TAPER: A Two-Step Approach for All-strong-pairs Correlation Query in Large Databases

Hui Xiong, *Student Member, IEEE,* Shashi Shekhar, *Fellow, IEEE,* Pang-Ning Tan, *Member, IEEE,* and Vipin Kumar, *Fellow, IEEE*

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. The goal of this paper is to provide computationally efficient algorithms to answer the all-strong-pairs correlation query. Indeed, 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 special 2D-monotone properties 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-world data sets exhibit similar trends and show that the TAPER algorithm can be an order of magnitude faster than brute-force alternatives. Finally, we demonstrate that the algorithmic ideas developed in the TAPER algorithm can be extended to efficiently compute negative correlation and uncentered Pearson's correlation coefficient.

Hui Xiong, Shashi Shekhar, and Vipin Kumar are with the Department of Computer Science and Engineering, University of Minnesota, 200 Union Steet SE, Minneapolis, MN 55455, USA. E-mail: {huix, shekhar, kumar}@cs.umn.edu.

Pang-Ning Tan is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824-1226, USA. E-mail: ptan@cse.msu.edu.

Corresponding Author: Hui Xiong. Phone: 612-626-8084. Fax: 612-626-1596. Email: huix@cs.umn.edu.

I. Introduction

Given a large set of items and observation data sets about co-occurring items, association analysis is concerned with identification of strongly related (e.g. as measured by Pearson's correlation coefficient [22]) subsets of items. Association analysis is a core problem in data mining and databases. It plays an important role in many application domains such as market-basket analysis [2], climate studies [25], public health [8], and bioinformatics [17]. For instance, association analysis in market basket study can reveal how the sales of a product are related with the sales of other products. This type of information can be useful for sales promotions, catalog design, and store layout.

The focus of this paper is on computing an *all-strong-pairs correlation query* that returns pairs of high positively correlated items (or binary attributes). The *all-strong-pairs correlation query* 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 in the data set 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 pairs, particularly when the number of transactions in the data set is also very large.

A. Related Work

Jermaine [13] investigated the implication of incorporating chi-square (χ^2) [22] 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. Jermaine [14] also presented an iterative procedure for high-dimensional correlation analysis by shaving off part of the database via feedback from human experts. Finally, Brin [4] proposed a χ^2-based correlation rule mining strategy. However, χ^2 does not possess a desired upward closure property for exploiting efficient computation [9].

This paper focuses on the efficient computation of statistical correlation for all pairs of items. 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 a statistical software package as the "correlation matrix" [15] 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. Recently, Ilyas et al. [12] proposed a more efficient method for identifying correlated pairs. In this method, the sampling techniques are applied to exploit efficient computation. As a result, this method cannot avoid false-positive and false-negative correlations.

In contrast, unlike the correlation matrix approach, the method 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, including commercial retail data, Web click-streams, and telecommunication data. Also, we show that our method is complete and correct, since we do not apply any approximation schemes, such as sampling techniques.

B. Contributions

In our preliminary work [28], 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 1-D 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 $supp(x_k) < supp(x_j)$.

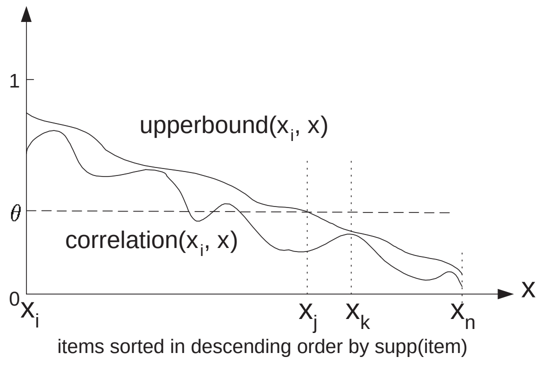


Fig. 1. Illustration of the Filtering Techniques. (The curves are only used for illustration purposes.)

A **T**wo-step **A**ll-strong-**P**airs corr**E**lation que**R**y (TAPER) algorithm is proposed to exploit this 1-D monotone property 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 TAPER and provided an algebraic cost model to quantify the 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 [29] or linear rank-support distributions.

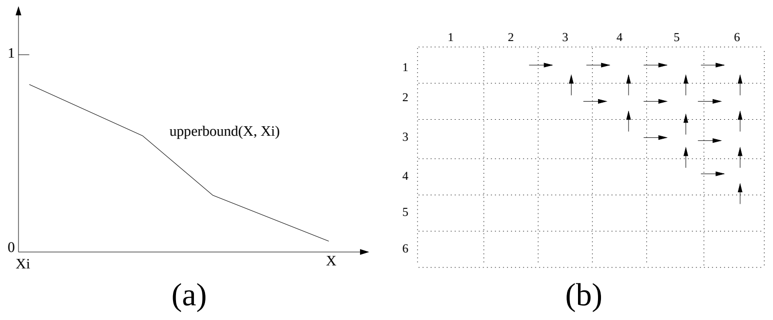


Fig. 2. (a) A Monotone Property of the Upper Bound (b) Illustration of 2-D Monotone Properties of the Upper Bound

In this paper, we identify that the upper bound of Pearson's correlation coefficient for binary variables has special 2-D monotone properties. Indeed, besides the 1-D monotone property mentioned above, the upper bound has another 1-D monotone property as shown in Figure 2 (a). The x-axis in the figure represents a sorted item list in ascending order of their individual support values. The y-axis indicates the correlation between item x_i and x. In this case, the item x_i is fixed and the upper bound of $correlation(x, x_i)$ is monotonically decreasing with the increase of the support value of item x.

Figure 2 shows 2-D monotone properties of the upper bound. In the figure, for an item list $\{$1, 2, 3, 4, 5, 6$\}$, which is sorted by item support in non-increasing order, the upper bound of item pairs is decreasing following the arrow direction. For instance, the upper bound of item pair $\{$5, 6$\}$ is greater than that of item pair $\{$4, 5$\}$. Also, the upper bound of item pair $\{$1, 2$\}$ is greater than that of item pair $\{$1, 3$\}$.

With 2-D monotone properties of the upper bound, we can further refine the TAPER algorithm by reducing the upper-bound computation in the coarse filter step. We show that the number of upper-bound computations is reduced from $\frac{n(n-1)}{2}$ to $2n-3$ for the worst case. In addition, we present experimental results to show this computational improvement.

Finally, the method proposed in this paper is not limited to finding all pairs of high positively correlated pairs. We show that the algorithmic ideas developed in the TAPER algorithm can also be extended for identifying pairs of high negatively correlated pairs and for efficiently computing uncentered Pearson's correlation coefficient. To this end, we provide the theoretical basis, algorithmic ideas, and experimental results for such an extension.

C. Scope and Outline

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.

Note that the all-strong-pairs correlation query problem is different from the standard association-rule mining problem [1], [3], [4], [5], [6], [7], [10], [11], [19], [20], [21], [26]. 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.

The remainder of this paper is organized as follows. Section II presents basic concepts. In section III, we introduce the upper bound of Pearson's correlation coefficient for binary variables. Section IV proposes the TAPER algorithm. In section V, we analyze the TAPER algorithm in the areas of completeness, correctness, and computation gain. Section VI discusses how to generalize our method. In section VII, we present the experimental results. Finally, we draw conclusions and suggest future work in section VIII.

II. Pearson's Correlation Coefficient

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 [16] and Spearman's Rank Correlation Coefficient [18]. In contrast, relationships among nominal variables can be analyzed with nominal measures of association such as Pearson's Correlation Coefficient and measures based on Chi Square [22].

The ϕ correlation coefficient [22] 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 $\times$ 2 two-way table shown in Table I, the calculation of the ϕ correlation coefficient reduces to

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

		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

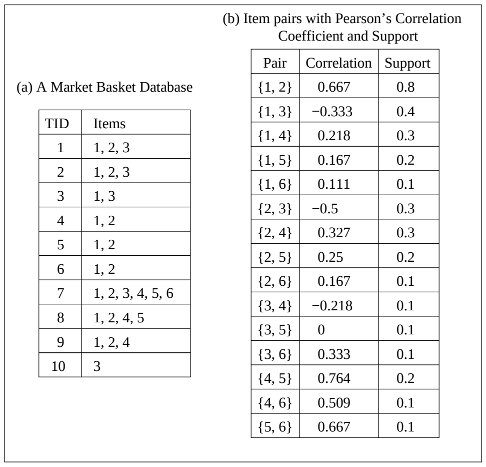
TABLE I
A TWO-WAY TABLE OF ITEM A AND ITEM B.

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, $P_{(i+)} = \sum_{j=0}^{1} P_{(ij)}\ \ and\ \ P_{(+j)} = \sum_{i=0}^{1} P_{(ij)}$. In the two-way table, given that N is the total number of samples, we can transform Equation 1 as follows.

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

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.

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


(a) A Market Basket 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

(b) Item pairs with Pearson's Correlation Coefficient and Support

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

Fig. 3. An Example Database.

Example 1: Consider the market basket database shown in Figure 3. For item 1 and item 2 in the database, we can construct a two-way table as shown in Table II. Then, by Equation 1, we get $\phi =$

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

TABLE II
A TWO-WAY TABLE OF ITEM 1 AND ITEM 2.

$\frac{1\times 8-1\times 0}{\sqrt{1\times 9\times 2\times 8}}=\frac{2}{3}$. Also, since $supp(1,2)=0.8$, $supp(1)=0.9$, and $supp(2)=0.8$, by Equation 2, we get $\phi=\frac{0.8-0.8\times 0.9}{\sqrt{0.8\times 0.9\times 0.1\times 0.2}}=\frac{0.08}{0.12}=\frac{2}{3}$. As can be seen, the results from the two equations are identical. Finally, ϕ correlation coefficients for other item pairs can be computed similarly. Figure 3 (b) shows ϕ correlation coefficients for all item pairs.

III. 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 queries.

A. An Upper Bound

In this subsection, we propose an upper bound of the ϕ correlation coefficient for a given pair $\{A,\ B\}$ in terms of the support value of item A and the support value of item B.

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 $\{A,\ B\}$ can be obtained when $supp(A,B)=supp(B)$ and

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

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

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.

Pair	UPPER (Φ)	Correlation
{1, 2}	0.667	0.667
{1, 3}	0.333	−0.333
{1, 4}	0.218	0.218
{1, 5}	0.167	0.167
{1, 6}	0.111	0.111
{2, 3}	0.5	−0.5
{2, 4}	0.327	0.327
{2, 5}	0.25	0.25
{2, 6}	0.167	0.167
{3, 4}	0.655	−0.218
{3, 5}	0.5	0
{3, 6}	0.333	0.333
{4, 5}	0.764	0.764
{4, 6}	0.509	0.509
{5, 6}	0.667	0.667

Fig. 4. All Pairs in the Example Database.

Example 2: Figure 4 shows all item pairs with their upper bound values for the example data set shown in Figure 3. If we consider item pair $\{1,\, 2\}$, then by Equation 3, we have $upper(\phi_{\{1,2\}}) = \sqrt{\frac{supp(2)}{supp(1)}}\sqrt{\frac{1-supp(1)}{1-supp(2)}} = \sqrt{\frac{0.8}{0.9}}\sqrt{\frac{1-0.9}{1-0.8}} = 0.667$ as shown in Figure 4. Finally, upper bounds of the ϕ correlation coefficient for other item pairs can be computed in a similar manner. Figure 4 shows the upper bounds and ϕ correlation coefficients for all item pairs.

B. 2-D Monotone Properties

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 an item pair $\{A,\, B\}$, if we let $supp(A) > supp(B)$ and fix item A, the upper bound $upper(\phi_{\{A,B\}})$ of $\{A,\, B\}$ 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. $\frac{upper(\phi_{\{A,B_1\}})}{upper(\phi_{\{A,B_2\}})} = \sqrt{\frac{supp(B_1)}{supp(B_2)}}\sqrt{\frac{1-supp(B_2)}{1-supp(B_1)}} > 1$. This follows the given condition that $supp(B_1) > supp(B_2)$ and $(1 - supp(B_1)) < (1 - supp(B_2))$.

Along the same line of Lemma 2, we can also derive another conditional monotone property of the upper bound of the ϕ correlation coefficient as follows.

Lemma 3: For a pair of items $\{A,\, B\}$, let $supp(A) > supp(B)$ and fix the item B, the upper bound $upper(\phi_{\{A,B\}})$ of $\{A,\, B\}$ is monotone increasing with the decreasing of the support value of item A.

Example 3: This example illustrates Lemma 2 and Lemma 3. As shown in Figure 4, if item 1 is fixed, the upper bounds of item pairs {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6} are in a decreasing order, which follows Lemma 2. Also, the upper bounds of item pairs {1, 6}, {2, 6}, {3, 6}, {4, 6}, {5, 6} are in an increasing order. This follows Lemma 3.

Lemma 2 and Lemma 3 are 2-D monotone properties of the upper bound. These two lemmas allow us to push the upper bound into the search algorithm, thus efficiently pruning the search space. In the following section, we will introduce how this pruning works.

IV. ALGORITHM DESCRIPTIONS

Here, we first present the **T**wo-step **A**ll-strong-**P**airs corr**E**lation que**R**y (TAPER) algorithm to exploit the proposed upper bound and its monotone properties.

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

CoarseFilter$(S', \theta)$ //The filtering Step
            1DFiltering$(S', \theta)$ or 2DFiltering$(S', \theta)$
Refine(A, B, $\theta$) //The Refinement Step
 1.         Get the support supp(A, B) of item set {A, B}
 2.         $\phi = \frac{supp(A,B) - supp(A)supp(B)}{\sqrt{supp(A)supp(B)(1-supp(A))(1-supp(B))}}$
 3.         if $\phi < \theta$ then
 4.             return $\emptyset$ //return NULL
 5.         else
 6.             return $\{\{A, B\}, \phi\}$
```

Fig. 5. The TAPER Algorithm

A. Overview

The TAPER algorithm is a 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 special monotone properties of the upper bound of the ϕ correlation coefficient.

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.

Figure 5 shows the pseudocode of the TAPER algorithm, which includes the $CoarseFilter$ and $Refine$ procedures. In the $CoarseFilter$ procedure, monotone properties of the upper bound of the ϕ correlation coefficient can be used in two different ways, namely 1D filtering and 2D filtering. The details of these two filtering techniques are introduced in the following subsections.

Procedure $Refine$ works as follows. Line 1 gets the support for the item pair $\{$A, B$\}$. Note that the I/O cost can be very expensive for line 1 when the number of items is large since we cannot store the support of all item pairs in the memory. Line 2 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 6. Otherwise, the procedure returns NULL in line 4.

B. *TAPER 1D: TAPER with 1D filter*

In this subsection, we illustrate TAPER with 1D filter, denoted as TAPER 1D. The working mechanism of 1D filtering is illustrated as the following Corollary.

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.

```
1DFiltering($S', \theta$) //The Filtering Step
1.    n = size($S'$), P = $\emptyset$
2.    for i from 1 to n-1
3.        A = $S'[i]$
4.        for j from i+1 to n
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 $\cup$ Refine(A, B, $\theta$)
```

Fig. 6. TAPER 1D: TAPER with a 1D filter

Figure 6 shows the pseudocode of 1D filtering, which works as follows. Line 1 initializes the variables and creates an empty query result set P. Lines 2 - 10 iteratively enumerate candidate pairs and filter out item pairs whose correlations are obviously less than the user-specified correlation threshold θ. This is implemented by two loops. Line 2 starts an outer loop. 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 loop 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. 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.

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)	Computed	Correlation	Computed
{1, 2}	0.667	Yes	0.667	Yes
{1, 3}	0.333	Yes	−0.333	No
{1, 4}	0.218	No	0.218	No
{1, 5}	0.167	No	0.167	No
{1, 6}	0.111	No	0.111	No
{2, 3}	0.5	Yes	−0.5	Yes
{2, 4}	0.327	Yes	0.327	No
{2, 5}	0.25	No	0.25	No
{2, 6}	0.167	No	0.167	No
{3, 4}	0.655	Yes	−0.218	Yes
{3, 5}	0.5	Yes	0	Yes
{3, 6}	0.333	Yes	0.333	No
{4, 5}	0.764	Yes	0.764	Yes
{4, 6}	0.509	Yes	0.509	Yes
{5, 6}	0.667	Yes	0.667	Yes

(c)

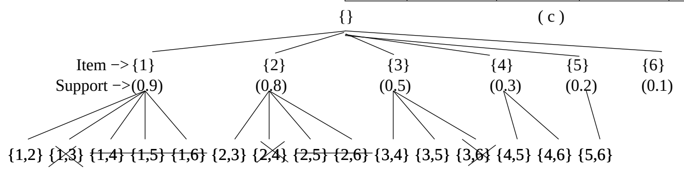


Fig. 7. Illustration of the filter-and-refine strategy of the TAPER algorithm with 1D filter.

Example 4: To illustrate the TAPER algorithm with 1D filter, consider a database shown in Figure 7. 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 [24] to demonstrate how 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 7 (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. Please note that the Rymon's set-enumeration tree is used for illustration purposes. In our algorithm, there is no requirement to construct such a tree structure.

C. TAPER 2D: TAPER with 2D Filter

The coarse filter step of TAPER 1D can be improved by reducing the number of upper bounds to be computed; this leads to TAPER 2D. Indeed, we can reduce the computation of upper bounds in each inner loop and produce an improved coarse filter step as shown in Figure 8. The key difference between TAPER 1D and TAPER 2D is that TAPER 2D records the break point in the last inner loop and starts computing upper bounds from the recorded point instead of going through every candidate pair. The correctness of this additional filtering is guaranteed by the following Corollary.

```
**2DFiltering**$(S', \theta)$ //The Filtering Step
1.      n = size($S'$), P = $\emptyset$
2.      **startposi = 2**
3.      for i from 1 to n-1
4.          A = $S'[i]$
5.          for j from i+1 to n
6.              flag=0
7.              B = $S'[j]$
8.              if(j $\geq$ startposi)
9.                  $upper(\phi) = \sqrt{\frac{supp(B)}{supp(A)}}\sqrt{\frac{1-supp(A)}{1-supp(B)}}$
10.                 **if**($upper(\phi) < \theta$) **then** //Pruning by the monotone property
11.                     **if** (j $>$ i+1 || startposi == n) **then** //Reducing the Computation of Upper Bounds
12.                         startposi = j
13.                     **else**
14.                         startposi = j + 1
15.                     break from inner loop
16.             P=P $\cup$ Refine(A, B, $\theta$)
17.         if(startposi == (i+1) && flag ==0) startposi ++
```

Fig. 8. TAPER 2D: TAPER with a 2D filter

Corollary 2: Given an item list $\{i_1, i_2, \ldots, i_n\}$, which is sorted by item supports in non-increasing order, the upper-bound computation of an item pair $\{i_a, i_c\}$ with $supp(\{i_a\}) > supp(\{i_c\})$ can be saved if $upper(\phi\{i_{a-1}, i_c\}) > \theta$, where $supp(\{i_{a-1}\}) > supp(\{i_c\})$.

Proof: First, we have $supp(\{i_{a-1}\}) > supp(\{i_a\})$, since items are sorted by support in non-increasing order. Also, by Lemma 3, we have $upper(\phi\{i_{a-1}, i_c\}) < upper(\phi\{i_a, i_c\})$. Therefore, $upper(\phi\{i_a, i_c\}) > \theta$. As a result, we do not need to compute the upper bound of $\{i_a, i_c\}$.

TAPER 2D is an improvement over TAPER 1D. The following Lemma 4 shows that the number of upper bound computations is reduced from $\frac{n(n-1)}{2}$ in TAPER 1D to $2n$-3 in TAPER 2D for the worst case.

Lemma 4: The number of upper bounds required to compute in the coarse filter step of TAPER 2D is $2n - 3$, where n is the number of objects in the data set.

Proof: Figure 8 shows that the number of upper bounds to be computed is determined by the value k. For each outer loop, there is an inner loop where the upper bound is computed. k starts from 2 and ends at n. In each outloop i, there are at most $k_i - k_{i-1} + 1$ number of upper bound computations. In total, there are $\sum_{i=1}^{n-1}(k_i - k_{i-1} + 1) = 2n - 3$ number of upper bounds to be computed.

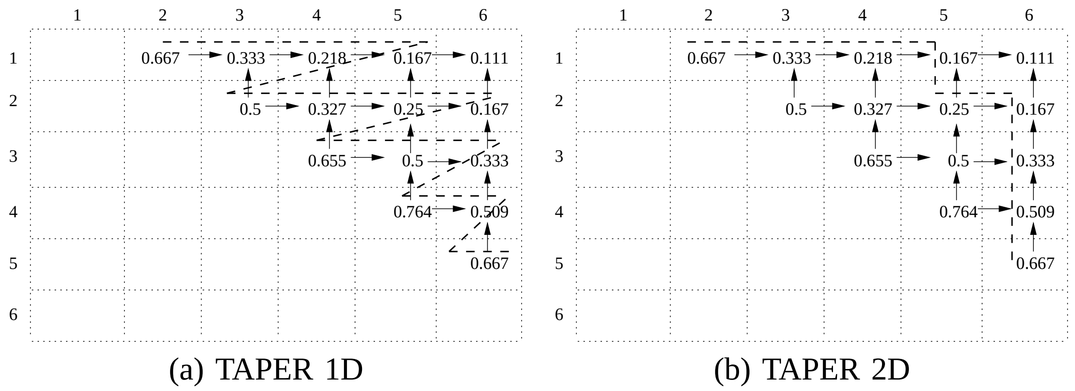


Fig. 9. TAPER 1D vs. TAPER 2D.

Example 5: This example illustrates the coarse filter steps of the TAPER algorithm with 1D and 2D filtering. Here, we use the data set shown in Figure 7. For the item list {1, 2, 3, 4, 5, 6}, which is sorted by item support in non-increasing order, if the correlation threshold is 0.2, the coarse filter step of TAPER 1D is illustrated in Figure 9 (a). In the figure, the dot line indicates the item pairs whose upper bounds need to be computed. As can be seen, there are 14 upper-bound computations. In contrast, as shown in Figure 9 (b), TAPER 2D computes upper bounds for 2n-3 = $2 \times 6 - 3$ = 9 number of item pairs, which are $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$, $\{1, 5\}$, $\{2, 5\}$, $\{2, 6\}$, $\{3, 6\}$, $\{4, 6\}$, and $\{5, 6\}$.

V. Analysis of the TAPER algorithm

In this section, we analyze TAPER in the areas of completeness, correctness, and computation savings. Note that TAPER here will stand for both TAPER 1D and TAPER 2D to simplify the discussion.

A. Completeness and Correctness

Lemma 5: 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 all item pairs in the database have the opportunity to be checked during the iteration 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 as well as Corollary 2. Also, the refinement step only prunes item pairs whose correlations are less than the user-specified correlation threshold. The second fact guarantees that all the pruned item pairs cannot have correlations above the user-specified minimum correlation threshold.

Lemma 6: The TAPER algorithm is correct. In other words, every pair that both algorithms find has a correlation above a user-specified minimum correlation threshold.

Proof: The correctness of the TAPER algorithm can be guaranteed by the refinement step of these two algorithms, since the exact correlation of each candidate pair is calculated in the refinement step and every pair with a correlation lower than the user-specified correlation threshold will be pruned.

B. 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 10 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 of the correlation. $upper(\phi_{\{x,y\}}) = \sqrt{\frac{supp(x)}{supp(y)}}\sqrt{\frac{1-supp(y)}{1-supp(x)}} < \theta \implies supp(x)(1 - supp(y)) < \theta^2 supp(y)(1 - supp(x))$.

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

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 10. 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.

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

As can be seen from Figure 10, 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.

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.

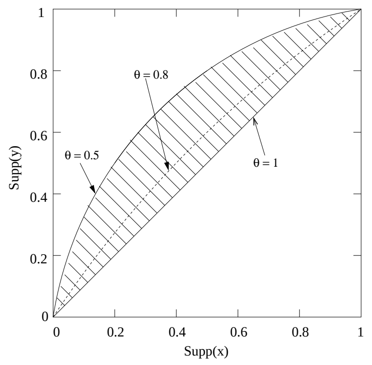


Fig. 10. An illustration of the reduced search space for the refinement step of the TAPER algorithm. Only item pairs within the shaded region must be computed for their correlation.

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

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

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 presents 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

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

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 achieved computation savings.

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

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 [29], 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 7: 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 $0 < \theta \leq 1$.

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 $f(k) = a - mk$, 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 $f^{-1}(y) = \frac{a-y}{m}$. Thus, $f^{-1}(\theta^2 f(j)) = \frac{a-\theta^2(a-mj)}{m} =$

$\frac{a}{m}(1-\theta^2) + j\theta^2$. According to Equation 7, we can get:

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

Since the pruning ratio $\gamma(\theta) = \frac{S(\theta)}{T}$ and $T = \frac{n(n-1)}{2}$, $\Rightarrow \;\; \gamma(\theta) = \frac{(n-2)-(2\frac{a}{m}-(n+2))(1-\theta^2)}{n}$. Also, we know $supp(A_n) = f(n) = a - mn > 0$, $\Rightarrow \;\; 2\frac{a}{m} > 2n \geq (n+2),\; when \;\; n \geq 2.$

Thus, we can derive three rules as follows:

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

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.

Lemma 8: 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:

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

Since the pruning ratio $\gamma(\theta) = \frac{S(\theta)}{T}$ and $T = \frac{n(n-1)}{2}$, $\Rightarrow \quad \gamma(\theta) = 2 - \frac{n+2}{n}\frac{1}{\theta^{\frac{2}{p}}}$.

Thus, we can derive three rules as follows:

$$\begin{aligned}
&rule\,1: \ \theta \nearrow \Rightarrow \ \frac{n+2}{n}\frac{1}{\theta^{\frac{2}{p}}} \searrow \Rightarrow \ \gamma(\theta) \nearrow \\
&rule\,2: \ p \nearrow \Rightarrow \ \frac{n+2}{n}\frac{1}{\theta^{\frac{2}{p}}} \searrow \Rightarrow \ \gamma(\theta) \nearrow \\
&rule\,3: \ n \rightarrow \infty \ \Rightarrow \ \lim_{n\rightarrow\infty}\frac{n+2}{n}\frac{1}{\theta^{\frac{2}{p}}} = \frac{1}{\theta^{\frac{2}{p}}}
\end{aligned}$$

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.

C. Dominance Zone Analysis

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

The main cost of the brute-force algorithm is the cost of computing $\frac{n(n-1)}{2}$ number of exact correlation coefficients. Let C_{corr} indicate the cost of computing the correlation coefficient for an item pair. The cost model of the brute-force algorithm is given as follows.

$$Cost_{Brute} = O(n^2) * C_{corr} \tag{8}$$

The main cost of the TAPER algorithm consists of three parts: the sorting cost denoted by C_{sort}, the cost of computing upper bounds denoted by C_{upper}, and the cost of computing exact correlation coefficients denoted by C_{tc}. The cost mode of the TAPER algorithm is given as follows.

$$Cost_{TAPER} = C_{sort} + C_{upper} + C_{tc} = O(nlogn) + O(n) * C_{upper} + (1 - \gamma(\theta))O(n^2) * C_{corr}$$

We have $Cost_{Brute} - Cost_{TAPER} = \gamma(\theta)O(n^2) * C_{corr} - O(nlogn) - O(n) * C_{upper}$.

With the increase of n, the computation savings can be more dramatic for the TAPER algorithm, particularly for data sets with non-uniform rank-support distributions, such as Zipf distributions. Note that C_{corr} is much larger than C_{upper}, since there is a very expensive computation cost of finding support values for item pairs when computing exact correlation coefficients. Indeed, when the number of objects is large, we cannot store the support values of all item pairs in the memory; that is, we may need to scan the whole data set once in order to find the support value of every item pair.

VI. Discussion

In this section, we extend our algorithm for finding item pairs with strong negative correlations and demonstrate that the algorithmic ideas developed here can also be applied to some other association measures, such as uncentered Pearson's correlation coefficients.

A. Negative Correlations

In this paper, our focus is to find all pairs of high positively correlated items. However, in some application domains, there may be interest in knowing pairs of high negatively correlated items [27]. In the following, we present a lower bound of the ϕ correlation coefficient.

Lemma 9: Given a pair of items $\{A,\, B\}$, without loss of generality, let $supp(A) \geq supp(B)$. The lower bound, $lower(\phi_{\{A,B\}})$, of the ϕ correlation coefficient is equal to

$$\begin{cases} -\frac{\sqrt{supp(A)supp(B)}}{\sqrt{(1-supp(A))(1-supp(B))}} & \text{if } supp(A) + supp(B) \leq 1 \\ \\ -\frac{\sqrt{(1-supp(A))(1-supp(B))}}{\sqrt{supp(A)supp(B)}} & \text{if } supp(A) + supp(B) > 1 \end{cases}$$

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 decreasing with the decrease of $supp(A,B)$. Let us consider the following two cases:

CASE 1: $if\, supp(A) + supp(B) \leq 1$

The minimum possible value of $supp(A, B)$ is zero. Hence,

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

CASE 2: $if\, supp(A) + supp(B) > 1$

The minimum possible value of $supp(A, B)$ is equal to $supp(A) + supp(B) - 1$. Hence,

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

From the above, this Lemma holds.

We also present a conditional monotone property of the lower bound of the ϕ correlation coefficient.

Lemma 10: For a pair $\{A, B\}$, let $supp(A) > supp(B)$. We have the following two cases: 1) If $supp(A)+supp(B) \leq 1$ and $supp(A)$ is fixed, the $lower(\phi_{\{A,B\}})$ is monotone increasing with the decrease of $supp(B)$. 2) If $supp(A)+supp(B) > 1$ and $supp(B)$ is fixed, the $lower(\phi_{\{A,B\}})$ is monotone increasing with the increase of $supp(A)$.

Proof: Let us consider the following two cases:

CASE 1: $supp(A) + supp(B) \leq 1$ and $supp(A)$ is fixed. By Lemma 9, we have $lower(\phi_{\{A,B\}}) = -\frac{\sqrt{supp(A)supp(B)}}{\sqrt{(1-supp(A))(1-supp(B))}}$. For any given two items B_1 and B_2 with $supp(A) > supp(B_1) > supp(B_2)$, we need to prove $lower(\phi_{\{A,B_1\}}) < lower(\phi_{\{A,B_2\}})$. This claim can be proved as follows.

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

The above follows the given condition that $supp(B_1) > supp(B_2)$ and $(1 - supp(B_1)) < (1 - supp(B_2))$. Since $lower(\phi_{\{A,B\}}) < 0$, we have $lower(\phi_{\{A,B_1\}}) < lower(\phi_{\{A,B_2\}})$.

CASE 2: $supp(A) + supp(B) > 1$ and $supp(B)$ is fixed. By Lemma 9, we have $lower(\phi_{\{A,B\}}) = -\frac{\sqrt{(1-supp(A))(1-supp(B))}}{\sqrt{supp(A)supp(B)}}$. For any given two items A_1 and A_2 with $supp(A_1) > supp(A_2) > supp(B)$, we need to prove $lower(\phi_{\{A_1,B\}}) > lower(\phi_{\{A_2,B\}})$. This claim can be proved as follows:

$$\frac{|\,lower(\phi_{\{A_1,B\}})|}{|\,lower(\phi_{\{A_2,B\}})|} = \sqrt{\frac{supp(A_2)}{supp(A_1)}}.\sqrt{\frac{1-supp(A_1)}{1-supp(A_2)}} < 1$$

The above follows the conditions that $supp(A_1) > supp(A_2)$ and $(1 - supp(A_1)) < (1 - supp(A_2))$. Since $lower(\phi_{\{A,B\}}) < 0$, we get $lower(\phi_{\{A_1,B\}}) > lower(\phi_{\{A_2,B\}})$.

Based on Lemmas 9 and 10, we can extend the TAPER algorithm to find all pairs of high negatively correlated items as well. Note that Pearson's correlation coefficient has some limitations when capturing negative correlation between items with low support. However, in this paper, our focus is on the computational perspective of Pearson's correlation coefficient.

B. An Extension to Uncentered Pearson's Correlation Coefficient

Here, we present how to extend the algorithmic ideas developed in this paper for uncentered Pearson's correlation coefficient, also known as the cosine measure [23]. Using the support notation, uncentered Pearson's correlation coefficient is defined as the following equation.

$$uncentered(\{A,B\}) = \frac{supp(\{A,B\})}{\sqrt{supp(A)supp(B)}} \tag{9}$$

Indeed, similar to the ϕ correlation coefficient, uncentered Pearson's correlation coefficient has an upper bound and this upper bound has a conditional monotone property as shown in the following.

Lemma 11: 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 $upper(uncentered$ $(\{A, B\}))$ of uncentered Pearson's correlation coefficient for an item pair $\{A, B\}$ can be obtained when $supp(A,\ B) = supp(B)$ and $upper(uncentered(\{A, B\})) = \sqrt{\frac{supp(B)}{supp(A)}}$

Proof: According to Equation 9, for an item pair {A, B}, $uncentered(\{A, B\}) = \frac{supp(\{A,B\})}{\sqrt{supp(A)supp(B)}}$. By the given condition $supp(A) \geq supp(B)$ and the anti-monotone property of the support measure, the upper bound upper(uncentered({A, B})) of uncentered Pearson's correlation coefficient can be obtained when

$supp(A, B) = supp(B)$. Hence, $upper(uncentered(\{A, B\})) = \sqrt{\frac{supp(B)}{supp(A)}}$.

Lemma 12: For a pair of items $\{A, B\}$, if we let $supp(A) > supp(B)$ and fix item A, the upper bound of uncentered Pearson's correlation coefficient $upper(uncentered(\{A, B\})$ for item pair $\{A, B\}$ is monotone decreasing with the decrease of the support value of item B.

Proof: Since $upper(uncentered(\{A, B\})) = \sqrt{\frac{supp(B)}{supp(A)}}$, if item A is fixed, the $upper(uncentered(\{A, B\}))$ becomes one variable function with supp(B) as the variable. It can be seen that this function is monotone decreasing with the decrease of supp(B).

Lemma 11 and Lemma 12 are analogous to Lemma 1 and Lemma 2, which form the basis of the TAPER algorithm. Therefore, the algorithmic ideas in the TAPER algorithm can also be applied to efficiently compute uncentered Pearson's correlation coefficient.

VII. Experimental Results

In this section, we present the results of experiments to evaluate the performance of the TAPER algorithm. Specifically, we demonstrate: (1) a performance comparison between TAPER 1D and a brute-force approach, (2) the effectiveness of the proposed algebraic cost model, (3) the scalability of the TAPER algorithm, (4) a performance evaluation of the choices between 1-D and 2-D monotone properties in the coarse filter step of TAPER, and (5) the extension of algorithmic ideas developed in TAPER for computing negative correlation as well as the uncentered Pearson's correlation coefficient.

TABLE III
Parameters of the 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

A. The Experimental Setup

Our experiments were performed on both real-life and synthetic data sets. Synthetic data sets were generated such that the rank-support distributions follow Zipf's law, as shown in Figure 11. Note that, in

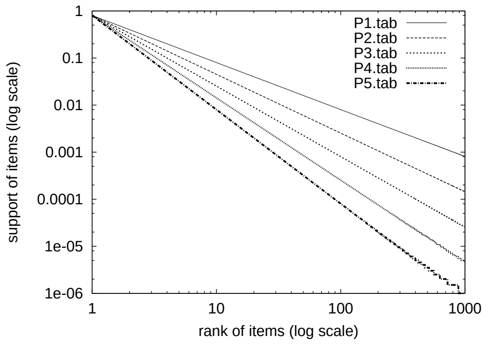


Fig. 11. The plot of the Zipf rank-support distributions of synthetic data sets in log-log scale.

TABLE IV

REAL-LIFE DATA SET CHARACTERISTICS.

Data set	#Item	#Transaction	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

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 III, 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-life data sets were obtained from several different application domains. Table IV 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 benchmarks for evaluating the performance of association rule algorithms on dense data sets. The `pumsb` 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 the 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.

[1]These data sets were obtained from IBM Almaden at http://www.almaden.ibm.com/cs/quest/demos.html (June 2005)

[2]These data sets and data content descriptions are available at http://www.ics.uci.edu/∼mlearn/MLRepository.html (June 2005)

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.

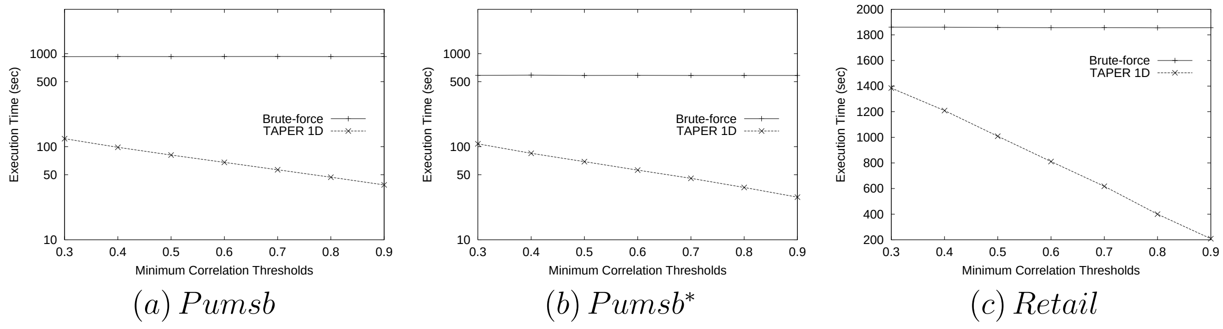


Fig. 12. TAPER 1D vs. a brute-force approach on the $Pumsb$, $Pumsb^*$, and retail data sets.

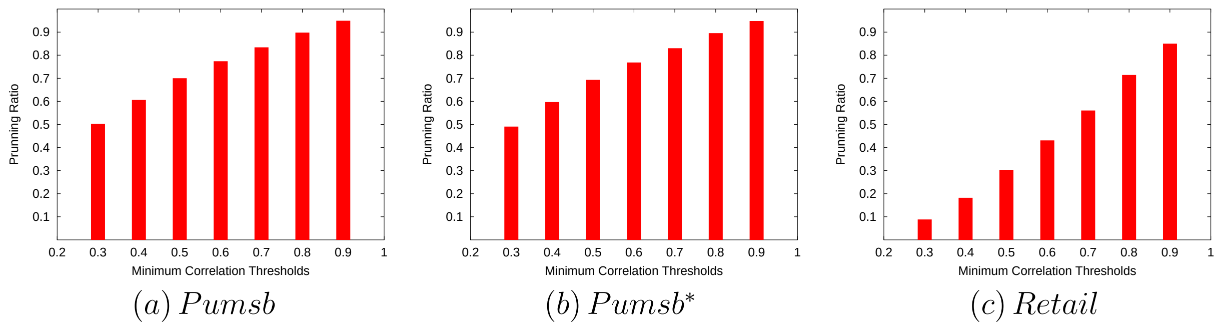


Fig. 13. The pruning effect of TAPER on $Pumsb$, $Pumsb^*$, and retail data sets.

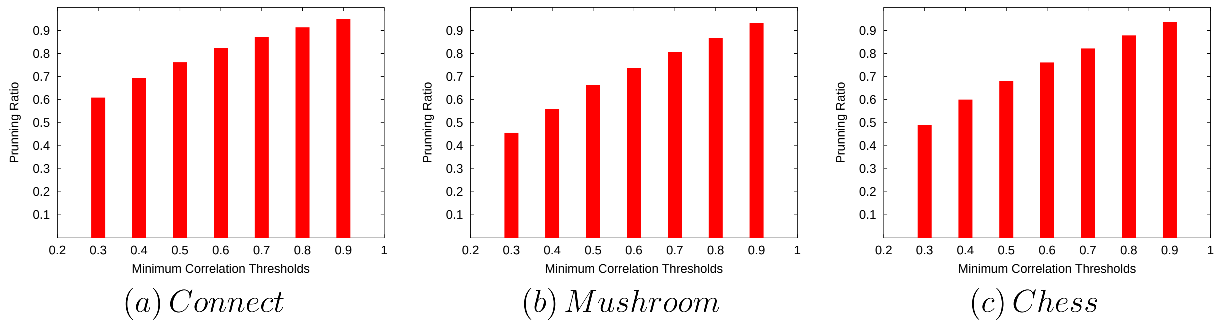


Fig. 14. The pruning effect of TAPER on UCI $Connect$, $Mushroom$, $Chess$ data sets.

B. TAPER 1-D vs. the Brute-force Approach.

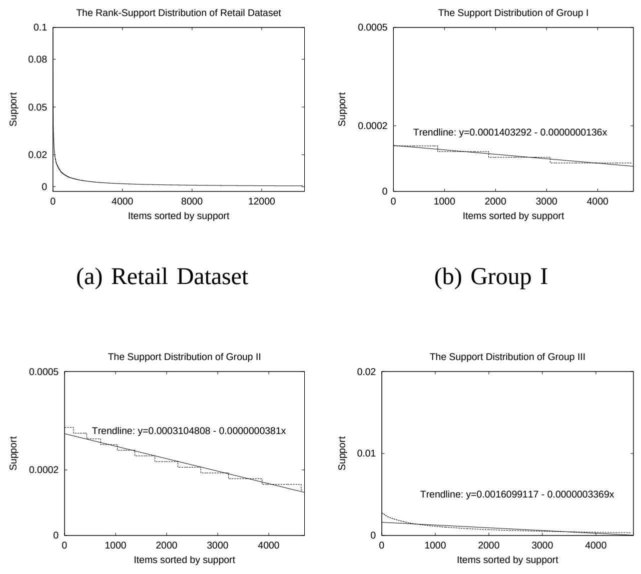
In this subsection, we present a performance comparison between TAPER 1D and a brute-force approach using several benchmark data sets from IBM, a UCI machine learning repository, and some other sources, such as retail stores. The implementation of the brute-force approach is similar to that of TAPER 1D except that the filtering mechanism implemented in TAPER 1D is not included in the brute-force approach.

Figure 12 shows the relative computation performance of TAPER 1-D 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 TAPER 1-D 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 12 (a), the execution time of TAPER 1D 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 1D 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 TAPER (both TAPER 1D and TAPER 2D) on these data sets. As illustrated in Figure 13, 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 20 shows the pruning ratio of TAPER on the UCI `connect`, `mushroom`, and `chess` data sets. The pruning ratios 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.

C. The Effect of Correlation Thresholds

In this subsection, we present the effect of correlation thresholds on the computation savings of TAPER (both TAPER 1D and TAPER 2D). Recall that our algebraic cost model shows that the pruning ratio of TAPER increases with increases of the correlation thresholds for data sets with linear and Zipf-like distributions. Figure 13 shows such an increasing trend of the pruning ratio on the `pumsb`, `pumsb`*, and `retail` data sets as correlation thresholds increase. Also, Figure 20 shows a similar increasing trend of the pruning ratio on the UCI benchmark datasets including `mushroom`, `chess`, and `connect`.



(a) Retail Dataset (b) Group I

(c) Group II (d) Group III

Fig. 15. The rank-support distributions of the retail data set and its three item groups with a linear regression fitting line (trendline).

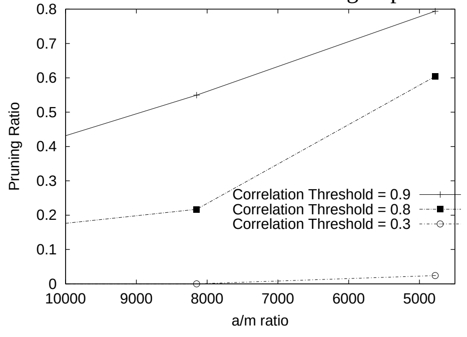


Fig. 16. Pruning ratios with the decrease of a/m for data sets with linear rank-support distribution.

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 trend similar to that of the proposed algebraic cost model although the rank-support distributions of these data sets do not follow Zipf's law exactly.

TABLE V
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

D. 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 TAPER (both TAPER 1D and TAPER 2D) 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 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 V. Figure 15 (a) shows the plot of the rank-support distribution of the retail data set and Figure 15 (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 V 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 16 shows the pruning ratio of the TAPER algorithm on the data set with linear rank-support distributions. As expected, 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 as shown in Lemma 7.

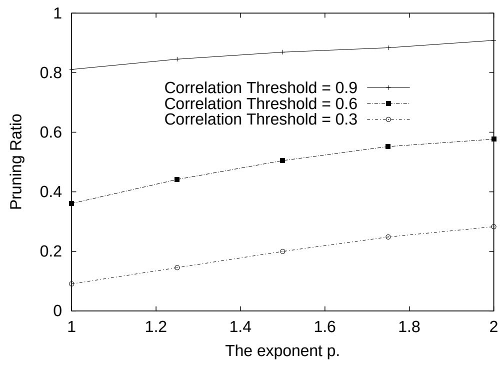


Fig. 17. The increase of pruning ratios with the increase of p for data sets with Zipf-like distribution.

E. The Effect of the Exponent p

In this subsection, we examine the effect of the exponent P on the performance of TAPER (both TAPER 1D and TAPER 2D) for data sets with a generalized Zipf rank-support distribution. We used the synthetic data sets presented in Table III 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 17 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.

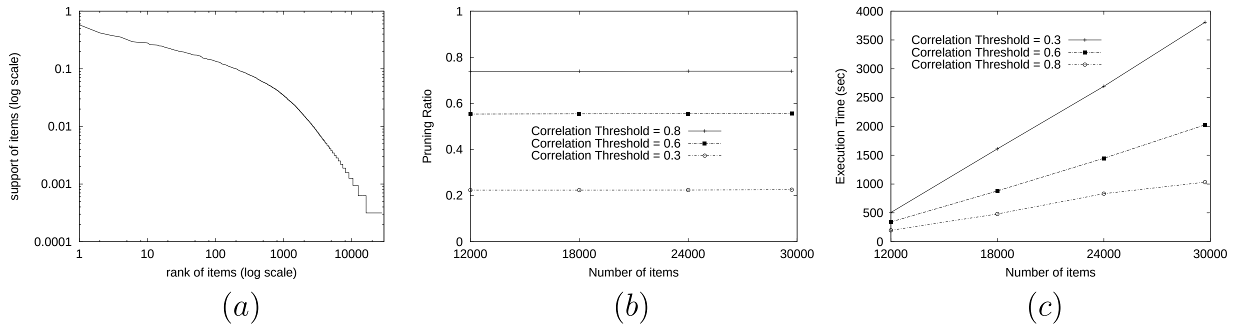


Fig. 18. (a) The plot of the rank-support distribution of the LA1 data set in log-log scale. (b) The effect of database dimensions on the pruning ratio for data sets with Zipf-like rank-support distributions. (c) The effect of database dimensions on the execution time for data sets with Zipf-like rank-support distributions.

F. 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 18 (a) 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 generated data sets and the LA1 data set for our scale-up experiments.

For data sets with Zipf-like rank-support distributions, Figure 18 (b) 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 18 (c), 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.

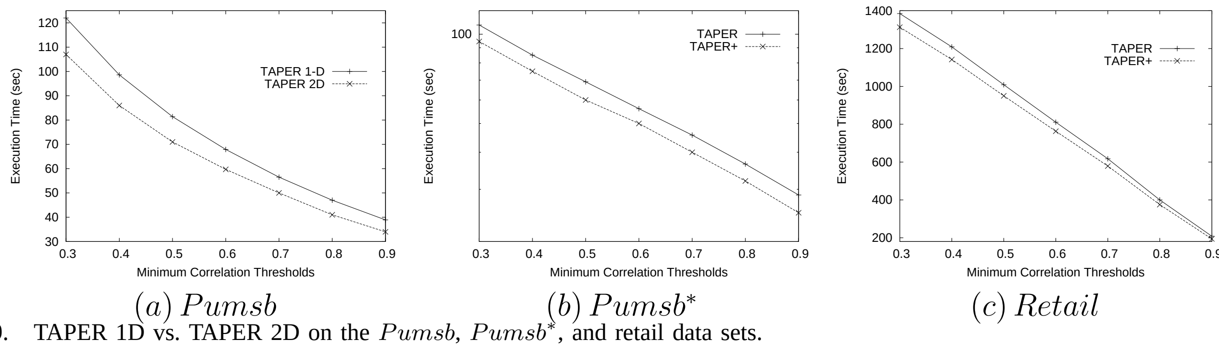


Fig. 19. TAPER 1D vs. TAPER 2D on the $Pumsb$, $Pumsb^*$, and retail data sets.

G. *Evaluation of the choices between 1D and 2D Monotone Properties*

Here, we present a performance evaluation of the Choices between 1D and 2D monotone properties in the coarse filter step of the TAPER algorithm. Figure 19 shows the execution time of TAPER 1D and TAPER 2D on the UCI `connect`, `mushroom`, and `chess` data sets. In the figure, we can see that the execution time of TAPER 2D can be 10-15% less than that of TAPER 1D for the various different correlation thresholds. This computation savings is due to the fact that TAPER 2D reduces the number of upperbounds that need to be computed as demonstrated by Lemma 4.

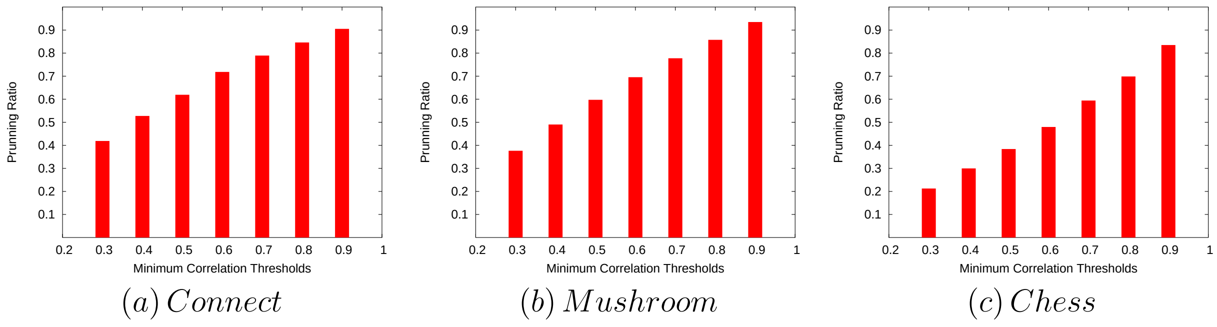


Fig. 20. The pruning effect for uncentered Pearson's correlation coefficient on UCI $Connect$, $Mushroom$, $Chess$ data sets.

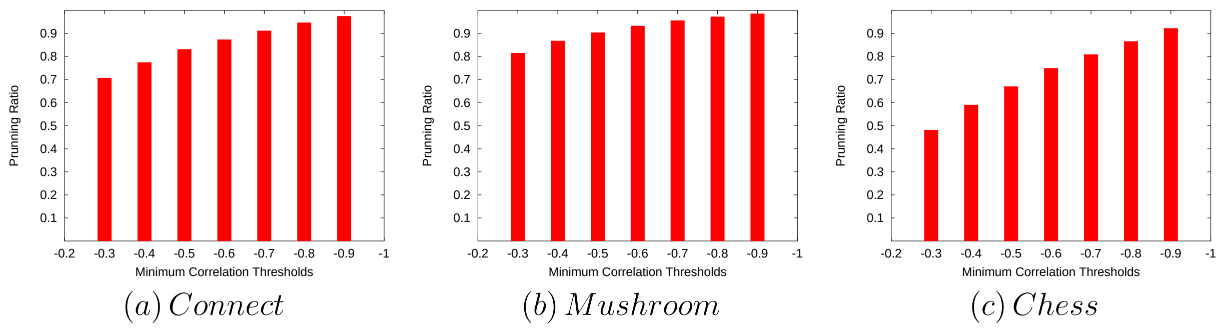


Fig. 21. The pruning effect for negative correlation computation on UCI $Connect$, $Mushroom$, $Chess$ data sets.

H. The Extension of Algorithmic Ideas Developed in TAPER

In this subsection, we present experimental results to show the effectiveness of extending algorithmic ideas developed in the TAPER algorithm for computing the cosine measure as well as negative correlation coefficients. Figure 20 shows the pruning ratios for computing the cosine measure on the UCI `connect`, `mushroom`, and `chess` data sets. Once again, the pruning ratios achieved on these data sets are quite significant. This indicates that the algorithmic ideas developed in TAPER can also achieve good computation performance for computing the cosine measure. Finally, Figure 21 shows the pruning ratios for computing negative correlation on the UCI `connect`, `mushroom`, and `chess` data sets. The pruning ratios achieved in this case are even better than those for computing positive correlation.

VIII. Conclusions and Future Work

In this paper, 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. TAPER uses an upper bound of the ϕ correlation coefficient, which shows 2-D monotonic properties. In addition, we provided algebraic cost models to measure the computation performance of the TAPER algorithms. 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. Finally, we showed that the algorithmic ideas developed in the TAPER algorithm can be extended to efficiently compute uncentered Pearson's correlation coefficients as well as negative correlation.

There are several potential directions for future research. First, we are interested in studying how to specify statistically meaningful correlation thresholds. Second, we plan to efficiently compute the top-k correlated item pairs. Finally, we will investigate an efficient solution for correlation detection using Bayesian and Markov models.

Acknowledgments

We thank Professor Joydeep Ghosh (University of Texas - Austin, USA) and Professor Ke Wang (Simon Fraser University, Canada) for valuable comments. Also, we are grateful to 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 *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

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

[3] 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.

[4] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 265–276, 1997.

[5] C. Bucila, J. Gehrke, D. Kifer, and Walker M. White. Dualminer: a dual-pruning algorithm for itemsets with constraints. In *Data Mining and Knowledge Discovery Journal*, pages 241–272, 2003.

[6] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, pages 443–452, 2001.

[7] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J.D. Ullman, and C. Yang. Finding interesting associations without support pruning. In *Proceedings of the 16th International Conference on Data Engineering (ICDE)*, pages 489–499, 2000.

[8] P. Cohen, J. Cohen, Stephen G. West, and Leona S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Science*. Lawrence Erlbaum Assoc; 3rd edition, 2002.

[9] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 67–76, 2001.

[10] G. Grahne, Laks V. S. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. In *Proceedings of the 16th International Conference on Data Engineering*, 512-521 2000.

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

[12] Ihab F. Ilyas, V. Markl, Peter J. Haas, P. Brown, and A. Aboulnaga. Cords: Automatic discovery of correlations and soft functional dependencies. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, 2004.

[13] C. Jermaine. The computational complexity of high-dimensional correlation search. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM)*, pages 249–256, 2001.

[14] C. Jermaine. Playing hide-and-seek with correlations. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

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

[16] M. Kendall and J. D. Gibbons. *Rank Correlation Methods*. Oxford University Press (fifth edition)., 1990.

[17] W.P. Kuo, T. Jenssen, A.J. Butte, L. Ohno-Machado, and I.S. Kohane. Analysis of matched mrna measurements from two different microarray technologies. *Bioinformatics*, 18(3), 2002.

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

[19] B. Liu, W. Hsu, and Y. Ma. Mining association rules with multiple minimum supports. In *Proc. of ACM SIGKDD*, 1999.

[20] R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining via constrained frequent set queries. In *Proceedings ACM SIGMOD International Conference on Management of Data*, 1999.

[21] R. Rastogi and K. Shim. Mining optimized association rules with categorical and numeric attributes. *IEEE Transactions on Knowledge and Data Engineering*, 14(1), January 2002.

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

[23] C. J. Van Rijsbergen. *Information Retrieval (2nd Edition)*. Butterworths, London, 1979.

[24] R. Rymon. Search through systematic set enumeration. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 539–550, 1992.

[25] H. V. Storch and F. W. Zwiers. *Statistical Analysis in Climate Research*. Cambridge University Press; Reprint edition, February 2002.

[26] K. Wang, Y. He, D. Cheung, and Y.L. Chin. Mining confident rules without support requirement. In *Prof. of the 2001 ACM CIKM Intĺ conf. Information and Knowledge Management*, 2001.

[27] X. Wu, C. Zhang, and S. Zhang. Mining both positive and negative association rules. In *Proceedings of the Nineteenth International Conference (ICML)*, pages 658–665, 2002.

[28] H. Xiong, S. Shekhar, P. Tan, and V. Kumar. Exploiting a support-based upper bound of pearson's correlation coefficient for efficiently identifying strongly correlated pairs. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 334–343, August 2004.

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