Paradoxical Correlation Pattern Mining

Wenjun Zhou, *Member, IEEE*, Hui Xiong, *Senior Member, IEEE*, Lian Duan, Keli Xiao, and Robert Mee

Abstract—Given a large transactional database, correlation computing/association analysis aims at efficiently finding strongly correlated items. For traditional association analysis, relationships among variables are usually measured at a global level. In this study, we investigate confounding factors that can help to capture abnormal correlation behaviors at a local level. Indeed, many real-world phenomena are localized to specific markets or subpopulations. Such local relationships may not be visible or may be miscalculated when collectively analyzing the entire data. In particular, confounding effects that change the direction of correlation are a most severe problem because the global correlations alone leads to errant conclusions. To this end, we propose CONFOUND, an efficient algorithm to identify paradoxical correlation patterns (i.e., where controlling for a third item changes the direction of association for strongly correlated pairs) using effective pruning strategies. Moreover, we also provide an enhanced version of this algorithm, called CONFOUND+, which substantially speeds up the confounder search step. Finally, experimental results showed that our proposed CONFOUND and CONFOUND+ algorithms can effectively identify confounders and the computational performance is orders of magnitude faster than benchmark methods.

Index Terms—Correlation coefficient, correlation computing, partial correlation, local patterns, confounder

1 Introduction

Given a large transactional database, correlation computing is concerned with efficient identification of strongly correlated items. As a special case of association analysis, correlation computing has been a core problem in data mining and statistical modeling, and has been successfully applied to various application domains, including market-basket analysis [1], climate studies [2], public health [3], [4], and bioinformatics [5].

Previous work in association mining mainly focused on identifying association patterns in the entire dataset at a global level [6], [7], and little attention has been paid to finding associations being confounded by other items. Ignoring this problem can easily lead to the identification of false negatives and false positives. *False negatives* are patterns that are insignificant globally, but significant locally. For example, one drug might have a significant adverse drug reaction only in a certain subpopulation, but not be significant in the entire population. Ignoring such patterns, we may miss potential new discovery opportunities [8], [9]. *False positives* are patterns that are significant globally, but insignificant locally. Making decisions based on false positive global patterns will lead to ineffective, sometimes harmful, real-world decisions. One example of false positives is known as the confounding effect, where the observed pattern is spurious and attributable to their correlation to some other factors. This problem has been found in many applications, including biomedical and social network analysis [10], [11]. Of particular potential harm are confounders that reverse the correlation's direction. For example, when assessed globally, two items are positively correlated; but when controlling for a third item, the two items are in fact negatively correlated. (A more detailed example will be discussed in Section 2.1.) In this case, failing to find this control item leads to misunderstanding the true association between item pairs.

To this end, we study the problem of identifying *paradoxical correlation patterns*, which are defined as reversing confounders that change the sign of the correlation between a pair of other items. Having these patterns identified efficiently and making them available would be useful in a number of applications that involve large-scale data. A good example may be described in the context of recommender systems: when a customer is buying an MP3 player, the same MP3 player in a different color may be recommended because, in the past, many people bought them together as presents for their multiple children. However, if the customer also showed interest in a matching memory card, there is high chance that the MP3 is for personal use. In this case, instead of recommending the same MP3 player with a different color, we may have a better chance recommending a protector or ear phone. Having paradoxical correlation patterns found may help us make more relevant, context-aware recommendations. Our goal is to provide a computational method to efficiently discover a substantially reduced number of probable confounders, which may be further investigated or exploited by domain experts.

- *W. Zhou and R. Mee are with the Department of Business Analytics and Statistics, University of Tennessee, Knoxville, TN 37996-0532. E-mail: {wzhou4, rmee}@utk.edu.*
- *H. Xiong is with the Management Science and Information Systems Department, Rutgers University, Newark, NJ 07102. E-mail: hxiong@rutgers.edu.*
- *L. Duan is with the Department of Information Systems and Business Analytics, Hofstra University, Hempstead, NY 11549. E-mail: lian.duan@hofstra.edu.*
- *K. Xiao is with the College of Business, Stony Brook University, Stony Brook, NY 11794-3775. E-mail: keli.xiao@stonybrook.edu.*

Manuscript received 31 July 2017; revised 27 Nov. 2017; accepted 2 Jan. 2018. Date of publication 18 Jan. 2018; date of current version 5 July 2018. (Corresponding author: Hui Xiong.) Recommended for acceptance by Q. He. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2018.2791602

In this study, we use the ϕ correlation coefficient [12] as the measure of association. Traditional correlation computing techniques aimed at finding all pairs of items that are considered positively correlated when their ϕ correlation is above a user-specified threshold θ [13]. We tend to leverage this high correlation for recommendation since co-purchase is promising. However, that global correlation may be misleading because a pair $\{A, B\}$ may be negatively correlated when controlling for a third item C. In this case, the partial correlation of $\{A, B\}$ controlled for C may be below $-\theta$. Practically, this means that for those who bought C, they were unlikely to buy both A and B; and for those who did not buy C, they were unlikely to buy both A and B simultaneously, either. This is known as the Simpson's Paradox. Since C has changed the direction of the correlation between A and B, it is called a *reversing confounder* of item pair $\{A, B\}$. Our paradoxical correlation pattern mining problem is then executed as finding triplets of $\{A, B|C\}$, where C is a reversing confounder of strong pair $\{A, B\}$.

It is challenging to effectively and efficiently identify reversing confounders in large-scale datasets. Considering the correlation between any two items, controlled for a third item, we may need to check all possible three-item tuples, making the problem complexity as $O(n^3)$, and so a brute-force approach will not be scalable when the number of items n is very large. Since identifying pair co-occurrence is relatively more expensive, and the correlation coefficient of a pair may be required multiple times during the process of computing partial correlation, we may save pair-wise correlation values as intermediate results for future look-up. This can save substantial computation time at the cost of $O(n^2)$ memory space. However, for a data set with a large number of items, it may not be practical to meet such a huge memory need.

To discover potential confounders in a more effective and efficient manner (in terms of both time and space), in this paper, we studied properties of the partial correlation coefficient, and used these properties to evaluate the effect of potential confounders. We first described three necessary conditions for identifying confounders: the "more-extreme" condition, the "all-strong" condition, and the "compatible-sign" condition. These conditions enabled powerful pruning to reduce the number of triplets to check. Based on these conditions, a confounder search algorithm, called CONFOUND, was developed and published in a preliminary version of this paper [14]. The CONFOUND algorithm had two basic steps: strong pairs search and confounder search. With improvements in each step, this paper provided further enhancements to CONFOUND, as described in the CONFOUND+ algorithm. First, as the "all-strong" condition leads to the need of efficiently finding both positive and negative strong pairs, we provide an analysis of the search space for bidirectional all-strong pairs, which reduces the number of bounds to compute. Second, we derive a tight bound for reversing confounders, which helps improve the confounder search efficiency. Finally, we also store the positive and negative strong pairs separately, which fully leverages the power of compatible-sign condition in the confounder search step without repeated checking. Using a number of benchmark datasets, we show that both CONFOUND and CONFOUND+ can effectively and efficiently identify confounders while striking a balance between the use of time and memory space.

TABLE 1
Medical Example of Simpson's Paradox

	Trmt A ($T = 1$)		Trmt B ($T = 0$)		Corr.
	Treated	Recovered	Treated	Recovered	
More Sick	80	38%	200	50%	-0.11
Less Sick	200	85%	80	94%	-0.12
Any	280	71%	280	63%	0.09

The rest of this paper is organized as follows. In Section 2, we provide preliminaries, such as concepts, examples, notations, and computation formula, based on which we formulate the paradoxical correlation pattern mining problem. Section 3 focuses on the detection of confounders by studying the properties of the correlation and partial correlation coefficients. Based on such properties, our newly developed algorithms along with the baseline methods are described in Section 4. Section 5 summarizes experimental results with a focus on scalability. In Section 6, we discuss related work briefly. Finally, we draw conclusions and suggest future work in Section 7.

2 Paradoxical Correlation Patterns

This section will introduce the preliminaries, including concepts and notations, computational formula, and examples.

2.1 Simpson's Paradox

Simpson's paradox refers to the phenomenon where the overall association pattern is different from its counterpart in *each* local segment [15], [16]. Assuming that A, B, C are three different events (a bar on top meaning its complement), $P(\cdot)$ represents a probability, and $P(\cdot|\cdot)$ represents a conditional probability, the mathematical definition for Simpson's paradox may be written as follows.

Definition 1 (Simpson's Paradox). *It is possible to have* $P(A|B) < P(A|\bar{B})$, *and at the same time both* $P(A|BC) \geq P(A|\bar{B}C)$ *and* $P(A|B\bar{C}) \geq P(A|\bar{B}\bar{C})$.

Example 1 (Treatment Selection). Suppose that for a medical condition, there are two possible treatments A and B. We are primarily interested in their success rates as basis for deciding which treatment to give to a new patient. More specifically, we are looking at the correlation between two binary variables: treatment T (1, if using Treatment A; and 0 if using Treatment B) and outcome O (1, if recovered; and 0 otherwise). As illustrated in Table 1, considering all patients' data (see the last row), the recovery rate of Treatment A is 71 percent, and of Treatment B is 63 percent. Therefore, one may conclude that Treatment A is better than Treatment B.

However, if we divide the data by the severity status of the patient S (1, if more sick; and 0 if less sick), for either group we would find that Treatment B is in fact better than Treatment A (50 versus 38 percent for more sick patients and 94 versus 85 percent for less sick patients).

This example shows that it is possible that correlation have different directions on the aggregated data from local segments. In practice, it is essential that we identify such patterns and avoid making wrong choices.

TABLE 2
The Two-Way Contingency Table

	$B=1$	$B=0$	Total
$A=1$	N_{AB}	$N_{A\bar{B}}$	N_A
$A=0$	$N_{\bar{A}B}$	$N_{\bar{A}\bar{B}}$	$N_{\bar{A}}$
Total	N_B	$N_{\bar{B}}$	N

2.2 Correlation and Partial Correlation

Suppose we have a large, binary market-basket database $\mathcal{D} = \{T_1, T_2, \ldots, T_N\}$, where T_k is a transaction ($k = 1, 2, \ldots, N$). Each transaction includes a set of items from the item space $I = \{I_1, I_2, \ldots, I_M\}$. We have $T_k \subseteq I$ ($k = 1, 2, \ldots, N$), and $I = \bigcup_{k=1}^{N} T_k$. So in $\mathcal{D}$, there are N transactions involving M items.

Then, for any two items A and B, we can produce a two-way contingency table, as shown in Table 2. We can see that among the N transactions in $\mathcal{D}$, there are N_{AB} containing both items A and B, $N_{A\bar{B}}$ containing item A but not item B, so on and so forth. With such notations, the ϕ correlation coefficient between items A and B can be computed as

$$\phi_{AB} = \frac{NN_{AB} - N_A N_B}{\sqrt{N_A(N - N_A)N_B(N - N_B)}}. \quad (1)$$

Note that ϕ_{AB} is not defined if $N_A \in \{0, N\}$ or $N_B \in \{0, N\}$. This ϕ coefficient ranges between -1 and 1. Its sign indicates the direction of association, whether it is positive or negative. A higher absolute value indicates a stronger correlation, whereas a lower absolute value indicates a weaker correlation. Given a user-specified minimum correlation threshold $\theta, \theta \in (0, 1)$, we say items A and B are (strongly) correlated if $\phi_{AB} \geq \theta$ or $\phi_{AB} \leq -\theta$. Translated into correlation terms, the Simpson's paradox may be expressed as follows.

Definition 2 (Simpson's Paradox in Correlation). *For any three binary variables A, B, and C, it is possible to have $\phi_{AB} > 0$, and at the same time both $\phi_{AB|C=0} < 0$ and $\phi_{AB|C=1} < 0$.*

In Table 1, there were three binary attributes to be considered: treatment (T), outcome (O) and sickness status (S). As shown in the last row of the table, considering all patients, the correlation between treatment (whether receiving Treatment A) and outcome (whether recovered) is $\phi_{TO} = 0.09$. This indicates that the choice of Treatment A is positively correlated to recovery. When looking at the data by patient status, the correlation between treatment and outcome for more sick patients and less sick patients became $\phi_{TO|S=1} = -0.11$ and $\phi_{TO|S=0} = -0.12$, respectively. This indicates that the choice of Treatment A is negatively correlated to recovery for either group.

An effective measure for discovering patterns showing Simpson's paradox is the partial correlation [17]. Take the three-item scenario as an example: controlling for an item C, the partial correlation between items A and B can be computed as

$$\phi_{AB|C} = \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{\sqrt{(1 - \phi_{AC}^2)(1 - \phi_{BC}^2)}}. \quad (2)$$

Note that $\phi_{AB|C}$ is not defined if $\phi_{AC} = \pm 1$ or $\phi_{BC} = \pm 1$.

In fact, the correlation between the treatment and sickness status in Table 1 is $\phi_{TS} = -0.43$, and. the correlation between the outcome and sickness status is $\phi_{OS} = -0.44$. If we calculate the partial correlation between treatment and outcome, controlled for patient status, we would get $\phi_{TO|S} = -0.11$, which is negative. That means the sickness status serves as a confounding factor that would reverse the correlation between treatment and outcome.

2.3 Problem Formulation and Scope

In this paper, we formulate the problem of paradoxical correlation pattern mining as follows. Given the data described in Section 2.2, our goal was to find all patterns in the form of $\{A, B|C\}$, such that $A, B, C \in I$, and item C is a reversing confounder of item pair $\{A, B\}$. Formally, we have the following definition.

Definition 3 (Reversing Confounder). *Given the threshold $\theta, \theta \in (0, 1)$, item C is called a reversing confounder of $\{A, B\}$, if $\phi_{AB} \geq \theta$ and $\phi_{AB|C} \leq -\theta$.*

Even though the above definition may be easily extended to the opposite direction (i.e., $\phi_{AB} \leq \theta$ and $\phi_{AB|C} > -\theta$), we focus on reversing confounders for positively correlated pairs only for practical importance and for brevity. The extension to consider reversing confounders for negatively correlated pairs is relatively straightforward given symmetry of the problem.

Similar to traditional correlation computing and pattern mining settings, we assume that all items are coded as binary. We also assume that the ϕ correlation is used for measuring item correlation. Computational methods developed in this paper are based on specific properties of this measure, which may be a limitation. Extension to other association measures will be interesting for future work.

Moreover, we limit our scope to confounders identified by first order constraints on a binary database. In other words, the confounder of an item pair $\{A, B\}$ is identified by a single item C instead of a group of items. Higher order constraints may be developed by using AND and OR conjunctions when needed, however, speeding up that process is beyond the scope of this study. Association among multiple items (i.e., beyond item pairs) is not well-defined for the correlation measure, making the extension to multi-item correlation and its confounding non-trivial, and therefore, is also beyond the scope of this study.

3 Detection of Confounders

In this section, we develop useful results that will help us speed up the search of paradoxical correlation patterns.

3.1 Necessary Conditions of Reversing Confounders

In this section, we study the properties of reversing confounders. Specially, for an item pair $\{A, B\}$, we identify necessary conditions for any item C to be its confounder.

Lemma 1 (The "More-Extreme" Condition). *Suppose that an item pair $\{A, B\}$ is positively correlated (i.e., $\phi_{AB} \geq \theta$, $0 < \theta < 1$). An item C may be a reversing confounder of $\{A, B\}$ only if either*

$$\begin{cases} \phi_{AC} \geq \phi_{AB} \\ \phi_{BC} \geq \phi_{AB} \end{cases} \text{or} \begin{cases} \phi_{AC} \leq -\phi_{AB} \\ \phi_{BC} \leq -\phi_{AB}. \end{cases} \quad (3)$$

Proof. For a positively correlated item pair $\{A, B\}$, we have $\phi_{AB} \geq \theta$. If C is a reversing confounder of $\{A, B\}$, we have

$$\phi_{AB|C} = \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{\sqrt{(1-\phi_{AC}^2)(1-\phi_{BC}^2)}} \leq -\theta, \quad (4)$$

from which we have

$$\frac{\phi_{AC}\phi_{BC} - \phi_{AB}}{\sqrt{(1-\phi_{AC}^2)(1-\phi_{BC}^2)}} \geq \theta.$$

Therefore

$$\phi_{AC}\phi_{BC} - \phi_{AB} \geq \theta\sqrt{(1-\phi_{AC}^2)(1-\phi_{BC}^2)} > 0. \quad (5)$$

Since $\phi_{AC}\phi_{BC} \in (-1, 1)$ and $\phi_{AB} \geq \theta > 0$, from (5) we have

$$0 < \phi_{AB} < \phi_{AC}\phi_{BC} < 1.$$

Therefore, a necessary condition for (4) would be (3). □

Lemma 2 (The "All-Strong" Condition). *Item C may be a reversing confounder of $\{A, B\}$ only if A, B, and C are pairwise correlated (either positively or negatively).*

Proof. If C is a reversing confounder of $\{A, B\}$, then by definition we have $\phi_{AB} \geq \theta$. In addition, from the necessary condition derived in Lemma 1, we have either

$$\begin{cases} \phi_{AC} \geq \phi_{AB} \geq \theta \\ \phi_{BC} \geq \phi_{AB} \geq \theta \end{cases} \text{or} \begin{cases} \phi_{AC} \leq -\phi_{AB} \leq -\theta \\ \phi_{BC} \leq -\phi_{AB} \leq -\theta \end{cases}.$$

So pairs $\{A, C\}$ and $\{B, C\}$ are also correlated. □

Lemma 3 (The "Compatible-Sign" Condition). *For any three items A, B, and C that are pairwise correlated (either positively or negatively), none of them will be a reversing confounder if $\phi_{AB}\phi_{AC}\phi_{BC} \leq 0$.*

Proof. If C is a reversing confounder of $\{A, B\}$, then by definition we have $\phi_{AB} \geq \theta > 0$. In addition, from the necessary condition derived in Lemma 2, we have either

$$\begin{cases} \phi_{AC} \geq \phi_{AB} > 0 \\ \phi_{BC} \geq \phi_{AB} > 0 \end{cases} \text{or} \begin{cases} \phi_{AC} \leq -\phi_{AB} < 0 \\ \phi_{BC} \leq -\phi_{AB} < 0 \end{cases}.$$

In either case, we have $\phi_{AB}\phi_{AC}\phi_{BC} > 0$. □

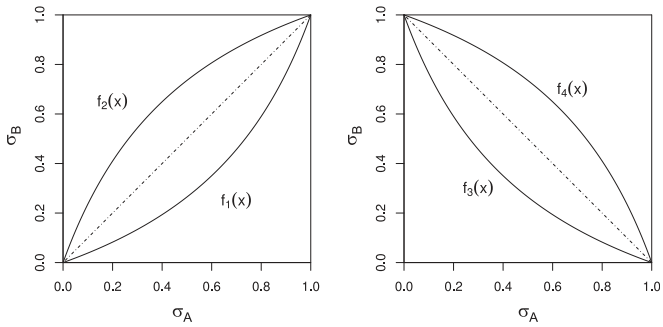
These conditions will be useful for effective pruning when designing efficient algorithms.

3.2 Bidirectional All-Strong-Pairs (BASP) Search

The "All-Strong" condition would require the availability of all strong pairs, including positive and negative pairs. We call the problem of finding both positive and negative strong pairs as a bidirectional all-strong-pairs search. Formally, we have the following definition.

Definition 4 (BASP Search). *Given θ, find all pairs that either $\phi \geq \theta$ or $\phi \leq -\theta$.*

In [13], the all-strong-pairs (ASP) query problem was formulated as finding all positive strong pairs only. An extension is needed when we need to include strong negative pairs. To achieve this goal, a few results regarding correlation range query may be useful. These results are summarized in Lemma 4, and the proof and other details may be found in [18].



(a) $upper(\phi_{AB}) \geq \theta$ (b) $lower(\phi_{AB}) \leq -\theta$

Fig. 1. Search space for all strong pairs.

Lemma 4 (Correlation Range Query). *Given a user-specified minimum correlation threshold θ ($0 < \theta < 1$), for any two items A and B, whose support values are σ_A and σ_B, respectively, item B may be positively strongly correlated with A only if*

$$f_1(\sigma_A) \leq \sigma_B \leq f_2(\sigma_A), \quad (6)$$

where

$$f_1(\sigma_A) = \frac{\theta^2\sigma_A}{(1-\sigma_A) + \theta^2\sigma_A}, \quad (7)$$

$$f_2(\sigma_A) = \frac{\sigma_A}{\sigma_A + \theta^2(1-\theta^2)}; \quad (8)$$

and negatively strongly correlated with A only if

$$f_3(\sigma_A) \leq \sigma_B \leq f_4(\sigma_A), \quad (9)$$

where

$$f_3(\sigma_A) = \frac{\theta^2(1-\sigma_A)}{\sigma_A + \theta^2(1-\sigma_A)}, \quad (10)$$

$$f_4(\sigma_A) = \frac{1-\sigma_A}{(1-\sigma_A) + \theta^2\sigma_A}. \quad (11)$$

Using σ_A as the x-axis and σ_B as the y-axis, we may visualize item pairs in a $[0, 1] \times [0, 1]$ region. Specifically, the ranges mentioned in Lemma 4 may be visualized as the solid lines in Fig. 1. The region between the solid lines would be the search space for strong pairs. The extent of bending (and thus the size of the search space) is determined by the parameter θ. In Fig. 1a, any item pair that lies between the $f_1(x)$ and $f_2(x)$ curves is a candidate positive strong pair. The dashed line represents the case when $\sigma_A = \sigma_B$, about which the search space is symmetric. In Fig. 1b, any item pair that lies between the $f_3(x)$ and $f_4(x)$ curves is a candidate negative strong pair. The dashed line represents the case when $\sigma_A + \sigma_B = 1$, about which the search space is symmetric.

When searching for BASPs, we overlay the positive bounds and negative bounds and illustrated the overall search space (shaded) in Fig. 2. Note that we only shaded

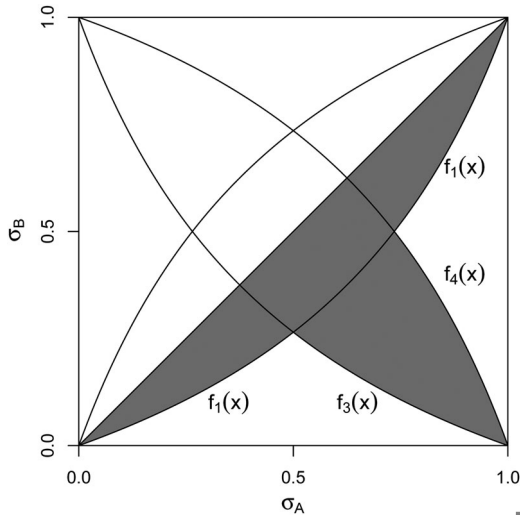


Fig. 2. Search space for all strong pairs.

regions in the lower right half of the square, because this is sufficient for enumerating all candidate pairs due to symmetry (i.e., $\phi_{AB} = \phi_{BA}$). It is then easy to see Theorem 1.

Theorem 1 (Search Regions for BASP). *For any given item A, whose support value is σ_A, the search space for item B is all those such that*

$$s_B \in \begin{cases} [f_1(\sigma_A), \sigma_A] & \text{if } \sigma_A \le 0.5; \\ [f_3(\sigma_A), \sigma_A] & \text{if } 0.5 < \sigma_A \le \frac{1}{1+\theta^2}; \\ [f_3(\sigma_A), f_4(\sigma_A)] & \\ \quad \cup [f_1(\sigma_A), \sigma_A] & otherwise. \end{cases} \tag{12}$$

Proof. The proof is omitted since the search space is easy to tell from Fig. 2, and the cutoff values may be easily solved from the equations of bounds in Lemma 4. □

3.3 Tight Bound for Reversing Confounders

The "More-Extreme" condition may be considered as a loose bound for pruning. In this section, we will identify a tight bound and derive a few more pruning rules.

Lemma 5 (First Lower Bound of Partial Correlation). *A lower bound for $\phi_{AB|C}$ is*

$$lower(\phi_{AB|C}) = \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{1 - \phi_{AC}\phi_{BC}}. \tag{13}$$

Proof. Since $(\phi_{AC} - \phi_{BC})^2 = \phi_{AC}^2 + \phi_{BC}^2 - 2\phi_{AC}\phi_{BC} \ge 0$, we can see that $\phi_{AC}^2 + \phi_{BC}^2 \ge 2\phi_{AC}\phi_{BC}$. We have

$$\begin{aligned} \phi_{AB|C} &= \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{\sqrt{(1-\phi_{AC}^2)(1-\phi_{BC}^2)}} \\ &= \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{\sqrt{1 - (\phi_{AC}^2 + \phi_{BC}^2) + \phi_{AC}^2\phi_{BC}^2}} \\ &\ge \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{\sqrt{1 - 2\phi_{AC}\phi_{BC} + \phi_{AC}^2\phi_{BC}^2}} \\ &= \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{1 - \phi_{AC}\phi_{BC}}, \end{aligned}$$

the right hand side of which is a lower bound for $\phi_{AB|C}$. □

Lemma 6 (Second Lower Bound of Partial Correlation). *Assume that $|\phi_{AC}| \le |\phi_{BC}|$ without loss of generality, a lower bound for $\phi_{AB|C}$ is*

$$lower(\phi_{AB|C}) = \frac{\phi_{AB} - \phi_{BC}^2}{1 - \phi_{BC}^2}. \tag{14}$$

Proof. Since $|\phi_{AC}| \le |\phi_{BC}|$, it is easy to see that $\phi_{AC}^2 \le \phi_{BC}^2$. Moreover, since $(\phi_{AC} - \phi_{BC})^2 = \phi_{AC}^2 + \phi_{BC}^2 - 2\phi_{AC}\phi_{BC} \ge 0$, we can see that $\phi_{AC}\phi_{BC} \le \frac{1}{2}\left(\phi_{AC}^2 + \phi_{BC}^2\right) \le \phi_{BC}^2$. We have

$$\begin{aligned} \phi_{AB|C} &\ge \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{1 - \phi_{AC}\phi_{BC}} \text{ (by Lemma 5)} \\ &= 1 - \frac{1 - \phi_{AB}}{1 - \phi_{AC}\phi_{BC}} \\ &\ge 1 - \frac{1 - \phi_{AB}}{1 - \phi_{BC}^2} = \frac{\phi_{AB} - \phi_{BC}^2}{1 - \phi_{BC}^2}, \end{aligned}$$

the right hand side of which is a lower bound for $\phi_{AB|C}$. □

Theorem 2 (Tight Bounds for Reversing Confounder of Positive Pairs). *Suppose that an item pair $\{A, B\}$ is positively correlated (i.e., $\phi_{AB} \ge \theta$) ($0 < \theta < 1$). Assuming that $|\phi_{AC}| \le |\phi_{BC}|$ without loss of generality, an item C may be a reversing confounder of $\{A, B\}$ only if either*

$$\begin{cases} \phi_{AC} \ge \delta_{AB}^2 \\ \phi_{BC} \ge \delta_{AB} \end{cases} \text{ or } \begin{cases} \phi_{AC} \le -\delta_{AB}^2 \\ \phi_{BC} \le -\delta_{AB} \end{cases}, \tag{15}$$

where $\delta_{AB} = \sqrt{\frac{\phi_{AB}+\theta}{1+\theta}}$.

Proof. By definition, in order for C to be a confounder of $\{A, B\}$, we require $\phi_{AB|C} < -\theta$. As a result, the lower bound of $\phi_{AB|C}$ (e.g., as derived in Lemmas 5 or 6) has to be less than $-\theta$.

First, using the bound derived in Lemma 6, we have

$$lower(\phi_{AB|C}) = \frac{\phi_{AB} - \phi_{BC}^2}{1 - \phi_{BC}^2} < -\theta,$$

from which we have

$$\phi_{BC}^2 > \frac{\phi_{AB} + \theta}{1 + \theta}. \tag{16}$$

Furthermore, using the bound derived in Lemma 5, we have

$$lower(\phi_{AB|C}) = \frac{\phi_{AB} - \phi_{AC}\phi_{BC}}{1 - \phi_{AC}\phi_{BC}} < -\theta,$$

from which we have

$$\phi_{AC}\phi_{BC} > \frac{\phi_{AB} + \theta}{1 + \theta}.$$

If $\phi_{BC} > 0$, then

$$\phi_{AC} > \frac{\phi_{AB} + \theta}{(1+\theta)\phi_{BC}} \ge \frac{\phi_{AB} + \theta}{1 + \theta}; \tag{17}$$

If $\phi_{BC} < 0$, then

$$\phi_{AC} < \frac{\phi_{AB} + \theta}{(1+\theta)\phi_{BC}} \le -\frac{\phi_{AB} + \theta}{1 + \theta}. \tag{18}$$

Combining (16), (17), and (18), we can come to (15). □

Based on the tight bound derived above, we prove the following corollaries, which will be used for speeding up our algorithm.

Corollary 1 (Reversible). *If C is a reversing confounder of $\{A, B\}$, then among the pairwise correlations ϕ_{AB}, ϕ_{AC}, and ϕ_{BC}, ϕ_{AB} has the smallest absolute value.*

Proof. This corollary follows directly from the necessary condition derived in Lemma 1. □

Corollary 1 indicates that for any triplets, only the pair with the least absolute of correlation may be reversed by the third item. Therefore, whenever we know that three items are pairwise strongly correlated, we only need to check whether the weakest pair is reversed by the third item. We do not need to check the other two possibilities.

Corollary 2 (Irreversible Triplet). *Suppose that A, B, and C are pairwise positively correlated, and that ϕ_{AB} is not the smallest among the pairwise correlations. If $\phi_{AB} < \frac{2\theta}{1+\theta}$, then no reversing confounders will be found among A, B, and C.*

Proof. From Corollary 1 we know that ϕ_{AB} will not be reversed since it does not have the smallest absolute value. If $\phi_{AC} < \phi_{BC}$, we shall only have to check if ϕ_{AC} is reversed by item B; otherwise we shall only have to check if ϕ_{BC} is reversed by item A.

From Theorem 2, we know that item B may be a reversing confounder of $\{A, C\}$ only if either

$$\begin{cases} \phi_{AB} \geq \delta_{AC}^2 \\ \phi_{BC} \geq \delta_{AC} \end{cases} \text{ or } \begin{cases} \phi_{AB} \geq \delta_{AC} \\ \phi_{BC} \geq \delta_{AC}^2 \end{cases}, \tag{19}$$

where

$$\delta_{AC} = \sqrt{\frac{\phi_{AC} + \theta}{1+\theta}} \geq \sqrt{\frac{2\theta}{1+\theta}}. \tag{20}$$

From Eqs. (19) and (20), we find the following necessary condition:

$$\phi_{AB} \geq \delta_{AC}^2 \geq \frac{2\theta}{1+\theta}. \tag{21}$$

Symmetrically, we can find that a necessary condition for item A to be a reversing confounder of $\{B, C\}$ will be

$$\phi_{AB} \geq \delta_{BC}^2 \geq \frac{2\theta}{1+\theta}. \tag{22}$$

The proof detail is omitted due to similarity. □

Corollary 3 (Irreversible Pair). *Suppose that $\{B, C\}$ is a positive strong pair, and $\phi_{AB} \leq \phi_{AC} \leq -\theta$. Then the pair $\{B, C\}$ cannot be reversed if $\phi_{AB} > -\sqrt{\frac{2\theta}{1+\theta}}$ or $\phi_{AC} > -\frac{2\theta}{1+\theta}$.*

Proof. From Theorem 2, we know that item A may be a reversing confounder of $\{B, C\}$ only if

$$\begin{cases} \phi_{AC} \leq -\delta_{BC}^2 \\ \phi_{AB} \leq -\delta_{BC} \end{cases}, \tag{23}$$

where $\delta_{BC} = \sqrt{\frac{\phi_{BC}+\theta}{1+\theta}} \geq \sqrt{\frac{2\theta}{1+\theta}}$. Therefore, the necessary conditions are that

$$\phi_{AB} \leq -\delta_{BC} \leq -\sqrt{\frac{2\theta}{1+\theta}}, \tag{24}$$

and that

$$\phi_{AC} \leq -\delta_{BC}^2 \leq -\frac{2\theta}{1+\theta}. \tag{25}$$

Note that both conditions have to be met. □

4 Algorithm Descriptions

In this section, we will describe algorithms for identifying confounders from a transactional database. We will first describe two straightforward solutions as the baseline, then the CONFOUND algorithm we have designed based on the initial set of necessary conditions. Finally, we will describe CONFOUND+, an enhanced version of CONFOUND with further pruning due to the newly found tight bound.

For each algorithm described in this section, we assume that the input variables are I, the reverse indexed transactional database (i.e., a list of item IDs followed by all transaction IDs in which the item was included); and θ, the minimum correlation threshold. The output are patterns in the format $\{A, B|C\}$, where item C is a reversing confounder for positively correlated item pair $\{A, B\}$. For brevity, by default A, B, C represent items, and a, b, and c represent their respective indices in I.

4.1 Baseline Algorithms

In this section, we describe two straightforward solutions as the baseline.

First, a "brute-force" approach is described in Algorithm 1. The loops starting in Lines 2 and 3 ensure that we iterates on each possible item pair $\{A, B\}$. The fact that items were sorted by decreasing support (Line 1) ensures that $\sigma_A \geq \sigma_B$. Despite the name "brute-force", we performed basic pruning in Line 4, which is based on a computation-friendly upper bound of ϕ [19]. If the upper bound is above the threshold θ, we compute the exact correlation ϕ_{AB} (Line 5); otherwise, there is no need to compute the exact correlation. In fact, if the upper bound of ϕ_{AB} is below the threshold θ, no strong pair will be found between A and B', $\forall B'$ s.t. $\sigma_{B'} \leq \sigma_B$ [13]. This is why we may break from the inner loop in Line 4. When A and B are not positively correlated, there is no need to search for their confounders (Line 6). Otherwise, a third loop starting from Line 7 iterates on each remaining item C, compute its correlations to items A and B, respectively (Lines 9-10), and then check the partial correlations to see if the pair $\{A, B\}$ is reversed (Lines 11-12).

Note that the computation of pairwise correlation in Algorithm 1 has lots of overlap. For instance, the correlation of ϕ_{AC} in Line 9 may have to be computed again in Line 5 in a later iteration. The computation of the exact value of ϕ correlation proves to be expensive [19]. Thus, it is desirable to store pairwise correlation for repeated use, which motivates us to design a save-and-lookup approach, as described in Algorithm 2.

Algorithm 2 has two phases. In the initialization phase (Phase I, Lines 1-4), the correlation of each possible item pair is computed and saved for later use. Then, in the confounder search phase (Phase II, Lines 5 through 14), we iterate over each possible item pair $\{A, B\}$ and check against each

remaining item C, to see if C is a confounder of $\{A, B\}$. The difference of this process from Algorithm 1 is that, instead of re-computing the ϕ correlations, we simply look them up from the pre-computed results. This is an approach to alleviate the computation cost by making use of more memory space.

Algorithm 1. The Brute-Force Approach

Input: I: an item list representing a reverse indexed transactional database; θ: a user-specified minimum correlation threshold.
Output: all reversing confounder patterns.

1 sort I by non-increasing item support;
2 **for** $a \leftarrow 1$ to $(M-1)$ **do**
3 **for** $b \leftarrow (a+1)$ to M **do**
4 **if** $upper(\phi_{AB}) < \theta$ **then break**;
5 find N_{AB} and calculate ϕ_{AB};
6 **if** $\phi_{AB} < \theta$ **then continue**;
  // Find all reversers of strong pair $\{A, B\}$
7 **for** $c \leftarrow 1$ to M **do**
8 **if** $C = A$ or $C = B$ **then continue**;
9 find N_{AC} and calculate ϕ_{AC};
10 find N_{BC} and calculate ϕ_{BC};
11 calculate $\phi_{AB|C}$;
12 **if** $\phi_{AB|C} \leq -\theta$ **then** output $\{A, B|C\}$;
13 **end**
14 **end**
15 **end**

Algorithm 2. The Save-and-Lookup Approach

Input: I: an item list representing a reverse indexed transactional database; θ: a user-specified minimum correlation threshold.
Output : all reversing confounder patterns.

  // Phase I: Compute and save pair-wise correlations
1 pre-allocate fixed storage space for all pairs;
2 **foreach** item pair $\{A, B\}$ **do**
3 find N_{AB}, compute ϕ_{AB}, and save it;
4 **end**
  // Phase II: Search for reversing confounder patterns
5 **foreach** item pair $\{A, B\}$ **do**
6 look up ϕ_{AB};
7 **if** $\phi_{AB} < \theta$ **then continue**;
  // Find all reversers of strong pair $\{A, B\}$
8 **for** $c \leftarrow 1$ to M **do**
9 **if** $c = a$ **or** $c = b$ **then continue**;
10 look up ϕ_{AC} and ϕ_{BC} by index;
11 calculate $\phi_{AB|C}$;
12 **if** $\phi_{AB|C} \leq -\theta$ **then** output $\{A, B|C\}$;
13 **end**
14 **end**

4.2 The CONFOUND Algorithm

The above two baseline methods present the two extremes in time and space usage for discovering confounders. With the simple necessary conditions developed in Section 3.1, we were able to design a new algorithm, called CONFOUND, which provides a balance between computation time and memory space.

The CONFOUND algorithm is described in Algorithm 3. Similar to Algorithm 2, CONFOUND has two phases: correlation computing (Lines 1 through 10) and confounder searching (Lines 11 through 36). In the correlation computing phase, our goal was to find and save all (i.e., positively or negatively) correlated pairs. Here we make use of both the upper and lower bounds of ϕ [13] for speeding up the computing (Line 3), as we need to find not only positive but also negative pairs. Specifically, if a pair's upper and lower bounds are not large enough (by absolute value), there is no chance for its actual correlation value to be strong. In this case, we can safely skip the computation of the exact correlation value for the pair.

Algorithm 3. CONFOUND

Input: I: an item list representing a reverse indexed transactional database; θ: a user-specified minimum correlation threshold.
Output: all reversing confounder patterns.

  // Phase I: Identify all strong pairs
1 initialize L as an empty linked list;
2 **foreach** item pair $\{A, B\}$ **do**
3 **if** $upper(\phi_{AB}) \geq \theta$ or $lower(\phi_{AB}) \leq -\theta$ **then**
4 find N_{AB} and compute ϕ_{AB};
5 **if** $\phi_{AB} = \pm 1$ **then continue**;
6 **if** $\phi_{AB} \geq \theta$ or $\phi_{AB} \leq -\theta$ **then**
7 save pair $\{A, B\}$ to L;
8 **end**
9 **end**
10 **end**
  // Phase II: Identify reversing confounders
11 **foreach** $a \in L.keys()$ **do**
12 $l \leftarrow L[a].length()$;
13 **if** $l < 2$ **then continue**;
14 **for** $i \leftarrow 1$ to $(l-1)$ **do**
15 $(b, \phi_{AB}) \leftarrow L[a][i]$;
16 **if** $b \notin L.keys()$ **then continue**;
17 **for** $j \leftarrow (i+1)$ to l **do**
18 $(c, \phi_{AC}) \leftarrow L[a][j]$;
19 look up ϕ_{BC} from L;
20 **if** ϕ_{BC} is not found **then continue**;
21 **if** $\phi_{AB}\phi_{BC}\phi_{AC} \leq 0$ **then continue**;
22 calculate $\phi_{AB|C}$;
23 **if** $\phi_{AB} \geq \theta$ and $\phi_{AB|C} \leq -\theta$ **then**
24 **output** $\{A, B|C\}$
25 **end**
26 calculate $\phi_{AC|B}$;
27 **if** $\phi_{AC} \geq \theta$ and $\phi_{AC|B} \leq -\theta$ **then**
28 output $\{A, C|B\}$
29 **end**
30 calculate $\phi_{BC|A}$;
31 **if** $\phi_{BC} \geq \theta$ and $\phi_{BC|A} \leq -\theta$ **then**
32 output $\{B, C|A\}$
33 **end**
34 **end**
35 **end**
36 **end**

In addition, we save pairwise correlations in a linked list structure, L (Line 7). For an item pair $\{A, B\}$, where $A = I[a]$ and $B = I[b]$, we save ϕ_{AB} in $L[b]$, if $b > a$.[1] In practice, strongly correlated pairs tend to be a small fraction of all possible pairs. Therefore, the compactness of a linked list

1. Here we are using lexical order. Using other ordering is also possible as long as being consistent.

will save space when only strong pairs need to be saved. This compactness can save us time both in pairwise correlation computing and confounder searching.

In the confounder search phase, we iterate on the pairwise correlated triplets only, based on the necessary conditions stated in Lemmas 2 and 3. First, we iterate on the keys of L (Line 11). For key a, the iteration is skipped unless there are at least two elements in $L[a]$ (Line 13). The reason is that if items $A = I[a]$, $B = I[b]$ and $C = I[c]$ are pairwise correlated, assuming $a > b > c$ without loss of generality, we should have (b, ϕ_{AB}) and (c, ϕ_{AC}) in $L[a]$ and (c, ϕ_{BC}) in $L[b]$. If b is not strongly correlated with anyone, break from all pairs that may involve B (Line 16). Then, in Line 19 we skip the iteration if ϕ_{BC} is not found in $L[b]$, which means that A and B are not strongly correlated. Finally, additional pruning can be done in Line 21, which is based on Lemma 3.

4.3 The CONFOUND+ Algorithm

In this section, we describe an enhanced version of the CONFOUND algorithm, named CONFOUND+, which is based on additional screening rules derived in this paper.

As described in Algorithm 4, the CONFOUND+ algorithm also has two phases: identifying all strong pairs (Line 2), and searching for reversing confounders (Lines 3-4).

Algorithm 4. CONFOUND+

Input: I: an item list representing a reverse indexed transactional database; θ: a user-specified minimum correlation threshold.
Output: all reversing confounder patterns.
// Phase I: Identify all strong pairs, saved by direction
1 initialize L^+ and L^- as two empty linked lists;
2 $(L^+, L^-) \leftarrow$ TraverseSearchRegion(I);
// Phase II: Identify reversing confounders
3 CheckTripletsPos(L^+); // positive pairs only
4 CheckTripletsNeg(L^-, L^+); // negative pairs

When identifying the strong pairs, different from the basic CONFOUND algorithm, we save the positively strong pairs and negatively strong pairs separately. Moreover, we developed a sequential traversal procedure to maximally reduce unnecessary computing. The pseudocode is detailed in Appendix A. The strength of this sequential search procedure, as compared to the direct search procedure in Algorithm 2, is that we can avoid computing too many bounds. The search procedure in Algorithm 3 needs to compute both the upper and the lower bounds for each pair of items, whereas in Algorithm 4, the search range for item B is computed only once for each given item A, and then we can deterministically iterate item B within the specified range sequentially.

We were able to search among positive pairs and negative pairs separately because of Lemma 3. There are two situations when a reversing confounder could be found. First, when all three items are positively correlated pairwise. We only need to search in L^+, the linked list of positive pairs (see Procedure CheckTripletsPos). Second, one pair is positive and the other two pairs were negatively strong. We will iterate over each two strong negative pairs, and then check if the third pair can be found in L^+ (i.e., is positively strong) and reversed (see Procedure CheckTripletsNeg).

Procedure. CheckTripletsPos(L^+)

```
1  foreach a ∈ L+.keys() do
2      l ← L+[a].length();
3      if l < 2 then continue;
4      sort L+[a] by decreasing φ;
5      for i ← 1 to (l − 1) do
6          (b, φ_AB) ← L+[a][i];
7          if φ_AB < 2θ/(1+θ) then break;
8          for j ← (i + 1) to l do
9              (c, φ_AC) ← L+[a][j];
10             look up φ_BC from L+;
11             if φ_BC is not found then continue;
12             if φ_BC < φ_AC then
                   // Check if {B, C} is reversed by A
13                 calculate φ_BC|A;
14                 if φ_BC|A ≤ −θ thenoutput {B, C|A};
15             else
                   // Check if {A, C} is reversed by B
16                 calculate φ_AC|B;
17                 if φ_AC|B ≤ −θ then output {A, C|B};
18             end
19         end
20     end
21 end
```

In Procedure CheckTripletsPos, for each backbone item A, we require at least two correlated items (Line 3). This is because if A is only correlated with one item, the All-Strong condition cannot be met. Then we iterate over each pair of items (B and C) that are correlated to A (see the two levels of loops starting in Lines 5 and 8, respectively). Since the correlated items are sorted by decreasing correlation (Line 4), we know $\phi_{AB} \geq \phi_{AC}$, and therefore, ϕ_{AB} is not reversible (Corollary 1). If ϕ_{AB} is less than $\phi_{AB} < \frac{2\theta}{1+\theta}$, then the search related to item A ends (Line 7). This is made possible by Corollary 2. The pruning on Line 11 is guaranteed by the all-strong condition. Since only the smallest correlation may be reversed (see Corollary 1), Lines 12 and 15 checks if the smaller of ϕ_{BC} and ϕ_{AC} was reversed. There is no need to check ϕ_{AB} because it is irreversible (see Corollary 2).

The second case of a reversing confounder is that one pair is positive and the other two pairs were negatively strong. The search procedure, outlined in Procedure CheckTripletsNeg, is similar to Procedure CheckTripletsPos, but there are a few differences. First, the pruning happens at two places (Lines 7 and 10). These conditions were proved in Corollary 3. Second, the negative pairs were saved with some redundancy for easy enumeration of candidates. More specifically, a strong pair $\{A, B\}$ is saved both as (a, ϕ_{AB}) in $L^-[b]$ and (b, ϕ_{AB}) in $L^-[a]$. Finally, we only need to check whether the positive pair $\{B, C\}$ was reversed by A.

5 Experimental Results

In this section, we first briefly summarize the experimental setup and then show the experimental results.

Datasets. The Frequent Itemset Mining Implementations (FIMI) repository (http://fimi.cs.helsinki.fi/data/) provides a collection of datasets that are often used as benchmarks for evaluating frequent pattern mining algorithms. Table 3 lists the names and sizes of these datasets.

TABLE 3
Experimental Datasets

Dataset	# Items	# Pairs	# Transactions
accidents	468	109,278	340,183
BMS-POS	1,657	1,371,996	3,367,020
BMS-WebView-1	497	123,256	149,639
BMS-WebView-2	3,340	5,576,130	358,278
pumsb	2,113	2,231,328	49,046
pumsb_star	2,088	2,178,828	49,046
T10I4D100K	870	378,015	100,000
T40I10D100K	942	443,211	100,000

Procedure. CheckTripletsNeg(L^-, L^+)

```
1  foreach a ∈ L⁻.keys() do
2      l ← L⁻[a].length();
3      if l < 2 then continue;
4      sort L⁻[a] by increasing φ;
5      for i ← 1 to (l − 1) do
6          (b, φ_AB) ← L⁻[a][i];
7          if φ_AB > −√(2θ/(1+θ)) then break;
8          for j ← (i + 1) to l do
9              (c, φ_AC) ← L⁻[a][j];
10             if φ_AC > −2θ/(1+θ) then break;
11             look up φ_BC from L⁺[b];
12             if φ_BC is not found then continue;
               // Check if {B, C} is reversed by A
13             calculate φ_BC|A;
14             if φ_BC|A ≤ −θ then output {B, C|A};
15         end
16     end
17 end
```

Platform. All the experiments were performed on a Dell Optiplex 980 desktop PC, with 3.47 GHz Intel Core i5 CPU and 8 GB of RAM. The operating system is Microsoft Windows 10 Pro. All programs were implemented in C++, and were compiled and run in Cygwin.

Regarding the evaluation of effectiveness, we compared the output from each algorithm to ensure that the patterns found were correct and complete. Since all algorithms described in this paper gave us the same output (given dataset and parameter), the remaining evaluations will focus on scalability, including both computation and memory efficiency. To control measurement errors of running time, for each experiment setting (i.e., dataset, parameter, algorithm), we ran five runs and took the median.

5.1 Computation Efficiency of CONFOUND

In this section, we focus on a comparison of the computation efficiency. Fig. 3 shows a comparison of the total running time (seconds, in logarithm) of CONFOUND and the two baseline methods for each dataset under various threshold values. We showed a range of lower θ's, from 0.00 to 0.30, because for higher thresholds, the running time of CONFOUND and CONFOUND+ gets close to zero, making it hard to measure reliably and show on logarithm. We can see that for any given θ, the save-and-lookup approach (SL) and the CONFOUND algorithm (CF) are both significantly faster than the brute-force approach (BF). Moreover, the CONFOUND algorithm is consistently faster than the save-and-lookup approach. The clear separation between their curves on the log scale indicates that the performance gap is high.

The performance gap between CONFOUND and the brute-force algorithm varied by threshold. The smaller θ, the more computational savings, as the number of strong pairs grows quickly with lower θ. The running time of the save-and-lookup algorithm is expected to be relatively stable across different parameters. For some datasets, it running time elevated for lower thresholds for the need of checking confounders for more strong pairs. It is surprising that the save-and-lookup algorithm is slower than CONFOUND even though we expected it to be memory intensive but computing efficient. The fact that CONFOUND has outperformed a memory intensive approach shows that our proposed method is highly scalable.

5.2 Number of Pairs to Compute and Save

After seeing that the brute-force method was unscalable (in spite of being frugal in memory use), Fig. 4 shows a comparison of space usage among the two-phase methods (i.e., save-and-lookup, CONFOUND, and CONFOUND+), which all required saving pairs in the memory. The number of all pairs

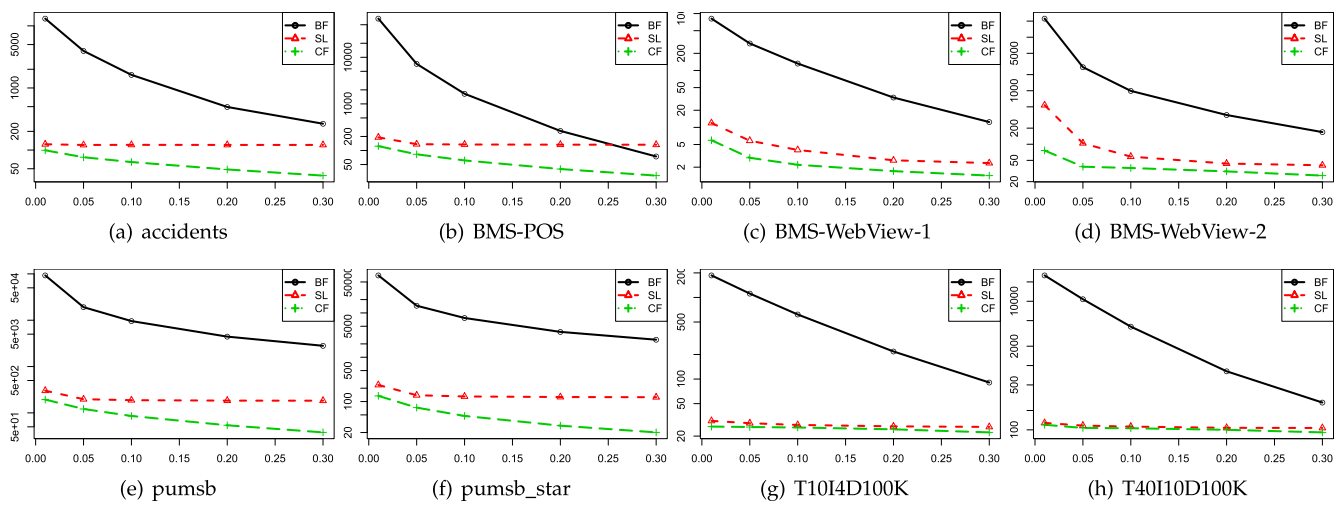


Fig. 3. Comparisons of running time: x-axis: θ; y-axis: Time in seconds, shown in logarithm.

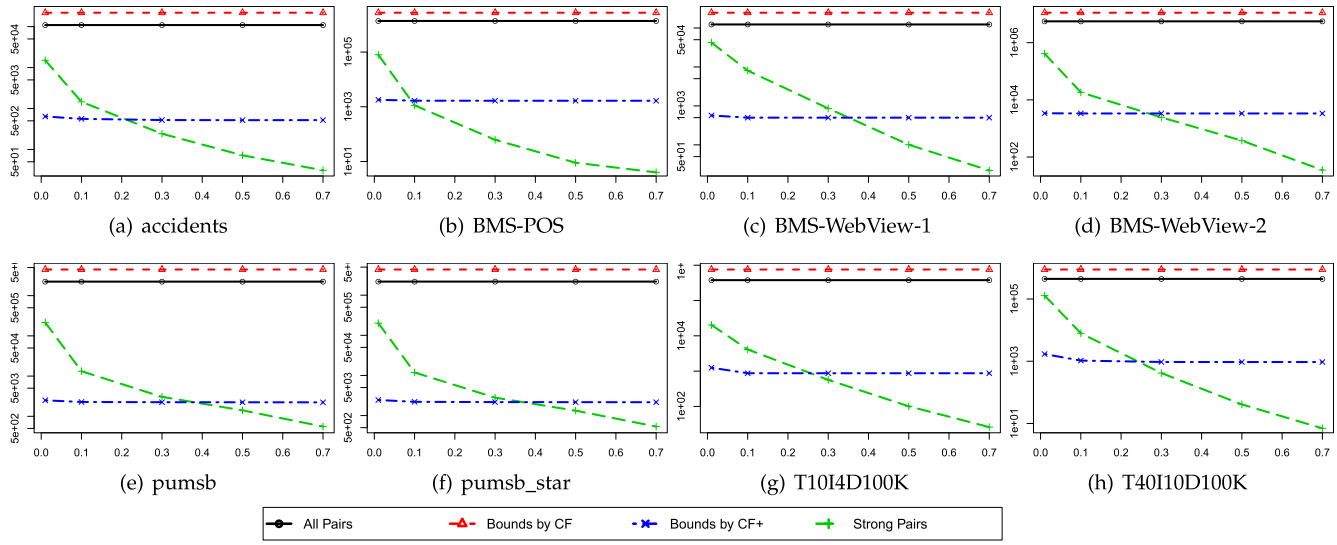


Fig. 4. Comparisons of use of space and bound computing: x-axis: θ; y-axis: Number of pairs, shown in logarithm.

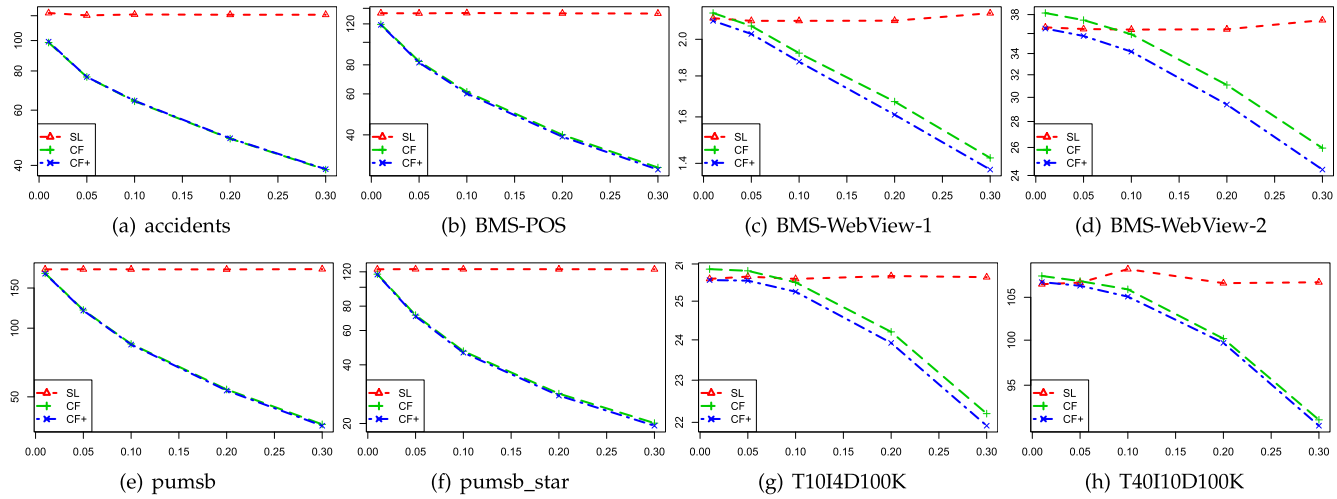


Fig. 5. Comparisons of BASP search time: x-axis: θ; y-axis: Time in seconds, shown in logarithm.

(see lines in black) is the number of pairs the save-and-lookup algorithm needed to save. It is determined as $M(M-1)/2$, where M is the number of unique items. Therefore, it does not depend on threshold. The number of strong pairs (see lines in green) is the number of pairs the CONFOUND and CONFOUND+ algorithms needed to save. Note that the y-axis representing the number of pairs is shown in logarithm. It is clear that there is a substantial gap between the black and green lines. When θ increases, the number of strong pairs decreases quickly. We can conclude that the CONFOUND and CONFOUND+ algorithms are memory efficient.

Also shown in Fig. 4 are the number of bounds computed by CONFOUND (red lines) and CONFOUND+ (blue lines), respectively, during the strong pairs search stage. We can see that they are almost constant across thresholds. The number of bounds by CONFOUND is even more than the total number of pairs (black line), because for each pair of items, we had to compute both the upper and the lower bounds of the correlation, making the number of bounds twice as many. In contrast, the number of bounds computed by CONFOUND+, thanks to our sequential traversal procedure, reduced this number to a small fraction.

5.3 A Comparison of the Two-Phase Methods

As mentioned earlier, the save-and-lookup, CONFOUND, CONFOUND+ algorithms all have a two-phase structure: strong pair search and reversing confounder search. In this section, we will compare the running time by phase.

Fig. 5 shows a comparison of the running time (seconds, in logarithm) of Phase I (i.e., BASP search). We can see that the save-and-lookup algorithm is the slowest and is almost constant across all thresholds. CONFOUND and CONFOUND+ are more efficient, and the running time decreases sharply as the threshold increases.

From Fig. 5 we can see that the BASP search time of CONFOUND+ is only slightly shorter than CONFOUND. For some datasets, their curves almost overlap. Since CONFOUND+ used sequential search and CONFOUND computes two bounds for all pairs, when the two lines overlap, it means the savings from avoiding computing bounds did not matter, as the computation of pair correlations took much longer. Indeed, as mentioned earlier, the computation of the exact correlation is time consuming, whereas computing a bound is relatively effortless. The size of the performance gap depends on the relative cost of computing the

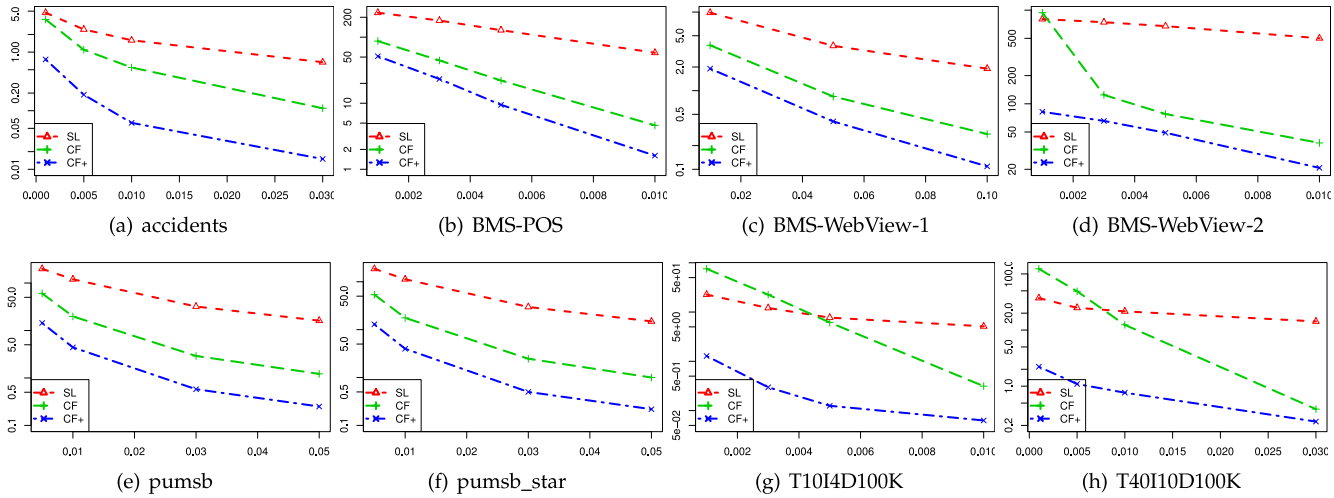


Fig. 6. Comparisons of confounder search time: x-axis: θ; y-axis: Time in seconds, shown in logarithm.

correlation versus computing the bounds. In Fig. 5, the save-and-lookup method may take longer than others when the number of pairs saved is so large that looking them up takes longer than computing them. This typically happens with dense data at lower threshold.

Fig. 6 shows a comparison of the running time (seconds, in logarithm) of Phase II (i.e., confounder searching). We can see that the searching time by CONFOUND (green lines) is much shorter than the save-and-lookup (red lines) algorithm for most datasets and parameters, and the performance gap increases as the threshold θ increases. This is significant because the save-and-lookup algorithm uses a pre-allocated (fixed) memory space, which is efficient for lookup; whereas CONFOUND, which uses a linked list (variable structure), was expected to be more complex for lookup. This shows that selectively saving just the strong pairs rather than all pairs paid off not only for space efficiency but also computation efficiency. Note that we are showing a range of even lower θ's because for higher thresholds, the confounder search time gets close to zero (for lack of strong pairs), making it hard to show on logarithm. In Fig. 6, the confounder search time of CONFOUND may be longer than the save-and-lookup method for certain datasets, especially for lower threshold values. The reason is that the CONFOUND algorithm involves computing overhead such as bound calculation and linked list maintenance.

Fig. 6 also shows that CONFOUND+ (blue lines) is much faster than CONFOUND in the confounder search phase, thanks to the separate search procedure among positive pairs and among negative pairs, and the more powerful pruning. Even though the confounder search time is relatively short compared to the strong pairs computing time, the saving in Phase II would be very helpful in situations where the strong pairs were incrementally maintained [20] and the confounder search is frequent.

6 Related Work

Correlation computing has been an active research topic in the past few decades. Using the Pearson's ϕ coefficient, a computation friendly upper bound was discovered and first utilized in [13], [19] for identifying all strong pairs. Leveraging this bound, efficient correlation range queries were studied in [18], [21] Extension to dynamic data environment was studied leveraging projected bounds of ϕ anticipating a buffer of future transactions. This has resulted in efficient volatile correlation computing [20], [22]. Other association measures have been studied in the data mining literature as well, such as confidence [7], lift [6], and the cosine measure [23].

The Simpson's paradox generally applies to most association measures, including those mentioned above. Therefore, it is important to identify the discrepancy between local and global patterns. Along this line, we found the following interesting studies with related concepts.

Contrast sets [24] are subgroups of the market-basket data, such that the same pattern behaves differently in different subgroups. The problem of finding group differences has been studied intensively in [25], [26], and [27] has applied the contrast set mining method to analyzing brain Ischaemina data. A study discovered that contrast-set mining is a special case of the rule-discovery framework [28]. However, contrast sets emphasize between-group differences, but not necessarily local versus global differences. Specially, contrast sets cannot discover local patterns showing Simpson's paradox [29]. In other words, some patterns behave differently in each local segment from the global level but appear similar among the local segments.

Niches [30] are defined as surprising association rules that contradict set routines, which consist of a number of dominant trends. Both the dominant trends and the niches can be captured by emerging patterns [31], which present significant differences between different classes. Although the concept of emerging patterns is closely related to contrast sets, the niche mining problem tries to discover a small number of exceptions (niches) that contradict the majority (set routines), which reflect the idea of discovering significant local patterns that are different from global ones. However, the problem setting is different from ours, since it focuses on one attribute as the class label (e.g., risky customers versus non-risky customers).

7 Conclusions

In this paper, we investigated how to efficiently discover paradoxical correlation patterns. We first identified three

necessary conditions of reversing confounders, which helped us prune the search space and develop an algorithm called CONFOUND. CONFOUND proves to be efficient in both memory space and computing when compared to two baseline methods. Moreover, we added further enhancements to CONFOUND and put forward a CONFOUND+ algorithm, which substantially improves the confounder search efficiency. The experimental results on benchmark datasets showed that the proposed CONFOUND and CONFOUND + algorithms can effectively identify confounders, and the computational performance can be orders of magnitude faster than the baseline methods.

Many extensions are possible for future work. First, it will be interesting to extend the discovery of reversing confounders to other types of confounders, including false positives (i.e., correlated globally but not correlated when controlled by this confounding variable) and false negatives (i.e., uncorrelated globally but correlated locally). Second, as a common challenge for all work in association pattern mining, it will be interesting to investigate how to integrate the discovered patterns into an overall predictive or decision optimization model. For example, the paradoxical correlation patterns may be used to adjust algorithms in the recommender systems. Finally, as the ϕ correlation coefficient only measures strength of correlation between binary variables, it would be useful to extend our framework into other association measures, including those for other variable types, and to consider statistical significance of the correlation patterns.

Appendix A
Traversing the Search Regions

Even though the search region may be expressed with three ranges as shown in Eq. (12), the actual computation of positive or negative strong pairs will be further sliced into five ranges of σ_A, or ten ranges of (σ_A, σ_B) combination, as annotated in Fig. 7.

Procedure. TraverseSearchRegion(I)

1 Sort I by non-decreasing item support;
2 $a \leftarrow 2$ // Initialize the index of item A
3 RegionOne // a increases until $\sigma_A \geq \frac{\theta}{1+\theta}$
4 RegionTwo // a increases until $\sigma_A \geq 0.5$
5 RegionThree // a increases until $\sigma_A \geq \frac{1}{1+\theta}$
6 RegionFour // a increases until $\sigma_A \geq \frac{1}{1+\theta^2}$
7 RegionFive // all others

The overall process of traversing the search region is outlined in Procedure TraverseSearchRegion. Since we only need to search for the region where $\sigma_A \geq \sigma_B$, in Line 2 we initialize the index of item A as the second least frequent item. Note that unlike Algorithm 1 that sorts I by non-increasing order of item support, here we choose to start from the least frequent, as most real-world data are highly skewed to the right. In other words, there is a large number of infrequent items, whereas the number of high-support items tends to be very small or none. Therefore, if we traverse from the least frequent items, it is likely to get the majority of the work done early on, and the search can stop early if there is no high-support item.

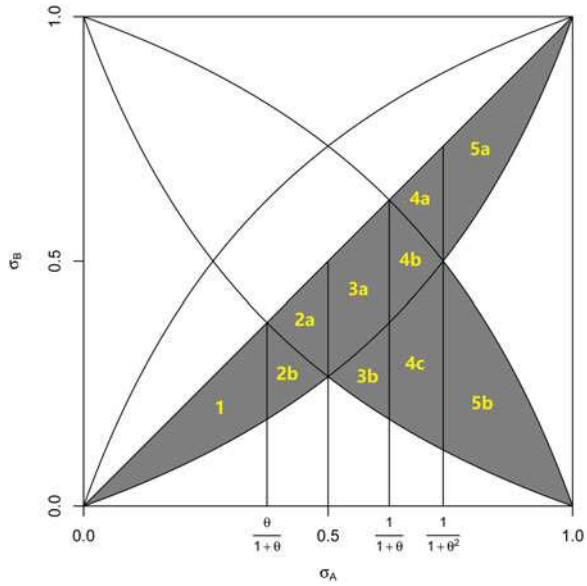


Fig. 7. Search space for all strong pairs.

In Lines 3 through 7, we traverse the five regions by σ_A sequentially. The detailed traversing pseudocode within each region is provided from Procedure RegionOne through Procedure RegionFive.

Procedure. RegionOne

1 **while** $a \leq M$ and $\sigma_A \leq \frac{\theta}{1+\theta}$ **do**
2 $\quad b \leftarrow a - 1$;
3 $\quad$ **while** $\sigma_B \geq f_1(\sigma_A)$ **do**
4 $\quad\quad$ find N_{AB} and compute ϕ_{AB};
5 $\quad\quad$ **if** $\phi_{AB} \geq \theta$ **then** save pair $\{A, B\}$ to L^+;
6 $\quad\quad b \leftarrow b - 1$;
7 $\quad$ **end**
8 $\quad a \leftarrow a + 1$;
9 **end**

Line 1 in Procedure RegionOne specifies the range of σ_A that defines Region 1 in Fig. 7. M is the total number of unique items, and the upper bound of Region 1, $\frac{\theta}{1+\theta}$, is the value of σ_A such that $f_3(\sigma_A) = \sigma_A$. In this region, all item pairs are candidates for positive strong pairs but not negative strong pairs. Therefore, we only need to check whether the correlation of each pair is above θ and insert the pair into L^+ if true (Line 5).

As the index of item A (i.e., a) continues to increment, the search continues into Region 2 (see Procedure RegionTwo). The upper bound for Region 2, 0.5, is the value of σ_A such that $f_1(\sigma_A) = f_3(\sigma_A)$, and is easy to tell in Fig. 7 by symmetry. Region 2 is further divided into two subregions, 2a and 2b, based on candidacy status. In Region 2a, all item pairs are candidates for both positive strong pairs and negative strong pairs. Therefore, we need to check whether the pair's correlation is above θ and insert it into L^+ if true (Line 5); and below $-\theta$ and insert it into L^- if true (Line 6). In Region 2b, all item pairs are candidates for positive strong pairs but not negative strong pairs. Therefore, we only need to check whether the correlation is above θ and insert the pair into L^+ if true (Line 11).

Procedure RegionThree

```
1  while a ≤ M and σ_A ≤ 1/(1+θ) do
2      b ← a − 1;
       // Region 3a
3      while σ_B ≥ f_1(σ_A) do
4          find N_AB and compute φ_AB;
5          if φ_AB ≥ θ then save pair {A, B} to L+ ;
6          if φ_AB ≤ −θ then save pair {A, B} to L− ;
7          b ← b − 1;
8      end
       // Region 3b
9      while σ_B ≥ f_3(σ_A) do
10         find N_AB and compute φ_AB;
11         if φ_AB ≤ −θ then save pair {A, B} to L− ;
12         b ← b − 1;
13     end
14     a ← a + 1;
15 end
```

Procedure RegionFour

```
1  while a ≤ M and σ_A ≤ θ/(1+θ^2) do
2      b ← a − 1;
       // Region 4a
3      while σ_B ≥ f_4(σ_A) do
4          find N_AB and compute φ_AB;
5          if φ_AB ≥ θ then save pair {A, B} to L+ ;
6          b ← b − 1;
7      end
       // Region 4b
8      while σ_B ≥ f_1(σ_A) do
9          find N_AB and compute φ_AB;
10         if φ_AB ≥ θ then save pair {A, B} to L+ ;
11         if φ_AB ≤ −θ then save pair {A, B} to L− ;
12         b ← b − 1;
13     end
       // Region 4c
14     while σ_B ≥ f_3(σ_A) do
15         find N_AB and compute φ_AB;
16         if φ_AB ≤ −θ then save pair {A, B} to L− ;
17         b ← b − 1;
18     end
19     a ← a + 1;
20 end
```

Procedure RegionFive

```
1  while a ≤ M do
2      b ← a − 1;
       // Region 5a
3      while σ_B ≥ f_1(σ_A) do
4          find N_AB and compute φ_AB;
5          if φ_AB ≥ θ then save pair {A, B} to L+ ;
6          b ← b − 1;
7      end
       // Skipping the irrelevant space in between
8      while σ_B ≥ f_4(σ_A) do
9          b ← b − 1;
10     end
       // Region 5b
11     while σ_B ≥ f_3(σ_A) do
12         find N_AB and compute φ_AB;
13         if φ_AB ≤ −θ then save pair {A, B} to L− ;
14         b ← b − 1;
15     end
16     a ← a + 1;
17 end
```

Fig. 8. Pseudocode for regions 3, 4, and 5.

Procedure. RegionTwo

```
1  while a ≤ M and σ_A ≤ 0.5 do
2      b ← a − 1;
       // Region 2a in Fig. 7
3      while σ_B ≥ f_3(σ_A) do
4          find N_AB and compute φ_AB;
5          if φ_AB ≥ θ then save pair {A, B} to L+;
6          if φ_AB ≤ −θ then save pair {A, B} to L−;
7          b ← b − 1;
8      end
       // Region 2b in Fig. 7
9      while σ_B ≥ f_1(σ_A) do
10         find N_AB and compute φ_AB;
11         if φ_AB ≥ θ then save pair {A, B} to L+;
12         b ← b − 1;
13     end
14     a ← a + 1;
15 end
```

Similarly, the search continues into Regions 3, 4, and 5 (see Fig. 8). Each of these regions was further divided into subregions based on candidacy status. In particular, the boundary point between Regions 3 and 4 is the value of σ_A such that $f_2(\sigma_A) = \sigma_A$; and the boundary point between Regions 4 and 5 is the value of σ_A such that $f_2(\sigma_A) = f_3(\sigma_A)$.

Acknowledgments

This research was partially supported by the National Science Foundation (NSF) via the grant number IIS-1648664. Also, it was supported in part by the Natural Science Foundation of China (71329201, 71531001).

References

[1] C. Alexander, *Market Models: A Guide to Financial Data Analysis*. Hoboken, NJ, USA: Wiley, 2001.

[2] H. V. Storch and F. W. Zwiers, *Statistical Analysis in Climate Research*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[3] L. Duan, M. Khoshneshin, W. N. Street, and M. Liu, "Adverse drug effect detection," *IEEE J. Biomed. Health Inform.*, vol. 17, no. 2, pp. 305–311, Mar. 2013.

[4] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Science*, 3rd ed. Hillsdale, NJ, USA: Lawrence Erlbaum, 2002.

[5] W. P. Kuo, T. K. Jenssen, A. J. Butte, L. Ohno-Machado, and I. S. Kohane, "Analysis of matched MRNA measurements from two different microarray technologies," *Bioinf.*, vol. 18, no. 3, pp. 405–412, 2002.

[6] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1997, pp. 265–276.

[7] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1993, pp. 207–216.

[8] X. Zhang, F. Pan, and W. Wang, "CARE: Finding local linear correlations in high dimensional data," in *Proc. IEEE Int. Conf. Data Eng.*, 2008, pp. 130–139.

[9] G. Dong, J. Han, J. M. W. Lam, J. Pei, and K. Wang, "Mining multi-dimensional constrained gradients in data cubes," in *Proc. Int. Conf. Very Large Data Bases*, 2001, pp. 321–330.

[10] K. A. Bollen, "Political democracy: Conceptual and measurement traps," *Stud. Comparative Int. Develop.*, vol. 25, no. 1, pp. 7–24, 1990.

[11] J. A. Delaney, L. Opatrny, J. M. Brophy, and S. Suissa, "Drug-drug interactions between antithrombotic medications and the risk of gastrointestinal bleeding," *Can. Med. Assoc. J.*, vol. 177, no. 4, pp. 347–351, 2007.

[12] H. T. Reynolds, *The Analysis of Cross-Classifications*. New York, NY, USA: Free Press, 1977.

[13] H. Xiong, S. Shekhar, P.-N. Tan, and V. Kumar, "TAPER: A two-step approach for all-strong-pairs correlation query in large databases," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 4, pp. 493–508, Apr. 2006.

[14] W. Zhou and H. Xiong, "Efficient discovery of confounders in large data sets," in *Proc. 9th IEEE Int. Conf. Data Mining*, 2009, pp. 647–656.

[15] C. H. Wagner, "Simpson's paradox in real life," *Amer. Statistician*, vol. 36, no. 1, pp. 46–48, 1982.

[16] C. R. Blyth, "On simpson's paradox and the sure-thing principle," *J. Amer. Statistical Assoc.*, vol. 67, no. 338, pp. 364–366, 1972.

[17] K. Baba, R. Shibata, and M. Sibuya, "Partial correlation and conditional correlation as measures of conditional independence," *Australian New Zealand J. Statist.*, vol. 46, no. 4, pp. 657–664, 2004.

[18] W. Zhou and H. Zhang, "Correlation range query for effective recommendations," *World Wide Web*, vol. 18, no. 3, pp. 709–729, 2015.

[19] H. Xiong, S. Shekhar, P. Tan, and V. Kumar, "Exploiting a support-based upper bound of Pearson's correlation coefficient for efficiently identifying strongly correlated pairs," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 334–343.

[20] W. Zhou and H. Xiong, "Checkpoint evolution for volatile correlation computing," *Mach. Learn.*, vol. 83, no. 1, pp. 103–131, 2011.

[21] W. Zhou and H. Zhang, "Correlation range query," in *Proc. 14th Int. Conf. Web-Age Inf. Manage.*, 2013, pp. 490–501.

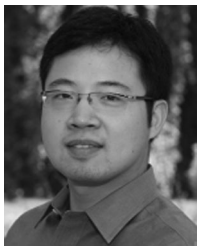
[22] W. Zhou and H. Xiong, "Volatile correlation computation: A checkpoint view," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 848–856.

[23] J. Wu, S. Zhu, H. Liu, and G. Xia, "Cosine interesting pattern discovery," *Inf. Sci.*, vol. 184, no. 1, pp. 176–195, 2012.
[24] P. Kralj, N. Lavrac, D. Gamberger, and A. Krstacic, "Contrast set mining for distinguishing between similar diseases," in *Proc. 12th Conf. Artif. Intell. Med.*, 2007, pp. 109–118.
[25] S. Bay and M. Pazzani, "Detecting group differences: Mining contrast sets," *Data Mining Knowl. Discovery*, vol. 5, no. 3, pp. 213–246, 2001.
[26] S. D. Bay and M. J. Pazzani, "Detecting change in categorical data: Mining contrast sets," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 1999, pp. 302–306.
[27] P. Kralj, N. Lavrac, D. Gamberger, and A. Krstacic, "Contrast set mining through subgroup discovery applied to brain ischaemina data," in *Proc. Pacific-Asia Conf. Advances Knowl. Discovery Data Mining*, 2007, pp. 579–586.
[28] G. I. Webb, S. M. Butler, and D. A. Newlands, "On detecting differences between groups," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2003, pp. 256–265.
[29] M. L. Samuels, "Simson's paradox and related phenomena," *J. Amer. Statistical Assoc.*, vol. 88, pp. 81–88, 1993.
[30] G. Dong and K. Deshpande, "Efficient mining of niches and set routines," in *Proc. Pacific-Asia Conf. Advances Knowl. Discovery Data Mining*, 2001, pp. 234–246.
[31] G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 1999, pp. 43–52.

Wenjun Zhou (S'11–M'12) received the BS degree from the University of Science and Technology of China (USTC), the MS degree from the University of Michigan-Ann Arbor, and the PhD degree from Rutgers University. She is currently an assistant professor in the Department of Business Analytics and Statistics, Haslam College of Business, University of Tennessee, Knoxville. Her general research interests include data mining, business analytics, and statistical computing. She was nominated R. Stanley Bowden II Faculty research fellow in 2016 and Roy & Audrey Fancher Faculty Research Fellow in 2017. She was the recipient of the Best Paper Award at WAIM 2013, Best Student Paper Runner-Up Award at KDD 2008, and Best Paper Runner-Up Award at ICTAI 2006. She was among the top five finalists for the ACM SIGKDD Doctoral Dissertation Award in 2011. She is a member of the IEEE.

Hui Xiong (SM'07) received the BE degree from the University of Science and Technology of China (USTC), the MS degree from the National University of Singapore, and the PhD degree from the University of Minnesota. He is currently a full professor and vice chair of the Management Science and Information Systems Department, and the director of the Center for Information Assurance, Rutgers University, where he received a two-year early promotion/tenure (2009), the Board of Trustees Research Fellowship for Scholarly Excellence (2009), and the ICDM-2011 Best Research Paper Award (2011). His general area of research is data and knowledge engineering, with a focus on developing effective and efficient data analysis techniques for emerging data intensive applications. He has published prolifically in refereed journals and conference proceedings (four books, 70+ journal papers, and 100+ conference papers). He is a co-editor-in-chief of the *Encyclopedia of GIS*, an associate editor of the *IEEE Transactions on Data and Knowledge Engineering (TKDE)*, the *IEEE Transactions on Big Data*, the *ACM Transactions on Knowledge Discovery from Data (TKDD)*, and the *ACM Transactions on Management Information Systems*. He has served regularly on the organization and program committees of numerous conferences, including as a program co-chair of the Industrial and Government Track for the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), a program co-chair for the IEEE 2013 International Conference on Data Mining (ICDM), and a general co-chair for the IEEE 2015 International Conference on Data Mining (ICDM). For his outstanding contributions to data mining and mobile computing, he was elected an ACM distinguished scientist in 2014. He is a senior member of the IEEE.

Lian Duan received the PhD degree in management sciences from the University of Iowa and the PhD degree in computer sciences from the Chinese Academy of Sciences. He is an assistant professor in the Department of Information Systems and Business Analytics with Hofstra University. His research interests include correlation analysis, health informatics, and social networks. He has published in the *ACM Transactions on Knowledge Discovery from Data*, KDD, ICDM, *Information Systems*, the *Annals of Operations Research*, etc.



Keli Xiao received the BS degree in computer science from the Wuhan University of Technology, in 2006, the MA degree in computer science from Queens College, City University of New York, in 2008, and the master's degree in quantitative finance and the PhD degree in management (finance) from Rutgers University, in 2009 and 2013, respectively. He is an assistant professor in the College of Business, Stony Brook University. His research interests include business analytics, data mining, quantitative finance, and economic bubbles. He has published papers in refereed journals and conference proceedings, such as the *IEEE Transactions on Knowledge and Data Engineering*, the *ACM Transactions on Knowledge Discovery from Data*, KDD, and ICDM.

Robert Mee received the BS degree in management science from the Georgia Institute of Technology, and the MS and PhD degrees in statistics from Iowa State University. He is currently a professor of business analytics and statistics, University of Tennessee's Haslam College of Business and a lecture professor with the Nankai University's Institute of Statistics in Tianjin, China. He is a fellow of the American Statistical Association and has authored more than 50 refereed journal articles. He is also the author of *A Comprehensive Guide to Factorial Two-Level Experimentation* (Springer, 2009).