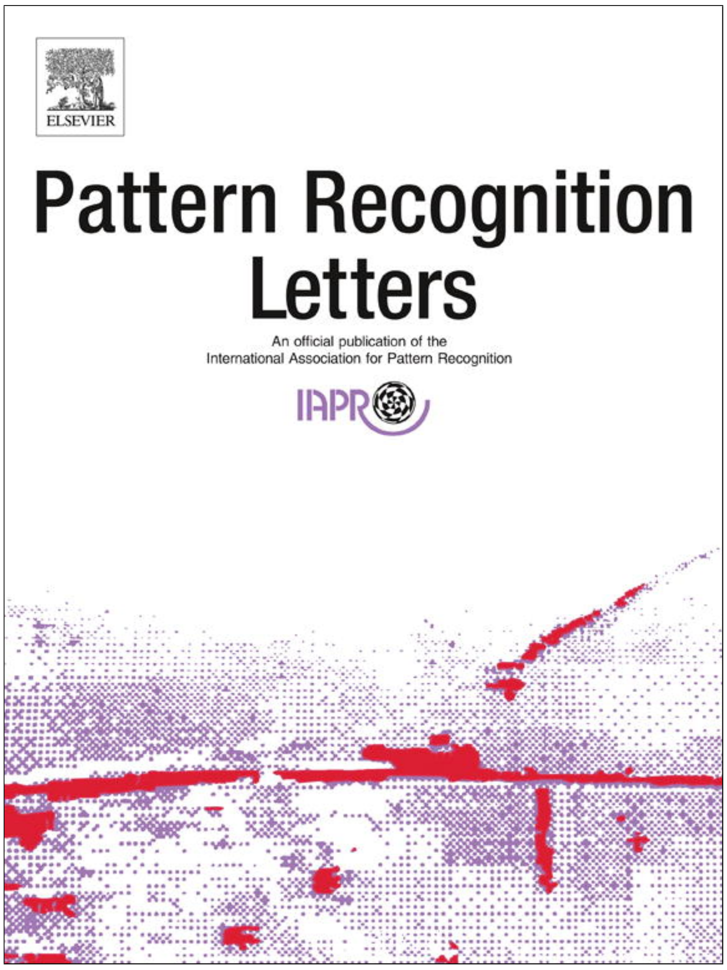
ELSEVIER
Pattern Recognition Letters
An official publication of the International Association for Pattern Recognition
IAPR

Pattern Recognition Letters 34 (2013) 1499–1507

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

# Speeding up correlation search for binary data

Lian Duan [a,*], W. Nick Street [b], Yanchi Liu [a]

[a] Department of Information Systems, New Jersey Institute of Technology, United States
[b] Department of Management Sciences, The University of Iowa, United States

ARTICLE INFO

*Article history:*
Available online 11 June 2013

Communicated by A. Petrosino

*Keywords:*
Correlation search
Correlation upper bound
Token-ring algorithm

ABSTRACT

Searching correlated pairs in a collection of items is essential for many problems in commercial, medical, and scientific domains. Recently, a lot of progress has been made to speed up the search for pairs that have a high Pearson correlation ($\phi$-coefficient). However, $\phi$-coefficient is not the only or the best correlation measure. In this paper, we aim at an alternative task: finding correlated pairs of any "good" correlation measure which satisfies the three widely-accepted correlation properties in Section 2.1. In this paper, we identify a 1-dimensional monotone property of the upper bound of any "good" correlation measure, and different 2-dimensional monotone properties for different types of correlation measures. We can either use the 2-dimensional search algorithm to retrieve correlated pairs above a certain threshold, or our new token-ring algorithm to find top-$k$ correlated pairs to prune many pairs without computing their correlations. The experimental results show that our robust algorithm can efficiently search correlated pairs under different situations and is an order of magnitude faster than the brute-force method.

## 1. Introduction and related work

It is very important to analyze the relationship among items for many data mining problems. For example, association analysis (Agrawal et al., 1993) is to discover a set of binary variables (called items) that co-occur frequently in a transaction database. Correlation search is useful for sales promotions (Lin et al., 2011), website catalog design, store shelf layout, and adverse drug effect detection (Duan et al., 2013). No matter how the relationships are defined, we need a proper measure to evaluate the patterns. Support and confidence (Agrawal et al., 1993) are often used to represent the significance. However, these can produce misleading results because of the lack of comparison to the expected probability under the assumption of independence. To overcome this, many correlation measures (Tan et al., 2004; Geng and Hamilton, 2006; Duan and Street, 2012) have been proposed and studied.

However, as the number of items and transactions in the dataset increases, calculating all the pair correlation values is computationally expensive. Since there is no monotone property for most of correlation measures to help prune the search space, the brute-force method is straightforward. When we conducted a blank search of books in Amazon.com on Jan. 5, 2013, there were 21,735,144 paperback books available. In order to find the correlated pairs of paperback books, a brute-force approach requires computing the correlation value of $2.36 * 10^{14}$ pairs. Even worse, when the co-occurrence of all the pairs cannot be loaded into the memory, the IO cost for retrieving co-occurrences is much more expensive than the computational cost for calculating correlations.

The most significant progress on correlated pair search was made by Xiong et al. (2008, 2006). He made use of the upper bound of the Pearson correlation coefficient ($\phi$-coefficient) for binary variables. The computation of this upper bound is much cheaper than the computation of the exact correlation because this upper bound is a function of single item supports. In addition, the upper bound has special 1-dimensional and 2-dimensional properties that prune many pairs from the search space without the need to compute their actual upper bounds. The algorithm TAPER (Xiong et al., 2006) makes use of the 1-dimensional and 2-dimensional properties to retrieve correlated pairs above a given threshold. The algorithm TOP-COP (Xiong et al., 2008) uses the 1-dimensional property and a diagonal traversal method, combined with a refine-and-filter strategy, to efficiently find the top-$k$ pairs. However, this work is only related to the $\phi$-coefficient, which is not the only or the best correlation measure. We extend the ideas of the upper bound, 1-dimensional and 2-dimensional properties for $\phi$-coefficient to any "good" correlation measures.

Given a pair $\{I_1, I_2\}$ in a dataset, the actual probability is $tp = P(I_1 \cap I_2)$, the expected probability under the assumption of independence is $ep = P(I_1) \cdot P(I_2)$. Since itemset mining has a long history from 1993 when frequent itemset mining Agrawal et al. (1993) was proposed, a lot of related concepts are proposed such as closed itemset (Yahia et al., 2006), contrast itemset (Aggarwal

* Corresponding author. Tel.: +1 973 596 5481; fax: +1 973 596 2986.
*E-mail addresses:* lian.duan@njit.edu (L. Duan), nick-street@uiowa.edu (W.N. Street), yl473@njit.edu (Y. Liu).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.05.027

et al., 2006), and discriminative itemset (Cheng et al., 2007). These concepts are related to each other in a certain degree; however, in this paper, we only focus on the performance issue and the general framework of finding correlated pairs where the correlation measure must explicitly have the comparison to the assumption of independence. In addition, there are several other issues related to correlation that we also do not address here, such as effectiveness of correlation measures, statistical type-1 and type-2 error reduction, error-tolerant methods using sampling, and high dimensional correlation search. In Duan and Street (2012), we carefully studied 19 correlation measures and provided four extra properties for correlation measures from the statistical point of view which can help users to choose the correlation measure retrieving results closer to human intuition. Webb (2007) proposed a framework of reducing the type-1 and type-2 error of itemset mining which can be applied to our pattern search framework. Zhang et al. (2006) studied the distribution of the $\phi$-coefficient and relaxed the upper bound in TAPER in order to speed up search. We proposed a maximal fully-correlated itemset (Duan et al., 2009) framework for high dimensional correlation search. Guns et al. (2011) formulated the traditional itemset mining problem, such as frequent, closed, and discriminative itemset mining, as constraint programming problems, and then applied an existing solver for constraint programming to speed up the search.

The rest of this paper is organized as follows. Section 2 presents basic notions of correlation properties, correlation upper bound, 1-dimensional and 2-dimensional properties. We propose several methods to speed up correlated pair search in Section 3. Section 4 shows the experimental results. Finally, we draw a conclusion in Section 5.

## 2. Basic properties

In this section, some basic properties of correlation are introduced to better explain the improved performance of correlation search.

### 2.1. Correlation measure properties

To find highly correlated pairs, we should find a good correlation measure first. All the correlation measures should satisfy the following three properties proposed by Piatetsky-Shapiro (1991).

Given an itemset $S = \{I_1, I_2, \ldots, I_m\}$, a correlation measure $M$ must satisfy (Piatetsky-Shapiro, 1991):

- P1: $M$ is equal to a certain constant number $C$ when all the items in the itemset are statistically independent.
- P2: $M$ monotonically increases with the increase of $P(S)$ when all the $P(I_i)$ remain the same.
- P3: $M$ monotonically decreases with the increase of any $P(I_i)$ when the remaining $P(I_k)$ and $P(S)$ remain unchanged.

The first property requires a fixed reference for independence. The last two properties are based on the following intuition. The correlation value increases when the expected probability stays the same while the actual probability goes up. The correlation value decreases when the expected probability goes up while the actual probability stays the same. Since it is impossible to compare against every possible measure (Geng and Hamilton, 2006), we use the following three commonly accepted properties to generalize different correlation measures, such as $\phi$-coefficient, the simplified $\chi^2$-statistic, probability ratio, leverage, and likelihood ratio (Duan and Street, 2012). For example, leverage is calculated as $P(S) - \prod_{i=1}^{m} P(I_i)$, and it is easy for readers to check its satisfaction for the above three properties. In the following sub-sections, we will make use of the above three properties to infer correlation upper bound properties. Interested readers can find more details related to correlation properties in Duan and Street (2012).

### 2.2. Correlation upper bound for pairs

**Theorem 1.** *Given any pair $\{I_i, I_j\}$ and support values $P(I_i)$ for item $I_i$ and $P(I_j)$ for item $I_j$, the correlation upper bound $CUB(I_i, I_j)$, i.e. the highest possible correlation value of the pair $\{I_i, I_j\}$, is the correlation value for $\{I_i, I_j\}$ when $P(I_i \cap I_j) = min\{P(I_i), P(I_j)\}$.*

**Proof.** The Support of the itemset $S$, $P(S)$ , is the proportion of transactions that contain $S$. $P(I_i \cap I_j)$ must be less or equal to $P(I_i)$. Similarly, we have $P(I_i \cap I_j) \leqslant P(I_j)$. Therefore, the upper bound of the support value $P(I_i \cap I_j)$ for the 2-itemset $\{I_i, I_j\}$ is $min\{P(I_i), P(I_j)\}$. For the given $P(I_i)$ and $P(I_j)$, any correlation measure reaches its upper bound when $P(I_i \cap I_j) = min\{P(I_i), P(I_j)\}$ according to correlation Property 3. □

The calculation of correlation upper bound (CUB) for pairs only needs the support of each item which can be saved in memory even for large datasets; however, the calculation of correlation for pairs needs the support of pairs, incurring a high IO cost for large datasets. Given a 2-itemset $\{I_i, I_j\}$, if its correlation upper bound is lower than the correlation threshold we specify, there is no need to retrieve the support of the 2-itemset $\{I_i, I_j\}$, because the correlation value for this pair is definitely lower than the threshold no matter what the support is. If we only retrieve the support of a given pair in order to calculate its correlation when its upper bound is greater than the threshold, we will save a lot of unnecessary IO cost when the threshold is high.

### 2.3. 1-Dimensional property

Although the correlation upper bound calculation can save a lot of unnecessary IO cost, it still requires the correlation upper bound calculation for all the possible pairs. Therefore, we make use of the 1-dimensional property to eliminate unnecessary upper bound checks. It is motivated by the search algorithm TAPER (Xiong et al., 2006) for the $\phi$-coefficient. If we sort the items according to their supports in decreasing order as in TAPER, different measures have different monotone patterns, which we will discuss in Section 2.4. In order to fit the same decreasing monotone pattern for any good correlation measure, we sort the items according to their supports in increasing order instead of decreasing order.

**Theorem 2.** *Given a user-specified threshold $\theta$ and an item list $\{I_1, I_2, \ldots, I_n\}$ sorted by support in increasing order, the correlation upper bound of $\{I_i, I_k\}$ is less than $\theta$ if the correlation upper bound of $\{I_i, I_j\}$ is less than $\theta$ and $i < j < k$.*

**Proof.** Since $i < j < k$ and the item list $\{I_1, I_2, \ldots, I_m\}$ is sorted by support in increasing order, $P(I_i) \leqslant P(I_j) \leqslant P(I_k)$. Then, the support upper bound of both $\{I_i, I_j\}$ and $\{I_i, I_k\}$ is equal to $P(I_i)$. For the pair $\{I_i, I_j\}$, $P_{upper}(I_i \cap I_j) = P(I_i)$ and $P_{expected}(I_i \cap I_j) = P(I_i)P(I_j)$. For the pair $\{I_i, I_k\}$, $P_{upper}(I_i \cap I_k) = P(I_i)$ and $P_{expected}(I_i \cap I_k) = P(I_i)P(I_k)$. Therefore, $P_{upper}(I_i \cap I_k) = P_{upper}(I_i \cap I_j)$, and $P_{expected}(I_i \cap I_k) \geqslant P_{expected}(I_i \cap I_j)$ because $P(I_k) \geqslant P(I_j)$. According to correlation property 3, we get $CUB(I_i, I_k) \leqslant CUB(I_i, I_j)$. Since $CUB(I_i, I_j) < \theta$, $CUB(I_i, I_k) < \theta$. □

To get all the pair correlation upper bounds, we need to calculate an $n \times n$ matrix for an item list sorted by support as shown in Table 1(a). Since this matrix is symmetrical, we only need to calculate the upper part above the diagonal. If the data set contains $n$ items, $(n - 1)$ branches (rows) need to be calculated. The pairs in

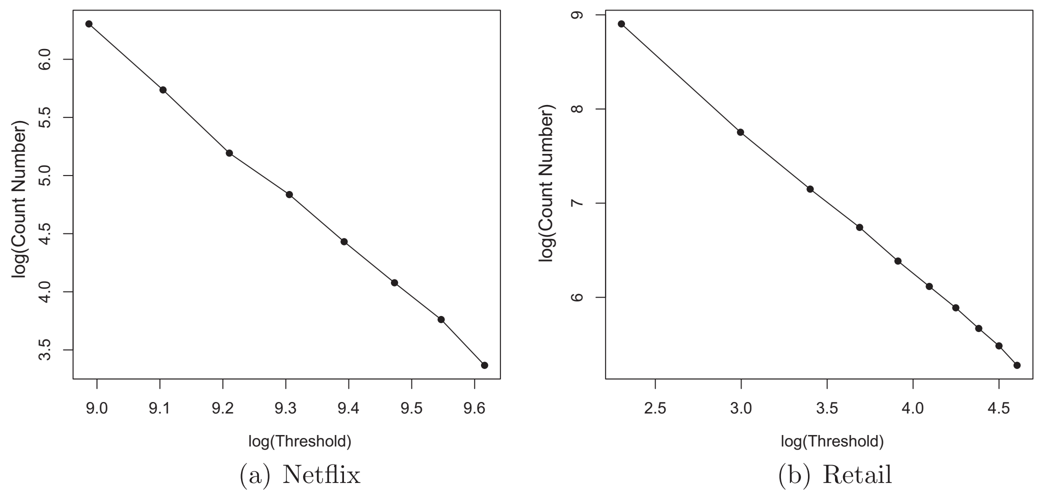

**Fig. 1.** The number of correlated pairs under different likelihood ratio thresholds.

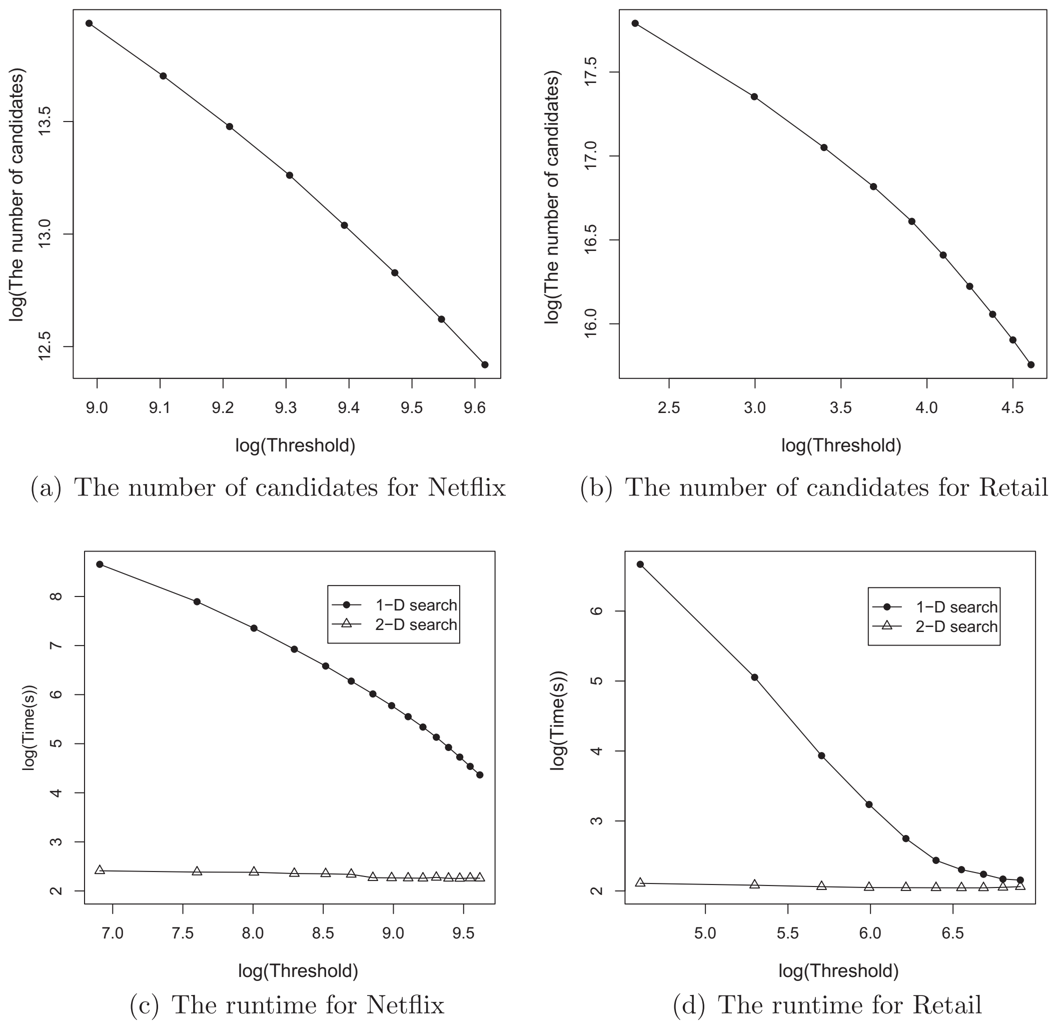

**Fig. 2.** The number of candidates and the runtime under different thresholds.

branch $i$ are $\{I_i, I_j\}$ where $i+1 \leqslant j \leqslant n$. The reference item $I_i$ is fixed in each branch $i$ and it has the minimum support value due to the way we construct the branch. Since items in each branch are also sorted based on their support in increasing order, the correlation upper bound of $\{I_i, I_j\}$ monotonically decreases with the increase of $j$ by Theorem 2. In other words, $CUB(I_i, I_k) < \theta$ when $CUB(I_i, I_j) < \theta$ and $j+1 \leqslant k \leqslant n$.

### 2.4. 2-Dimensional property

When the threshold is low, we might still calculate a lot of correlation upper bounds by using the 1-dimensional property. In order to avoid too many correlation upper bound checks, we try to check the upper bound change in both horizontal and vertical dimensions. However, different correlation measures have different 2-dimensional properties and require different search strategies.

Given three items $I_i,\ I_j$, and $I_k$ with $P(I_i) \geqslant P(I_j) \geqslant P(I_k)$, the correlation measure $M$ is.

- Type 1 if $CUB(I_i, I_j) \geqslant CUB(I_i, I_k)$.
- Type 2 if $CUB(I_i, I_j) \leqslant CUB(I_i, I_k)$.
- Type 3 if $CUB(I_i, I_j) = CUB(I_i, I_k)$.

The simplified $\chi^2$-statistic, $\phi$-coefficient, and likelihood ratio are type 1 correlation measures. Although we know of no existing type 2 correlation measure, we can construct a type 2 correlation measure which satisfies all the three correlation properties like $Correlation_{type2} = \frac{\sqrt{P(I_i \cap I_j) - P(I_i) * P(I_j)}}{P(I_i) * P(I_j)}$. Probability ratio is a type 3 correlation measure. For different types of correlation measures, we get different 2-dimensional properties. For type 1 correlation measures, the upper bound value decreases from left to right and from bottom to top. If the upper bound of the current cell $\{I_i, I_j\}$ is below $\theta$, then $CUB(I_p, I_q) < \theta$ when $1 \leqslant p \leqslant i$ and $j \leqslant q \leqslant n$. For type 2 correlation measures, the upper bound value decreases from left to right and from top to bottom. If the upper bound of the current cell $\{I_i, I_j\}$ is below $\theta$, then $CUB(I_p, I_q) < \theta$ when $i \leqslant p \leqslant n-1$ and $max(p+1, j) \leqslant q \leqslant n$. For type 3 correlation measures, the rightmost column has the lowest upper bound value. If the upper bound of the current cell $\{I_i, I_j\}$ is below $\theta$, then the upper bounds of any cell to its right is below $\theta$.

## 3. Search algorithm

In general, there are two types of correlated pair searches: (1) correlated pairs above a certain threshold, and (2) top-$k$ correlated pairs. Although the correlation itself does not have a monotone property to help prune, the correlation upper bound does. Consequently, we make use of the 1-dimensional monotone property of the upper bound of any good correlation measure, and different 2-dimensional monotone properties for different types of correlation measures. We can either use the 2-dimensional search algorithm to retrieve correlated pairs above a certain threshold, or our new token-ring algorithm to find the top-$k$ correlated pairs, to prune many pairs without the need to compute their correlations.

### 3.1. Correlated pairs above a certain threshold

In this subsection, we focus on task 1: finding correlated pairs above a certain threshold. The easiest way of speeding up search is calculating the upper bound before the real correlation. If the upper bound is already below the threshold, there is no need to retrieve the pair support to calculate the real correlation. Therefore, we greatly reduce IO cost for high thresholds. Upper bound checking needs to calculate the upper bound of all the possible $n \cdot (n-1)/2$ pairs which is still computationally expensive. The 1-dimensional property can be used to save a lot of unnecessary upper bound checks for very high thresholds. We specify the reference item $A$ and start a search within each branch. The reference item $A$ is fixed in each branch and it has the minimum support value due to the way we construct the branch. Items in each branch are also sorted based on their support in increasing order. By Theorem 2, the correlation upper bound of $\{A, B\}$ monotonically decreases with the increase of the support of item $B$. Therefore, if we find the first item $B$, the turning point, which results in an correlation upper bound less than the user-specified threshold, we can stop the search for the current branch. If the upper bound is greater than the user-specified threshold, we calculate the exact correlation and check whether this pair is really satisfied. Furthermore, the 2-dimensional property can be used to prune cells faster. The key difference between 1-dimensional search and 2-dimensional search is that the 2-dimensional search algorithm records the turning point in the previous branch and starts computing from that point instead of the beginning of the current branch. But we will use different search sequences for three different types of correlation measures. For type 1 correlation measures, we start the search from the upper-left corner. If the upper bound of the current cell is above the threshold $\theta$, we will check the cell to the right of it; else we will instead check the cell under it. For type 2 correlation measures, we start the search from the upper-right corner. If the upper bound of the current cell is greater than $\theta$, we check the cell below the current cell; else, we check the cell to the left of the current cell. For type 3 correlation measures, the rightmost column has the lowest upper bound value. We only need to search the first branch. If the current column is above the threshold, we continue the search of the right-side column until the current column is below the threshold.

**Theorem 3.** *The number of upper bound calculations in the 2-dimensional search is between $n-1$ and $2n-3$ for the first type, $n-1$ for the second type, and between 1 and $n-1$ for the third type.*

**Proof.** For the 2-dimensional search in type 1, the number of calculations is determined by the cell where we stop on the right most column. If the algorithm stops at the upper-right corner, the whole search moves from left to right $n-1$ times. If the algorithm stops at the lower-right corner, the whole search moves from left to right $n-1$ times and from up to down $n-2$ times, so there are $n-1+n-2 = 2n-3$ movements. Since the search might stop at any cell on the most right column, the calculation for type 1 is between $n-1$ and $2n-3$.

For the second type, the search starts at the upper-right corner and stops at one of the border cells along the diagonal, that is, cell $C_{i,i+1}$ where $i = 1, 2, \ldots, n-1$. From the upper-right corner cell $C_{1,n}$ to the cell $C_{i,i+1}$, we have to make $i$ movements from up to down and $n-i-1$ movements from right to left. In all, we need to make $i + n - i - 1 = n - 1$ movements no matter in which cell we stop the search. For the third type, we stop the search between the calculation for the first column and that of the last column, so the number of calculation is between 1 and $n-1$. □

Different methods of facilitating correlated pair search above a certain threshold are discussed in this section. If we do not make use of correlation upper bound at all, we need to calculate the correlation for all the possible pairs to find correlated pairs above a threshold. This brute-force method will have $n(n-1)/2$ support IO cost and $n(n-1)/2$ correlation calculations. Here, we call a pair a candidate if its correlation upper bound is greater than the user-specified threshold $\theta$. For a given dataset and $\theta$, the number of candidates, $\alpha$, is fixed. For each candidate, we have to calculate its true correlation to determine whether its correlation is above the threshold or not. No matter which method we use, the IO cost of retrieving support and the computing cost of correlation for $\alpha$ candidates are inevitable. The only difference is the number of correlation upper bound checks. If it is just upper bound calculation, $n(n-1)/2$ upper bounds need to be calculated. If it is 1-dimensional search, $\alpha$ plus an extra $\beta$ upper bounds need to be calculated where $\beta$ is between 0 and $n-1$ depending on in how many branches we check all the cells. If it is 2-dimensional search, $\gamma$ upper bounds need to be calculated which is between $n-1$ and $2n-3$ for type 1, $n-1$ for type 2, and between 1 and $n-1$ for type 3.

Here, we further study the difference between 1-dimensional and 2-dimensional search. If we want to find correlated pairs above a given threshold, $\alpha$ IO cost and correlation calculation is inevitable because each candidate must be checked. The difference is that 1-dimensional has $\alpha + \beta$ upper bound calculations and 2-dimensional has $\gamma$ upper bound calculations. Since $\alpha$ is much greater than $\beta$ and $\gamma$, and IO is much more expensive than calculation, the $\alpha$ IO cost dominates the computational cost. There is no significant difference between 1-dimensional and 2-dimensional search with regard to finding pairs above a threshold. However, it is hard to determine which threshold is proper for a given dataset. Normally, this threshold is determined by intuition. But we can use the number of candidates for the given threshold to do the evaluation. For example, we may know how much time we spend on retrieving support and calculating correlation for a pair and how long we are allowed to search. We can estimate how many candidates we can afford to search. When just finding the number of candidates, we can avoid the $\alpha$ support IO cost and $\alpha$ correlation calculation. In this case, 2-dimensional search can find the number of candidates in linear time which is much better than 1-dimensional search.

### 3.2. Top-k correlated pairs

Although 2-dimensional search can efficiently find correlated pairs above the threshold $\theta$, it is hard for users to provide a proper correlation threshold in many applications. Instead, finding top-$k$ correlated pairs is more meaningful for users.

#### 3.2.1. TOP-COP search

The original TOP-COP algorithm (Xiong et al., 2008) introduced a diagonal traversal method for efficiently searching the top-$k$ correlated pairs of $\phi$-coefficient. While it exploits the 2-dimensional monotone property of the upper bound of the $\phi$-coefficient, it needs $O(n^2)$ space to save pair status indicating whether or not the pair needs to be checked. Saving pair status only takes 3% of the space of saving support, but it is still not feasible for large datasets. If items are sorted by their support in increasing order, the upper bound of pairs for any good correlation measure decreases from left to right for each row. The search starts from the diagonal consisting of $\{I_i, I_{i+1}\}$ which is the closest to the main diagonal, then goes to the diagonal consisting of $\{I_i, I_{i+2}\}$ which is the second closest to the main diagonal, and so on. During the iterative search process, this method maintains a top-$k$ list and a pair is pushed into this list if its correlation coefficient is greater than the current minimum correlation coefficient in the top-$k$ list. The search stops if the maximal upper bound of all the pairs in a diagonal is less than the current minimum correlation coefficient in the top-$k$ list.

#### 3.2.2. Token-ring search

TOP-COP calculates all the pairs in a diagonal to find the maximal upper bound. In fact, the upper bound calculation of some pairs in the diagonal can be avoided if the upper bound of their left pair is already less than the current minimum correlation coefficient $\tau$ in the top-$k$ list. In order to achieve that, we propose a token-ring search algorithm, shown in Algorithm 1. We treat all the $n-1$ branches as nodes in the token-ring. Only the branch that gets the token can calculate the upper bound of the leftmost uncalculated pair in this branch and compare its upper bound against $\tau$. If the upper bound is above $\tau$, we will check the correlation value of this pair. This pair will be pushed into this list if its correlation coefficient is greater than $\tau$. If the upper bound is below $\tau$, this branch will be removed from the token-ring because the upper bounds of the uncalculated pairs in this branch must be less than $\tau$ according to the 1-dimensional property. In addition, a branch will also be removed from the token-ring if all the pairs in this branch are calculated. The algorithm will stop when there is no branch in the token-ring. According to the way token-ring search works, the upper bound of all the pruned pairs must be less than the current minimum correlation coefficient in the top-$k$ list, so the current top-$k$ list contains the pairs with the $k$ highest correlation values in the data set. The token-ring algorithm only saves the status of $n-1$ branches while TOP-COP needs $O(n^2)$ space to save pair status.

**Algorithm 1.** Token-ring search

```
Main: TokenRing (SortList, k)
 for $i = 0;\ i < SortList.length - 1;\ i++$ do
   BranchInformation[0]=i; //save the id of the branch
 that have potential candidates
   BranchInformation[1]=i + 1; //save the leftmost
 uncalculated pair id in this branch
   LivingBranch.add (BranchInformation);
 end for
 for $i = 0;\ i < SortList.length - 1;\ i++$ do
   //We at most search the whole LivingBranch $(n-1)$ times
   CurrentCorrelationThreshold = 0;
   for $j = 0;\ j < LivingBranch.length;\ j++$ do
     Set the CurrentPair as the leftmost uncalculated
 pair in LivingBranch.get (j)
     CurrentCorrelationUpperbound =
 getPairCorrelationUpperbound (CurrentPair)
     Change the leftmost uncalculated pair id in
 LivingBranch.get (j)
     if CurrentCorrelationUpperbound>Current
 CorrelationThreshold then
       if LivingBranch.get (j) has no uncalculated pair then
           LivingBranch.remove (j);
       end if
     CurrentCorrelation = getPairCorrelation (CurrentPair);
     if CurrentCorrelation>CurrentCorrelationThreshold
 then
       Push CurrentPair into the top-k array;
       kth-correlation = the correlation of the kth pair
 in the current top-k array;
       if kth-correlation>CurrentCorrelationThreshold then
       CurrentCorrelationThreshold = kth-correlation;
     end if
   end if
   else
     LivingBranch.remove (j);
     j–;
   end if
   end for
   if LivingBranch.isEmpty () then
   break; //No branch has the potential candidates.
   end if
       end for
```

We coined a data set with 6 items. The correlation upper bounds and correlations of all the pairs are shown in Table 1(a) and tab:cubp(b) respectively. An example of the token-ring search for the top-5 pairs in this data set is shown as follows. After traveling the first diagonal from $\{I_1, I_2\}$ to $\{I_5, I_6\}$, all the five pairs are pushed into the top-5 list, and the current minimum correlation coefficient in the top-5 list is 15. Since the current $\tau = 15$ is less than the maximal upper bound in the current diagonal 49, we continue the search. When checking $\{I_1, I_3\}$, the upper bound 14

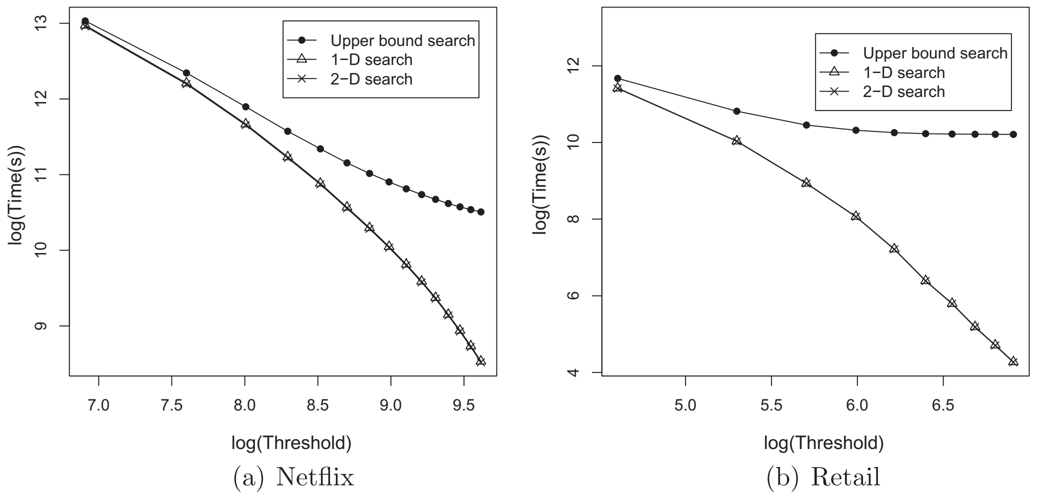

**Fig. 3.** The runtime for retrieving the satisfied pairs.

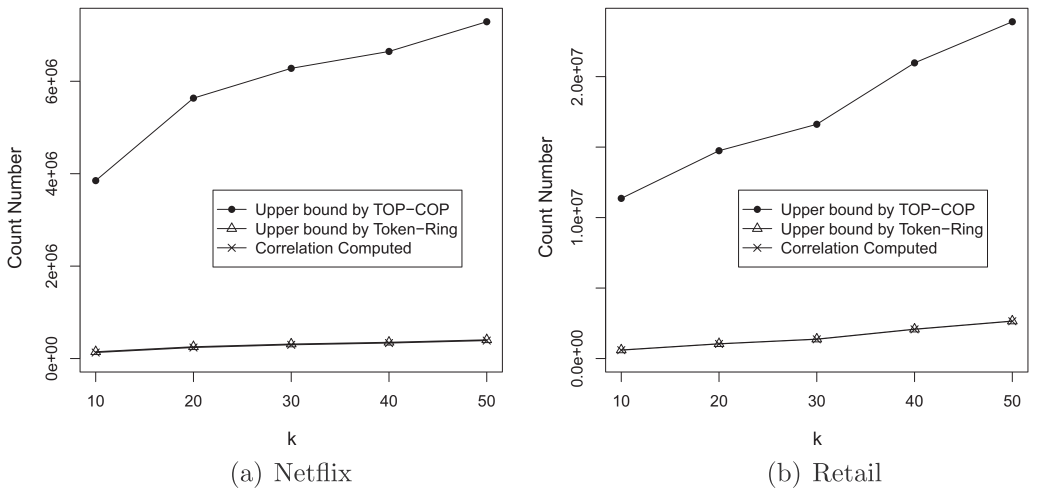

**Fig. 4.** The number of correlation and correlation upper bound checks.

is less than the current minimum correlation coefficient in the top-5 list 15, so the first branch quits the token-ring. The upper bound of $\{I_2, I_4\}$ is 23 which is greater than 15, but its correlation is less than 15. Therefore, the second branch stays in the token-ring and the current top-5 list is not changed. A similar thing happens to $\{I_3, I_5\}$. After checking $\{I_4, I_6\}$, the current minimum correlation coefficient in the top-5 list is 20 and the maximal upper bound in the current diagonal is 31. Therefore, we continue the search in the third diagonal. Branch 1 is already pruned, so $\{I_1, I_4\}$ would not be checked. We only check $\{I_2, I_5\}$ and $\{I_3, I_6\}$. After traveling the third diagonal, the current minimum correlation coefficient in the top-5 list is 20 and the maximal upper bound in the current diagonal is 19. Finally, we stop the search at cell $\{I_3, I_6\}$.

**Table 1**
A coined example.

| | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ |
|---|---|---|---|---|---|---|
| *(a) Correlation upper bound* | | | | | | |
| $I_1$ | | 15 | 14 | 13 | 8 | 5 |
| $I_2$ | | | 36 | 23 | 14 | 9 |
| $I_3$ | | | | 48 | 31 | 19 |
| $I_4$ | | | | | 49 | 31 |
| $I_5$ | | | | | | 45 |
| $I_6$ | | | | | | |
| *(b) Correlation* | | | | | | |
| $I_1$ | | 15 | 14 | 3 | −2 | 1 |
| $I_2$ | | | 36 | 3 | 12 | 0 |
| $I_3$ | | | | 48 | 2 | 10 |
| $I_4$ | | | | | 36 | 29 |
| $I_5$ | | | | | | 20 |
| $I_6$ | | | | | | |

**Theorem 4.** *Both TOP-COP and token-ring calculate the same number of correlations. The only performance difference between TOP-COP and token-ring is that token-ring reduces the number of correlation upper bound computations.*

**Proof.** The correlation upper bound check for the cell $\{I_i, I_{i+j}\}$ in the $i$th branch and the $j$th diagonal line can be avoided if the correlation upper bound of the cell $\{I_i, I_{i+j-1}\}$ in the $i$th branch and the $(j-1)$th diagonal line is below the minimum correlation coefficient in the top-$k$ list at that time. Therefore, token-ring avoids the upper bound checks of TOP-COP for those pairs that cannot affect the current top-$k$ list, but the current top-$k$ list changes at the same cells for both token-ring and TOP-COP. In all, both TOP-COP and token-ring calculate the same number of correlations, but they check a different number of correlation upper bounds.

**Theorem 5.** *For the token-ring algorithm, the difference between the number of computed correlation upper bounds and the number of computed correlations is no more than* $n - 1$.

**Proof.** For the token-ring algorithm, there are $n - 1$ branches at the beginning. If the $i$th branch is pruned because all the pairs in this branch are calculated, we calculate the same number of upper bounds as correlations. If the $i$th branch is pruned because the upper bound is lower than the current minimum correlation coefficient in the top-$k$ list, we calculate one more upper bound than correlations for this branch. Therefore, the difference between the number of computed correlation upper bounds and the number of computed correlations is no more than $n - 1$. $\square$

### 3.3. Combination of two tasks

Although the 2-dimensional search algorithm can efficiently find correlated pairs above a given threshold, it is difficult for users to provide an appropriate correlation threshold in real-world applications since different data sets have different characteristics. Instead, we try to find the top-$k$ correlated pairs. However, when using the token-ring algorithm to find the top-$k$ correlated pairs, we could spend a lot of time on calculation and end up with trivial results in a data set that does not contain strong positive correlations. Therefore, we propose a user procedure to combine 2-dimensional search and token-ring search which can stop searching wisely on its way finding the top-$k$ correlated pairs when there are few strong positive correlations in the data set.

Before we try to find the correlated pairs, we can calculate the number of candidates for a given threshold and use this number to estimate the time we will spend. For example, if we hope the algorithm is completed in one day and the time for processing one candidate is 20 ms, then we should choose the threshold under which the number of candidates is less than 4 million. Due to the existence of 2-dimensional search, we can get the number of candidates for any threshold in the linear time $O(n)$. We can choose the lowest threshold $\theta_{min}$ under which the number of candidates is less than 4 million, and then search the correlated pairs by using 2-dimensional search. The advantage is that the algorithm will stop on time if the data set contains few strong positive correlations. However, there are two disadvantages. First, we might retrieve too many pairs and we are only interested in the top-$k$ correlated pairs. Second, we will definitely spend one day to search the correlated pairs, while we might only need to spend one hour to find the top-$k$ correlated pairs by using the token-ring algorithm. Instead of searching the correlated pairs above $\theta_{min}$ by the 2-dimensional search algorithm, we integrate the threshold $\theta_{min}$ into the token-ring algorithm. We treat all the $n - 1$ branches as nodes in the token-ring. Only the branch which gets the token can calculate the upper bound of the leftmost uncalculated pair in this branch and compare its upper bound against the value $\psi = max\{\theta_{min}, \tau\}$ where $\tau$ is the current minimum correlation coefficient in the top-$k$ list. If the upper bound is above $\psi$, we will check the correlation value of this pair. This pair will be pushed into this list if its correlation coefficient is greater than $\psi$. If the upper bound is below $\psi$, this branch will be removed from the token-ring. The algorithm will stop when there is no node in the token-ring. In that way, even if the current minimum correlation coefficient $\tau$ in the top-$k$ list never exceeds the threshold $\theta_{min}$, the algorithm will process all the candidates under the threshold $\theta_{min}$ which would not cost more than the maximal time we allow. It stops searching wisely on its way to find the top-$k$ correlated pairs when there are few strong positive correlations in the data set. If there are a lot of strong positive correlations in the data set, the algorithm will find the top-$k$ list and stop. For example, if we try to find the top-5 pairs and set up the threshold 40 in the coined data set shown in Table 1(a) and tab:cubp(b), the search sequence is as follows: $\{I_1, I_2\}$, $\{I_2, I_3\}$, $\{I_3, I_4\}$, $\{I_4, I_5\}$, $\{I_5, I_6\}$, $\{I_3, I_5\}$, and $\{I_4, I_6\}$. We end up with the only strong pair $\{I_3, I_4\}$.

## 4. Experiments

The efficiency performance of our methods was tested on several real-life datasets. Since the results were similar for all the datasets, we only present results on two typical datasets. The first is the Netflix data set which contains 17,770 movies and 480,000 transactions. The second is a retail data set from the FIMI repository containing 16,470 items and 88,000 transactions. Netflix contains many correlated patterns because of movies in the same series and TV shows of many episodes, while the retail data set contains fewer correlated patterns because those highly correlated items might be already offered by manufactures as a package. Since the pattern of all the correlation measures are similar to each other, we only show the result of likelihood ratio due to the limitation of the pages. The number of correlated pairs under different thresholds for these two data sets is shown in Fig. 1. We transformed both $x$-axis and $y$-axis into the log scale, and it is easy to see that the number of correlated pairs and the likelihood ratio threshold follow a power-law distribution. We implemented our algorithm using Java 1.6.0 on a Dell workstation with 2.4 GHz Dual CPU and 4G memory running on the Vista operating system. In the following, we will check the performance improvement from each algorithm.

### 4.1. Finding correlated pairs above a certain threshold

Here, we focus on task 1 of correlated pair search: finding correlated pairs above a given threshold $\theta$.

#### 4.1.1. Count the number of pair candidates

Before we determine the threshold to search the satisfied pairs, we can count how many pairs' CUBs are above a tentative threshold. This number can help us to estimate the time we will spend on the satisfied pair search because the support of the pair needs to be retrieved if its CUB is above the threshold. The number of candidates and the time spent on counting candidates under different thresholds are shown in Fig. 2. The number of candidate for different thresholds is transformed to the log–log scale and roughly follows a power-law distribution. To get the number of pair candidates, 2-dimensional search is linear and 1-dimensional search is exponential with the threshold. When the threshold is 0, the 1-dimensional search will check all the pairs. Therefore, the runtime of 1-dimensional search with threshold 0 is equal to the runtime of the upper bound calculation method. The 1-dimensional search takes less time than the upper bound calculation method when the threshold is high, but the 2-dimensional search is always an order of magnitude faster than the just upper bound calculation method.

#### 4.1.2. Search the satisfied pairs

Since the IO cost for calculating correlation of pairs is much higher than the time cost for calculating CUB, the CUB calculation can save a lot of unnecessary IO cost when the threshold is high. The log runtime of retrieving the satisfied pairs by calculating CUB first given different log correlation thresholds is shown in Fig. 3. The 1-dimensional and 2-dimensional search are almost identical. The runtime of the CUB, 1-dimensional, and 2-dimensional search all decreases drastically as we increase the threshold. Both 1-dimensional and 2-dimensional search take much less time than the CUB search when the threshold is high because

1-dimensional and 2-dimensional search take less time to find the pairs whose CUB is higher than the threshold. For the dataset with less correlations, the correlation upper bound check for high thresholds dominates the calculation. In that case, the speed-up of 1-dimensional and 2-dimensional search comparing to the CUB search is more obvious like in the retail dataset. However, the CUB, 1-dimensional, and 2-dimensional search do not make too much difference when the threshold is low.

### 4.2. Finding top-k correlated pairs

For finding the top-*k* correlated pairs, the runtime of the brute-force method is determined by the number of correlations we need to compute, while the runtime of TOP-COP or token-ring is determined by the number of correlations and the number of CUBs we need to compute. The brute-force method will calculate the correlation of all the possible pairs which is not feasible for large datasets. According to Theorem 4, token-ring computes fewer CUBs than TOP-COP, but computes the same number of correlations. Therefore, the runtime difference between token-ring and TOP-COP is determined by the difference between the number of CUB calculations. The number of correlation calculations and CUB calculations under different *k* is shown in Fig. 4. According to Theorem 5, the number of correlation calculations is almost equal to that of CUB calculations in the token-ring algorithm which is verified in Fig. 4. Token-ring can save a huge number of unnecessary CUB checks compared to TOP-COP. The runtime of TOP-COP and token-ring for top-*k* correlated pairs is shown in Fig. 5. When the IO cost of retrieving pair support is much more expensive than the CUB calculation for a dense dataset like Netflix, the total runtime is dominated by the time spent on correlation calculation and there is little difference between token-ring and TOP-COP. When the IO cost is not that expensive compared to the CUB calculation for a sparse dataset like retail, we can see token-ring took significantly less time than TOP-COP. Besides reducing a large amount of memory usage, token-ring is significantly faster than TOP-COP where the gap between the true correlation and the correlation upper bound is large.

The top-*k* search method is more flexible for finding the desired patterns compared to the 2-dimensional method. Assuming we can get the top-*k* pair correlation ahead of time and use it to search the top-*k* pairs with the 2-dimensional method, Fig. 6 shows the number of correlation calculations using 2-dimensional search and those using our top-*k* search method. The gap between the 2-dimensional method and the top-*k* search method is not huge and the gap gradually increases with the *k*. The top-*k* search

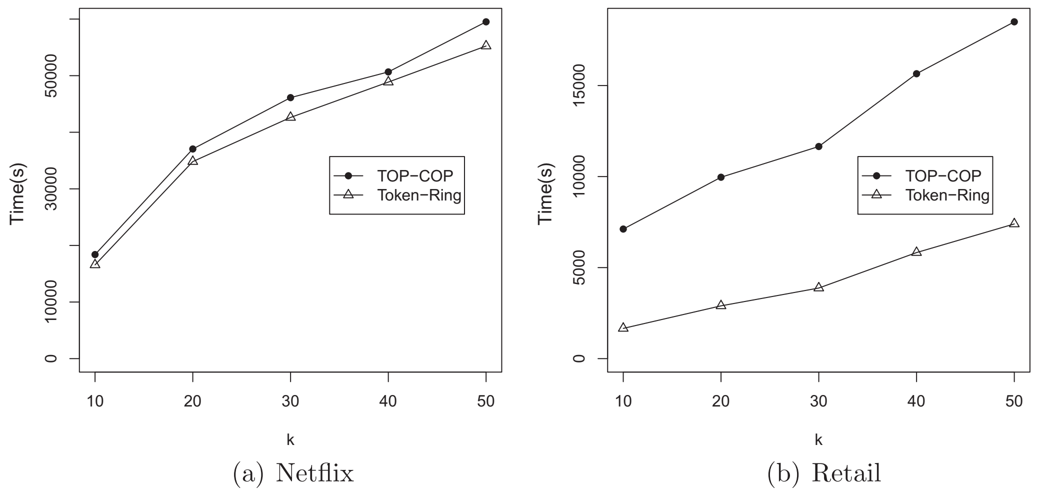

**Fig. 5.** The runtime for top-*k* algorithms.

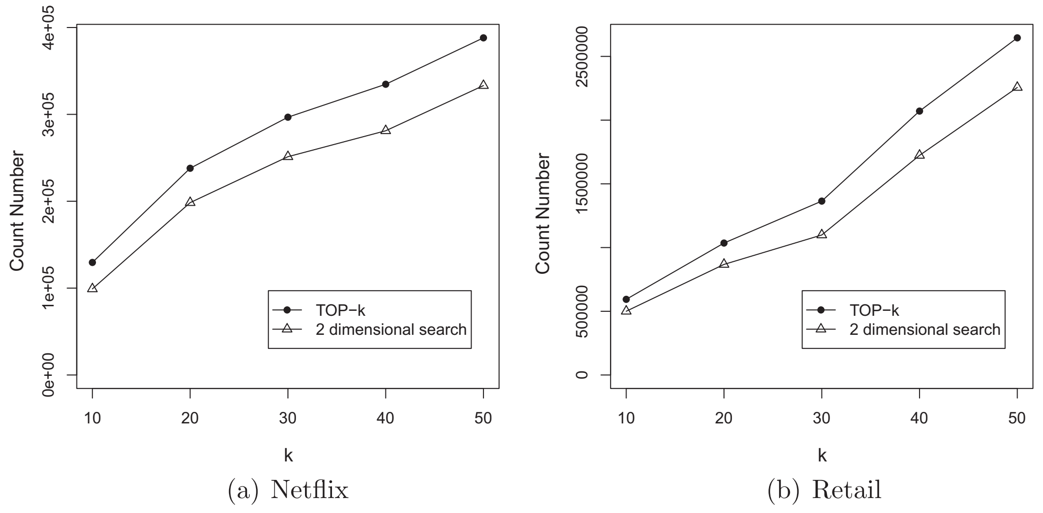

**Fig. 6.** Correlation checks for top-*k* search and threshold search.

method is more flexible and does not require much more time than the threshold search method.

### 4.3. Combination of two tasks

We proposed a user procedure to combine the 2-dimensional search algorithm and the token-ring algorithm to reconcile the two tasks of searching for pairs above a certain threshold and for top-$k$ pairs. The performance of the user procedure really depends on the parameters we set. If the threshold $\theta$ is so high that the $k$th correlation in the dataset is less than $\theta$, the runtime is equal to the 1-dimensional search under the threshold $\theta$ because the modified token-ring checks the same number of correlations and the same number of CUBs as the 1-dimensional search algorithm. If $k$ is small and the data set contains many strong positive correlations, the runtime is close to the original token-ring algorithm. From Figs. 3 and 5, we can estimate the runtime of the modified token-ring for different parameters. For example, when the threshold $\theta$ is 1000 and $k$ is 10 in Netflix, the runtime will be close to 18,000 s.

## 5. Conclusions

We made several significant progresses on correlated pair search in the paper. First, we propose a framework to search correlated pairs for any "good" correlation measure. Second, a new token-ring algorithm is proposed for the top-$k$ search. Comparing to the state of art TOP-COP algorithm, our token-ring algorithm reduces a large amount of memory usage and is significantly faster when the gap between the true correlation and the correlation upper bound is large. Third, we propose a user procedure to combine the correlated pair search above a certain threshold and the top-$k$ search. The combination is more robust for the threshold specification and avoids the situation of spending too much time to get trivial results in a data set that does not contain too many strong positive correlations. The experimental results show that our robust algorithm can efficiently search correlated pairs under different situations and is an order of magnitude faster than the brute-force method.

## References

Aggarwal, C.C., Pei, J., Zhang, B., 2006. On privacy preservation against adversarial data mining. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'06. ACM, New York, NY, USA, pp. 510–516.

Agrawal, R., Imieliński, T., Swami, A., 1993. Mining association rules between sets of items in large databases. In: SIGMOD'93: Proceedings of the ACM SIGMOD International Conference on Management of Data. ACM, New York, NY, USA, pp. 207–216.

Cheng, H., Yan, X., Han, J., Hsu, C.-W., 2007. Discriminative frequent pattern analysis for effective classification. In: ICDE'07, pp. 716–725.

Duan, L., Khoshneshin, M., Street, W.N., Liu, M., 2013. Adverse drug effect detection. IEEE Journal of Biomedical and Health Informatics (in press).

Duan, L., Street, W.N., 2009. Finding maximal fully-correlated itemsets in large databases. In: ICDM'09: Proceedings of the International Conference on Data Mining, Miami, FL, USA, pp. 770–775.

Duan, L., Street, W.N., 2012. Selecting the right correlation measure for binary data. <http://papers.ssrn.com/sol3/papers.cfm?abstractid=2035491>.

Geng, L., Hamilton, H.J., 2006. Interestingness measures for data mining: a survey. ACM Computing Surveys 38 (3), 9.

Guns, T., Nijssen, S., De Raedt, L., 2011. Itemset mining: a constraint programming perspective. Artificial Intelligence 175, 1951–1983.

Lin, K.-H., Chiu, Y.-S., Chen, J.-S., 2011. An adaptive correlation-based group recommendation system. In: 2011 International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), pp. 1–5.

Piatetsky-Shapiro, G., 1991. Discovery Analysis and Presentation of Strong Rules. AAAI/MIT Press.

Tan, P.-N., Kumar, V., Srivastava, J., 2004. Selecting the right objective measure for association analysis. Information Systems 29 (4), 293–313.

Webb, G.I., 2007. Discovering significant patterns. Machine Learning 68, 1–33.

Xiong, H., Shekhar, S., Tan, P.-N., Kumar, V., 2006. TAPER: a two-step approach for all-strong-pairs correlation query in large databases. IEEE Transaction on Knowledge and Data Engineering 18 (4), 493–508.

Xiong, H., Zhou, W., Brodie, M., Ma, S., 2008. Top-k $\phi$ correlation computation. INFORMS Journal on Computing 20 (4), 539–552.

Yahia, S.B., Hamrouni, T., Nguifo, E.M., 2006. Frequent closed itemset based algorithms: a thorough structural and analytical survey. SIGKDD Exploration Newsletter 8, 93–104.

Zhang, J., Feigenbaum, J., 2006. Finding highly correlated pairs efficiently with powerful pruning. In: CIKM'06: Proceedings of the ACM CIKM International Conference on Information and Knowledge Management. ACM, New York, NY, USA, pp. 152–161.