum.ai-directory.com/read.php?5,33

Domadiya, N. H., & Rao, U. P. (2013, February). Hiding sensitive association rules to maintain privacy and data quality in database. In Advance Computing Conference (IACC), 2013 IEEE 3rd International (pp. 1306-1310). IEEE.

Evfimievski, A., Srikant, R., Agrawal, R., & Gehrke, J. (2004). Privacy preserving mining of association rules. *Information Systems,* 29(4), 343-364. Oxford, UK: Elsevier Science Ltd.

Flouvat, F. (2013). Special Implementation of the iZi project that provides positive and negative border. Retrieved through an attachment to LaMacchia, C. May 6, 2013 student email account.

Flouvat,F, De Marchi, F., Petit, JM (2009). The iZi project: easy prototyping of interesting pattern mining algorithms. *New Frontiers in Applied Data Mining, PAKDD 2009 International Workshops,* Revised Selected Papers, LNCS 5669, ©Springer-Verlag, p. 1-15, 2009.

Frequent Itemset Mining Database Repository. Retrieved June 4, 2014 from: http:// fimi.ua.ac.be/data/

Giannotti, F., Lakshmanan, L. V., Monreale, A., Pedreschi, D., & Wang, H. (2013). Privacy-preserving mining of association rules from outsourced transaction databases. Systems Journal, IEEE, 7(3), 385-395.

Gkoulalas-Divanis, A. & Verykios, V. S. (2010). *Association Rule Hiding for Data Mining*. New York: Springer

Gkoulalas-Divanis, A. & Verykios, V. S. (2009a). An overview of privacy preserving data mining. *Crossroads*, 15(4), Article No. 6. New York: ACM.

Gkoulalas-Divanis, A. & Verykios, V. S. (2009b). Exact Knowledge Hiding through Database Extension. *IEEE Transaction on Knowledge and Data Engineering*, 21(5), 699–713.

IBM Academic Initiative. (2014). IBM ILOG CPLEX Optimization Studio, V12.4. Retrieved September 08, 2014 from: http://www 01.ibm.com/support/docview.wss?uid=swg21419058

Lin, C. W., Zhang, B., Yang, K. T., & Hong, T. P. (2014). Efficiently hiding sensitive itemsets with transaction deletion based on genetic algorithms. The Scientific World Journal, 2014.

Menon, S. & Sarkar, S. (2007). Minimizing Information Loss and Preserving Privacy. *Management Science*, 53(1), 101–116.

Menon, S., Sarkar, S., & Mukherjee. S. (2005). Maximizing accuracy of shared databases when concealing sensitive patterns. *Information Systems Research*, 16(3), 256–270.

Oliveira, S. & Zaïane, O. (2002). Privacy preserving frequent item set mining. *Proceedings of the IEEE international conference on Privacy, security and data mining – Volume 14.* (Maebashi City, Japan) (pp. 43 – 54). Darlinghurst, Australia: Australian Computer Society, Inc.

Oliveira, S. & Zaïane, O. (2003). Protecting Sensitive Knowledge by Data Sanitization. *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03).* (Melbourne, FL). (pp. 99-106). Washington, D.C.: IEEE computer Society.

Oliveira, S. & Zaïane, O. (2004). Toward standardization in privacy preserving data mining. *ACB SIGKDD 3rd Workshop on Data Mining Standards*. (pp. 7-17).

Saygin, Y., Verykios, V.S., & Clifton, C. (2001). Using Unknowns to Prevent Discovery of Association Rules. *ACM SIGMOD Record.* 30(4), 45-54.

Stack Overflow (2013). Hints to Improve Java Performance. Stack Exchanged Inc. Retrieved January 4, 2013 from: http://stackoverflow.com/questions/5660247/hints-to-improve-eclipse-performance

Stavropoulos, E. C., Verykios, V. S., & Kagklis, V. (2015). A transversal hypergraph approach for the frequent itemset hiding problem. Knowledge and Information Systems, 1-21.

Swamp, V., Seligman, L., & Rosenthal, A. (2006). Specifying data sharing agreements. *Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06).* (pp. 157-162).

Sun, X. & Yu, P.S. (2007). Hiding sensitive frequent item sets by a border-based approach. *Computer Science and Engineering.*1(1), (pp. 74-94).

Sun, X. & Yu, P.S. (2005). A Border-based Approach for Hiding Sensitive Frequent Item sets. *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05).* (pp. 426-433).

Tsai, J. Y., Raghu, T.S. & Shao, B.M. (2013). Information systems and technology sourcing strategies of e-Retailers for value chain enablement." Journal of Operations Management 31.6 (2013): 345-362.

Verykios, V., Elmagarmid, E., Bertino, E., Saygin, Y., & Dasseni, E. (2004). Association rule hiding. *IEEE Transactions on Knowledge and Data Engineering*. 16(4), 434-447 Piscataway, NJ, USA: IEEE Educational Activities Department.

* Acknowledgement: Sumitra Mukherjee, Ph.D, Dissertation Chair, Francisco Mitropoulos, Ph.D., Dissertation Committee Member, & Michael J. Lazio, Ph.D., Dissertation Committee Member, Nova Southeastern University, Ft. Lauderdale, FL, USA

APPENDIX

Table 1. Problem instances used in the experiments.

			Sensitive Item Sets		
No.	**Dataset**	***mfreq***	**Number**	**Items in item set**	**Support Category**
1	Chess	.6	5	7	Low
2	Chess	.6	10	7	Low
3	Chess	.6	15	7	Low
4	Chess	.6	5	7	High
5	Chess	.6	10	7	High
6	Chess	.6	15	7	High
7	Mushroom	.1	5	5	Low
8	Mushroom	.1	10	5	Low
9	Mushroom	.1	15	5	Low
10	Mushroom	.1	5	5	High
11	Mushroom	.1	10	5	High
12	Mushroom	.1	15	5	High
13	BMS	.005	5	1	Low
14	BMS	.005	10	1	Low
15	BMS	.005	15	1	Low
16	BMS	.005	5	1	High
17	BMS	.005	10	1	High
18	BMS	.005	15	1	High

Table 2. Data collected for each problem instance.

Algorithm identification: (Exact Algorithm or Substitution Algorithm)
Problem Instance number:
n_c: number of constraints in COP
s: solver solution status
t: solver run time
C_n: number of composite variables (Substitution Algorithm)
$\mathcal{D}x$: number of generated transactions
n_s: number of sensitive item sets that are not hidden
n_f: number of frequent item sets that became lost/infrequent
n_i: number of infrequent item sets that became lost/frequent

Table 3 – Substitution Algorithm example

Step	Candidate Generation Process
0	Given: • $\mathcal{N}$=150 • mfreq = .3 • Sensitive item set $\mathcal{S}_{\max}$ = {a,e} • $\mathcal{Q}$ is calculated as 11. • Partial representation of border of frequent and infrequent item sets with support appearing as superscript $\mathcal{Bd}^{+}(\mathcal{F}'_{D})$: $\{a,b\}^{52}$, $\{b, c, d, e\}^{54}$, $\{b, d, e, f, g\}^{63}$ $\mathcal{Bd}^{-}(\mathcal{F}'_{D})$: $\{a,c\}^{36}$, $\{a,d\}^{49}$, $\{a,e\}^{48}$, $\{a, f\}^{24}$, $\{a,g\}^{17}$, $\{c,f\}^{15}$, $\{c,g\}^{18}$, $\{h\}^{1}$, $\{i\}^{3}$, $\{j\}^{1}$
1	Items *f* and *g* follow a *consistent pattern* because they appear together in the {b, d, e, f, g} item set of the positive border and, when appearing separately in the negative border in item sets {a, f}, {a,g}, {c,f}, {c,g}, they are always paired with the same items: *a* and *c*. In addition, items h, i, and j are consider candidates; these include only one item and therefore, contain no supersets in the revised positive border.
2	Implementing the Substitution Heuristic, items *f* and *g* are represented by composite variable C_1 in the COP formulation and items *h*, *i*, and *j* are represented by composite variable, C_2, in the COP formulation. Within the revised positive border item set that includes C_1, the support value remains 63. The support value of the revised negative border item sets that includes C_1 are 17 and 15. The support value of the revised negative border item sets that include C_2 is 1.
3	Including the composite variables, the border of frequent and infrequent item sets for the COP formulation process becomes: $\mathcal{Bd}^{+}(\mathcal{F}'_{D})$: $\{a,b\}^{52}$, $\{b, c, d, e\}^{54}$, $\{b, d, e, C_1\}^{63}$ $\mathcal{Bd}^{-}(\mathcal{F}'_{D})$: $\{a,c\}^{36}$, $\{a,d\}^{49}$, $\{a,e\}^{48}$, $\{a, C_1\}^{17}$, $\{c, C_1\}^{15}$, $\{C_2\}^{1}$
	Mapping Process
0	After the solver process with a smaller COP formulation, all occurrences of C_1 and C_2 in the solver set of transactions are mapped back to the original item sets.
1	53% ((24 + 15) / (24 + 15 + 17 + 18)) of the C_1 variables set to one in the solver solution are mapped to item *f* and 47% ((17 + 18) / (24 + 15 + 17 + 18)) are mapped to item *g*.
2	20% (1 / (1 + 3 + 1) of the C_2 variables set to one in the solver solution are mapped to item *h*, 60% (3/ (1 + 3 + 1) to *i*, and 20% (1 / (1 + 3 + 1) to item *j*.

Table 4 Comparison of the Substitution Algorithm with Exact Algorithm

Problem Instance	Candidates	Constraints Δ	Solver Run Time Δ	Size of $\mathcal{D}x$ Δ
1: Chess, .6, 5 × 7, *Low*	13	(6 %)	(1 %)	1 %
2: Chess, .6, 10 × 7, *Low*	41	(3 %)	(1 %)	.5 %
3: Chess, .6, 15 × 7, *Low*	67	(2 %)	(3 %)	.9 %
4: Chess, .6, 5 × 7, *High*	126	(12 %)	(1)	(1)
5: Chess, .6, 10 × 7, *High*	151	(8 %)	(1)	(1)
6: Chess, .6, 15 × 7, *High*	164	(7 %)	(1)	(1)
7: Mushroom, .1, 5 × 5, *Low*	64	(1.7 %)	(1.8 %)	(1.6 %)
8: Mushroom, .1, 10 × 5, *Low*	86	(3.1 %)	(1.7 %)	(1.0 %)
9: Mushroom, .1, 15 × 5, *Low*	103	(2.2 %)	(1.3 %)	(1.0 %)
10: Mushroom, .1, 5 × 5, *High*	146	(2.4 %)	(1.9 %)	(1.0 %)
11: Mushroom, .1, 10 × 5, *High*	171	(2.2 %)	(0.1 %)	(1.6 %)
12: Mushroom, .1, 15 × 5, *High*	192	(2.9 %)	(0.3 %)	(1.0 %)
13: BMS-WebView1, .005, 5 × 1, *Low*	1937	(89.8%)	(1)	(1)
14: BMS-WebView1, .005, 10 × 1, *Low*	8436	(89.9%)	(1)	(1)
15: BMS-WebView1, .005, 15 × 1, *Low*	16031	(90.6%)	(1)	(1)
15: BMS-WebView1, .005, 5 × 1, *High*	35194	(91.1%)	(1)	(1)
17: BMS-WebView1, .005, 10 × 1, *High*	46357	(91.4%)	(1)	(1)
18: BMS-WebView1, .005, 15 × 1, *High*	58148	(92.1%)	(1)	(1)

(1) Solver did not process either Exact Algorithm or the Substitution Algorithm