TOP-K ϕ Correlation Computation

Hui Xiong, Wenjun Zhou
Management Science and Information Systems Department
Rutgers, the State University of New Jersey
hxiong@andromeda.rutgers.edu, wjzhou@pegasus.rutgers.edu

Mark Brodie
IBM TJ Watson Research Center
mbrodie@us.ibm.com

Sheng Ma
DoubleClick Inc.
shengma2005@gmail.com

Recently, there has been considerable interest in efficiently computing strongly correlated pairs in large databases. Most previous studies require the specification of a minimum correlation threshold to perform the computation. However, it may be difficult for users to provide an appropriate threshold in practice, since different data sets typically have different characteristics. To this end, in this paper, we propose an alternative task: finding the top-k strongly correlated pairs. Consequently, we identify a 2-D monotone property of an upper bound of ϕ correlation coefficient and develop an efficient algorithm, called TOP-COP to exploit this property to effectively prune many pairs even without computing their correlation coefficients. Our experimental results show that TOP-COP can be an order of magnitude faster than alternative approaches for mining the top-k strongly correlated pairs. Finally, we show that the performance of the TOP-COP algorithm is tightly related to the degree of data dispersion. Indeed, the higher the degree of data dispersion, the larger the computational savings achieved by the TOP-COP algorithm.

Key words: ϕ Correlation Coefficient, Pearson's Correlation Coefficient, Statistical Computing

---

1. Introduction

Given a large set of items (objects) and observation data about co-occurring items, association analysis is concerned with the identification of strongly related subsets of items. Association analysis is a core problem in data mining and databases. It plays an important role in many application domains such as market-basket analysis (Alexander, 2001), climate studies (Storch and Zwiers, 2002), public health (Cohen et al., 2002), and bioinformatics (Kuo et al., 2002; Xiong et al., 2005). For example, in market-basket study, the association analysis can be used for sales promotion,

shelf management, and inventory management. Also, in medical informatics, association analysis is used to find combination of patient symptoms and complaints associated with certain diseases.

The focus of this paper is on computing a *top-k correlated-pairs query* that returns the top k pairs of positively correlated items. As a motivating example, the top-k correlated-pairs query can reveal information about how the sales of a product are related to the sales of other products. This type of information can be useful for sales promotions, catalog design, and store layout. However, as the number of items and transactions (observations) in the data set increases, the computational cost of the top-k correlated-pairs query becomes prohibitively expensive. For example, if a database contains 10^6 items, which may represent the collection of books available at an e-commerce Web site, a brute-force approach to answering the top-k correlated-pairs query requires computing the correlations of $\binom{10^6}{2} \approx 0.5 \times 10^{12}$ possible item pairs. Thus, it may not be computationally feasible to apply a brute-force approach.

1.1. Contributions and Scope

The *top-k correlated-pairs query* problem can be described as follows: Given a user-specified k and a market basket database with N items and T transactions, a top-k correlated-pairs query finds the top k item pairs with the highest positive correlations. The scope of the top-k correlated-pairs query problem proposed in this paper is restricted to market basket databases with binary variables, and the form of correlation is Pearson's correlation coefficient (Reynolds, 1977) for binary variables, also called the ϕ correlation coefficient.

The main contribution of this work is the development and analysis of the **TOP**-k **CO**rrelated-**P**airs (TOP-COP) query algorithm. We prove this algorithm to be complete and correct, and show that it can be faster than alternative approaches by an order of magnitude. The TOP-COP algorithm exploits a 2-D monotone property of the upper bound of Pearson's correlation coefficient. By interpreting this property geometrically, we obtain an algorithm that uses a diagonal traversal method, combined with a refine-and-filter strategy, to efficiently find the top-k pairs.

1.2. Related Work

The top-k correlated-pairs query problem is different from the standard association-rule mining problem (Agrawal et al., 1993; Bayardo et al., 2000; Brin et al., 1997; Bucila et al., 2003; Burdick et al., 2001; Cohen et al., 2000; Grahne et al., 2000; Liu et al., 1999; Ng et al., 1999; Rastogi and Shim, 2002; Wang et al., 2001). Let $I = \{i_1, i_2, ..., i_m\}$ be a set of distinct items. Each transaction

T in database D is a subset of I. We call $X \subseteq I$ an itemset. The support of X, denoted by $supp(X)$, is the fraction of transactions containing X. If $supp(X)$ is no less than a user-specified threshold, X is called a frequent itemset. The confidence of association rule $X_1 \rightarrow X_2$ is defined as $conf(X_1 \rightarrow X_2) = supp(X_1 \cup X_2)/supp(X_1)$.

Existing association-rule mining methods rely on the support and confidence measures to find groups of highly-correlated objects from massive data. However, the notions of support and correlation may not necessarily agree with each other. This is because item pairs with high support may be poorly correlated while those that are highly correlated may have low support. For instance, suppose we have an item pair {A, B}, where $supp(A) = supp(B) = 0.8$ and $supp(A, B) = 0.64$. Both items are uncorrelated, since $supp(A, B) = supp(A)supp(B)$. In contrast, an item pair {A, B} with $supp(A) = supp(B) = supp(A, B) = 0.001$ is perfectly correlated despite its low support. Patterns with low support but high correlation are useful for capturing interesting associations among rare anomalous events or rare but expensive items such as gold necklaces and earrings.

In addition, the notion of confidence and correlation also may not agree with each other. Let us illustrate this with the following example (Tan et al., 2005).

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Figure 1: A Tea-Coffee Example

In Figure 1, we can derive an association rule {Tea} $\rightarrow$ {coffee} with a reasonably high confidence value (75%). This rule leads to the conclusion that people who drink tea will more likely drink coffee. However, the fraction of coffee drinkers, regardless of whether they drink tea, is 90%. In contrast, the fraction of tea drinkers who drink coffee is only 75%; that is, knowing that a person who drink tea actually decreases his/her probability of being a coffee drinker from 90% to 75%. If we compute correlation between Tea and $Coffee$, we can find Tea and $Coffee$ are actually negatively correlated in the given example. Indeed, the problem of association-rule mining was motivated by the difficulty of efficiently identifying highly correlated objects using traditional statistical correlation measures. This has led to the use of alternative interest measures, such as support and confidence, despite the lack of a precise relationship between these new interest measures

and statistical correlation measures. However, as illustrated above, the support-confidence framework tends to generate too many spurious patterns involving objects which are poorly correlated. This, in turn, motivates the work in this paper.

Recently, Xiong et al. (2004) proposed the TAPER algorithm to efficiently compute the all-strong-pairs correlation query. Given a user-specified minimum correlation threshold θ and a market basket database with N items and T transactions, the all-strong-pairs correlation query finds all item pairs with correlations above the threshold θ. However, it is difficult for users to provide an appropriate correlation threshold for the all-strong-pairs correlation query in real-world applications, since different data sets typically have different characteristics.

Along the same line of the all-strong-pairs correlation query, Ilyas et al. (2004) also proposed a method for efficiently identifying correlated pairs. In this method, sampling techniques are applied to exploit efficient computation. Due to the nature of sampling, this method cannot avoid finding false-positive and false-negative correlations. Furthermore, this method also requires users to specify a correlation threshold.

Additionally, Brin et al. (1997) proposed a χ^2-based correlation mining strategy; however χ^2 does not possess an upward closure property for exploiting efficient computation (DuMouchel and Pregibon, 2001). Also, Jermaine (2001) investigated the implication of incorporating chi-square (χ^2) (Reynolds, 1977) based queries to data cube computations. He showed that finding the subcubes that satisfy statistical tests such as χ^2 are inherently NP-hard, but can be made more tractable using approximation schemes. Finally, Jermaine (2003) presented an iterative procedure for correlation analysis by shaving off part of the database via feedback from human experts.

1.3. Outline

The remainder of this paper is organized as follows. Section 2 presents the basic concepts of Pearson's correlation coefficient. In Section 3, we introduce 2-D monotone properties of the upper bound of Pearson's correlation coefficient for binary variables. Section 4 proposes an alternative approach based on the TAPER algorithm for finding top-k correlated pairs. In Section 5, we describe the TOP-COP algorithm. Section 6 presents the analysis of the TOP-COP algorithm in the areas of completeness, correctness, and computational cost, and Section 7 shows the experimental results. Finally, we draw conclusions and suggest future work in section 8.

2. Basic Concepts

In statistics, a measure of association is a numerical index which describes the strength or magnitude of a relationship among variables. Although literally dozens of measures exist, they can be categorized into two broad groups: ordinal and nominal. Relationships among ordinal variables can be analyzed with ordinal measures of association such as Kendall's Tau (Kendall and Gibbons, 1990) and Spearman's Rank Correlation Coefficient (Lehmann and D'Abrera, 1998). In contrast, relationships among nominal variables can be analyzed with measures of association such as Pearson's Correlation Coefficient and measures based on Chi Square (Reynolds, 1977).

2.1. The ϕ Correlation Coefficient

The ϕ correlation coefficient (Reynolds, 1977) is the computational form of Pearson's Correlation Coefficient for binary variables. Here, we first describe some basic concepts related to the ϕ correlation coefficient.

		B		Row Total
		0	1	
A	0	$P_{(00)}$	$P_{(01)}$	$P_{(0+)}$
	1	$P_{(10)}$	$P_{(11)}$	$P_{(1+)}$
Column Total		$P_{(+0)}$	$P_{(+1)}$	N

Figure 2: A Two-Way Table of Item A and Item B

In a 2 $\times$ 2 two-way table shown in Figure 2, the calculation of the ϕ correlation coefficient reduces to

$$\phi_{\{A,B\}} = \frac{P_{(00)}P_{(11)} - P_{(01)}P_{(10)}}{\sqrt{P_{(0+)}P_{(1+)}P_{(+0)}P_{(+1)}}}, \tag{1}$$

where $P_{(ij)}$, for i = 0, 1 and j = 0, 1, denote the number of samples which are classified in the ith row and jth column of the table. Furthermore, we let $P_{(i+)}$ denote the total number of samples classified in the ith row and $P_{(+j)}$ denote the total number of samples classified in the jth column. Thus $P_{(i+)} = \sum_{j=0}^{1} P_{(ij)}$ *and* $P_{(+j)} = \sum_{i=0}^{1} P_{(ij)}$.

When adopting the support measure of association rule mining (Agrawal et al., 1993), for two items A and B in a market basket database, we have $supp(A) = P_{(1+)}/N$, $supp(B) = P_{(+1)}/N$, and $supp(A, B) = P_{(11)}/N$, where N is the total number of samples in the two-way table. With support notations, as illustrated in (Xiong et al., 2004), we have the support form of Equation (1) shown below as Equation (2).

$$\phi_{\{A,B\}} = \frac{supp(A,B) - supp(A)supp(B)}{\sqrt{supp(A)supp(B)(1-supp(A))(1-supp(B))}} \tag{2}$$

(a) A Market Basket Database

TID	Items
1	a, b, c
2	a, b, c
3	a, c
4	a, b
5	a, b
6	a, b
7	a, b, c, d, e, f
8	a, b, d, e
9	a, b, d
10	c

(b) Item Pairs with Upper Bounds and Correlation Coefficients

Pair	Upper Bound	Correlation
{a, b}	0.667	0.667
{a, c}	0.333	−0.333
{a, d}	0.218	0.218
{a, e}	0.167	0.167
{a, f}	0.111	0.111
{b, c}	0.5	−0.5
{b, d}	0.327	0.327
{b, e}	0.25	0.25
{b, f}	0.167	0.167
{c, d}	0.655	−0.218
{c, e}	0.5	0
{c, f}	0.333	0.333
{d, e}	0.764	0.764
{d, f}	0.509	0.509
{e, f}	0.667	0.667

Figure 3: An Example Data Set

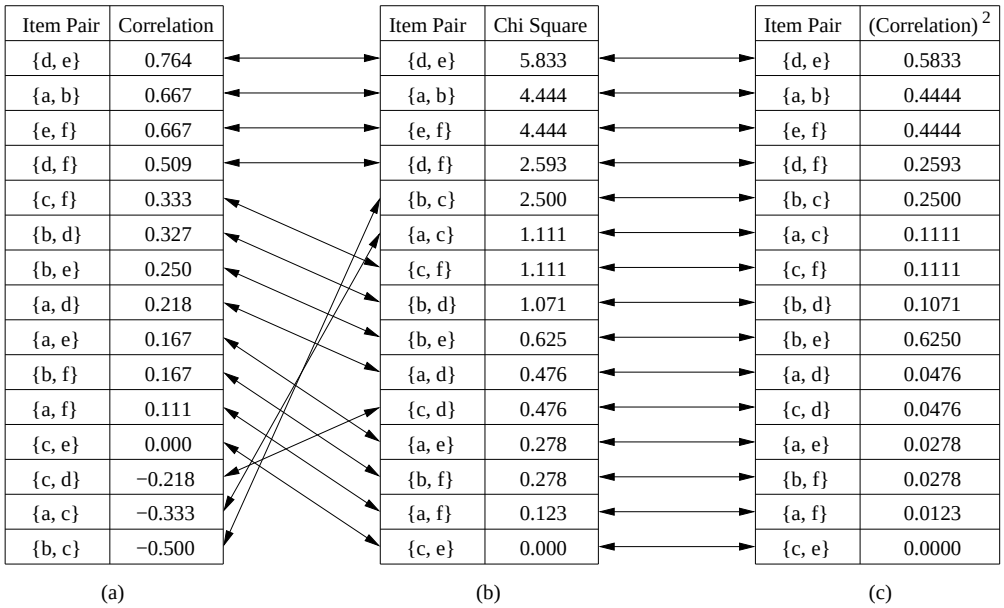
For example, consider the example data set shown in Figure 3 (a). To compute $\phi_{\{a,b\}}$, we note that $supp(a) = 9/10$, $supp(b) = 8/10$, and $supp(a, b) = 8/10$. Direct calculation shows that $\phi_{\{a,b\}} = 0.08/0.12 = 2/3$, confirming that a and b are strongly correlated.

Given an item pair $\{A,\ B\}$, the support value $supp(A)$ for item A, and the support value $supp(B)$ for item B, we can suppose without loss of generality that $supp(A) \geq supp(B)$. By the anti-monotone property of the support measure (Agrawal et al., 1993), we have $supp(A,\ B) \leq supp(B) \leq supp(A)$. In other words, the maximum possible value of $supp(A, B)$ is $supp(B)$. As a result, the upper bound upper($\phi_{\{A,B\}}$) of the ϕ correlation coefficient for an item pair $\{$A, B$\}$ can be obtained when $supp(A, B) = supp(B)$. Hence,

$$\begin{aligned} upper(\phi_{\{A,B\}}) &= \frac{supp(B) - supp(A)supp(B)}{\sqrt{supp(A)supp(B)(1-supp(A))(1-supp(B))}} \\ &= \sqrt{\frac{supp(B)}{supp(A)}}\sqrt{\frac{1-supp(A)}{1-supp\,(B)}} \end{aligned} \tag{3}$$

2.2. The ϕ Correlation Coefficient versus the Chi Square (χ^2) Statistic

In this subsection, we illustrate the relationship between the ϕ correlation coefficient and the chi square (χ^2) statistic.



Item Pair	Correlation
{d, e}	0.764
{a, b}	0.667
{e, f}	0.667
{d, f}	0.509
{c, f}	0.333
{b, d}	0.327
{b, e}	0.250
{a, d}	0.218
{a, e}	0.167
{b, f}	0.167
{a, f}	0.111
{c, e}	0.000
{c, d}	−0.218
{a, c}	−0.333
{b, c}	−0.500

(a)

Item Pair	Chi Square
{d, e}	5.833
{a, b}	4.444
{e, f}	4.444
{d, f}	2.593
{b, c}	2.500
{a, c}	1.111
{c, f}	1.111
{b, d}	1.071
{b, e}	0.625
{a, d}	0.476
{c, d}	0.476
{a, e}	0.278
{b, f}	0.278
{a, f}	0.123
{c, e}	0.000

(b)

Item Pair	(Correlation) ^{2}
{d, e}	0.5833
{a, b}	0.4444
{e, f}	0.4444
{d, f}	0.2593
{b, c}	0.2500
{a, c}	0.1111
{c, f}	0.1111
{b, d}	0.1071
{b, e}	0.6250
{a, d}	0.0476
{c, d}	0.0476
{a, e}	0.0278
{b, f}	0.0278
{a, f}	0.0123
{c, e}	0.0000

(c)

Figure 4: An Illustration of the Relationship between χ^2 and the ϕ Correlation Coefficient

For the example data set shown in Figure 3 (a), we compute the ϕ correlation coefficients and the chi square (χ^2) statistics for all item pairs. Figure 4 (a), (b), and (c) show the ranking of item pairs based on the ϕ correlation coefficient, the chi square statistic, and the square of the ϕ correlation coefficient respectively. In the figure, we can observe that the ranking difference by the ϕ correlation coefficient and the chi square statistic is caused by item pairs with negative correlation coefficients. Also, the rankings based on the chi square statistic and the square of the ϕ correlation coefficient are exactly the same. Indeed, the chi square statistic only measures the *likelihood* of the existence of association (Agresti, 2002). In other words, an item set $\{A, B\}$ having high ranking by the chi square test only means that there is relatively strong association between item A and item B, but there is no indication whether the existence of A (or B) leads to the existence or absence of B (or A).

Indeed, in a 2×2 two-way table shown in Figure 2, the goodness-of-fit chi square is computed by the following equation (Reynolds, 1977):

$$\chi^2_{\{A,B\}} = \frac{N(P_{(00)}P_{(11)} - P_{(01)}P_{(10)})^2}{P_{(0+)}P_{(1+)}P_{(+0)}P_{(+1)}}, \tag{4}$$

where N is the total number of samples in the two-way table. According to Equation (1) and Equation (4), we have the following:

$$\frac{\chi^2_{\{A,B\}}}{N} = (\phi_{\{A,B\}})^2 \tag{5}$$

Equation (5) indicates that dividing chi square by N leads to the square of the ϕ correlation coefficient. In summary, according to Equation (4), we know that the chi square may not be a good measure of association, since its numerical magnitude depends partly on the sample size. In contrast, the ϕ correlation coefficient is not depend on the sample size. Also, similar to the χ^2 statistic, the ϕ correlation coefficient measures the deviance from independence. In addition to this, the ϕ correlation coefficient is able to capture both positive and negative correlations between two variables. Therefore, the ϕ correlation coefficient is a more suitable measure of association.

3. A 2-D Monotone Property of the Upper Bound of ϕ Correlation Coefficient

In this section, we present a 2-D monotone property of the upper bound of the ϕ correlation coefficient. This monotone property can be exploited to develop a diagonal traversal method for efficiently mining the top-k correlated pairs.

Using Equation (3) above, Xiong et al. (Xiong et al., 2004) also derived a 1-D monotone property of the upper bound of the ϕ correlation coefficient:

Lemma 1 *For an item pair $\{A, B\}$, let $supp(A) > supp(B)$ and fix item A. The upper bound, $upper(\phi_{\{A,B\}})$, is monotone decreasing with decreasing support of item B.*

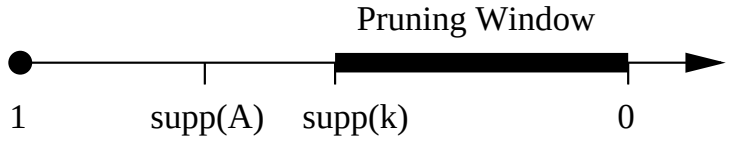


Figure 5: The Computational Exploration of the 1-D Monotone Property

Figure 5 shows a geometric interpretation of the computational exploration of the 1-D monotone property of the upper bound. In the figure, all items are sorted according to item support values in non-increasing order. Let us consider the all-strong-pairs correlation query with a user-specified correlation threshold θ. If we identify an item k such that the upper bound $upper(\phi_{\{A,k\}})$ is less

than the threshold θ, then any item pair {A, f} with $supp(f) < supp(k)$ can be safely pruned, since the upper bound $upper(\phi_{\{A,f\}})$ is guaranteed to be less than $upper(\phi_{\{A,k\}})$ according to Lemma 1. In other words, for any item A, we can generate a one-dimensional pruning window for efficiently eliminating item pairs which do not satisfy the correlation threshold θ.

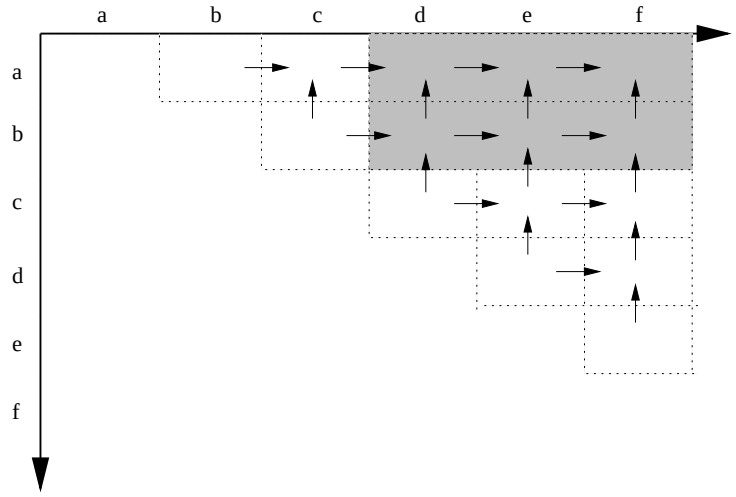


Figure 6: The Computational Exploration of the 2-D Monotone Property

Lemma 2 *For a pair of items $\{A, B\}$, let $supp(A) > supp(B)$ and fix item B. The upper bound $upper(\phi_{\{A,B\}})$ is monotone increasing with decreasing support of item A.*

Proof: Given two items A_1 and A_2 with $supp(A_1) > supp(A_2) > supp(B)$, we need to prove that $upper(\phi_{\{A_2,B\}}) > upper(\phi_{\{A_1,B\}})$. According to Equation (3), $upper(\phi_{\{A,B\}}) = \sqrt{\frac{supp(B)}{supp(A)}}\sqrt{\frac{1-supp(A)}{1-supp(B)}}$, with $A = A_1$ and $A = A_2$, we see that $\frac{upper(\phi_{\{A_2,B\}})}{upper(\phi_{\{A_1,B\}})} = \sqrt{\frac{supp(A_1)}{supp(A_2)}}\sqrt{\frac{1-supp(A_2)}{1-supp(A_1)}} > 1$ because $supp(A_1) > supp(A_2)$ and $(1 - supp(A_1)) < (1 - supp(A_2))$.

Lemma 1 and Lemma 2 form the basis of the 2-D monotone property of the upper bound, illustrated in Figure 6. An item list $\{a, b, c, d, e, f\}$, is sorted by item support values in non-increasing order. The upper bound of item pairs decreases following the direction of the arrow. For instance, the upper bound of item pair $\{d, f\}$ is greater than that of item pair $\{c, f\}$ but smaller than that of item pair $\{d, e\}$.

In contrast to Figure 5, the 2-D monotone property can help prune item pairs from two dimensions instead of one dimension. For instance, if the upper bound of item pair $\{b, d\}$ indicates that

this pair is not strongly correlated, we can generate a rectangle pruning space for efficiently eliminating item pairs: $\{b, e\}$, $\{b, f\}$, $\{a, d\}$, $\{a, e\}$, and $\{a, f\}$. Since the upper bounds of all these five item pairs cannot be greater than the upper bound of item pair $\{b, d\}$, these pairs are also not strongly correlated.

The TAPER-TOP-K Algorithm

Input: S': an item list sorted by item supports in non-increasing order.
k: The number of most highly correlated pairs requested.
Output: The list of the top-k strongly correlated pairs.
Variables: L: the size of item set S'; TL: the current top-k list;
A: the item with larger support; B: the item with smaller support.
$minPair$: the item pair in TL with the smallest correlation.
$minCorr$: the correlation value of minPair.

TAPER-TOP-K(S', k)

1. L = size(S'), $TL = \emptyset$, $minCorr$ = -1
2. **for** i from 1 to L-1
3. $A = S'[i]$
4. **for** j from i+1 to L
5. $B = S'[j]$, $upper(\phi) = \sqrt{\frac{supp(B)}{supp(A)}}\sqrt{\frac{1-supp(A)}{1-supp(B)}}$
6. **if**($upper(\phi) < minCorr$) **then**
7. break from inner loop //pruning
8. **else**
9. Get $supp(A, B)$, the support of item set $\{A,\ B\}$
10. $\phi = \frac{supp(A,B)-supp(A)supp(B)}{\sqrt{supp(A)supp(B)(1-supp(A))(1-supp(B))}}$
11. **if**($\phi > minCorr$) **then**
12. **if**($size(TL) < k$) **then**
13. Push item pair $\{A, B, \phi\}$ into TL
14. **else**
15. Delete the current $minPair$ in TL and insert $\{A, B, \phi\}$ into TL
16. Update $minPair$ and $minCorr$ in TL
17. **end if**
18. **end if**
19. **end if**
20. **end for**
21. **end for**
22. Return TL

Figure 7: The TAPER-TOP-K Algorithm

4. TAPER-TOP-K: TAPER based Top-K Correlated Pairs Query

A brute-force approach to computing the top-k correlated pairs is to first compute the correlation coefficients for all item pairs, then sort all item pairs based on their correlation coefficients, and finally report the top-k strongly correlated pairs as the final result. Instead of the brute-force approach, an alternative more sophisticated method towards the top-k correlated-pairs query problem is to modify the TAPER algorithm proposed by Xiong et al. (2004) to do a coarse filtering and then compute exact correlation coefficients for all the item pairs which remain after the filtering. Note that the TAPER algorithm is designed to efficiently compute the all-strong-pairs correlation query. Given a user-specified minimum correlation threshold θ and a market basket database with N items and T transactions, the all-strong-pairs correlation query finds all item pairs with correlations above the threshold θ.

Algorithm Descriptions. Here, we describe the details of the TAPER-TOP-K algorithm. Essentially, the algorithm searches for the top-k strongly correlated item pairs by applying a pruning method which uses the upper bound of the ϕ correlation coefficient. In other words, if the upper bound of the ϕ correlation coefficient for an item pair is less than the minimum correlation coefficient in the current top-k list, we can prune this item pair right way.

The TAPER-TOP-K algorithm is shown in Figure 7. In the algorithm, Line 6 checks the upper bound of the correlation of the item pair to determine whether it can be pruned or not. If the upper bound of an item pair is less than the minimum correlation coefficient, this item pair will be pruned, as indicated in Line 7. Lines 12 -16 show how to update and maintain the top-k list. The computation savings of the TAPER-TOP-K algorithm are due to the fact that the cost for computing the upper bound is much cheaper than the cost for computing the exact correlation coefficient.

5. TOP-COP: TOP-K Correlated Pairs Query

We first introduce a diagonal traversal method which exploits the 2-D monotone property of the upper bound of the ϕ correlation coefficient for effciently computing the top-k correlated pairs.

5.1. A Diagonal Traversal Method

In contrast to the brute-force approach and the TAPER-TOP-K method, we propose a diagonal-traversal method to efficiently compute top-k correlated pairs.

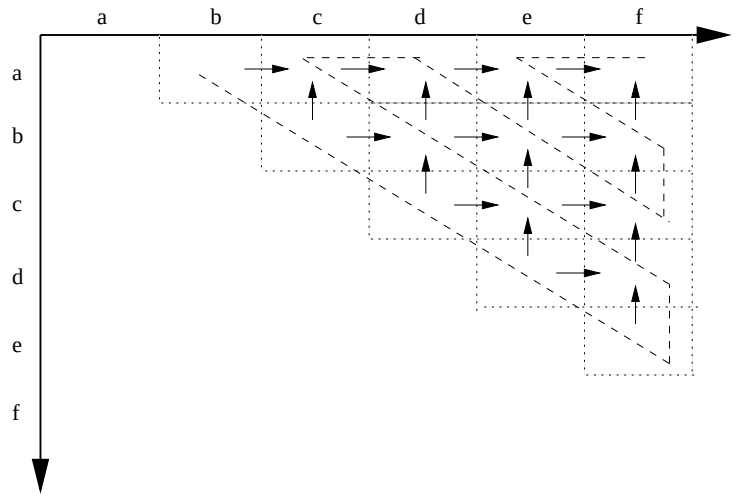


Figure 8: An Illustration of the Diagonal Traversal Method

Figure 8 illustrates the diagonal traversal method. In the figure, an item list $\{a, b, c, d, e, f\}$ is sorted by item support values in non-increasing order. The upper bound of item pairs decreases following the arrow direction according to the 2-D monotone property of the upper bound of the ϕ correlation coefficient as described in Lemma 1 and Lemma 2. The diagonal traversal method conducts a diagonal traverse to search for the top-k correlated pairs. The search starts from the principal diagonal which is traversed in the "southeast" direction, then goes to the diagonal above the main diagonal, and so on. During the iterative search process, this method maintains a top-k list and an item pair is pushed into this list if the correlation coefficient of this pair is greater than the current minimum correlation coefficient in the top-k list. The search stops if the maximal upper bound of all item pairs in a diagonal is less than the current minimum correlation coefficient in the top-k list.

Lemma 3 [**Diagonal Traversal Stop Criteria.**] *If the maximal upper bound of all item pairs in one diagonal is less than the current minimum correlation coefficient in the top-k list, then the top-k correlation pairs have been found.*

Proof. Let the minimum correlation coefficient in the current top-k list be t, and the maximal upper bound in one diagonal be max_bound. If $max\_bound < t$, we know that (1) all item pairs in this diagonal cannot have correlation coefficient above t, (2) according to Lemma 1 and Lemma 2, all the remaining item pairs that have not yet been traversed have correlation coefficient below max_bound, hence cannot have correlation coefficient above t. In other words, the current top-k list contains the item pairs with the k highest correlation coefficient in the data set.

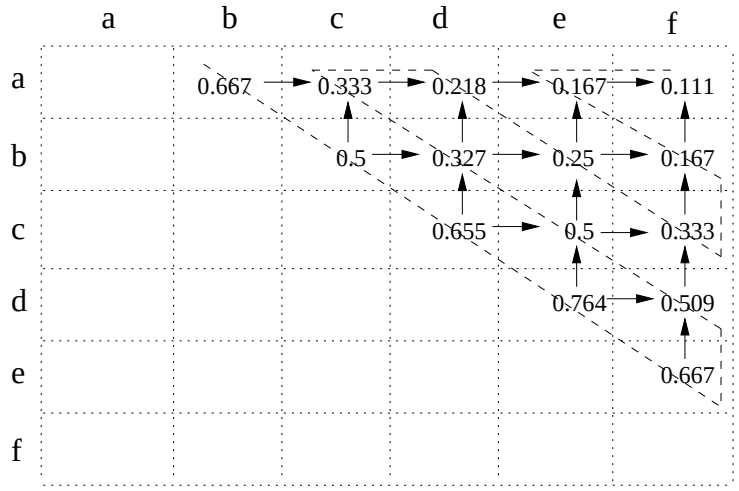


Figure 9: An Example of the Diagonal Traversal Method

Example 1 *To illustrate the diagonal traversal method, consider the data set shown in Figure 3 (a). To simplify the discussion, we use an item list* $\{a, b, c, d, e, f\}$ *which is sorted by item support values in non-increasing order. Figure 3 (b) shows the upper bound and correlation coefficients for every item pair. We can arrange all item pairs in the upper triangle of a matrix as shown in Figure 9. Suppose that we are interested in finding the top-3 strongly correlated item pairs from the data set. After traversing the main diagonal, we have a top-3 list containing three item pairs* $\{e, f\}$*,* $\{d, e\}$*, and* $\{a, b\}$*. The minimum correlation coefficient in this top-3 list is 0.667. Next, we search the super-diagonal. We can find that the maximal upper bound of all item pairs in the super-diagonal is 0.509 and is less than 0.667. Therefore, the search stops.*

5.2. Algorithm Description

In this subsection, we describe the algorithm details of the **TOP**-k **CO**rrelated **P**airs (TOP-COP) query algorithm. Essentially, the TOP-COP algorithm searches for the top-k strongly correlated item pairs in a diagonal traversal manner. During the diagonal traversal, TOP-COP applies a filter-and-refine query processing strategy.

The Filtering Step: In this step, the TOP-COP algorithm applies two pruning techniques. The first technique uses the upper bound of the ϕ correlation coefficient as a coarse filter, since the cost for computing an upper bound is much lower than that of computing the correlation coefficient. In other words, if the upper bound of the ϕ correlation coefficient for an item pair is less than the minimum correlation coefficient in the current top-k list, we can prune this item pair right way.

The second pruning technique prunes item pairs based on the 2-D monotone property of the upper bound of the ϕ correlation coefficient.

The TOP-COP Algorithm

Input: S': an item list sorted by item supports in non-increasing order.
k: The number of correlated pairs requested.

Output: The list of the top k most strongly correlated pairs.

Variables: L: the size of item set S'; TL: the current top-k list.
A: the item with larger support; B: the item with smaller support.
$status[i][j]$: the pruning status of pair $\{i, j\}$
$minPair$: the item pair in TL with the smallest correlation.
$minCorr$: the correlation value of $minPair$.

TOP-COP(S', k)

1. r = 1, $minCorr$=-1, L = size(S'), $TL = \emptyset$, $status[i][j] = 0$ for all i, j
2. **while** ($r \leq L - 1$)
3. i = 0, $flag$ = 0;
4. **for** j from r to $L - 1$
5. **if** ($status[i][j] = 0$) **then**
6. $A = S'[i]$, $B = S'[j]$, $upper(\phi) = \sqrt{\frac{supp(B)}{supp(A)}}\sqrt{\frac{1-supp(A)}{1-supp(B)}}$
7. **if**($upper(\phi) < minCorr$) **then**
8. **for** p from 0 to i
9. **for** q from j to $L - 1$
10. $status[p][q] = 1$
11. **end for**
12. **end for**
13. **else**
14. $flag = 1$, TL= Refine(A, B, TL, k)
15. **end if**
16. **end if**
17. $i = i + 1$
18. **end for**
19. **if** ($flag = 0$) **then** //search stop criterion.
20. **break** from the while loop
21. **end if**
22. $r = r + 1$
23. **end while**
24. **return** TL

Figure 10: The TOP-COP Algorithm

The Refinement Step: In the refinement step, the TOP-COP algorithm computes the exact correlation coefficient for each surviving pair from the filtering step and pushes the pair into the top-k list if the correlation coefficient of this pair is above the minimum correlation coefficient in the current top-k list.

The TOP-COP algorithm is shown in Figure 10. The diagonal traversal starts from Line 2. Line 5 checks the status array to determine whether an item pair can be pruned or not. The values stored in the status array are set based on the 2-D monotone property of the upper bound; if the upper bound of an item pair is less than the minimum correlation coefficient in the current top-k list, as indicated in Line 7, this item pair will be pruned, and Lines 8-12 will set the pruning status for all item pairs according to the 2-D monotone property. Finally, Line 14 calls the refinement procedure for all surviving item pairs in the filtering process.

Refine(A, B, TL, k) //The Refinement Step
1. Get the support $supp(A, B)$ of item set $\{A, B\}$
2. $\phi = \frac{supp(A,B) - supp(A)supp(B)}{\sqrt{supp(A)supp(B)(1-supp(A))(1-supp(B))}}$
3. **if** $\phi > minCorr$ **then**
4. **if**($size(TL) < k$) **then**
5. Push item pair $\{A, B, \phi\}$ into TL
6. **else**
7. Delete the current $minPair$ in TL and insert $\{A, B, \phi\}$ into TL
8. Update $minPair$ and $minCorr$ in TL
9. **end if**
10. **end if**
11. **return** TL

Figure 11: The Refinement Procedure

The pseudocode of the refinement procedure is shown in Figure 11. Line 1 gets the support for the item pair $\{$A, B$\}$. Line 2 calculates the correlation coefficient of this item pair. If the correlation is greater than the minimum correlation coefficient in the current top-k list, this item pair is pushed into the top-k list in line 5. Lines 4 - 9 show how to update and maintain the top-k list.

Example 2 *Figure 12 illustrates the pruning process based on the 2-D monotone property. In this example, we use the example data set shown in Figure 3. For this data set, the item list* $\{a, b, c, d, e, f\}$ *is sorted by item support in non-increasing order. Suppose that we want to find the top-3 correlated item pairs. When traversing the item pair* $\{d, e\}$*, we know that the correlation of this item pair is 0.764 (Figure 3 (b)), which is greater than the minimum correlation coefficient in the current top-3 list. Therefore, the item pair* $\{b, c\}$ *is deleted from the current top-3 list and eight item pairs* $\{a, c\}, \{a, d\}, \{a, e\}, \{a, f\}, \{b, c\}, \{b, d\}, \{b, e\}, \{b, f\}$ *can be deleted even without computing their upper bound based on the 2-D monotone property. These item pairs are presented in the rectangle as shown in Figure 12 (a). Similarly, when traversing the item pair*

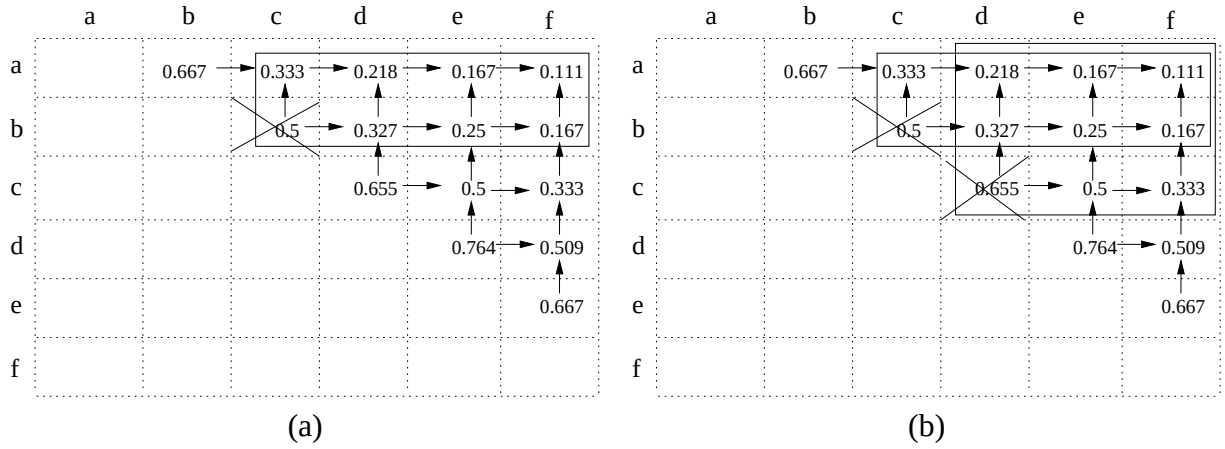


Figure 12: Illustration of the Pruning Process based on the 2-D Monotone Property

$\{e, f\}$*, we can delete the item pair* $\{c, d\}$ *and the corresponding other item pairs as shown in Figure 12 (b).*

6. Analysis of The TOP-COP Algorithm

In this section, we analyze the TOP-COP algorithm in the areas of completeness, correctness, and computational cost.

6.1. The Completeness and Correctness of the TOP-COP Algorithm

Lemma 4 *The TOP-COP algorithm is complete and correct. In other words, this algorithm finds the* k *item pairs with the* k *highest correlation coefficients among all item pairs in the data set.*

Proof: The completeness and correctness of the TOP-COP algorithm can be shown by the following three facts. The first is that all item pairs in the database have the opportunity to be checked during the iteration process due to the diagonal traversal method. The second fact is that the filtering step only prunes item pairs based on the 2-D monotone property or if the upper bounds of the ϕ correlation coefficient for these pairs are less than the minimum correlation coefficient in the current top-k list. Therefore any item pair whose correlation coefficient is among the top-k will be checked at some point, placed in the top-k list, and will not be removed from the list later. Finally, all item pairs that are not checked before the search stops cannot be in the top-k list. This is guaranteed by Lemma 3.

6.2. Computational Cost Analysis

Here, we provide simple algebraic cost models for the computational cost of the brute-force method and the TOP-COP algorithm. We assume that the total number of objects in the data set is n.

The cost of the brute-force algorithm includes the cost of computing $\frac{n(n-1)}{2}$ exact correlation coefficients and the cost of maintaining the top-k list. Let C_{corr} indicate the cost of computing the correlation coefficient for an item pair. The cost model of the brute-force algorithm is given by:

$$Cost_{Brute} = O(n^2) * C_{corr} + O(k^2) \tag{6}$$

The cost of the TOP-COP algorithm consists of four parts: the sorting cost, denoted by C_{sort}, the cost of computing the upper bounds, denoted by C_{bound}, the cost of computing the exact correlation coefficients, denoted by C_{exact}, and the cost for maintaining the top-k list. Let $\gamma_1(k)$ be the percentage of all item pairs whose upper bounds need to be computed, $\gamma_2(k)$ be the percentage of all item pairs whose exact correlation coefficients need to be computed, and C_{upper} be the cost of computing the upper bound for an item pair. Then the cost model of the TOP-COP algorithm is given by:

$$\begin{aligned} &Cost_{TOP-COP} \\ &= C_{sort} + C_{bound} + C_{exact} + O(k^2) \\ &= O(nlogn) + \gamma_1(k)O(n^2)C_{upper} + \gamma_2(k)O(n^2)C_{corr} + O(k^2) \end{aligned}$$

Thus the difference in computation cost between the brute-force method and the top-cop algorithm is:

$$\begin{aligned} &Cost_{Brute} - Cost_{TOP-COP} \\ &= (1-\gamma_2(k))O(n^2)C_{corr} - \gamma_1(k)O(n^2)C_{upper} - O(nlogn) \end{aligned}$$

Note that C_{corr} is much larger than C_{upper}, because of the high computation cost of finding support values for item pairs when computing exact correlation coefficients. In general, since the sorting cost is not significant compared to the cost for computing correlation coefficients, for large n we have:

$$\begin{aligned} &Cost_{Brute} - Cost_{TOP-COP} \\ &\approx (1-\gamma_2(k))O(n^2)C_{corr} - \gamma_1(k)O(n^2)C_{upper} \end{aligned}$$

In practice the savings will thus depend primarily on how many exact correlation coefficients need to be computed. This of course depends on the inherent structure of the data set, as we will explore further in the experiments.

A Special Case. To demonstrate the computational savings of the TOP-COP algorithm, let us consider a special case where $k << n$ and the top k item pairs are identified in the main diagonal. In other words, the search will stop after traversing the super-diagonal. For this case, in the worst scenario, TOP-COP needs to compute n upper bounds and exact correlation coefficients when traversing the main diagonal. However, according to the 2-D monotone property, the algorithm can prune $n - k + 1$ pairs along the super-diagonal since $n - k$ pairs along the main diagonal are not in the top-k list. Therefore, in the worst case, the algorithm needs to compute upper bounds and correlation coefficients for $k - 2$ pairs along the super-diagonal. Hence $\gamma_1(k) = \gamma_2(k) = (n + k - 2)/\frac{n(n-1)}{2}$ and the computational savings are:

$$
\begin{aligned}
& Cost_{Brute} - Cost_{TOP-COP} \\
& \approx (1 - (n + k - 2)/\frac{n(n-1)}{2})O(n^2)C_{corr} \\
& \quad - (n + k - 2)/\frac{n(n-1)}{2}O(n^2)C_{upper} \\
& = (\frac{n(n-1)}{2} - (n + k - 2))C_{corr} - (n + k - 2)C_{upper}
\end{aligned}
$$

Since C_{corr} is much larger than C_{upper}, the computational savings of TOP-COP will be quite large ($O(n^2)C_{corr}$) when the number of objects n in the data set is very large.

7. Experimental Results

In this section, we present the results of experiments to evaluate the performance of the TOP-COP algorithm. Specifically, we study: (1) the performance of TOP-COP compared with the brute-force approach as well as the TAPER-TOP-K method, (2) the performance of TOP-COP with respect to the dispersion of the data, (3) the performance of TOP-COP on data sets with Zipf-like distributions, and (4) the scalability of the TOP-COP algorithm.

7.1. The Experimental Setup

Our experiments were performed on both real-life and synthetic data sets, described further below. All experiments were performed on a PC with a 1.66 GHz CPU and 1 GB of RAM running the Microsoft Windows XP operating system.

Table 1: Some Characteristics of Real-life Data Sets

Data set	#Item	#Transaction	Source
Pumsb	2113	49046	IBM Almaden
Pumsb-I	2088	49046	IBM Almaden
Pumsb-II	2074	49046	IBM Almaden
Pumsb-III	2042	49046	IBM Almaden
LA1	29704	3204	TREC-5
Retail	14462	57671	Retail Store

Real-life Data Sets. The real-life data sets used in our experiments were obtained from several different application domains. Table 1 shows some characteristics of these data sets. `Pumsb` is often used as the benchmark for evaluating the performance of association rule algorithms on dense data sets. The `pumsb` data set corresponds to binarized versions of a census data set from IBM (available at http://fimi.cs.helsinki.fi/data/). The difference between `Pumsb` and `Pumsb-I`, `Pumsb-II` as well as `Pumsb-III` is that `Pumsb`$-I$ does not contain items with support greater than 80%, `Pumsb`$-II$ does not contain items with support greater than 60%, and `Pumsb`$-III$ does not contain items with support greater than 40%. The `LA1` data set is part of the TREC-5 collection (http://trec.nist.gov) and contains news articles from the Los Angeles Times. Finally, `retail` is a masked data set obtained from a large mail-order company.

7.2. A Performance Comparison of TOP-COP with the Brute-force Approach and the TAPER-TOP-K Method

First, we present a performance comparison of the TOP-COP algorithm with the brute-force approach and the TAPER-TOP-K method using several benchmark data sets from IBM, TREC, and some other sources, such as retail stores. The implementation of the brute-force approach is similar to that of the TAPER-TOP-K algorithm except that the filtering mechanism implemented in the TAPER-TOP-K algorithm is not included in the brute-force approach. The TAPER-TOP-K method is built on top of the TAPER (Xiong et al., 2004) algorithm.

Figure 13 shows the relative computation performance of TOP-COP, the brute-force approach, and TAPER-TOP-K on the `Pumsb`, `Pumsb-I`, `Retail` and `LA1` data sets. As can be seen, the performance of the brute-force approach does not change much for any of the four data sets at different k values. The execution time of the TOP-COP algorithm can be an order of magnitude faster than the brute-force approach if k is small, and TOP-COP has much better performance than TAPER-TOP-K. For instance, as shown in Figure 13 (a), the execution time of TOP-COP

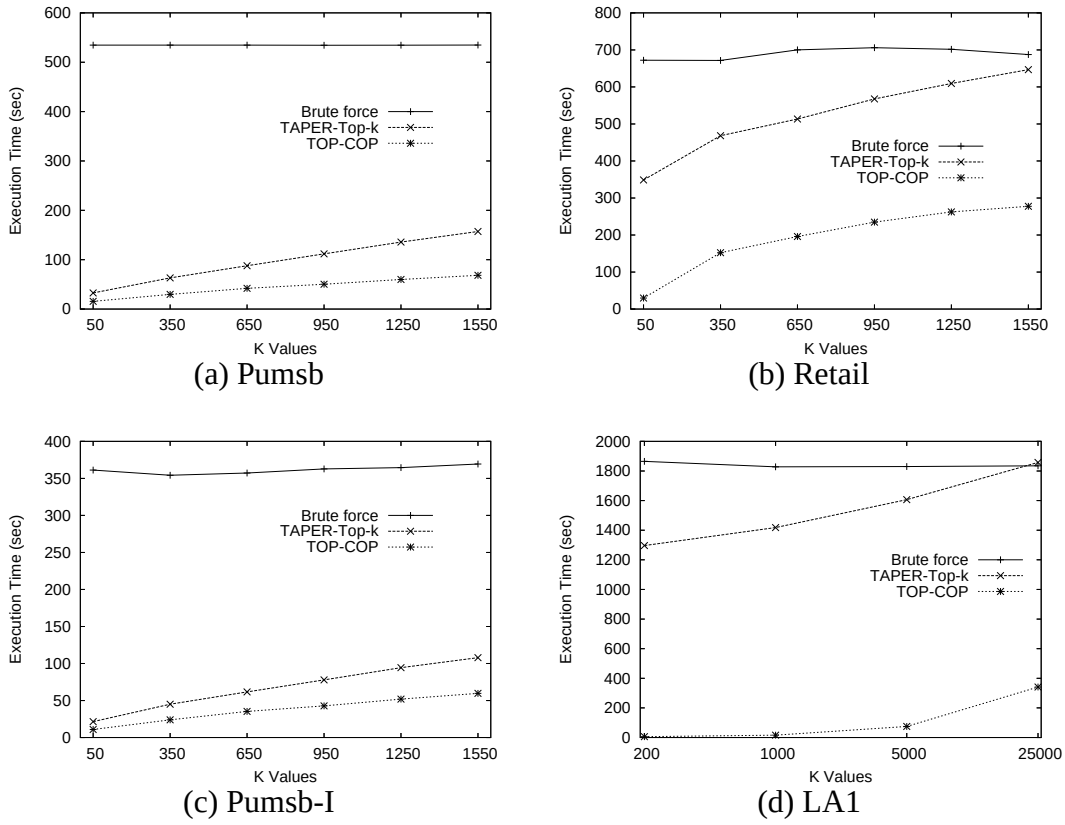


Figure 13: A Performance Comparison

on the `Pumsb` data set is one order of magnitude less than that of the brute-force approach when the k values are less than 950. Also, as the value of k increases, the execution time of TOP-COP increases accordingly. Similar computation effects can also be observed on the `Pumsb-I` , `Retail`, and `LA1` data sets, while the computation savings on these data sets are not as significant as on the $Pumsb$ data set. In the following subsection, we provide a detailed analysis to show the performance of the TOP-COP algorithm with respect to the dispersion of the data.

7.3. The Performance of the TOP-COP Algorithm with respect to the Dispersion of the Data

Here, we are interested in investigating the relationship between the performance of TOP-COP and the dispersion of the data. To this end, we first introduce the coefficient of variation (CV) (DeGroot and Schervish, 2001), which is a measure of the dispersion of a data distribution. The CV is defined as the ratio of the standard deviation to the mean. Given a set of objects $X = \{x_1, x_2, \ldots, x_n\}$, (in our case X_i will be the support of the $i'th$ data item) we have

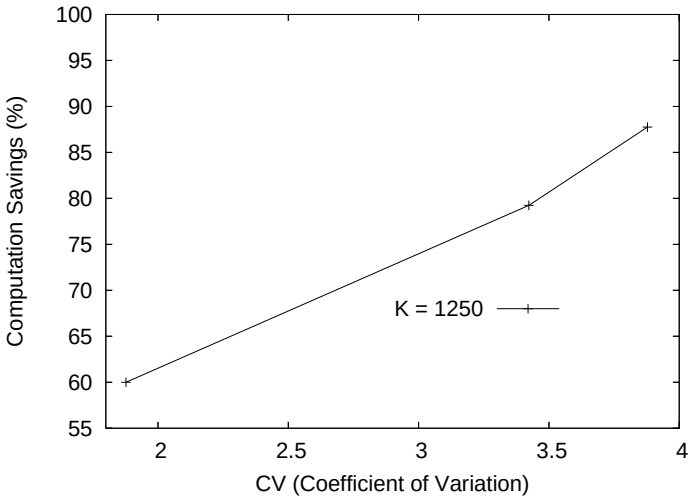


Figure 14: The Computation Savings of TOP-COP on the Retail, LA1, and Pumsb Data Sets

$$CV = \frac{s}{\bar{x}}$$

where

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \quad s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$$

Note that there are some other statistics, such as standard deviation and skewness (DeGroot and Schervish, 2001), which can also be used to characterize the dispersion of a data distribution. However, the standard deviation has no scalability; that is, the dispersion of the original data and stratified sample data is not equal if the standard deviation is used. Indeed, this does not agree with our intuition. Meanwhile, skewness cannot catch the dispersion in the situation that the data is symmetric but has high variance. In contrast, the CV is a dimensionless number that allows comparison of the variation of populations that have significantly different mean values. In general, the larger the CV value, the greater the variability is in the data.

Table 2: The Spread of Data Distributions

Dataset Name	CV (Coefficient of Variation)
Pumsb	3.878
Pumsb-I	3.857
Pumsb-II	3.803
Pumsb-III	3.645
LA1	3.423
Retail	1.877

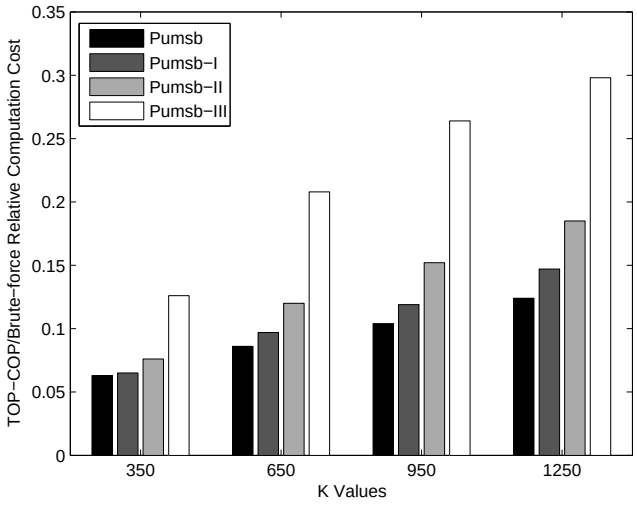


Figure 15: TOP-COP/Brute-Force Relative Computation Cost

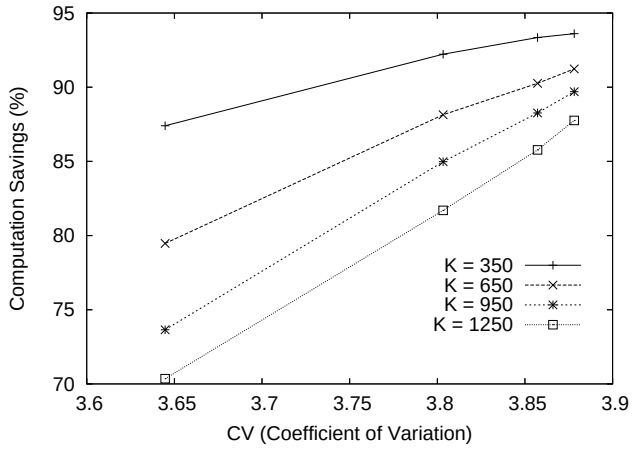


Figure 16: The Performance of TOP-COP with respect to the Dispersion of the Data Distributions

Table 2 shows the CV values of the individual item support for various data sets. In the table, we can see that $Pumsb$ has the largest CV value and $Retail$ data the smallest; i.e. the support variability of the $Pumsb$ data set is the greatest among all these data sets.

The results in Figure 14 show that the dispersion of support of individual items has a direct impact on the computation performance of the TOP-COP algorithm. Increasing the degree of dispersion of support, as measured by the increase of the CV values, increases the computation savings (defined as the number of pruned item pairs divided by the total number of item pairs) achieved by TOP-COP. For example, the TOP-COP algorithm has the highest computation savings on the Pumsb data set, since the support variability of the $Pumsb$ data set is the greatest among all these data sets.

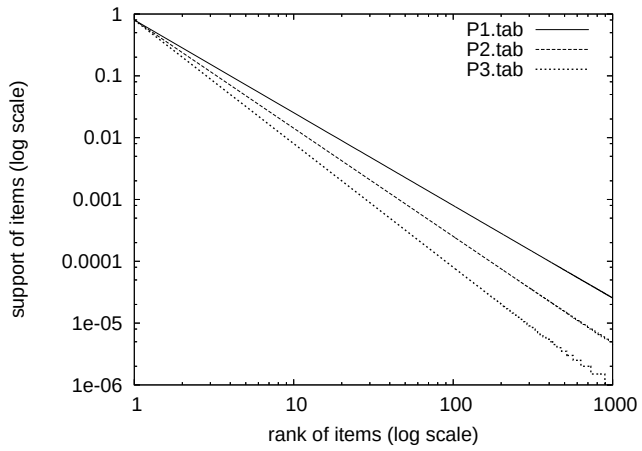


Figure 17: The Plot of the Zipf Rank-Support Distributions of Synthetic Data in Log-Log Scale

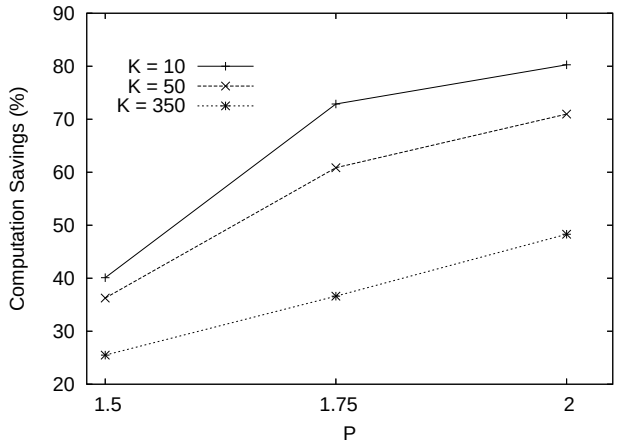


Figure 18: The Computation Savings of TOP-COP on Data Sets with Zipf Distributions

Next, to better illustrate the relationship between the performance of TOP-COP and the degree of dispersion of the data, we have conducted experiments on the Pumsb, Pumsb-I, Pumsb-II, and Pumsb-III series data sets. These data sets have exactly the same number of transactions. The difference among them is that Pumsb-I does not contain items with support greater than 80%, Pumsb-II does not contain items with support greater than 60%, and Pumsb-III does not contain items with support greater than 40%.

Figure 15 shows the relative computational cost (defined as the computational cost of TOP-COP divided by the computational cost of the brute-force method) between TOP-COP and brute-force at different k values. In the figure, we observe that the relative computational cost of TOP-COP on Pumsb is the smallest (thus the savings are the highest), since the CV value of the individual item support of Pumsb is the highest among these four data sets. Also, as we saw before, the relative computational cost increases as the value of k increases.

Finally, Figure 16 shows the corresonding computation savings of TOP-COP on these four data sets as a function of CV. The results also confirm that TOP-COP has a better computational performance on data sets with higher degrees of dispersion as measured by the larger CV values.

7.4. The Performance of TOP-COP on Data Sets with Zipf Distributions

Synthetic Data Sets. In this subsection we examine the performance of the TOP-COP algorithm on synthetic data sets with a generalized Zipf rank-support distribution. For a sorted item list, the rank-support function $f(r)$ is a discrete function which presents the support in terms of the rank r; thus $f(1)$ is the support of the item with highest support, $f(2)$ is the support of the item with second-highest support, and so on. A generalized Zipf distribution has the rank-support function $f(r) = \frac{C}{r^P}$, where C and P are constants and $P \geq 1$. When P is equal to 1, the rank-support function has a Zipf distribution. In the real world, Zipf-like distributions have been observed in a variety of application domains, including commercial retail data, Web click-streams, and telecommunication data.

Table 3: Parameters of the Synthetic Data Sets

Data set name	T	N	C	P
P1.tab	2000000	1000	0.8	1.5
P2.tab	2000000	1000	0.8	1.75
P3.tab	2000000	1000	0.8	2

In our experiments, synthetic data sets were generated such that the rank-support distributions follow Zipf's law. All the synthetic data sets have the same number of transactions and items. The rank-support distributions of these data sets follow Zipf's law but with different exponent P. A summary of the parameter settings used to generate the synthetic data sets is presented in Table 3, where T is the number of transactions, N is the number of items, C is the constant of a generalized Zipf distribution, and P is the exponent of a generalized Zipf distribution.

Figure 17 shows the rank-support plots of synthetic data sets, on a log-log scale. As can be seen, all rank-support plots are straight lines. In other words, the synthetic data sets have Zipf rank-support distributions. Also, Figure 18 displays the computational savings of the TOP-COP algorithm compared to the brute-force approaches on data sets with different exponent P. As can be seen, the computational savings of the TOP-COP algorithm increase with the increase of the exponent P at different k values.

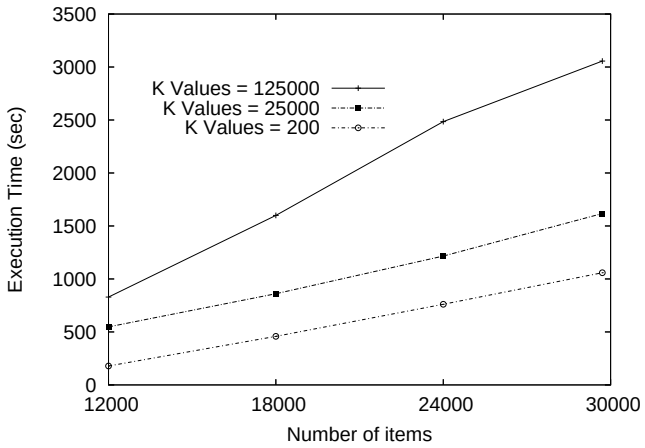


Figure 19: The Scalability of the TOP-COP Algorithm with respect to the Data Dimension

7.5. The Scalability of the TOP-COP algorithm

Finally, we show the scalability of the TOP-COP algorithm with respect to the number of items in the databases. In this experiment, we generated three data sets with the number of items equal to 12000, 18000, and 24000 from the LA1 data set by randomly sampling on all items in the LA1 data set. The above three data sets, generated by random sampling, have almost the same rank-support distributions as the LA1 data set. As a result we used these three generated data sets and the LA1 data set itself for our scale-up experiments.

Figure 19 shows the execution time of the TOP-COP algorithm for our scale-up experiments. As can be seen, the execution time increases linearly with the increase of the number of items at several different top-k values.

8. Conclusions

In this paper, we identified a 2-D monotone property of the upper bound of Pearson's correlation coefficient and provided a geometric interpretation of the computational exploration of the 2-D monotone property for efficiently computing the top-k correlated pairs. With this 2-D monotone property, we designed a **TOP**-k **CO**rrelated **P**airs (TOP-COP) query algorithm that uses a diagonal traversal method, combined with a refine-and-filter strategy, to efficiently find the top-k correlated pairs. We also proved the completeness and correctness of the TOP-COP algorithm. In addition, as demonstrated by our experiments, the TOP-COP algorithm can be an order of magnitude faster than the alternative approaches for mining the top-k correlated pairs. Finally, we showed that the performance of the TOP-COP algorithm is tightly related to the degree of data dispersion as

measured by the CV (coefficient of variation). In general, the larger the CV value of the data, the larger the computational savings achieved.

For future work, we are interested in extending the ideas developed in this paper for computing Pearson's correlation coefficient among continuous variables. Also, we are planning to design algorithms for efficiently computing some other statistical correlation measures.

Acknowledgement

This research was partially supported by a Faculty Research Grant from Rutgers Business School and an internal research grant from Vice President for research at Rutgers University. Also, we would like to thank Professor Shashi Shekhar and Professor Vipin Kumar at University of Minnesota for their valuable comments. Finally, we are grateful to Area Editor Amit Basu and the anonymous referees for their constructive comments on the paper.

References

Agrawal, R., T. Imielinski, A.N. Swami. 1993. Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD Internat. Conf. Management of Data*. ACM Press, 207–216.

Agresti, A. 2002. *Categorical Data Analysis*. Wiley Series in Probability and Statistics, Wiley Interscience.

Alexander, C. 2001. *Market Models: A Guide to Financial Data Analysis*. John Wiley & Sons.

Bayardo, R.J., R. Agrawal, D. Gunopulos. 2000. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery* **4** 217–240.

Brin, S., R. Motwani, C. Silverstein. 1997. Beyond market baskets: Generalizing association rules to correlations. *Proc. ACM SIGMOD Internat. Conf. Management of Data*. ACM Press, 265–276.

Bucila, C., J. Gehrke, D. Kifer, W.M. White. 2003. Dualminer: a dual-pruning algorithm for itemsets with constraints. *Data Mining and Knowledge Discovery* **7** 241–272.

Burdick, D., M. Calimlim, J. Gehrke. 2001. Mafia: A maximal frequent itemset algorithm for transactional databases. *Proc. Internat. Conf. Data Engrg.*. IEEE Computer Society, 443–452.

Cohen, E., M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J.D. Ullman, C. Yang. 2000. Finding interesting associations without support pruning. *Proc. Internat. Conf. Data Engrg.*. IEEE Computer Society, 489–499.

Cohen, P., J. Cohen, S.G. West, L.S. Aiken. 2002. *Applied Multiple Regression/Correlation Analysis for the Behavioral Science*. 3rd ed. Lawrence Erlbaum.

DeGroot, M., M. Schervish. 2001. *Probability and Statistics*. 3rd ed. Addison Wesley.

DuMouchel, W., D. Pregibon. 2001. Empirical bayes screening for multi-item associations. *Proc. ACM SIGKDD Internat. Conf. Knowledge Discovery and Data Mining*. ACM Press, 67–76.

Grahne, G., L.V.S. Lakshmanan, X. Wang. 2000. Efficient mining of constrained correlated sets. *Proc. Internat. Conf. Data Engrg.*. IEEE Computer Society, 512–521.

Ilyas, I.F., V. Markl, P.J. Haas, P. Brown, A. Aboulnaga. 2004. Cords: Automatic discovery of correlations and soft functional dependencies. *Proc. ACM SIGMOD Internat. Conf. Management of Data*. ACM Press, 647–658.

Jermaine, C. 2001. The computational complexity of high-dimensional correlation search. *Proc. IEEE Internat. Conf. Data Mining*. IEEE Computer Society, 249–256.

Jermaine, C. 2003. Playing hide-and-seek with correlations. *Proc. ACM SIGKDD Internat. Conf. Knowledge Discovery and Data Mining*. ACM Press, 559–564.

Kendall, M., J.D. Gibbons. 1990. *Rank Correlation Methods*. 5th ed. Oxford University Press.

Kuo, W.P., T.-K. Jenssen, A.J. Butte, L. Ohno-Machado, I.S. Kohane. 2002. Analysis of matched mrna measurements from two different microarray technologies. *Bioinformatics* **18** 405–412.

Lehmann, E.L., H.J.M. D'Abrera. 1998. *Nonparametrics: Statistical Methods Based on Ranks*. Prentice Hall.

Liu, B., W. Hsu, Y. Ma. 1999. Mining association rules with multiple minimum supports. *Proc. ACM SIGKDD Internat. Conf. Knowledge Discovery and Data Mining*.

Ng, R., L.V.S. Lakshmanan, J. Han, A. Pang. 1999. Exploratory mining via constrained frequent set queries. *Proc. ACM SIGMOD Internat. Conf. Management of Data*. ACM Press, 556–558.

Rastogi, R., K. Shim. 2002. Mining optimized association rules with categorical and numeric attributes. *IEEE Trans. Knowledge and Data Engrg.* **14** 29–50.

Reynolds, H.T. 1977. *The Analysis of Cross-Classifications*. The Free Press, New York.

Storch, H.V., F.W. Zwiers. 2002. *Statistical Analysis in Climate Research*. Reprint ed. Cambridge University Press.

Tan, P.-N., M. Steinbach, V. Kumar. 2005. *Introduction to Data Mining*. Addison-Wesley.

Wang, K., Y. He, D.W.-L. Cheung, F.Y.L. Chin. 2001. Mining confident rules without support requirement. *Proc. Internat. Conf. Information and Knowledge Management*. ACM Press, 89–96.

Xiong, H., X. He, C.H.Q. Ding, Y. Zhang, V. Kumar, S.R. Holbrook. 2005. Identification of functional modules in protein complexes via hyperclique pattern discovery. *Proc. Pacific Sympos. Biocomputing*. World Scientific.

Xiong, H., S. Shekhar, P. Tan, V. Kumar. 2004. Exploiting a support-based upper bound of pearson's correlation coefficient for efficiently identifying strongly correlated pairs. *Proc. ACM SIGKDD Internat. Conf. Knowledge Discovery and Data Mining*. ACM Press, 334–343.