Exploiting A Support-based Upper Bound of Pearson's Correlation Coefficient for Efficiently Identifying Strongly Correlated Pairs

Hui Xiong
Computer Science
University of Minnesota
huix@cs.umn.edu

Shashi Shekhar
Computer Science
University of Minnesota
shekhar@cs.umn.edu

Pang-Ning Tan
Computer Science
Michigan State University
ptan@cse.msu.edu

Vipin Kumar
Computer Science
University of Minnesota
kumar@cs.umn.edu

ABSTRACT

Given a user-specified minimum correlation threshold θ and a market basket database with N items and T transactions, an 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.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms

Keywords

Pearson's Correlation Coefficient, Statistical Computing

1. INTRODUCTION

With the wide spread use of statistical techniques for data analysis, it is expected that many such techniques will be made available in a database environment where users can apply the techniques more flexibly, efficiently, easily, and with minimal mathematical assumptions. Our research is directed towards developing such techniques.

More specifically, this paper examines the problem of computing correlations efficiently from large databases. Correlation analysis plays an important role in many application domains such as market-basket analysis, climate studies, and public health. Our focus, however, is on computing an *all-strong-pairs correlation query* that returns pairs of high positively correlated items (or binary attributes). This problem can be formalized as follows: Given a user-specified minimum correlation threshold θ and a market basket database with N items and T transactions, an all-strong-pairs correlation query finds all item pairs with correlations above the minimum correlation threshold, θ.

However, as the number of items and transactions increases, the computation cost for an all-strong-pairs correlation query becomes prohibitively expensive. For example, consider a database of 10^6 items, which may represent the collection of books available at an e-commerce Web site. Answering the all-strong-pairs correlation query from such a massive database 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 to compute correlations for all half trillion item pairs, particularly when the number of transactions is also large.

Note that the all-strong-pairs correlation query problem is different from the standard association-rule mining problem [1, 3, 5, 9, 14]. Given a set of transactions, the 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 particular subset of items. 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 very 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 because $supp(A, B) = supp(A)supp(B)$. 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 this paper, we focus on the efficient computation of statistical correlation for all pairs of items with high pos-

KDD'04, August 22–25, 2004, Seattle, Washington, USA.

itive correlation. More specifically, we provide an upper bound of Pearson's correlation coefficient for binary variables. The computation of this upper bound is much cheaper than the computation of the exact correlation, since this upper bound can be computed as a function of the support of individual items. 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 1. The x-axis in the figure represents the set of items having a lower level of support than the support for item x_i. These items are sorted from left to right in decreasing order of their individual support values. The y-axis indicates the correlation between each item x and item x_i. $Upperbound(x_i, x)$ represents the upper bound of $correlation(x_i, x)$ and has a monotone decreasing behavior. This behavior guarantees that an item pair (x_i, x_k) can be pruned if there exists an item x_j such that $upperbound(x_i, x_j) < \theta$ and $supp(x_k) < supp(x_j)$.

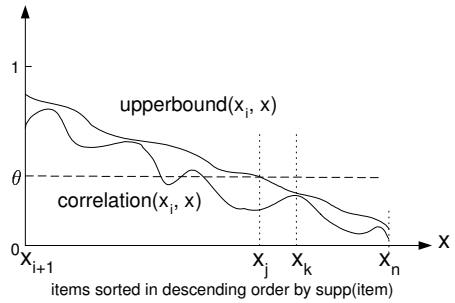


Figure 1: Illustration of the Filtering Techniques. (The curves are only used for illustration purposes.)

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 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 final query results.

In addition, we have proved the completeness and correctness of the TAPER algorithm and provided an algebraic cost model to quantify its computational 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 [18] or linear rank-support distributions.

1.1 Related Work

Related literature can be grouped into two categories. One category has focused on statistical correlation measures.

Jermaine [10] investigated the implication of incorporating chi-square (χ^2) [15] 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, Jermaine presented an iterative procedure for high-dimensional correlation analysis by shaving off part of the database via feedback from human experts [11]. Finally, Brin [3] proposed a χ^2-based correlation rule mining strategy. However, χ^2 does not possess a desired upward closure property for exploiting efficient computation [7].

In this paper, we focus on the efficient computation of statistical correlation for all pairs of items with high positive correlation. Given n items, a traditional 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" [12] 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 market-basket-type data sets. The approach proposed in this paper does not need to compute all $\binom{n}{2}$ pairs. In particular, for market-basket-type data sets with a Zipf-like rank-support distribution, we show that only a small portion of the item pairs needs to be examined. In the real world, Zipf-like distributions have been observed in a variety of application domains, such as retail data and Web click-streams.

Another category of related work is from the association-rule mining framework [1], namely constraint-based association pattern mining [2, 4, 6, 8, 13]. Instead of using statistical correlation measures as the constraints, these approaches use some other measures (constraints), such as support, lift, and the Jaccard measure, for efficiently pruning the pattern search space and identifying interesting patterns.

1.2 Overview and Scope

The remainder of this paper is organized as follows. Section 2 presents 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 restricted to market basket databases with binary variables, and the correlation computational 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

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. 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 [15].

The ϕ correlation coefficient [15] 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 2, the calculation of the ϕ correlation coefficient reduces to

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

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

		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 the two-way table, N is the total number of samples. Furthermore, we can transform Equation 1 as follows.

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

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

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

Hence, when adopting the support measure of association rule mining [1], for two items A and B in a market basket 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.

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

3. PROPERTIES OF ϕ CORRELATION

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

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 the ϕ correlation coefficient for an item pair $\{A, B\}$ can be obtained when $supp(A, B) = supp(B)$ and*

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

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 monotone 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 the ϕ correlation coefficient for an item pair {A, B} can be obtained when supp(A, B) = supp(B). Hence,

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

As can be seen in Equation 3, the upper bound of the ϕ correlation coefficient for an item pair {A, B} relies only 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 already noted, when the number of items N becomes very large, it is 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 to store the support of individual items in the main 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 values 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 $upper(\phi_{\{A,B\}})$ of pair $\{A, B\}$ is monotone decreasing with the decrease of the support value of item B.*

Proof: By Lemma 1, we get:

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

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:

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

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

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. THE TAPER ALGORITHM

In this section, we present the **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.

The Filtering Step: In this 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 1 as previously noted in introduction. In summary, the purpose of the filtering step is to reduce false positive item pairs and further processing cost.

The 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 minimum correlation threshold as the query results.

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.

CoarseFilter(S', θ) //The Filtering Step
1. L = size(S'), P = $\emptyset$
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. $upper(\phi) = \sqrt{\frac{supp(B)}{supp(A)}}\sqrt{\frac{1-supp(A)}{1-supp(B)}}$
7. **if**($upper(\phi) < \theta$) **then**
//Pruning by the monotone property
8. break from inner loop
9. **else**
10. P=P ∪ Refine(A, B, θ)

Refine(A, B, θ) //The Refinement Step
11. Get the support supp(A, B) of item set {A, B}
12. $\phi = \frac{supp(A,B) - supp(A)supp(B)}{\sqrt{supp(A)supp(B)(1-supp(A))(1-supp(B))}}$
13. **if** $\phi < \theta$ **then**
14. **return** $\emptyset$ //return NULL
15. else
16. **return** $\{\{A, B\}, \phi\}$

Figure 3: The TAPER Algorithm

Figure 3 shows the pseudocode of the TAPER algorithm, including the $CoarseFilter$ 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 [16] 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 monotone decreasing with the decrease of the support of item B. Hence, if we find the first target item B which results in an upper bound $upper(\phi_{\{A,B\}})$ that 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 11 gets the support for the item pair {A, B}. Note that the I/O cost can be very expensive for line 11 when the number of items is large since we cannot store the support of all item pairs in the memory. Line 12 calculates the exact correlation coefficient of this item pair. If the correlation is greater than the user-specified minimum correlation threshold, this item pair is returned as a query result in line 16. Otherwise, the procedure returns NULL in line 14.

EXAMPLE 1. *To illustrate the TAPER algorithm, consider a database shown in Figure 4. 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 [16] 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 4 (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.*

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

(a)

Item	Support
1	0.9
2	0.8
3	0.5
4	0.3
5	0.2
6	0.1

(b)

Pair	UPPER (Φ)	Correlation
{1, 2}	0.667	0.667
{1, 3}	0.333	NC
{1, 4}	NC	NC
{1, 5}	NC	NC
{1, 6}	NC	NC
{2, 3}	0.5	-0.5
{2, 4}	0.327	NC
{2, 5}	NC	NC
{2, 6}	NC	NC
{3, 4}	0.655	-0.218
{3, 5}	0.5	0
{3, 6}	0.333	NC
{4, 5}	0.764	0.764
{4, 6}	0.509	0.509
{5, 6}	0.667	0.667

(c)

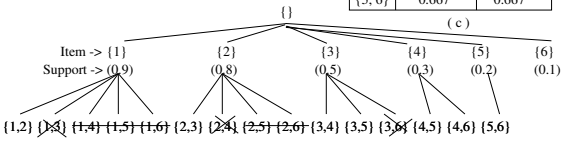


Figure 4: Illustration of the filter-and-refine strategy. NC means there is no computation required.

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: This lemma proof as well as some following lemma proofs are presented in our Technical Report [17]. □

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

5.2 Quantifying the Computation Savings

This section presents analytical results for the amount of computational savings obtained by TAPER. First, we illustrate the relationship between the choices of the minimum correlation threshold and the size of the reduced search space (after performing the filtering step). Knowing the relationship gives us an idea of the amount of pruning achieved using the upper-bound function of correlation.

Figure 5 illustrates a 2-dimensional plot for every possible combination of support pairs, $supp(x)$ and $supp(y)$. If we impose the constraint that $supp(x) \leq supp(y)$, then all item pairs must be projected to the upper left triangle since the diagonal line represents the condition $supp(x) = supp(y)$.

To determine the size of the reduced search space, let us start from the upper bound function of correlation.

$$upper(\phi_{\{x,y\}}) = \sqrt{\frac{supp(x)}{supp(y)}}\sqrt{\frac{1-supp(y)}{1-supp(x)}} < \theta$$

$$\Longrightarrow supp(x)(1-supp(y)) < \theta^2 supp(y)(1-supp(x))$$

$$\Longrightarrow supp(y) > \frac{supp(x)}{\theta^2 + (1-\theta^2)supp(x)} \quad (4)$$

The above inequality provides a lower bound on $supp(y)$ such that any item pair involving x and y can be pruned using the conditional monotone property of the upper bound function. In other words, any surviving item pair that undergoes the refinement step must violate the condition given in Equation 4. These item pairs are indicated by the shaded region shown in Figure 5. During the refinement step, TAPER has to compute the exact correlation for all item pairs that fall in the shaded region between the diagonal and the polyline drawn by Equation 5.

$$supp(y) = \frac{supp(x)}{\theta^2 + (1-\theta^2)supp(x)} \quad (5)$$

As can be seen from Figure 5, the size of the reduced search space depends on the choice of minimum correlation threshold. If we increase the threshold from 0.5 to 0.8, the search space for the refinement step is reduced substantially. When the correlation threshold is 1.0, the polyline from Equation 5 overlaps with the diagonal line. In this limit, the search space for the refinement step becomes zero.

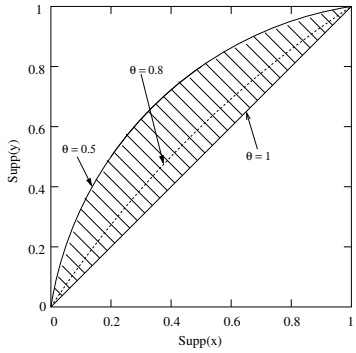


Figure 5: An illustration of the reduced search space for the refinement step of the TAPER algorithm. Only item pairs within the shaded region must be computed for their correlation.

The above analysis shows only the size of the reduced search space that must be explored during the refinement step of the TAPER algorithm. The actual amount of pruning achieved by TAPER depends on the support distribution of items in the database. To facilitate our discussion, we first introduce the definitions of several concepts used in the remainder of this section.

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

$$\gamma(\theta) = \frac{S(\theta)}{T}, \quad (6)$$

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.*

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 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 monotone 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 need to find only 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 a further computation requirement. According to Lemma 1, we get

$$upper(\phi_{\{A_j,A_l\}}) = \sqrt{\frac{supp(A_l)}{supp(A_j)}}\sqrt{\frac{1 - supp(A_j)}{1 - supp(A_l)}} < \sqrt{\frac{supp(A_l)}{supp(A_j)}} = \sqrt{\frac{f(l)}{f(j)}} < \theta$$

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

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

To make the computation simple, we let $l = f^{-1}(\theta^2 f(j)) + 1$. 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 7. Note that the computation savings shown in Equation 7 is an underestimated value of the real computation savings which can be achieved by the TAPER algorithm.

$$S(\theta) = \sum_{j=2}^{n}\{n - f^{-1}(\theta^2 f(j))\} \quad (7)$$

Finally, we conduct computation savings 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 [18], 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.

CASE II: A Linear Distribution.

In this case, the rank-support function has a linear distribution and $f(k) = a - mk$, 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 $f(k) = a - mk$ $(a > 0,\ m > 0)$, 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 $0 < \theta \leq 1$.*

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.

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 $0 < \theta \leq 1$. Furthermore, the pruning ratio is independent when the number of items is increased.*

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 7, we get:

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

Since the pruning ratio $\gamma(\theta) = \frac{S(\theta)}{T}$ and $T = \frac{n(n-1)}{2}$,

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

Thus, we can derive three rules as follows:

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

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

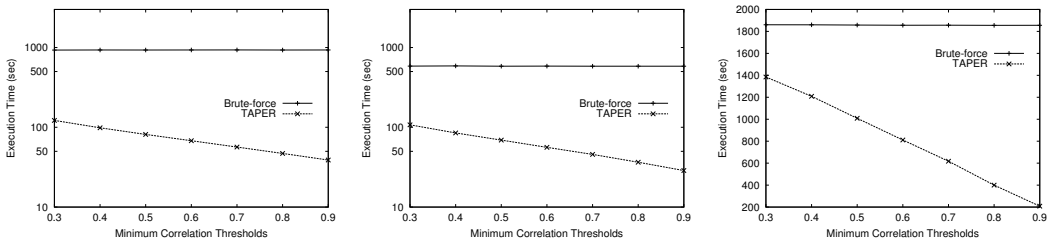
$$rule\,3:\ n \to \infty \Rightarrow \lim_{n\to\infty}\frac{n+2}{n}\frac{1}{\theta^{\frac{2}{p}}} = \frac{1}{\theta^{\frac{2}{p}}}$$

Therefore, the claim that the pruning ratio of the TAPER algorithm increases with the increase of p and the correlation threshold θ holds. Also, rule 3 indicates that the pruning ratio is independent when the number of items is increased in data sets with *Zipf* distributions. □

6. EXPERIMENTAL RESULTS

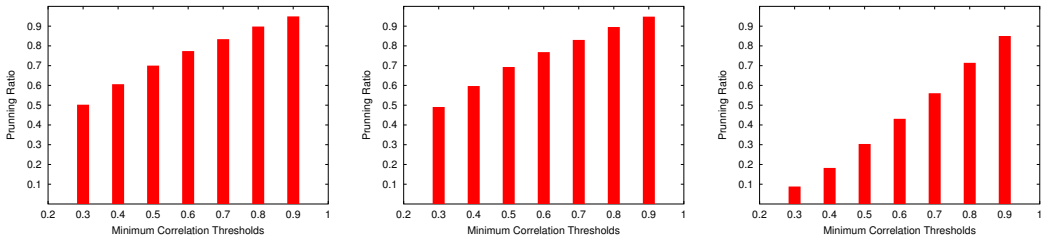
In this section, we present the results of extensive experiments to evaluate the performance of the TAPER algorithm. Specifically, we demonstrate: (1) a performance comparison between the TAPER algorithm and a brute-force approach, (2) the effectiveness of the proposed algebraic cost model, and (3) the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

Experimental Data Sets: 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 6. 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



(a) *Pumsb* (b) *Pumsb** (c) *Retail*

Figure 7: TAPER vs. a brute-force approach on the *Pumsb*, *Pumsb, and retail data sets.**



(a) *Pumsb* (b) *Pumsb** (c) *Retail*

Figure 8: The pruning effect of TAPER on *Pumsb*, *Pumsb, and retail data sets.**

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

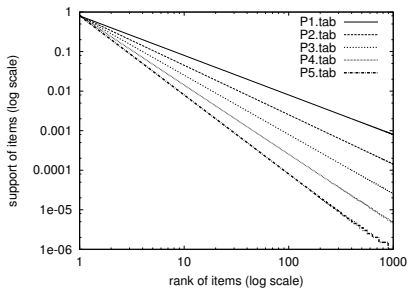


Figure 6: The plot of the Zipf rank-support distributions of synthetic data sets in log-log scale.

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.

The real data sets were obtained from several different application domains. Table 2 shows some characteristics of these data sets. The first five data sets in the table, i.e., `pumsb`, `pumsb*`, `chess`, `mushroom`, and `connect` are often used as benchmark for evaluating the performance of association rule algorithms on dense data sets. The `pumsb`

Table 2: Real Data Set Characteristics.

Data set	#Item	#Record	Source
Pumsb	2113	49046	IBM Almaden
Pumsb*	2089	49046	IBM Almaden
Chess	75	3196	UCI Repository
Mushroom	119	8124	UCI Repository
Connect	127	67557	UCI Repository
LA1	29704	3204	TREC-5
Retail	14462	57671	Retail Store

and `pumsb*` data sets correspond to binarized versions of a census data set from IBM[1]. The difference between them is that `pumsb*` does not contain items with support greater than 80%. The `chess`, `mushroom`, and `connect` data sets are benchmark data sets from UCI machine learning repository [2]. 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.

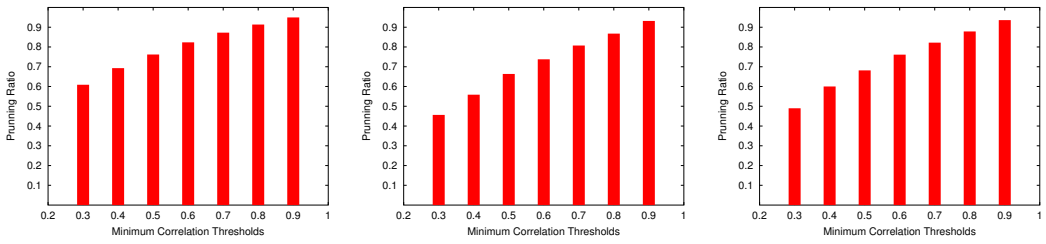
Experimental Platform: We implemented TAPER using C++ and all experiments were performed on a Sun Ultra 10 workstation with a 440 MHz CPU and 128 Mbytes of memory running the SunOS 5.7 operating system.

6.1 TAPER vs. the Brute-force Approach.

In this subsection, we present a performance comparison between the TAPER algorithm and a brute-force approach using several benchmark data sets from IBM, a UCI machine learning repository, and some other sources, such as retail stores. The implementation of the brute-force approach is

[1]These data sets are obtained from IBM Almaden at http://www.almaden.ibm.com/cs/quest/demos.html.

[2]These data sets and data content descriptions are available at http://www.ics.uci.edu/~mlearn/MLRepository.html



(a) *Connect* (b) *Mushroom* (c) *Chess*

Figure 9: The pruning effect of TAPER on UCI *Connect*, *Mushroom*, *Chess* data sets.

similar to that of the TAPER algorithm except that the filtering mechanism implemented in the TAPER algorithm is not included in the brute-force approach.

Figure 7 shows the relative computation performance of the TAPER algorithm and the brute-force approach on the `pumsb`, `pumsb*`, and `retail` data sets. As can be seen, the performance of the brute-force approach does not change much for any of the three data sets. However, the execution time of the TAPER algorithm can be an order of magnitude faster than the brute-force approach even if the minimum correlation threshold is low. For instance, as shown in Figure 7 (a), the execution time of TAPER on the `pumsb` data set is one order of magnitude less than that of the brute-force approach at the correlation threshold 0.4. Also, when the minimum correlation threshold increases, the execution time of TAPER dramatically decreases on the `pumsb` data set. Similar computation effects can also be observed on the `pumsb*` and `retail` data sets although the computation savings on the `retail` data set is not as significant as it is on the other two data sets.

To better understand the above computation effects, we also present the pruning ratio of the TAPER algorithm on these data sets in Figure 8. As can be seen, the pruning ratio of TAPER on the `retail` data set is much smaller than that on the `pumsb` and `pumsb*` data sets. This smaller pruning ratio explains why the computation savings on `retail` is less than that on the other two data sets. Also, Figure 9 shows the pruning ratio of TAPER on UCI `connect`, `mushroom`, and `chess` data sets. The pruning ratio achieved on these data sets are comparable with the pruning ratio we obtained on the `pumsb` data set. This indicates that TAPER also achieves much better computation performance than the brute-force approach on UCI benchmark data sets.

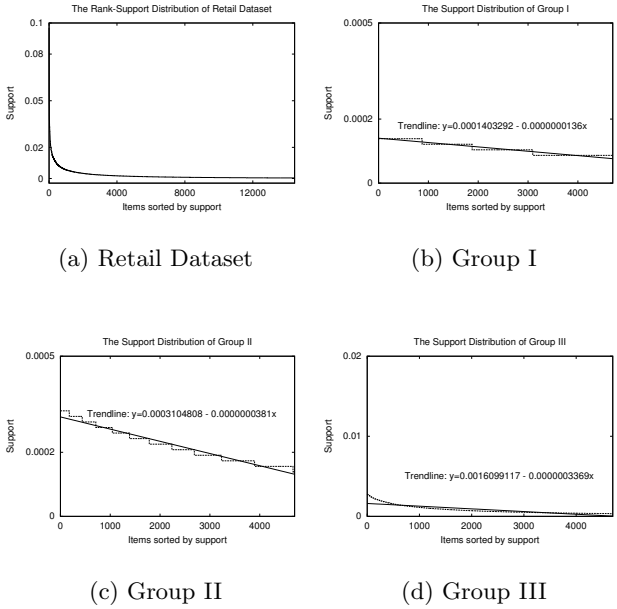
6.2 The Effect of Correlation Thresholds

In this subsection, 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 such an increasing trend of the pruning ratio on the `pumsb`, `pumsb*`, and `retail` data sets as correlation thresholds increase. Also, Figure 9 shows a similar increasing trend of the pruning ratio on the UCI benchmark datasets including `mushroom`, `chess`, and `connect`.

One common feature of all the above data sets is the skewed nature of their rank-support distributions. As a result, these experimental results still exhibit a similar trend as the proposed algebraic cost model although the rank-support distributions of these datasets do not follow Zipf's law exactly.

Table 3: 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 10: The plot of the rank-support distributions of the retail data set and its three item groups with a linear regression fitting line (trendline).

6.3 The Effect of the Slope m

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 increases 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

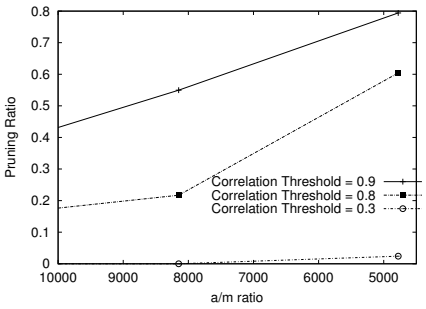


Figure 11: Pruning ratios with the decrease of a/m for data sets with linear rank-support distribution.

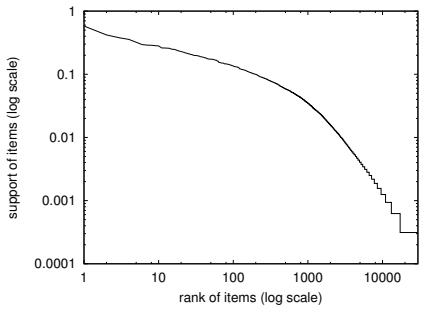


Figure 13: The plot of the rank-support distribution of the LA1 data set in log-log scale.

ratio a/m on the performance of the TAPER algorithm for data sets with a linear rank-support distribution.

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 partitioning 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 the group data sets shown in Table 3. Figure 10 (a) shows the plot of the rank-support distribution of the retail data set and Figure 10 (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 3 lists some of the characteristics of these data set groups. Each group 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 11 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 increased. These experimental results confirm the trend exhibited by the cost model.

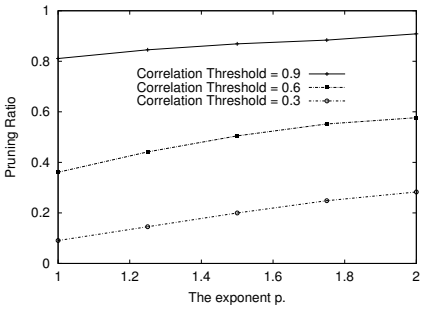


Figure 12: The increase of pruning ratios with the increase of p for data sets with Zipf-like distribution.

6.4 The Effect of the Exponent p

In this subsection, we examine the effect of the exponent P on the performance of the TAPER algorithm for data sets with a generalized Zipf rank-support distribution. We used 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 12 displays the pruning ratio of the TAPER algorithm on data sets with different exponent P. Again, the pruning ratios of the TAPER algorithm increase with the increase of the exponent P at different correlation thresholds. Also, we can observe that 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.

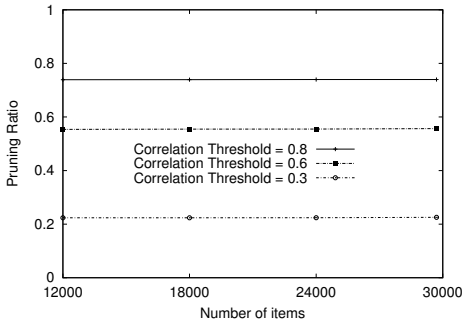


Figure 14: The effect of database dimensions on the pruning ratio for data sets with Zipf-like rank-support distributions.

6.5 The Scalability of TAPER

In this subsection, we show the scalability of the TAPER algorithm with respect to database dimensions. Figure 13 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 12000, 18000, and 24000 items respectively, from the LA1 data set by random sampling on the item set. Due to the random sampling, the three data sets can have almost the same rank-support distributions as the LA1 data set. As a result, we used these three

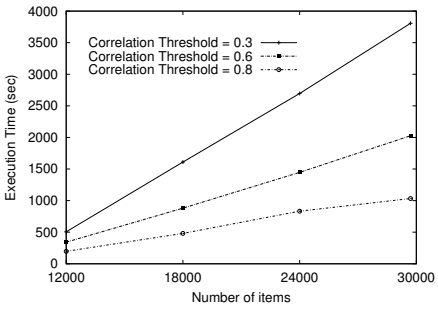


Figure 15: The effect of database dimensions on the execution time for data sets with Zipf-like rank-support distributions.

generated data sets and the LA1 data set for our scale-up experiments.

For data sets with Zipf-like rank-support distributions, Figure 14 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.

Finally, in Figure 15, we show that the execution time for our scale-up experiments increases linearly with the increase of the number of items at several different minimum correlation thresholds.

7. CONCLUSIONS 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 TAPER. As demonstrated by our experimental results on both real and synthetic data sets, the pruning ratio of TAPER can be maintained or even increases with the increase of database dimensions, and the performance of TAPER 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. Finally, we will extend the TAPER algorithm to find all pairs of high negatively correlated items.

8. ACKNOWLEDGMENTS

This work was partially supported by NASA grant # NCC 2 1231, DOE/LLNL W-7045-ENG-48, and by Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

9. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD*, 1993.

[2] R. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery Journal*, pages 217–240, 2000.

[3] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *ACM SIGMOD*, 1997.

[4] C. Bucila, J. Gehrke, D. Kifer, and W. M. White. Dualminer: a dual-pruning algorithm for itemsets with constraints. In *ACM SIGKDD*, 2002.

[5] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *ICDE*, 2001.

[6] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding interesting associations without support pruning. In *ICDE*, 2000.

[7] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In *ACM SIGKDD*, 2001.

[8] G. Grahne, L. V. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. In *ICDE*, 2000.

[9] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD*, 2000.

[10] C. Jermaine. The computational complexity of high-dimensional correlation search. In *ICDM*, 2001.

[11] C. Jermaine. Playing hide-and-seek with correlations. In *ACM SIGKDD*, 2003.

[12] S. K. Kachigan. *Multivariate Statistical Analysis: A Conceptual Introduction*. Radius Press, 1991.

[13] R. Ng, L. Lakshmanan, J. Han, and A. Pang. Exploratory mining via constrained frequent set queries. In *ACM SIGMOD*, 1999.

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

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

[16] R. Rymon. Search through systematic set enumeration. In *Int'l. Conf. on Principles of Knowledge Representation and Reasoning*, 1992.

[17] H. Xiong, , S. Shekhar, P. Tan, and V. Kumar. Taper: An efficient two-step approach for all-pairs correlation query in transaction databases. In *Technical Report 03-020, computer science and engineering, University of Minnesota - Twin Cities*, May 2003.

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