TAPER: An Efficient Two-Step Approach for All-strong-pairs Correlation Query in Transaction Databases

Hui Xiong, Shashi Shekhar, Pang-Ning Tan, Vipin Kumar
Department of Computer Science, University of Minnesota at Twin Cities
{huix, shekhar, ptan, kumar}@cs.umn.edu

Abstract

Given a user-specified minimum correlation threshold θ and a transaction database with N items, all-strong-pairs correlation query finds all item pairs with correlations above the threshold θ. However, when the number of items and transactions are large, the computation cost of this query can be very high. In this paper, we identify an upper bound of Pearson's correlation coefficient for binary variables. This upper bound is not only much cheaper to compute than Pearson's correlation coefficient but also exhibits a special monotone property which allows pruning of many item pairs even without computing their upper bounds. A **T***wo-step* **A***ll-strong-***P***airs corr***E***lation que***R***y (TAPER) algorithm is proposed to exploit these properties in a filter-and-refine manner. Furthermore, we provide an algebraic cost model which shows that the computation savings from pruning is independent or improves when the number of items is increased in data sets with common Zipf or linear rank-support distributions. Experimental results from synthetic and real data sets exhibit similar trends and show that the TAPER algorithm can be an order of magnitude faster than brute-force alternatives.*

1 Introduction

Correlation analysis is important for various application domains such as marketing data analysis [3], climatology [19], and public health [7]. For instance, correlation analysis in marketing data study can reveal how the sales of a product are related with the sales of other products. A major retailer found that the sales of beer were linked to the sales of diapers during weekday evenings in blue-collar areas. This type of information can be useful for sales promotions, catalog design, and store layout.

In this paper, we focus on all-strong-pairs correlation query which retrieves all pairs of items with high positive correlation in transaction databases. The problem can be formalized as follows. Given a user-specified minimum correlation threshold θ and a transaction database with N items and T transactions, all-strong-pairs correlation query finds all item pairs with correlations above the minimum correlation threshold θ. This query can be expressed in SQL syntax as follows:

select $I_1.id$, $I_2.id$
from $ITEM\ I_1$, $ITEM\ I_2$
where $I_1.id < I_2.id$ and $correlation(I_1, I_2) > \theta$;

All-strong-pairs Correlation Query

INPUT:

1) Transaction Database

TID	Items
1	1, 2, 3
2	1, 2, 3
3	1, 3
4	1, 2
5	1, 2
6	1, 2
7	1, 2, 3, 4, 5, 6
8	1, 2, 4, 5
9	1, 2, 4
10	3

2) Correlation threshold = 0.6

Pair	Correlation	Support
{1, 2}	0.667	0.8
{1, 3}	-0.333	0.4
{1, 4}	0.218	0.3
{1, 5}	0.167	0.2
{1, 6}	0.111	0.1
{2, 3}	-0.5	0.3
{2, 4}	0.327	0.3
{2, 5}	0.25	0.2
{2, 6}	0.167	0.1
{3, 4}	-0.218	0.1
{3, 5}	0	0.1
{3, 6}	0.333	0.1
{4, 5}	0.764	0.2
{4, 6}	0.509	0.1
{5, 6}	0.667	0.1

OUTPUT:
{1, 2}
{4, 5}
{5, 6}

Figure 1. All-strong-pairs Correlation Query Problem Illustration

Example 1 *Figure 1 illustrates the all-strong-pairs correlation query problem. One of the problem inputs is a transaction database with 10 transactions. Each transacion includes items bought by one customer. There are a total of six items in the database. In other words, $\binom{6}{2}$ = 15 pairs are candidates which are listed along with their correlations.*

For a correlation threshold of 0.6, the all-strong-pairs correlation query will return three pairs {1, 2}, {4, 5}, and {5, 6} out of 15 pairs as query results.

However, as the number of items and transactions become large, the computation cost of all-strong-pairs correlation query can be prohibitively expensive. Consider a set of 10^6 items which may represent a set of published book available on an e-commerce web site. There are a total of $\binom{10^6}{2} \approx 0.5 \times 10^{12}$ item pairs. It will be extremely difficult for a brute-force algorithm to compute correlations for all half trillion item pairs, particularly when the number of transactions are also large.

The all-strong-pairs correlation query problem is different from the association-rule mining problem [1, 2, 5, 6, 8, 13, 16]. Given a set of transactions, one objective of association rule mining is to extract all subsets of items that satisfy a minimum support threshold. Support measures the fraction of transactions that contain a given subset of items. Consider the transaction database shown in Figure 1. Let $supp(i)$ denote the support for item set i. Then $supp(1, 2) = 8/10 = 0.8$, which is the fraction of transactions that contain item 1 and item 2. Standard association rule mining algorithms rely on support-based pruning to find high support patterns. However, it is well-known that an item pair with high support may have a very low correlation. Additionally, some item pairs with high correlations may also have very low support. For instance, suppose we have an item pair {A, B} such that supp(A)=supp(B)= 0.8 and supp(A, B) = 0.64. Both items are uncorrelated because supp(A, B) = supp(A)supp(B). In contrast, we can also have an item pair {A, B} with supp(A) = supp(B) = supp(A,B) =0.001. Both items are perfectly correlated in this case. In fact, the patterns with low support but high correlation are useful for capturing interesting associations among rare but expensive items such as gold necklaces and earrings.

In this paper, we provide an upper bound of Pearson's correlation coefficient for binary variables. The computation of this upper bound is much cheaper than the compuation of the exact correlation, since this upper bound is computable by only individual item supports. Furthermore, we show that this upper bound has a special monotone property which allows elimination of many item pairs even without computing their upper bounds, as shown in Figure 2. The x-axis in the Figure represents the set of items with support that is lower than the support of item x_i. These items are sorted from left to right in decreasing order of their support. $Upperbound(x_i, x)$ represents the upper bound of $correlation(x_i, x)$ and has a monotone behavior. This guarantees that an item pair (x_i, x_k) can be pruned if $upperbound(x_i, x_j) < \theta$ and $supp(x_k) < supp(x_j)$. A **T**wo-step **A**ll-strong-**P**airs corr**E**lation que**R**y (TAPER) algorithm is proposed to exploit these properties in a filter-and-refine manner [4, 14, 15] which consists of two steps: filtering and refinement. In the filtering step, many item pairs are filtered out using the easy-to-compute $upperbound(x_i, x)$ and its monotone property. In the refinement step, the exact correlation is computed for remaining pairs to determine the query results.

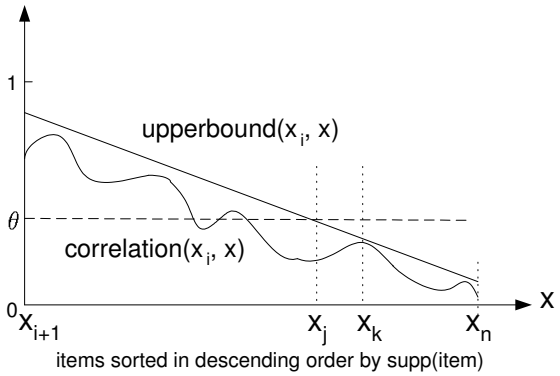


Figure 2. Illustration of the Filtering Techniques in the TAPER Algorithm.

Furthermore, we have proved the completeness and correctness of the TAPER algorithm and provided an algebraic cost model to quantify the computation savings. As demonstrated by our experiments on both real and synthetic data sets, TAPER can be an order of magnitude faster than brute-force alternatives and the computational savings by TAPER is independent or improves when the number of items is increased in data sets with common Zipf [20] or linear rank-support distributions.

1.1 Related Work

Related literature can be categorized by the correlation measures. Approaches to efficient computation of correlated variables measured by Chi-Square statistic [17] including Jermaine et.al [9] and Brin et.al [5].

Jermaine [9] investigated the implication of incorporating chi-square 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. Also, Brin [5] proposed a χ^2-based correlation rule mining strategy. The algorithm in this strategy is similar to the Apriori [2].

In this paper, we focus on efficiently computing Pearson's correlation coefficient for binary variables. Given n items, a brute force approach computes Pearson's correlation coefficient for all $\binom{n}{2} = \frac{n(n-1)}{2}$ item pairs. This approach is often implemented using matrix algebra in statistical software package as the "correlation matrix" [10] function, which computes Pearson's correlation coefficient

for all pairs of columns. This approach is applicable to but not efficient for the case of boolean matrices, which can model transaction databases. The approach proposed in this paper does not need to compute all $\binom{n}{2}$ pairs. In particular, for transaction databases with a Zipf-like rank-support distribution, we show that only a small portion of the item pairs needs to be examined.

1.2 Overview and Scope

The remainder of this paper is organized as follows. Section 2 introduce basic concepts. In section 3, we introduce the upper bound of Pearson's correlation coefficient for binary variables. Section 4 proposes the TAPER algorithm. In section 5, we analyze the TAPER algorithm in the areas of completeness, correctness, and computation gain. Section 6 presents the experimental results. Finally, in section 7, we draw conclusions and suggest future work.

The scope of the all-strong-pairs correlation query problem proposed in this paper is restrict to transaction databases with binary variables and the correlation compuation form is Pearson's correlation coefficient for binary variables, which is also called the ϕ correlation coefficient. Furthermore, we assume that the support of items is between 0 and 1 but not equal to either 0 or 1. These boundary cases can be handled separately.

2 Pearson's Correlation Coefficient for Binary Variables

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 and Spearman's Rank Correlation Coefficient [11, 12]. In contrast, relationships among nominal variables can be analyzed with nominal measures of association such as Pearson's Correlation Coefficient, the Odds Ratio, and measures based on Chi Square [17].

The ϕ correlation coefficient [17] is the computation form of Pearson's Correlation Coefficient for binary variables. In this section, we describe the ϕ correlation coefficient and show how it can be computed using the support measure of association-rule mining [1].

In a 2×2 two-way table shown in Figure 3, the calculation of the ϕ correlation coefficient reduces to

$$\phi = \frac{P_{(00)}P_{(11)} - P_{(01)}P_{(10)}}{\sqrt{P_{(0+)}P_{(1+)}P_{(+0)}P_{(+1)}}}, \quad (1)$$

		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 3. Two way table of item A and item B.

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 we let $P_{(+j)}$ denote the total number of samples classified in the jth column. Thus,

$$P_{(i+)} = \sum_{j=0}^{1} P_{(ij)} \quad and \quad P_{(+j)} = \sum_{i=0}^{1} P_{(ij)}$$

In the two-way table, N is the total number of samples and $N = P_{(0+)} + P_{(1+)} = P_{(+0)} + P_{(+1)}$. Furthermore, we can transform Equation 1 as follows.

$$\phi = \frac{(N - P_{(01)} - P_{(10)} - P_{(11)})P_{(11)} - P_{(01)}P_{(10)}}{\sqrt{P_{(0+)}P_{(1+)}P_{(+0)}P_{(+1)}}}$$
$$\phi = \frac{NP_{(11)} - (P_{(11)} + P_{(10)})(P_{(01)} + P_{(11)})}{\sqrt{P_{(0+)}P_{(1+)}P_{(+0)}P_{(+1)}}}$$
$$\phi = \frac{NP_{(11)} - P_{(1+)}P_{(+1)}}{\sqrt{P_{(0+)}P_{(1+)}P_{(+0)}P_{(+1)}}}$$
$$\phi = \frac{\frac{P_{(11)}}{N} - \frac{P_{(1+)}}{N}\frac{P_{(+1)}}{N}}{\sqrt{\frac{P_{(0+)}}{N}\frac{P_{(1+)}}{N}\frac{P_{(+0)}}{N}\frac{P_{(+1)}}{N}}}$$

Hence, when adopting the support measure of association rule mining [1], for two items A and B in a transaction database, we have $supp(A) = P_{(1+)}/N$, $supp(B) = P_{(+1)}/N$, and $supp(A, B) = P_{(11)}/N$. With support notations and the above new derivations of Equation 1, we can derive the support form of the ϕ correlation coefficient as shown below in Equation 2.

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

Example 2 *Consider the transaction database shown in Figure 1. For item 1 and item 2 in the database, we can construct a two-way table as shown in Figure 4. Then, by Equation 1, we get* $\phi = \frac{1\times8-1\times0}{\sqrt{1\times9\times2\times8}} = \frac{2}{3}$. *Also, since* $supp(1, 2) = 0.8$, $supp(1) = 0.9$, *and* $supp(2) = 0.8$, *by Equation 2, we get* $\phi = \frac{0.8-0.8\times0.9}{\sqrt{0.8\times0.9\times0.1\times0.2}} = \frac{0.08}{0.12} = \frac{2}{3}$. *As can be seen, the results from the two equations are identical.*

		{2}		Row Total
		0	1	
{1}	0	1	0	1
	1	1	8	9
Column Total		2	8	10

Figure 4. Two way table of item 1 and item 2.

3 Properties of the ϕ Correlation Coefficient

In this section, we present some properties of the ϕ correlation coefficient. These properties are useful for the efficient computation of all-strong-pairs correlation query.

3.1 An Upper Bound of the ϕ Correlation Coefficient

In this subsection, we reveal that the support measure is closely related with the ϕ correlation coefficient. Specifically, we prove that an upper bound of the ϕ correlation coefficient for a given pair {A, B} exists and is determined only by the support value of item A and the support value of item B as shown below in Lemma 1.

Lemma 1 *Given an item pair {A, B}, the support value supp(A) for item A, and the support value supp(B) for item B, without loss of generality, let $supp(A) \geq supp(B)$. The upper bound upper($\phi_{\{A,B\}}$) of an item pair {A, B} can be obtained when supp(A, B) = supp(B) and*

$$upper(\phi_{\{A,B\}}) = \sqrt{\frac{supp(B)}{supp(A)}} \cdot \sqrt{\frac{1-supp(A)}{1-supp(B)}} \quad (3)$$

Proof: According to Equation 2, for an item pair {A, B}:

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

When the support values supp(A) and supp(B) are fixed, $\phi_{\{A,B\}}$ is monotonically increasing with the increase of the support value supp(A, B). By the given condition $supp(A) \geq supp(B)$ and the anti-monotone property of the support measure, we get the maximum possible value of supp(A, B) is supp(B). As a result, the upper bound upper($\phi_{\{A,B\}}$) of an item pair {A, B} can be obtained when supp(A, B) = supp(B). Hence,

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

As can be seen in Equation 3, the upper bound of ϕ correlation coefficient for an item pair {A, B} only relies on the support value of item A and the support value of item B. In other words, there is no requirement to get the support value supp(A, B) of an item pair {A, B} for the calculation of this upper bound. As we know, when the number of items N becomes very large, it will be difficult to store the support of every item pair in the memory, since $N * (N-1)/2$ is a huge number. However, it is possible for us to store the support of individual item in the memory. As a result, this upper bound can serve as a coarse filter to filter out item pairs which are of no interest, thus saving I/O cost by reducing the computation of the support value of those pruned pairs.

3.2 Conditional Monotone Property

In this subsection, we present a conditional monotone property of the upper bound of the ϕ correlation coefficient as shown below in Lemma 2

Lemma 2 *For a pair of items {A, B}, if we let $supp(A) > supp(B)$ and fix the item A, the $\phi_{\{A,B\}_{MPCV}}$ of pair {A, B} is monotonically decreasing with the decrease of the support value of item B.*

Proof: By Lemma 1, we get:

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

For any given two items B_1 and B_2 with $supp(A) > supp(B_1) > supp(B_2)$, we need to prove $upper(\phi_{\{A,B_1\}}) > upper(\phi_{\{A,B_2\}})$. This claim can be proved as follows:

$$\frac{upper(\phi_{\{A,B_1\}})}{upper(\phi_{\{A,B_2\}})} = \sqrt{\frac{supp(B_1)}{supp(B_2)}} \cdot \sqrt{\frac{1-supp(B_2)}{1-supp(B_1)}} > 1$$

The above follows the given condition that $supp(B_1) > supp(B_2)$ and $(1-supp(B_1)) < (1-supp(B_2))$.

Lemma 2 allows us to push the upper bound of the ϕ correlation coefficient into the search algorithm, thus efficiently pruning the search space.

Corollary 1 *When searching for all pairs of items with correlations above a user-specified threshold θ, if an item list $\{i_1, i_2, \ldots, i_m\}$ is sorted by item supports in non-increasing order, an item pair $\{i_a, i_c\}$ with $supp(i_a) > supp(i_c)$ can be pruned if $upper(\phi(i_a, i_b)) < \theta$ and $supp(i_c) \leq supp(i_b)$.*

Proof: First, when $supp(i_c) = supp(i_b)$, we get $upper(\phi(i_a, i_c)) = upper(\phi(i_a, i_b)) < \theta$ according to

Equation 3 and the given condition $upper(\phi(i_a, i_b)) < \theta$, then we can prune the item pair $\{i_a, i_c\}$. Next, we consider $supp(i_c) < supp(i_b)$. Since $supp(i_a) > supp(i_b) > supp(i_c)$, by Lemma 2, we get $upper(\phi(i_a, i_c)) < upper(\phi(i_a, i_b)) < \theta$. Hence, the pair $\{i_a, i_c\}$ is pruned.

4 TAPER: A Two-Step All-strong-Pairs Correlation Query Algorithm

In this section, we present a **T**wo-step **A**ll-strong-**P**airs corr**E**lation que**R**y (TAPER) algorithm. The TAPER algorithm is a two-step filter-and-refine query processing strategy which consists of two steps: filtering and refinement.

A Filtering Step

In the filtering step, the TAPER algorithm applies two pruning techniques. The first technique uses the upper bound of the ϕ correlation coefficient as a coarse filter. In other words, if the upper bound of the ϕ correlation coefficient for an item pair is less than the user-specified correlation threshold, we can prune this item pair right way. The second pruning technique prunes item pairs based on the conditional monotone property of the upper bound of the ϕ correlation coefficient. The correctness of this pruning is guaranteed by Corollary 1 and the process of this pruning is illustrated in Figure 2 as previously noted in introduction section. In summary, the purpose of the filtering step is to reduce false positive item pairs and further processing cost.

A Refinement Step

In the refinement step, the TAPER algorithm computes the exact correlation for each surviving pair from the filtering step and retrieves the pairs with correlations above the user-specified correlation threshold as the query results.

Figure 5 shows the pseudocode of the TAPER algorithm, including $CoarseFilter$ and $Refine$ procedures.

Procedure $CoarseFilter$ works as follows. Line 1 initialize the variables and creates an empty query result set P. Lines 2 - 10 use Rymon's generic set-enumeration tree search framework [18] to enumerate candidate pairs and filter out item pairs whose correlations are obviously less than the user-specified correlation threshold θ. Line 2 starts an outer loop. Each outer loop corresponds to a search tree branch. Line 3 specifies the reference item A and Line 4 starts a search within each branch. Line 5 specifies the target item B and line 6 computes the upper bound of the ϕ correlation coefficient for item pair {A, B}. In line 7, if this upper bound is less than the user-specified correlation threshold θ, the search within this branch can stop by exiting from the inner loop as shown in line 8. The reason is as follows. First, the reference item A is fixed in each branch and it has the maximum support value due to the way we construct the branch. Also, items within each branch are sorted based on their support in non-increasing order. Then, by Lemma 2, the upper bound of the ϕ correlation coefficient for the item pair {A, B} is monotonically decreasing with the decrease of the support of item B. Hence, if we find the first target item B which results in the situation that the $upper(\phi_{\{A,B\}})$ is less than the user-specified correlation threshold θ, we can stop the search in this branch. Line 10 calls the procedure Refine to compute the exact correlation for each surviving candidate pair and continues to check the next target item until no target item is left in the current search branch.

Procedure $Refine$ works as follows. Line 12 gets the support for the item pair {A, B}. Note that the I/O cost can be very expensive for line 12 when the number of items is

TAPER ALGORITHM

Input: S': an item list sorted by item supports in non-increasing order.
θ: a user-specified minimum correlation threshold.
Output: P: the result of all-strong-pairs correlation query.
Variables: L: the size of item set S'.
A: the item with larger support.
B: the item with smaller support.

```
//The Filtering Step
CoarseFilter(S', θ)
1.  L = size(S'), P = ∅
2.  for i from 0 to L-1
3.      A = S'[i]
4.      for j from i+1 to L
5.          B = S'[j]
6.          upper(φ) = sqrt(supp(B)/supp(A)) · sqrt((1-supp(A))/(1-supp(B)))
7.          if(upper(φ) < θ) then
                //Pruning by the monotone property
8.              break from inner loop
9.          else
10.             P=P ∪ Refine(A, B, θ)
11.     end

//The Refinement Step
Refine(A, B, θ)
12.     Get the support supp(A, B) of item set {A, B}
13.     φ = (supp(A,B) - supp(A)supp(B)) / sqrt(supp(A)supp(B)(1-supp(A))(1-supp(B)))
14.     if φ < θ then
15.         return ∅ //return NULL
16.     else
17.         return {{A, B}, φ}
```

Figure 5. The TAPER Algorithm

large since we cannot store the support of all item pairs in the memory. Line 13 calculates the exact correlation coefficient of this item pair. If the correlation is greater than the user-specified correlation threshold, this item pair is returned as a query result in line 17. Otherwise, the procedure returns NULL in line 15.

Example 3 *To illustrate the TAPER algorithm, consider a database shown in Figure 6. To simplify the discussion, we use an item list* $\{1, 2, 3, 4, 5, 6\}$ *which is sorted by item support in non-increasing order. For a given correlation threshold 0.36, we can use Rymon's generic set-enumeration tree search framework [18] to demonstrate how two-step filter-and-refine query processing works. For instance, for the branch starting from item 1, we identify that the upper bound of the* ϕ *correlation coefficient for the item pair* $\{1, 3\}$ *is 0.333, which is less than the given correlation threshold 0.36. Hence, we can prune this item pair immediately. Also, since the item list* $\{1, 2, 3, 4, 5, 6\}$ *is sorted by item supports in non-increasing order, we can prune pairs* $\{1, 4\}$, $\{1, 5\}$, *and* $\{1, 6\}$ *by Lemma 2 without any further computation cost. In contrast, for the traditional filter-and-refine paradigm, the coarse filter can only prune the item pair* $\{1, 3\}$. *There is no technique to prune item pairs*$\{1, 4\}$, $\{1, 5\}$, *and* $\{1, 6\}$. *Finally, in the refinement step, only seven item pairs are required to compute the exact correlation coefficients, as shown in Figure 6 (c). More than half of the item pairs are pruned in the filter step even though the correlation threshold is as low as 0.36.*

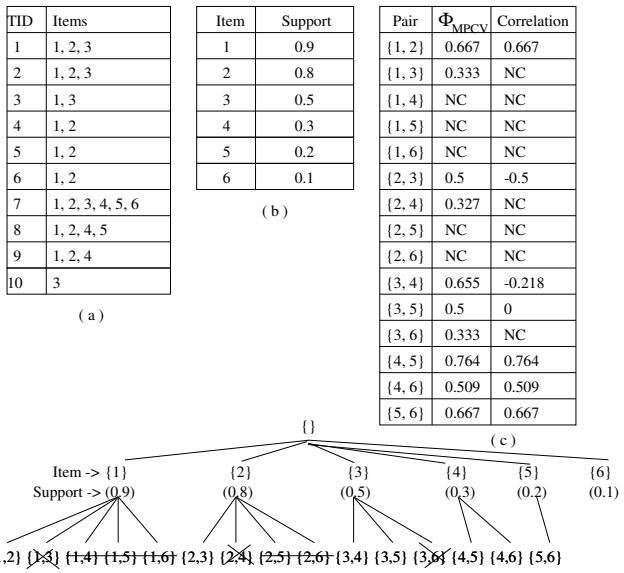


Figure 6. Illustration of the filter-and-refine strategy. Note that NC means there is no computation required.

5 Analysis of the TAPER algorithm

In this section, we analyze TAPER in the areas of completeness, correctness, and the computation savings.

5.1 Completeness and Correctness

Lemma 3 *The TAPER algorithm is complete. In other words, this algorithm finds all pairs which have correlations above a user-specified minimum correlation threshold.*

Proof: The completeness of the TAPER algorithm can be shown by the following two facts. The first is that a set-enumeration tree search [18] is complete; that is, all item pairs in the database are enumerated during the search process. The second fact is that the filtering step only prunes item pairs if the upper bounds of the ϕ correlation coefficient for these pairs are less than the user-specified correlation threshold. This is guaranteed by Corollary 1. Also, the refinement step only prunes item pairs whose correlations are less than the user-specified correlation threshold. The second fact can guarantee that all the pruned item pairs cannot have correlations above the user-specified minimum correlation threshold.

Lemma 4 *The TAPER algorithm is correct. In other words, every pair this algorithm finds has the correlation above a user-specified minimum correlation threshold.*

Proof: The correctness of the TAPER algorithm can be guaranteed by the refinement step, since the exact correlation of each candidate pair is calculated in the refinement step and every pair which has the correlation less than the user-specified correlation threshold will be pruned.

5.2 Quantifying the Computation Savings

In this subsection, we examine the computation savings of TAPER. To facilitate our discussion, we first introduce some definitions of notations and terms as follows.

Definition 1 *The pruning ratio of the TAPER algorithm is defined by the following equation.*

$$\gamma(\theta) = \frac{S(\theta)}{T}, \tag{4}$$

where θ *is the minimum correlation threshold,* $S(\theta)$ *is the number of item pairs which are pruned before computing their exact correlations at the correlation threshold* θ*, and* T *is the total number of item pairs in the database. For a given database,* T *is a fixed number and is equal to* $\binom{n}{2} = \frac{n(n-1)}{2}$*, where n is the number of items.*

Definition 2 *For a sorted item list, the rank-support function $f(k)$ is a discrete function which present the support in terms of the rank k.*

Next, we would like to quantify the computation savings in terms of the rank-support function. For a given database, let $I = \{A_1, A_2, \ldots, A_n\}$ be an item list sorted by item supports in non-increasing order. Then the item A_1 has the maximum support and the rank-support function $f(k) = supp(A_k)$, $\forall\ 1 \leq k \leq n$, which is monotonically decreasing with the increase of the rank k. To quantify the computation savings for a given item A_j $(1 \leq j < n)$ at the threshold θ, we only need to find the first item A_l $(j < l \leq n)$ such that $upper(\phi_{\{A_j,A_l\}}) < \theta$. By Lemma 2, if $upper(\phi_{\{A_j,A_l\}}) < \theta$, we can guarantee that $upper(\phi_{\{A_j,A_i\}})$, where $l \leq i \leq n$, is less than the correlation threshold θ. In other words, all these $n-l+1$ pairs can be pruned without further computation requirement. According to Lemma 1, we get

$$\begin{aligned} & upper(\phi_{\{A_j,A_l\}}) \\ &= \sqrt{\frac{supp(A_l)}{supp(A_j)}} \cdot \sqrt{\frac{1-supp(A_j)}{1-supp(A_l)}} \\ &< \sqrt{\frac{supp(A_l)}{supp(A_j)}} = \sqrt{\frac{f(l)}{f(j)}} < \theta \\ &\Rightarrow \frac{f(l)}{f(j)} < \theta^2 \end{aligned}$$

Since the rank-support function f(k) is monotonically decreasing with the increase of the rank k, we get

$$l > f^{-1}(\theta^2 f(l))$$

To make the computation simple, we let $l = f^{-1}(\theta^2 f(l)) + 1$. Therefore, for a given item A_j $(1 < j \leq n)$, the computation cost for $(n - f^{-1}(\theta^2 f(j)))$ item pairs can be saved. As a result, the total computation savings of the TAPER algorithm is shown below in Equation 5. Note that the computation savings shown in Equation 5 is an underestimated value of the real computation savings which can be achieved by the TAPER algorithm.

$$S(\theta) = \sum_{j=2}^{n} \{n - f^{-1}(\theta^2 f(j))\} \tag{5}$$

Finally, we would like to conduct computation saving analysis on the data sets with some special rank-support distributions. Specifically, we consider three special rank-support distributions: a uniform distribution, a linear distribution, and a generalized Zipf distribution [20], as shown in the following three cases.

CASE I: A Uniform Distribution.

In this case, the rank-support function $f(k) = C$, where C is a constant. According to Equation 3, the upper bound of the ϕ correlation coefficient for any item pair is 1, which is the maximum possible value for the correlation. Hence, for any given item A_j, we cannot find an item A_l $(j < l \leq n)$ such that $upper(\phi_{\{A_j,A_l\}}) < \theta$, where $\theta \leq 1$. As a result, the total computation savings $S(\theta)$ is zero in this case.

CASE II: A Linear Distribution.

In this case, the rank-support function has a linear distribution and $f(k) = a - m \times k$, where m is the absolute value of the slope and a is the intercept

Lemma 5 *When a database has a linear rank-support distribution $f(k)$ and $f(k) = a - m \times k$ $(a > 0,\ m > 0)$, for a user-specified minimum correlation threshold θ, the pruning ratio of the TAPER algorithm increases with the decrease of the ratio a/m, the increase of the correlation threshold θ, and the increase of the number of items, where $0 < \theta \leq 1$.*

Proof: For the given database, let $I = \{A_1, A_2, \ldots, A_n\}$ be the item list sorted by item support in non-increasing order. Then the item A_1 has the maximum support. Also, let the rank-support function $f(k) = a - m \times k$, where m is the absolute value of the slope and a is the intercept. From the rank-support function $f(k)$, we can derive the inverse function $f^{-1}(y) = \frac{a-y}{m}$. Accordingly,

$$f^{-1}(\theta^2 f(j)) = \frac{a - \theta^2(a - mj)}{m} = \frac{a}{m}(1-\theta^2) + j\theta^2$$

According to Equation 5, we can get:

$$\begin{aligned} S(\theta) &= \sum_{j=2}^{n} \{n - f^{-1}(\theta^2 f(j))\} \\ &= n(n-1) - \sum_{j=2}^{n} f^{-1}(\theta^2 f(j)) \\ &= n(n-1) - \sum_{j=2}^{n} \frac{a}{m}(1-\theta^2) - \sum_{j=2}^{n} j\theta^2 \\ &= n(n-1) - \frac{a}{m}(n-1)(1-\theta^2) - \frac{(n-1)(n+2)}{2}\theta^2 \\ &= (n-1)(n - \frac{a}{m}(1-\theta^2) - \frac{(n+2)}{2}\theta^2) \\ &= (n-1)(\frac{n-2}{2} - (\frac{a}{m} - \frac{n+2}{2})(1-\theta^2)) \end{aligned}$$

Since the pruning ratio $\gamma(\theta) = \frac{S(\theta)}{T}$,

$$\begin{aligned} \Rightarrow\ \gamma(\theta) &= \frac{(n-1)(\frac{n-2}{2} - (\frac{a}{m} - \frac{n+2}{2})(1-\theta^2))}{\frac{n(n-1)}{2}} \\ \Rightarrow\ \gamma(\theta) &= \frac{(n-2) - (2\frac{a}{m} - (n+2))(1-\theta^2)}{n} \end{aligned}$$

Also, we know $supp(A_n) = f(n) = a - m \times n > 0$,

$$\Rightarrow \frac{a}{m} > n$$
$$\Rightarrow 2\frac{a}{m} > 2n \geq (n+2),\ when\, n \geq 2$$

Thus, we can derive three rules as follows:

$$rule\,1:\ \theta \nearrow \Rightarrow\ (1-\theta^2) \searrow \Rightarrow\ \gamma(\theta) \nearrow$$
$$rule\,2:\ a/m \searrow \Rightarrow\ (2\frac{a}{m} - (n+2)) \searrow \Rightarrow\ \gamma(\theta) \nearrow$$
$$rule\,3:\ n \nearrow \Rightarrow\ (2\frac{a}{m} - (n+2)) \searrow \Rightarrow\ \gamma(\theta) \nearrow$$

Therefore, the claim that the pruning ratio of the TAPER algorithm is increased with the decrease of the ratio a/m, the increase of the correlation threshold θ, and the increase of the number of items holds .

CASE III: A Generalized Zipf Distribution.

In this case, the rank-support function has a generalized Zipf distribution and $f(k) = \frac{c}{k^p}$, 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 has been observed in a variety of application domains, including commercial retail data, Web click-streams, and telecommunication data.

Lemma 6 *When a database has a generalized Zipf rank-support distribution $f(k)$ and $f(k) = \frac{c}{k^p}$, for a user-specified minimum correlation threshold θ, the pruning ratio of the TAPER algorithm increases with the increase of p and the correlation threshold θ, where* $0 < \theta \leq 1$.

Proof: Since the rank-support function $f(k) = \frac{c}{k^p}$, the inverse function $f^{-1}(y) = (\frac{c}{y})^{\frac{1}{p}}$. Accordingly,

$$f^{-1}(\theta^2 f(j)) = (\frac{c}{\theta^2 \frac{c}{j^p}})^{\frac{1}{p}} = \frac{j}{(\theta^2)^{\frac{1}{p}}}$$

Applying Equation 5, we get:

$$\begin{aligned} S(\theta) &= \sum_{j=2}^{n} \{n - f^{-1}(\theta^2 f(j))\} \\ &= n(n-1) - \sum_{j=2}^{n} f^{-1}(\theta^2 f(j)) \\ &= n(n-1) - \sum_{j=2}^{n} \frac{j}{(\theta^2)^{\frac{1}{p}}} \\ &= n(n-1) - \frac{(n-1)(n+2)}{2} \frac{1}{\theta^{\frac{2}{p}}} \end{aligned}$$

Since the pruning ratio $\gamma(\theta) = \frac{S(\theta)}{T}$,

$$\Rightarrow\ \gamma(\theta) = \frac{n(n-1) - \frac{(n-1)(n+2)}{2} \frac{1}{\theta^{\frac{2}{p}}}}{\frac{n(n-1)}{2}}$$
$$\Rightarrow\ \gamma(\theta) = 2 - \frac{n+2}{n} \frac{1}{\theta^{\frac{2}{p}}}$$

Thus, we can derive two rules as follows:

$$rule\,1:\ \theta \nearrow \Rightarrow\ \frac{n+2}{n} \frac{1}{\theta^{\frac{2}{p}}} \searrow \Rightarrow\ \gamma(\theta) \nearrow$$
$$rule\,2:\ p \nearrow \Rightarrow\ \frac{n+2}{n} \frac{1}{\theta^{\frac{2}{p}}} \searrow \Rightarrow\ \gamma(\theta) \nearrow$$

Therefore, the claim that the pruning ratio of the TAPER algorithm increases with the increase of p and the correlation threshold θ holds.

6 Experimental Results

In this section, we present extensive experiments to evaluate the performance of the TAPER algorithm. Specifically, we demonstrate: (1) a performance comparison between the TAPER algorithm and a brute-force approach, (2) the effectiveness of the proposed algebraic cost model, and (3) the scalability of the TAPER algorithm.

6.1 The Experimental Setup

Our experiments were performed on both real and synthetic data sets. Synthetic data sets were generated such that the rank-support distributions follow Zipf's law, as shown in Figure 7. Note that, in log-log scales, the rank-support plot of a Zipf distribution will be a straight line with a slope equal to the exponent P in the Zipf distribution. A summary of the parameter settings used to generate the synthetic data sets is presented in Table 1, 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.

Table 1. Parameters of synthetic data sets.

Data set name	T	N	C	P
P1.tab	2000000	1000	0.8	1
P2.tab	2000000	1000	0.8	1.25
P3.tab	2000000	1000	0.8	1.5
P4.tab	2000000	1000	0.8	1.75
P5.tab	2000000	1000	0.8	2

The real data sets were obtained from two different application domains, one from journalism and one from the

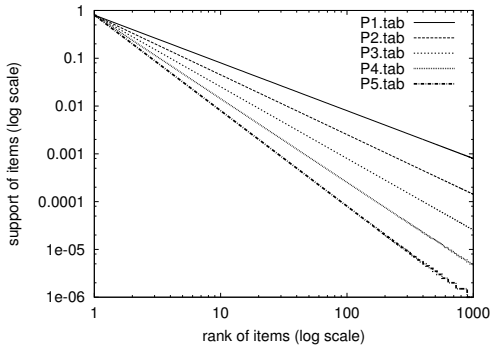


Figure 7. The plot of the Zipf rank-support distributions of synthetic data sets in log-log scale.

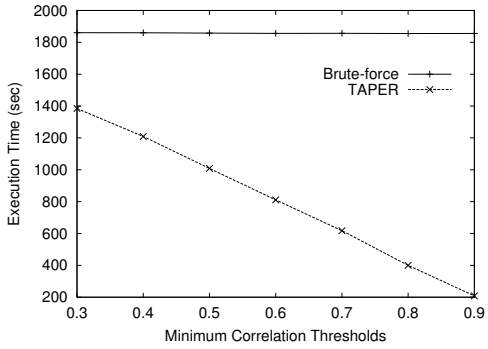


Figure 8. The performance comparison between TAPER and a brute-force approach on the retail data set.

retail industry. The LA1 data set is part of the TREC-5 collection [1] and contains news articles from the Los Angeles Times. This data set has 29704 items and 3204 transactions. In contrast, the real retail data set is a masked data set obtained from a large mail-order company. This data set has 14462 items and 57671 transactions.

The purpose of our experiments was to answer the following questions:

1. How does the relative performance of the TAPER algorithm compare with that of the brute-force approach?

2. How do the computation savings of the TAPER algorithm vary with correlation thresholds?

3. When data sets have a linear rank-support distribution, what is the effect of the slope m on the performance of the TAPER algorithm?

4. When data sets have a generalized Zipf rank-support distribution, what is the effect of the exponent p on the performance of the TAPER algorithm?

5. What is the scalability of the TAPER algorithm with respect to database dimensions?

6. How effective is the algebraic cost model for measuring computation savings?

We implemented the TAPER algorithm using C++ and all experiments were performed on a Sun Ultra 10 workstation with a 440 MHz CPU and 128 Mbytes of memory running the SunOS 5.7 operating system.

[1] The data set is available at http://trec.nist.gov.

6.2 How does the performance of TAPER compare with that of the brute-force approach?

In this subsection, we present a performance comparison between the TAPER algorithm and a brute-force approach using the retail data set. The implementation of the brute-force approach is similar to that of the TAPER algorithm except that the filtering mechanism implemented in TAPER is not included in the brute-force approach.

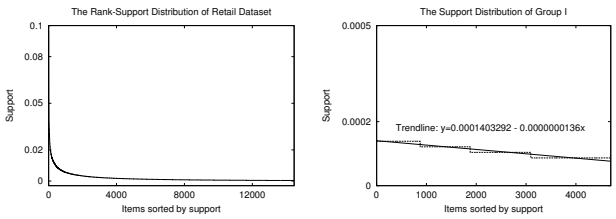
Figure 8 shows the execution time of the two algorithms as the correlation thresholds are increased. As can be seen, the performance of the brute-force approach does not change much. However, the execution time of the TAPER algorithm dramatically decreases with the increase of correlation thresholds. In addition, when the correlation threshold is high, the execution time of TAPER can be one order less than that of the brute-force approach. Finally, even with the correlation threshold as low as 0.3, TAPER still achieves much better performance than the brute-force approach.

6.3 How do the computation savings of TAPER vary with correlation thresholds?

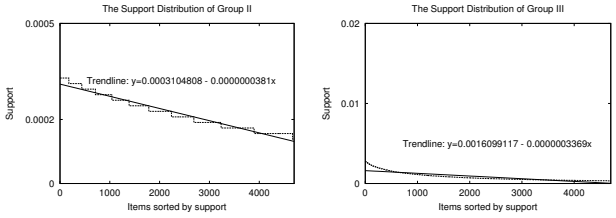
In this section, we present the effect of correlation thresholds on the computation savings of the TAPER algorithm. Recall that our algebraic cost model shows that the pruning ratio of the TAPER algorithm increases with increases of the correlation thresholds for data sets with linear and Zipf-like distributions. Figure 8 shows a decreasing trend of the execution time of the TAPER algorithm on the retail data set as correlation thresholds increase. Although the rank-support distribution of the retail data set does not follow Zipf's law exactly, these experimental results still exhibit a similar trend as the proposed algebraic cost model.

Table 2. Groups of items for the `Retail` data set

Group	I	II	III
# Items	4700	4700	4700
# Transactions	57671	57671	57671
a/m	10318	8149	4778



(a) Retail Dataset (b) Group I



(c) Group II (d) Group III

Figure 9. The plot of the rank-support distributions of the retail data set and its three item groups with a linear regression fitting line (trendline).

6.4 What is the effect of the slope m on the performance of the TAPER algorithm?

Recall that the algebraic cost model for data sets with a linear rank-support distribution provides rules which indicate that the pruning ratio of the TAPER algorithm will increase with the decrease of the ratio a/m and the pruning ratio increases with the increase of the correlation threshold. In this subsection, we empirically evaluate the effect of the ratio a/m on the performance of the TAPER algorithm for data sets with a linear rank-support distrition.

First, we generated three groups of data from the retail data set by sorting all the items in the data set in non-decreasing order and then partition them into four groups. Each of the first three groups contains 4700 items and the last group contains 362 items. The first three groups are

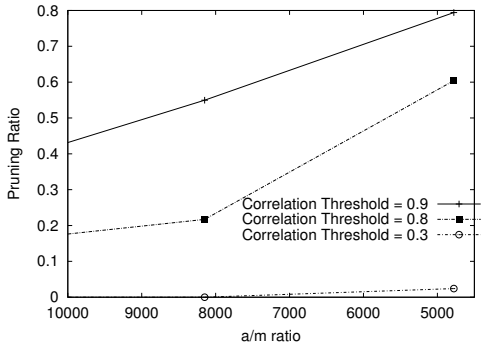


Figure 10. The increase of pruning ratios with the decrease of a/m for data sets with linear rank-support distribution.

the group data sets shown in Table 2. Figure 9 (a) shows the plot of the rank-support distribution of the retail data set and Figure 9 (b), (c), and (d) shows the plots of the rank-support distributions of three groups of data generated from the retail data set. As can be seen, the rank-support distributions of the three groups approximately follow a linear distribution. Table 2 shows some of the characteristics of these data set groups. As can be seen, each group data set has the same number of items and transactions but a different a/m ratio. Group I has the highest a/m ratio and Group III has the lowest a/m ratio. Since the major difference among these three data set groups is the ratio a/m, we can apply these data sets to show the impact of the a/m on the performance of the TAPER algorithm. Figure 10 shows the pruning ratio of the TAPER algorithm on the data set with linear rank-support distributions. As can be seen, the pruning ratio increases as the a/m ratio decreases at different correlation thresholds. The pruning ratio also increases as correlation thresholds are increases. These experimental results confirm the trend exhibited by the cost model.

6.5 What is the effect of the exponent p on the performance of the TAPER algorithm?

In this subsection, for data sets with a generalized Zipf rank-support distribution, we examine the effect of the exponent P on the performance of the TAPER algorithm. We use the synthetic data sets presented in Table 1 for this experiment. All the synthetic data sets in the table 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. Figure 11 shows the pruning ratio of the TAPER algorithm on data sets with different exponent P. As can be seen, the pruning ratios of the TAPER algorithm increase with the increase of the exponent P at different correlation thresholds. Also, we can observe that

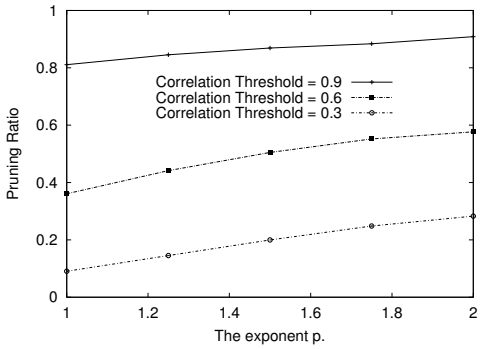


Figure 11. The increase of pruning ratios with the increase of p for data sets with Zipf-like distribution.

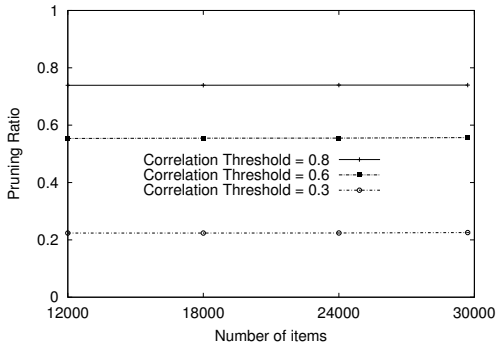


Figure 13. The effect of database dimensions on the pruning ratio for data sets with Zipf-like rank-support distributions.

the pruning ratios of the TAPER algorithm increase with the increase of the correlation thresholds. Recall that the proposed algebraic cost model for data sets with a generalized Zipf distributions provides two rules which confirm the above two observations.

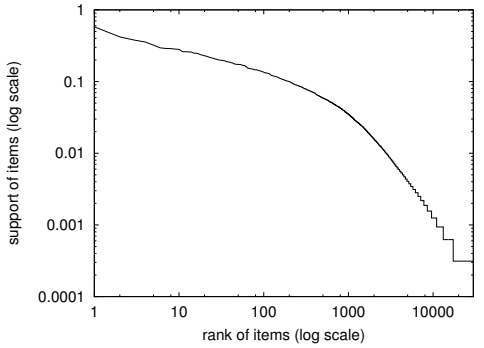


Figure 12. The plot of the rank-support distribution of the LA1 data set in log-log scale. Although this plot does not follow Zipf's law exactly, it does show Zipf-like behavior.

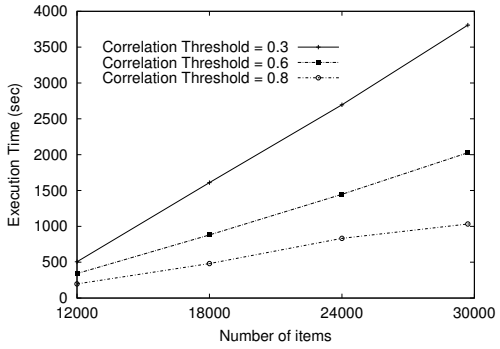


Figure 14. The effect of database dimensions on the execution time for data sets with Zipf-like rank-support distributions.

6.6 What is the scalability of the TAPER algorithm with respect to database dimensions?

In this subsection, we show the scalability of the TAPER algorithm with respect to database dimensions. Figure 12 shows the plot of the rank-support distribution of the LA1 data set in log-log scale. Although this plot does not follow Zipf's law exactly, it does show Zipf-like behavior. In other words, the LA1 data set has an approximate Zipf-like distribution with the exponent $P = 1.406$. In this experiment, we generated three data sets with the number of items equal to 12000, 18000, and 24000 from the LA data set by random sampling on the item set. The above three data sets generated by random sampling can 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 for our scale-up experiments.

For data sets with Zipf-like rank-support distributions, Figure 13 shows the effect of database dimensions on the performance of the TAPER algorithm. As can be seen the pruning ratios of the TAPER algorithm show almost no change or slightly increase at different correlation thresholds. This indicates that the pruning ratios of the TAPER algorithm can be maintained when the number of items is increased. Recall that the proposed algebraic cost model for data sets with a generalized Zipf distribution exhibits a similar trend as the result of this experiment.

In addition, Figure 14 shows the execution time 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 correlation thresholds.

6.7 Summary of the effectiveness of the algebraic cost model for measuring computation savings by TAPER

In this subsection, we summarize the experimental results. First, the algebraic cost model shows that the pruning ratio increases with increases of the correlation thresholds for data sets with linear and zipf-like distributions. As illustrated in Figures 8, 10, 11, and 13, our experiments do show this trend. Second, for data sets with linear rank-support distribution, the algebraic cost model shows that the pruning ratio increases with the decrease of the ratio a/m. Our experimental results also confirm this, as shown in Figure 10. Third, for data sets with Zipf-like rank-support distribution, the algebraic cost model shows that the pruning ratio increases with the exponent p. This is confirmed by our experimental results, as displayed in Figure 11. Finally, the algebraic cost model for linear and zipf rank-support distribution indicates that the pruning ratio is insensitive to the number of items. Our experimental results exhibit a similar trend, as shown in Figure 13.

7 Conclusion and Future Work

In this paper, we proposed using an upper bound of the ϕ correlation coefficient, which shows a conditional monotonic property. Based on this upper bound, we designed an efficient two-step filter-and-refine algorithm, called TAPER, to search all the item pairs with correlations above a user-specified minimum correlation threshold. In addition, we provided an algebraic cost model to quantify the computation savings of the TAPER algorithm. As demonstrated by our experimental results on both real and synthetic data sets, the pruning ratio of the TAPER algorithm can be maintained or even increases with the increase of database dimensions, and the performance of the TAPER algorithm confirms the proposed algebraic cost model.

There are several potential directions for future research. First, we plan to generalize the TAPER algorithm as a standard algorithm for efficient computation of other measures of association. In particular, we will examine the potential upper bound functions of other measures for their monotone property. Second, we propose to extend our methodology to answer correlation-like queries beyond pairs of items.

8 Acknowledgments

We thank Kim Koffolt for her timely and detailed feedback to help improve the readability of this paper.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD*, pages 207–216, May 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB*, 1994.

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

[4] Z. Aung, W. Fu, and K. Tan. An efficient index-based protein structure database searching method. In *International Conference on Database Systems for Advanced Applications*, 2003.

[5] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 265–276, 1997.

[6] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proc. of IEEE Conf. on Data Engineering (ICDE)*, 2001.

[7] P. Cohen, C. J., S. West, and L. Aiken. *Applied Multiple Regression:Correlation Analysis for the Behavioral Science*. Lawrence Erlbaum Assoc; 3rd edition, 2002.

[8] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Intl. Conference on Management of Data*, 2000.

[9] C. Jermaine. The computational complexity of high-dimensional correlation search. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, 2001.

[10] S. K. Kachigan. *Multivariate Statistical Analysis: A conceptual Introduction*. Radius Press, 2nd edition, 1991.

[11] M. Kendall and J. D. Gibbons. *Rank correlation methods*. Oxford University Press (fifth edition)., 1990.

[12] E. L. Lehmann and H. j. M. D'Abrera. *Nonparametrics: Statistical Methods Based on Ranks*. Prentice Hall, 1998.

[13] B. Liu, W. Hsu, and Y. Ma. Mining association rules with multiple minimum supports. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1999.

[14] B. Ooi, H. Pang, H. Wang, L. Wong, and C. Yu. Fast filter-and-refine algorithms for subsequence selection. In *Proc. of the Interational Database Engineering and Application Symposium*, 2002.

[15] J. Patel and D. DeWitt. Partition Based Spatial-Merge Join. In *Proc. of the ACM SIGMOD Conference on Management of Data*, June 1996.

[16] R. Rastogi and K. Shim. Mining optimized association rules with categorical and numeric attributes. *IEEE TKDE*, 14(1), January 2002.

[17] H. T. Reynolds. *The Analysis of Cross-classifications*. The Free Press, New York, 1977.

[18] R. Rymon. Search through systematic set enumeration. In *Proc. of Third Int'l. Conf. on Principles of Knowledge Representation and Reasoning*, pages 539–550, 1992.

[19] H. V. Storch and F. W. Zwiers. *Statistical Analysis in Climate Research*. Cambridge University Press, July 1999.

[20] G. Zipf. *Human Behavior and Principle of Least Effort: An Introudction to Human Ecology*. Addison Wesley, Cambridge, Massachusetts, 1949.