Chapter 4

Finding Strongly Correlated Item Pairs in Large Transaction Databases

Correlation mining is an approach that allows one to draw statistical relationships among items from transaction data. Existing techniques either generate large number of candidates or build huge trees and require multiple passes over the database. This chapter presents an effective and fast Strongly COrelated Pairs Extraction technique called SCOPE, and its extension to extract $k$ most strongly correlated pairs from large transaction databases. Many existing techniques use Pearson's correlation coefficient as a measure of correlation, which may not always perform well when data is noisy and binary. As an alternative to Pearson's correlation coefficient, we present a method of computing Spearman's rank order correlation coefficient from the transaction data. We find that the proposed technique performs satisfactorily in terms of execution time when tested with several real and synthetic datasets compared to other similar techniques.
4.1 Introduction

Starting from market basket data analysis, association mining is now applied in a wide variety of domains such as machine learning, soft-computing and computational biology. Standard association mining technique extracts all subset of items satisfying a minimum support constraint. The traditional association rule mining technique\cite{21,39} is based on a support-confidence framework. However, the support-confidence framework can be misleading; it can identify a rule \((A \Rightarrow B)\) as interesting (strong) when in fact, the occurrence of \(A\) might not imply the occurrence of \(B\). Thus, the support and confidence measures are insufficient in filtering out uninteresting association rules\cite{39,40}. It has been observed that item pairs with high support value may not imply high correlation. Similarly, a highly correlated item pair may exhibit low support value. To tackle this weakness, correlation analysis can be used to provide an alternative framework for finding statistically interesting relationships \cite{40}. It also improves the understanding of some association rules. In statistics, relationships among nominal variables can be analyzed with nominal measures of association such as Pearson’s correlation coefficient and measures based on Chi Square\cite{41}. The \(\phi\) correlation coefficient\cite{41} is a computational form of Pearson’s correlation coefficient for binary variables. An equivalent support measure based \(\phi\) correlation coefficient computation technique is introduced in\cite{42,43} to find correlation of item pairs in a transaction database based on their support count. For any two items \(X\) and \(Y\) in a transaction database, the support based \(\phi\) correlation coefficient can be calculated as:

\[
\phi(X, Y) = \frac{Sup(X, Y) - Sup(X) \times Sup(Y)}{\sqrt{Sup(X) \times Sup(Y) \times (1 - Sup(X)) \times (1 - Sup(Y))}} \tag{4.1}
\]

where \(Sup(X)\), \(Sup(Y)\), and \(Sup(X, Y)\) are the individual supports and the joint support of item \(X, Y\), respectively.

Unlike traditional association mining, the all-pair-strongly correlated query is to find statistical relationships among pair of items from a transaction database. The problem can be defined as follows.
**Definition 4.1.1 (Strongly correlated pair)**: Assume a market basket database $D$ with $T$ transactions and $N$ items. Each transaction, $T$ is a subset of $I$, where $I = \{X_1, X_2, \ldots, X_N\}$ is a set of $N$ distinct items. Given a user-specified minimum correlation threshold $\theta$, an all-strong-pairs correlation query (SC) finds a set of all item pairs $(X_i, X_j)$ (for $i, j = 1 \ldots N$) with correlation, $\text{Corr}(X_i, X_j)$, above the threshold $\theta$. Formally, it can be defined as:

$$SC(D, \theta) = \{\{X_i, X_j\}|\{X_i, X_j\} \subseteq I, X_i \neq X_j \land \text{Corr}(X_i, X_j) \geq \theta\}. \quad (4.2)$$

Besides providing a statistical meaning for the traditional association mining problem, correlation mining can play a major role in addressing various issues such as how sales of a product are associated with sales of other products, which in turn may help in designing sales promotion, catalog design and store layout. Correlation mining can be helpful in efficient finding of co-citations and term co-occurrences during document analysis. Functional relationship among pairs of genes based on gene expression profile and changes in functional relationship in different diseases and conditions may be indicative of disease mechanism for diseases like cancer. It has been observed that a simple pair-wise correlation analysis may be helpful in revealing new gene-gene relationship, which again in turn are useful in discovering gene regulatory pathways or gene interaction networks.

To determine the appropriate value of $\theta$, a prior knowledge of data distribution is required. Without specific knowledge of the target data, users will have difficulty in setting the correlation threshold to obtain required results. If the correlation threshold is set too large, there may be only a small number of results or even no result. In such a case, the user may have to guess a smaller threshold and perform the mining again, which may or may not give better result. If the threshold is too small, there may be too many results for the user; too many results need an exceedingly long time to compute, and also extra effort to filter the answers.

An alternative solution to this problem could be to change the task of mining
correlated item pairs under pre-specified threshold to mine top-k strongly correlated item pairs from transaction database, where k is the desired number of item pairs that have largest correlation values. Recently the idea of top-k strongly correlated pairs has been applied in graph databases to find top-k frequent correlated subgraph\(^{47}\). The top-k correlated-pairs query problem in market-basket can be defined as follows.

**Definition 4.1.2 (Top-k strongly correlated pairs):** Given a user-specified threshold \(k\) and a market basket database \(D\) of \(T\) transactions where each transaction \(I_i\) is a subset of \(I\) (set of \(N\) distinct items), a top-k correlated-pair query, \(TopK(D,k)\), finds list of \(k\) top most item pairs based on correlation coefficient value. Thus, \(TopK(D,k)\) can be represented as follows:

\[
TopK(D,k) = \{\{X_{i_1},X_{j_1}\}, \{X_{i_2},X_{j_2}\}, \ldots, \{X_{i_k},X_{j_k}\}\}
\]  
(4.3)

where, \(Corr(X_{i_1},X_{j_1}) \geq Corr(X_{i_2},X_{j_2}) \geq \ldots \geq Corr(X_{i_k},X_{j_k})\), for all \(\{X_{i_k},X_{j_k}\} \subseteq I\) and \(X_{i_k} \neq X_{j_k}\)

Thus, top-k is a sorted list of \(k\) item pairs based on any suitable correlation coefficient, \(Corr\).

**4.1.1 Computing support based correlated pairs: an illustration**

The task of strongly correlated item pair finding generates a list of pairs from the database where \(Corr\) value of a pair is greater than the user specified \(\theta\). Similarly, the task of top-k correlated-pair finding generates a sorted list of \(k\) pairs in the order of \(Corr\) from the database. An illustration of both correlated-pairs query problems is given in Figures 4.1 and 4.2. In the example, we denote \(Corr\) as correlation coefficient. The input to the strongly correlated query is a market basket database containing 8 transactions and 6 items. The value of \(\theta\) is set to 0.05. Similarly, for
the top-\(k\) problem the value of \(k\) is set to 8. Since the database has six items, there are \((\binom{6}{2}) = 15\) item pairs for which correlation coefficient \(\phi\) is calculated. To compute \(\phi(4, 5)\) using Equation (4.1), we need the single element supports \(\text{Sup}(4) = 4/8\) and \(\text{Sup}(5) = 3/8\), and joint support \(\text{Sup}(4, 5)=3/8\), to compute correlation coefficient, \(\phi(4, 5)\), which is 0.77. Finally, all pairs that satisfy \(\theta\) constraint are extracted, and the list of strongly correlated pairs is generated as output. Similarly, the list of \(k\) most strongly correlated pairs is generated as an output for the second problem (irrespective of any \(\theta\) value).

### 4.2 Related Work

We now discuss some of the state-of-the-art approaches towards finding strongly correlated item pairs and top \(k\) strongly correlated item pairs from transaction database.

#### 4.2.1 TAPER

TAPER\textsuperscript{42,43} is a candidate generation based technique for finding all strongly correlated item pairs. It consists of two steps: filtering and refinement. In the filtering
step, it applies two pruning techniques. The first technique uses an upper bound of the $\phi$ correlation coefficient as a coarse filter. The upper bound $upper(\phi(X,Y))$ of $\phi$ correlation coefficient for $(X,Y)$ is:

$$
\phi(X,Y) \leq upper(\phi(X,Y)) = \sqrt{\frac{sup(Y)}{sup(X)}} \sqrt{\frac{1 - sup(X)}{1 - sup(Y)}}.
$$ (4.4)

If the upper bound of the $\phi$ correlation coefficient for an item pair is less than the user-specified correlation threshold $\theta$, the item pair is pruned right away. To minimize the effort to compute upper bounds of all possible item pairs, TAPER applies the second pruning technique based on the conditional monotone property (1-D) of the upper bound of the $\phi$ correlation coefficient. For an item pair $(X,Y)$, if the upper bound is less than $\theta$, all item pairs involving item $X$ and rest of the target items having support less than $Y$ will also give upper bound less than $\theta$. In other words, for item pair $X,Y$, if $sup(X) > sup(Y)$ and we fix item $X$, the upper bounds $upper(\phi(X,Y))$, is monotone decreasing with decreasing support of item $Y$. Based on this 1-D monotone property, straightaway one can avoid computation of upper bound for other items. In the refinement step, TAPER computes the exact correlation for each surviving pair and retrieves the pairs with correlation above $\theta$.

It is understood that in comparison with single element item sets, usually the
two element candidate sets are huge. The upper bound based pruning technique is very effective in eliminating large numbers of item pairs during the candidate generation phase. However, when the database contains a large number of items and transactions, testing even the remaining candidate pairs is expensive.

4.2.2 Tcp

FP-tree\(^{36}\) based technique, Tcp\(^{48}\) is a milestone in strongly correlated item pair extraction, that overcome the bottlenecks of TAPER. Strongly correlated item pairs are generated without any candidate generation. Tcp includes two sub processes: (i) construction of the FP-tree, and (ii) computation of correlation coefficient of each item pair using the support count from the FP-tree and extraction of all strongly correlated item pairs with correlation greater than \(\theta\). The efficiency of the FP-tree algorithm can be justified as follows: (i) The FP-tree is a compressed representation of the original database, (ii) the algorithm scans the database twice only, and (iii) the support value of all item pairs is available in the FP-tree.

Although the algorithm is based on an efficient FP-tree data structure, it suffers from the following two significant disadvantages.

1. Tcp constructs the entire FP-tree with an initial support threshold of zero. The time taken to construct such an FP-tree is quite large, especially when the dimensions are large.

2. Moreover, it requires a large amount of space to store the entire FP-tree in the memory, particularly when the number of items is very large.

Below we discuss some of the top-\(k\) correlated pair finding techniques. Almost all the techniques proposed so far are minor extensions of strongly correlated pair finding techniques.

4.2.3 TOP-COP

TOP-COP\(^{49}\) is an upper bound based algorithm for finding top-\(k\) strongly correlated item pairs and is an extended version of TAPER. TOP-COP exploits a 2-D
monotone property of the upper bound of $\phi$ correlation coefficient for pruning non-potential item pairs, i.e., pairs which do not satisfy the correlation threshold $\theta$. The 2-D monotone property is as follows: For a pair of items $X, Y$, if $sup(X) > sup(Y)$ and we fix item $Y$, $upper(\phi(X, Y))$ is monotone increasing with decreasing support of item $X$. Based on the 2-D monotone property a diagonal traversal technique, combined with a refine-and filter strategy is used to efficiently mine top-$k$ strongly correlated pairs.

Like TAPER, TOP-COP is also a candidate generation based technique. The 1-D monotone property, used in TAPER provides a one dimensional pruning window for eliminating non-potential item pairs. Moving one step further, TOP-COP exploits the 2-D monotone property, which helps further in eliminating non-potential pairs from two dimensions instead of one dimension. Compared to 1-D monotone based pruning, the 2-D pruning technique is more effective in eliminating a large number of item pairs during the candidate generation phase. Like TAPER, TOP-COP also starts with a sorted list of items based on support in non-increasing order, which needs a scan of the database once for creating such a list. Since it is a candidate generation based technique and has structural similarity with TAPER, it also suffers from the drawback of expensive testing of remaining candidates after pruning and filtering steps.

4.2.4 Tkcp

Tkcp$^{50}$, is an extension of Tcp to extract top-$k$ strongly correlated item pairs using an FP-tree$^{36}$ based approach. Tkcp also includes two sub processes: $(a)$ construction of the FP-tree, and $(b)$ computation of correlation coefficient for each item pair using support count from the FP-tree and extraction of all the top-$k$ strongly correlated item pairs based on the correlation coefficient value, $\phi$. Tkcp are also suffers from the same limitations as Tcp.
4.3 Motivation

Existing correlated pair finding techniques require multiple passes over the database, which is too costly for large transaction databases. It would be more effective if both strongly correlated pairs as well as top-$k$ strongly correlated item pairs can be extracted using a single pass over the database and without generating any large tree or candidate itemsets. Majority of correlation mining techniques use Pearson's correlation coefficient for finding strongly correlated item pairs. These are parametric techniques that work well with continuous variables. Since typical market basket data are binary in nature, a parametric approach may not always perform well for binary data. Further, the parametric correlation coefficient is sensitive to outliers in a data set.

To address the above issues, we present two fast and effective techniques, (i) SCOPE, which extracts all strongly correlated item pairs, for any large database and (ii) $k$-SCOPE, an extension to SCOPE, to extract top-$k$ strongly correlated item pairs in only one pass over the database, without generating any candidate set. To overcome the problems associated with Pearson correlation, non-parametric techniques like Spearman's $ho$ \(^{51}\) can be considered as better alternative in this regard. Below, we present an approach for computing Spearman's $ho$ between an item pair from market basket data.

4.4 Computing Spearman’s Rank order correlation

Parametric techniques like Pearson's correlation are sensitive to the distribution of the data\(^{52,53}\). Parametric techniques may not be effective when data is noisy and binary in nature. The alternative solution is to apply non-parametric correlation. If two variables $X$ and $Y$ are metric (e.g., interval or ratio scale measures) and they are to be correlated, a parametric technique like Pearson's correlation coefficient is suitable. While desirable, it is not always possible to use a paramet-
ric test such as the Pearson's method. In case of non-parametric (e.g., nominal or ordinal measures) variables, correlation can be determined effectively by using a non-parametric correlation technique. Non-parametric correlation coefficients, such as Chi-square, Point biserial correlation\textsuperscript{54}, Spearman's $\rho$\textsuperscript{51}, Kendall's $\tau$\textsuperscript{55}, and Goodman and Kruskal's $\lambda$\textsuperscript{56} may perform better than parametric correlation coefficient when outliers are present. The most frequently used is the rank order based Spearman's $\rho$ correlation. In principle, Spearman's $\rho$ is simply a special case of Pearson's product-moment coefficient in which two sets of data $X_i$'s and $Y_i$'s are converted to rank $x_i$'s and $y_i$'s before calculating the coefficients. The raw scores are converted to ranks, and the differences $D_i$'s between the ranks of the observations on the two variables are calculated. If there are no tied ranks, then $\rho$ is given by:

$$\rho = 1 - \frac{6 \sum D_i^2}{N(N^2 - 1)}. \hspace{2cm} (4.5)$$

where, $D_i = x_i - y_i$, is the difference between the ranks of corresponding values $X_i$ and $Y_i$, and $N$ is the number of samples in each dataset (same for both sets).

If tied ranks exist, each tied score is assigned a rank equal to the average of all tied positions. For example, if a pair of scores are tied for the 2nd and 3rd rank, both scores are assigned a rank of 2.5 ((2+3)/2=2.5). In case of binary variables, a large number of tied cases are present. Binary market basket data contains score 0 and 1 only. To compute the ranks of 0 and 1, their natural ordering can be used to compute tied ranks. For computing the tied ranks, we assign 0 with greater priority than 1 and use simple frequency counts of 1 and 0. Below, we discuss the technique for calculating Spearman's $\rho$ with tied cases between binary variables.

Assume a binary variable $I$ with $N$ values. The frequency of score 1 in variable $I$ is denoted as $f(I)$. To determine the appropriate rank of tied cases, we need to add the rank positions and divide by the number of tied cases. Since the number of scores of a binary variable is only two, the tied rank of score 0 and 1 can be
calculated as:

\[
Rank_0 = \sum_{i=1}^{N-f(I)} \frac{i}{N - f(I)}. \tag{4.6}
\]

Similarly, the rank of score 1 can be calculated as:

\[
Rank_1 = \sum_{i=f(I)}^{N} \frac{i}{f(I)}. \tag{4.7}
\]

Once the ranks of 0 and 1 are calculated for the target variables (item pairs), say \(I_1\) and \(I_2\), whose correlation is to be calculated, the difference of their ranks is then used to compute \(D_i^2\). In case of two binary item sets, the only possible combination of scores are \((0,0), (0,1), (1,0)\) and \((1,1)\). Thus just by counting the frequencies of the above patterns and using the rank of 0 and 1 for both item sets, the sum of square differences of ranks \(D_i^2\) can be easily calculated as:

\[
\sum D_i^2 = P_{(00)}(Rank_0(I_1) - Rank_0(I_2))^2 + P_{(01)}(Rank_0(I_1) - Rank_1(I_2))^2 + P_{(10)}(Rank_1(I_1) - Rank_0(I_2))^2 + P_{(11)}(Rank_1(I_1) - Rank_1(I_2))^2 \tag{4.8}
\]

where, \(P_{(00)}, P_{(01)}, P_{(10)}\) and \(P_{(11)}\) are the frequencies of \((0,0), (0,1), (1,0)\) and \((1,1)\) patterns, respectively. \(Rank_0(I_1)\) and \(Rank_1(I_1)\) correspond to the ranks of 0's and 1's in item \(I_1\) and \(Rank_0(I_2)\) and \(Rank_1(I_2)\) are the corresponding 0's and 1's rank in item \(I_2\), respectively.

For a long sequence of binary data occurring in a large transaction dataset, sometimes it is costly to find \(P_{(00)}, P_{(01)}, P_{(10)}\) and \(P_{(11)}\) for each pair of items. It would be more effective if these can be computed with minimal information. One possible way is given below, especially when frequency of score 1 in item \(I_1\) and \(I_2\) (\(f(I_1)\) and \(f(I_2)\)) and joint occurrences in both item pairs \(f(I_1, I_2)\) are known.

\[
P_{(01)} = f(I_2) - f(I_1, I_2), P_{(10)} = f(I_1) - f(I_1, I_2), P_{(11)} = f(I_1, I_2),
\]

\[
P_{(00)} = N - (P_{(01)} + P_{(10)} + P_{(11)}).
\]
4.4.1. Computing Spearman’s $\rho$: an illustration

In this section we demonstrate how to compute Spearman’s $\rho$ with tied cases for binary market basket data. For example, let us consider the following item pairs $I_1$ and $I_2$ with $N = 6$ transactions $(T1, T2, \cdots, T6)$ with similar occurrence patterns.

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$I_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1: Sample market basket data with two items and six transactions

In the above data, it is obvious that the frequency of 1 in $I_1$ and $I_2$, and joint occurrences of 1 in both $I_1$ and $I_2$ are $f(I_1) = 3, f(I_2) = 3, f(I_1, I_2) = 3$, respectively. Using Equations (4.6) and (4.7) the value of $\text{Rank}_0(I_1)$ and $\text{Rank}_1(I_1)$ for $I_1$ become $\text{Rank}_0(I_1) = (1 + 2 + 3)/3 = 2$ and $\text{Rank}_1(I_1) = (4 + 5 + 6)/3 = 5$ Similarly, $\text{Rank}_0(I_2)$ and $\text{Rank}_1(I_2)$ of $I_2$ are 2 and 5, respectively.

The next step is to calculate $P_{(00)}, P_{(01)}, P_{(10)}$ and $P_{(11)}$ using joint frequency count $f(I_1, I_2)$.

$$
P_{(01)} = f(I_2) - f(I_1, I_2) = 3 - 3 = 0, 
P_{(10)} = f(I_1) - f(I_1, I_2) = 3 - 3 = 0, 
P_{(11)} = f(I_1, I_2) = 3, 
P_{(00)} = N - (P_{(01)} + P_{(10)} + P_{(11)}) = 6 - (0 + 0 + 3) = 3
$$

The summation of square rank difference $D_i^2$ is:

$$
\sum D_i^2 = P_{(00)}(\text{Rank}_0(I_1) - \text{Rank}_0(I_2))^2 + P_{(01)}(\text{Rank}_0(I_1) - \text{Rank}_1(I_2))^2 + P_{(10)}(\text{Rank}_1(I_1) - \text{Rank}_0(I_2))^2 + P_{(11)}(\text{Rank}_1(I_1) - \text{Rank}_1(I_2))^2.
$$

$$
= 3(2 - 2)^2 + 0(2 - 5)^2 + 0(5 - 2)^2 + 3(5 - 5)^2
$$

$$
= 0.
$$

Thus, Spearman's $\rho$ can be calculated as: $\rho = 1 - (6 \times 0)/6(6^2 - 1) = 1 - 0 = 1.$

58
In a transaction database, computing $f(I_1)$, $f(I_2)$ and $f(I_1, I_2)$ for any item pair is nothing but finding their individual and joint supports (1- and 2- element item sets).

A traditional approach for counting the support for item pairs need at least two scans over the entire dataset. We present a one-pass strongly correlated item pair finding technique called SCOPE and its extension $k$-SCOPE. We use a correlogram matrix for counting individual and joint supports for item pair in a single pass over the database. Using support count and the method discussed above, it is straightforward to compute the correlation coefficient $\phi$ or $\rho$ between an item pair.

### 4.5 SCOPE: Strongly COrelated Pair Extraction Technique

SCOPE attempt to find all strongly correlated item pairs and $k$-SCOPE extracts $k$ top most correlated pairs from any transaction database using a single scan over the database without generating any candidates. We use a correlogram matrix to store the support counts of all 1- and 2- element itemsets. Later, the matrix is used to calculate correlation coefficients of all item pairs.

SCOPE accepts the market-basket database $D$ and the correlation coefficient threshold $\theta$ as input, and generates all strongly correlated item pairs as output. Step 1 of SCOPE (see Algorithm 2) is dedicated to the construction of the correlogram matrix using a single scan of the original database. In step 3, the correlation coefficient of each item pair is computed and in step 5, all item pairs whose coefficient values are greater than or equal to $\theta$, are extracted. Finally, the algorithm returns a list of all strongly correlated item pairs.

An extended version of this algorithm for generating top-$k$ correlated pairs, viz, $k$-SCOPE is presented in Algorithm 3. The algorithm accepts the market-basket database $D$ and $k$ as input and generates a list of top-$k$ strongly correlated item pairs, $L$, as output. The first phase of the algorithm is the same as that of SCOPE.
input : \( D \) (Original Dataset), \( \theta \) (Correlation coefficient threshold)

output: \( L \) (List of strongly correlated item pairs)

1 Generate Correlogram Matrix \( M \) from \( D \);
2 for each item pair \((i, j) \in D\) do
3     Compute \( \text{Corr} (i, j) \) using support from \( M \);
4     if \( \text{Corr} (i, j) \geq \theta \) then
5         \( L := L \cup (i, j) \);
6     end
7 end
8 Return \( L \);

Algorithm 2: SCOPE: Strongly COrelated Pair Extraction

In steps 8 to 12, topmost \( k \) correlated item pairs are extracted and added to the top-\( k \) list. Top-\( k \) list \( L \) is a sorted list (descending order) of item pairs based on value of the correlation coefficient. Any pair whose correlation coefficient is lower than the \( k^{th} \) pair’s correlation coefficient is straightaway pruned. Otherwise, the algorithm updates the list by eliminating the \( k^{th} \) pair and inserting the new pair in its appropriate position in the list. Finally, the algorithm returns top-\( k \) list \( L \).

input : \( D \) (Original Dataset), \( k \)

output: \( L \) (List of \( k \) strongly correlated item pairs)

1 Generate Correlogram Matrix \( M \) from \( D \);
2 for each item pair \((i, j) \in D\) do
3     Compute \( \text{Corr} (i, j) \) using support from \( M \);
4     if \( |L| \leq k \) then
5         \( L := L \cup (i, j) \);
6     end
7 else
8     if \( \text{Corr} (i, j) \geq \text{Corr} (L[k]) \) then
9         \( L[k] := L[k] \cup (i, j) \);
10        Sort \( L \) in descending order on \( \text{Corr} \) of each pair;
11     end
12 end
13 end
14 Return \( L \);

Algorithm 3: \( k \)-SCOPE: Top \( k \) strongly correlated Pair Extraction
4.6 Analysis of Our Algorithms

Here, we analyse SCOPE and \( k \)-SCOPE in terms of completeness, correctness and computational complexity.

4.6.1 Completeness and correctness

Lemma 4.6.1. SCOPE is complete, i.e., SCOPE finds all strongly correlated pairs.

Proof. Since SCOPE is based on exhaustive search and computes correlation coefficients of all pairs without pruning any item pair, SCOPE extracts all strongly correlated item pairs with coefficient greater than the threshold \( \theta \). This fact ensures that SCOPE is complete in all respects.

Lemma 4.6.2. SCOPE is correct, i.e., the correlation coefficient of all pairs, extracted by SCOPE, is above threshold \( \theta \).

Proof. The correctness of SCOPE can be guaranteed by the fact that SCOPE calculates exact correlation of each pair present in the database and prunes all pairs whose correlation coefficient is lower than the user specified threshold \( \theta \).

Lemma 4.6.3. \( k \)-SCOPE is complete, i.e., \( k \)-SCOPE finds top-\( k \) strongly correlated pairs.

Proof. Like SCOPE, \( k \)-SCOPE is based on exhaustive search and computes the correlation coefficient of all pairs without pruning any item pairs. Therefore, \( k \)-SCOPE extracts \( k \) top most strongly correlated item pairs based on the value \( \phi \). This fact ensures that \( k \)-SCOPE is complete in all respects.

Lemma 4.6.4. \( k \)-SCOPE is correct, i.e., correlation coefficients of the extracted pairs are the \( k \) top most correlation coefficients.

Proof. The correctness of \( k \)-SCOPE can be guaranteed by the fact that, \( k \)-SCOPE calculates exact correlation of each pair present in the database and creates a sorted list (descending order) of item pairs based on the correlation coefficient and prunes all pairs whose correlation coefficient is lower than the \( k \)th pair's correlation coefficient.
4.6.2 Complexity analysis

Since k-SCOPE is an extension of SCOPE, we analyze only k-SCOPE in terms of space and time complexity.

4.6.2.1 Space complexity

TAPER and TOP-COP need memory for keeping the top-k list and support count of all items, and a huge number of candidate item pairs depending on the value of the \( \theta \) upper bound. TOP-COP maintains a list with the pruning status of all item pairs out of \( N \) items, requiring memory space of order \( (N^2) \). Tkcp creates an entire FP-tree in the memory with initial support threshold zero (0). This tree is normally huge when the number of transactions as well as the dimensions are large. Its size also depends on the number of unique patterns of items in the database. Sometimes it is difficult to construct such a tree in the memory. However, k-SCOPE always requires a fixed memory of size, \( N \times (N + 1)/2 \) to construct the correlogram matrix and array of size \( k \) to store top-\( k \) strongly correlated item pairs. Thus, the total space requirement is:

\[
SPACE_{k-SCOPE} = O(N \times (N + 1)/2) + O(k) \\
\approx O(N^2) + O(k).
\]

4.6.2.2 Time complexity

The computational cost for k-SCOPE consists of two parts: (i) correlogram matrix construction cost \( (C_{CM}) \) and (ii) the cost for extraction of top-\( k \) strongly correlated item pairs \( (C_{EX}) \).

a) Construction of correlogram matrix: Cost can be calculated as describe in section 3.5.2.2 of chapter 3.

b) Extraction of top-\( k \) strongly correlated item pairs: To calculate the correlation
of each pair, \(k\)-SCOPE must traverse the correlogram matrix once. Thus, the time requirement for extracting the correlation coefficient of all item pairs is \(O(N^2)\).

To create the top-\(k\) list, for each item pair the algorithm compares the correlation coefficient (\(Corr\)) of the new pair with \((k - 1)^{th}\) pair in the list. If \(Corr\) of the new pair is greater than that of the \(k^{th}\) pair, the \(k^{th}\) pair is eliminated from the list and a new pair is inserted and placed in the list in descending order of \(Corr\). Thus, for placing a new pair, it requires at most \(k\) number of comparison and swapping. Since, the problem is to find \(k\) top most item pairs out of \(N * (N - 1)/2\) item pairs, the time requirement for creating list of top \(k\) item pairs can be denoted as:

\[
\begin{align*}
C_{EX} & = O(N^2) + O(k * (N * (N - 1))/2) \\
& \approx O(N^2) + O(k * N^2) \\
& \approx O(k * N^2).
\end{align*}
\]

Thus, in the worst case total cost incurred by \(k\)-SCOPE is:

\[
\begin{align*}
COST_{k-SCOPE} & = C_{CM} + C_{EX} \\
& = O(T * N^2) + O(N^2) + O(k * (N * (N - 1))/2) \\
& \approx O(T * N^2) + O(N^2) + O(k * N^2).
\end{align*}
\]

The computational cost of the TOP-COP and TAPER algorithms are almost similar, except that the cost of computing the exact correlations for remaining candidates may be less in the case of TOP-COP, as it prunes more non-potential item pairs based on the 2-D monotone property. The cost of TOP-COP can be modeled as,

\[
COST_{TOP-COP} = C_{Sort} + C_{Bound} + C_{Exact} + C_{k-list}
\]

where \(C_{Sort}, C_{Bound}, C_{Exact}\) and \(C_{k-list}\) are the costs of creating a sorted list of items in non-increasing order of support, the cost of computing upper bounds, computing the cost of exact correlation of remaining pairs, and \(k\)-top list maintenance cost,
respectively. After simplifying the above cost computation, we get

\[ \text{COST}_{\text{TOP-COP}} = O(N \log N) + O(N^2) + O(N^2) + O(k^2). \]

However, this cost model, does not consider the cost of scanning the database. It requires one scan for creating the initial sorted item list and at least another whole scan (when any hash based data structure is used) of the database for computing exact correlation of existing pairs after pruning. After adding this cost, the total becomes

\[ \text{COST}_{\text{TOP-COP}} = O(T \ast N) + O(N \log N) + O(T \ast N) + O(N^2) + O(N^2) + O(k^2) \]

\[ \approx 2 \ast O(T \ast N) + O(N \log N) + 2 \ast O(N^2) + O(k^2). \]

Similarly the cost of Tkcp algorithm can be modeled as:

\[ \text{COST}_{\text{Tkcp}} = C_{\text{Sort}} + C_{\text{D.Sort}} + C_{\text{FP}} + C_{\text{k-list}} \]

\[ = (O(T \ast N) + O(N \log N)) + O(T \ast N^2)(O(T \ast N) + C_{\text{FP.Tree}}) \]

\[ + \ (O(N) \ast C_{\text{Cond.base}} + O(P \ast k^2)) \]

where \( C_{\text{Sort}} \) is the cost of creating the initial sorted list of items based on support count using one pass of the database, \( C_{\text{D.Sort}} \) is the cost incurred during sorting the database in descending order of item support, and \( C_{\text{FP}} \) is the total cost of creating the FP-tree. The creation of the complete FP-tree requires one complete scan over the database and the cost of creating the pattern tree is \( C_{\text{FP.Tree}} \). To compute the correlation of each pair and to maintain the \( k \)-top list, it requires additional cost \( C_{\text{Cond.base}} \) for creating the conditional pattern base (P) for each item. We see that the cost of scanning a database is much larger than the other computational parameters. So, the computational savings of \( k \)-SCOPE, i.e., \( O(T \ast N^2) \) is larger when the number of records in a transaction database is very high.
4.7 Performance Evaluation

To evaluate the performance of SCOPE and k-SCOPE and to compare them with other techniques, we test them using several synthetic as well as real-life datasets. Since, TAPER in its original form cannot generate the top-k list, we modified TAPER, so that it can generate such a top-k strongly correlated item pair list. As TAPER is dependent on the correlation threshold $\theta$, in order to generate the same result using TAPER we set $\theta$ as the correlation coefficient of the $k$-th pair from the top-k list generated by k-SCOPE. The ideal $\theta$ value for TAPER for different datasets are presented in Table 4.7.1. We also provide results showing the performance of Spearman’s $\rho$ as correlation coefficient, compared to Pearson’s $\phi$ when used with market basket data.

We implemented our techniques using Java 1.6 on Windows 7 platform running in 2.53 GHz machine. We used same environment for implementation of SCOPE, k-SCOPE, TAPER and the modified version of TAPER. For TOP-COP, we used the code as provided by the original author. Since performance of Tcp and Tkcp is highly dependent on FP-tree implementation, we use a third party FP-tree implementation from\(^5\) for Tcp and Tkcp to avoid any implementation bias.

4.7.1 Dataset used

To generate synthetic dataset, we used ARMiner\(^a\) software and generate several synthetic datasets. The details of the synthetic dataset is given in Table 4.2. We also used market basket version of three real datasets, Mushroom, Pumsb and Chess. Mushroom dataset\(^b\) is taken from FIMI\(^c\), the Pumsb\(^d\) dataset from IBM, corresponding to a binarized versions of a census dataset. Pumsb is often used as the benchmark for evaluating the performance of association mining algorithms on dense datasets. The details of real transaction datasets are given in Table 4.3.

\(^a\)http://www.cs.umb.edu/laur/ARMiner/
\(^b\)http://www.ics.uci.edu/rilearn/MLRRepository.html
\(^c\)http://fimi.ua.ac.be
\(^d\)http://fimi.cs.helsinki.fi/data/
Table 4.2: Synthetic Transaction Dataset

<table>
<thead>
<tr>
<th>Data Set</th>
<th>No.of Transactions</th>
<th>No. of Items</th>
<th>Avg. size of Transaction</th>
<th>No. of Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>T10I400D100K</td>
<td>100,000</td>
<td>400</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>T10I600D100K</td>
<td>100,000</td>
<td>600</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>T10I800D100K</td>
<td>100,000</td>
<td>800</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>T10I1000D100K</td>
<td>100,000</td>
<td>1000</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>T10P1000D100K</td>
<td>100,000</td>
<td>1000</td>
<td>10</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 4.3: Real Dataset

<table>
<thead>
<tr>
<th>Data Set (Market Basket)</th>
<th>No.of Transactions</th>
<th>No. of Items</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mushroom</td>
<td>8124</td>
<td>128</td>
<td><a href="http://fimi.ua.ac.be">http://fimi.ua.ac.be</a></td>
</tr>
<tr>
<td>Pumsb</td>
<td>49046</td>
<td>2113</td>
<td><a href="http://fimi.cs.helsinki.fi">http://fimi.cs.helsinki.fi</a></td>
</tr>
<tr>
<td>Chess</td>
<td>3196</td>
<td>75</td>
<td><a href="http://fimi.ua.ac.be">http://fimi.ua.ac.be</a></td>
</tr>
</tbody>
</table>

4.7.2 Experimental results

To evaluate the performance of the proposed algorithms, we compare them with other similar techniques in terms of execution time for different values of \( \theta \) and \( k \). We find that Tcp and Tkcp consume a lot more time compared to other two techniques, since Tcp and Tkcp generate the entire FP-tree with the initial minimum support value of 0. We also observe that Tcp and Tkcp do not work when the number of items is more than 1000. In case of T10P1000D100K dataset, both Tcp and Tkcp failed to mine, due to the large number of items and unique patterns. However, in all cases, SCOPE exhibits better performance than TAPER and Tcp. With decrease in the value of \( \theta \), the running time of TAPER also increases, since low \( \theta \) value generates a large number of candidate sets. But, SCOPE and Tcp keep stable running time for the whole range of correlation thresholds in different datasets. We further confirm the fact that like Tcp, SCOPE is also robust with respect to input parameters (Figure 4.3).

From the performance graph in Figures 4.3 and 4.4, we easily observe that modified TAPER performs much better than TOP-COP, even though TOP-COP is an improved and modified version of TAPER. It is because of the use of an efficient hash
Table 4.4: Suitable \( \theta \) value for different datasets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mushroom</td>
<td>0.49</td>
<td>0.37</td>
<td>0.31</td>
<td>0.25</td>
<td>0.23</td>
</tr>
<tr>
<td>Pumsb</td>
<td>0.97</td>
<td>0.869</td>
<td>0.764</td>
<td>0.703</td>
<td>0.647</td>
</tr>
<tr>
<td>T10I400D100K</td>
<td>0.51</td>
<td>0.027</td>
<td>-0.006</td>
<td>-0.011</td>
<td>-0.016</td>
</tr>
<tr>
<td>T10I600D100K</td>
<td>0.81</td>
<td>0.27</td>
<td>0.001</td>
<td>-0.006</td>
<td>-0.009</td>
</tr>
<tr>
<td>T10I800D100K</td>
<td>0.63</td>
<td>0.290</td>
<td>0.001</td>
<td>-0.003</td>
<td>-0.005</td>
</tr>
<tr>
<td>T10I1000D100K</td>
<td>0.96</td>
<td>0.95</td>
<td>0.94</td>
<td>0.93</td>
<td>0.89</td>
</tr>
<tr>
<td>T10P1000D100K</td>
<td>0.95</td>
<td>0.92</td>
<td>0.87</td>
<td>0.83</td>
<td>0.80</td>
</tr>
</tbody>
</table>

data structure, which is lacking in the original TOP-COP implementation. This further indicates that the performance of correlation mining algorithms can be improved through efficient implementation. However, in all cases, \( k \)-SCOPE exhibits better performance than TAPER (modified), TOP-COP and Tkcp. TOP-COP exhibits an exponential performance graph (Figures 4.3 and 4.4) as the number of items increases. But \( k \)-SCOPE and Tkcp maintain stable running time in different datasets, since both algorithms are independent of \( \theta \). It further confirms the fact that SCOPE and \( k \)-SCOPE are robust with respect to input parameters \( \theta \) and \( k \).

4.7.2.1 Scalability of \( k \)-SCOPE

The scalability of the \( k \)-SCOPE algorithm with respect to the number of transactions and number of items in the databases is shown in Figure 4.6. We used ARMMiner to generate four datasets with the number of transactions ranging from 1,00,000 to 5,00,000. In each case, we kept the number of test items at 1,000 as we tested for scalability in terms of number of transactions. To test scalability in terms of number of items, we generated another five transaction datasets with numbers of items ranging from 2,000 to 10,000 keeping number of transactions equal to 1,00,000. We observe the execution time increases linearly with increase in the number of transactions and items at different \( k \) values. Figure 4.6 shows the scalability test results for \( k \) ranging from 500 and 2500. From the graph, it is clear
Figure 4.3: Execution time comparison between SCOPE, Tcp and TAPER
Figure 4.4: Execution time comparison of $k$-SCOPE with TAPER (mod), Tkcp and TOPCOP on Synthetic dataset.
that the performance of $k$-SCOPE is not sensitive to input parameter $k$. Thus, $k$-SCOPE is robust in handling large transaction databases for different values of $k$.

4.7.2.2 Pearson’s $\phi$ vs. Spearman’s $\rho$ in correlated item pair findings

Now we provide a few results to establish that Spearman’s $\rho$ is superior in comparison to Pearson’s $\phi$ over market basket dataset in terms of (i) finding number of correlated item pairs and (ii) correlation coefficient values for different $k$ values.

For measuring the performance of $\phi$ and $\rho$ as correlation coefficient for finding correlated item pairs, we used the Mushroom and Chess datasets. We measured the number of possible correlated item pairs for various $\theta$ values, generated by both Pearson’s $\phi$ and Spearman’s $\rho$. In the Figure 4.7 and 4.8, we easily observe that Spearman’s $\rho$ generates more correlated pairs compared to Pearson’s $\phi$. Similarly, when we measure $k^{th}$ correlation coefficient value for different $k$ values, we find that Spearman’s $\rho$ gives higher values than Pearson’s $\phi$. We conclude that Spearman’s $\rho$ is able to detect hidden correlated pairs undetected by Pearson’s $\phi$. Moreover, in some cases, $\rho$ gives higher correlation value compared to $\phi$, for a particular pair. We feel that this is due to the problems associated with Pearson’s $\phi$, as already discussed.
Figure 4.6: Scalability of $k$-SCOPE algorithm.

Figure 4.7: Performance comparison of Pearson's $\phi$ and Spearman's $\rho$ on Mushroom dataset

4.8 Discussion

We have presented two effective techniques for finding strongly correlated item pairs and top-$k$ strongly correlated item pairs from market basket data, in this chapter. We have also presented an alternative way of measuring correlation coefficient between item pairs from market basket data using Spearman's rank order correlation. The advantages of these techniques compared to existing similar techniques are (i)
they require single pass over the whole database, and (ii) they require no candidate generation. We evaluated both the techniques using several synthetic and real life datasets and found that the results are quite satisfactory.

In the following chapters, we present potential application of above data mining techniques in gene expression data analysis. Next chapter presents a pattern based gene co-expression network finding technique using correlogram matrix.