Extracting Redundancy-Aware Top-K Patterns [*]

Dong Xin Hong Cheng Xifeng Yan Jiawei Han

Department of Computer Science
University of Illinois at Urbana-Champaign
{dongxin, hcheng3, xyan, hanj}@uiuc.edu

ABSTRACT

Observed in many applications, there is a potential need of extracting a small set of frequent patterns having not only high significance but also low redundancy. The significance is usually defined by the context of applications. Previous studies have been concentrating on how to compute top-k significant patterns or how to remove redundancy among patterns separately. There is limited work on finding those top-k patterns which demonstrate high-significance and low-redundancy simultaneously.

In this paper, we study the problem of extracting *redundancy-aware top-k patterns* from a large collection of frequent patterns. We first examine the evaluation functions for measuring the combined significance of a pattern set and propose the *MMS* (Maximal Marginal Significance) as the problem formulation. The problem is known as NP-hard. We further present a greedy algorithm which approximates the optimal solution with performance bound $O(\log k)$ (with conditions on redundancy), where k is the number of reported patterns. The direct usage of redundancy-aware top-k patterns is illustrated through two real applications: disk block prefetch and document theme extraction. Our method can also be applied to processing redundancy-aware top-k queries in traditional database.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining

General Terms: Algorithms

Keywords: Pattern Extraction, Significance, Redundancy

1. INTRODUCTION

Frequent patterns are widely used in sophisticated data mining and database applications, including association rule mining, classification, clustering, and indexing. Recent progress on frequent-pattern mining has seen two trends: (1) measuring *significance* of various kinds of patterns, such as *tf-idf* scores [23] for text topics and position-weighted matrix score [17] for biological motifs; and (2) eliminating *redundancy* among discovered patterns, *e.g.*, lossless compression using closed [18] or non-derivable [4] patterns, and lossy summarization using ordered patterns [16], cover-set [1], clustering [25], or pattern profiles [26]. These studies often emphasize significance and redundancy separately, while many applications need to consider these two measures together.

One interesting example is *correlation-directed disk block prefetch*. A disk access sequence is a sequence of blocks, e.g., $b_{35}, b_{100}, b_{9039}, ...$, where b_i represents the i^{th} block on the disk. Suppose an access to b_{35} is repeatedly followed by an access to b_{9039}, it may improve the I/O performance if these two blocks are arranged adjacent to each other and fetched together when block b_{35} is accessed. Li *et al.* [14] show that correlation-directed prefetch can improve the average I/O response time by up to 25%. The system uses association rules as a decision system: Whenever the left-hand side of a rule is satisfied, the blocks on the right-hand side are prefetched. However, there are considerable redundancy existing in association rules, for example, one can generate more than 200k rules for one I/O trace collected at the HP Lab [20]. Due to the resource limitation, a system may only want to pick a subset of important yet divergent rules. The significance of each rule can be measured by its additional value to the existing rules.

The second example is *document theme extraction* [3, 15], where each document (or each sentence) is treated as a transaction. The goal is to extract the frequent patterns of term occurrence, called *themes*, buried in a large set of documents. Given a document set, the top-k frequent patterns returned by a mining algorithm are not necessarily the best k themes one can find. Many frequent term sets could overlap significantly with each other. Such overlapping may render top-k important themes very redundant.

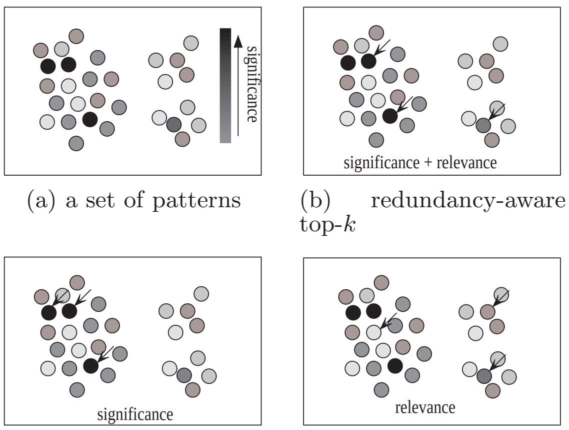
As shown in the above two applications, a useful compact pattern set should simultaneously demonstrate high significance and low-redundancy. We call this kind of patterns *redundancy-aware top-k patterns*.

Previous studies on pattern compression (summarization) [1, 16, 25, 26] are able to approximate a collection

[*]The work was supported in part by the U.S. National Science Foundation NSF IIS-03-08215/05-13678 and NSF BDI-05-15813.

of frequent patterns using a small pattern set, which aims to minimize the frequency restoration error for those patterns that are not selected. A close work to this paper is the pattern ordering problem studied in [16], where the authors rank patterns such that the top-k patterns are able to best summarize the whole set of frequent patterns. The major difference between our problem and all of the previous works is that we emphasize both significance and redundancy on the *selected* top-k patterns, and the pattern significance is defined by the context of the applications; while summarizing the whole collection of the patterns is not our goal. The previous works only consider pattern relevance rather than significance, thus may not provide a solution to redundancy-aware top-k pattern extraction.

Previous works on top-k frequent pattern mining [10] assume patterns are independent, which unfortunately is not the case. Figure 1(a) shows a set of frequent patterns where each circle represents one pattern whose significance is colored in gray scale, and the distance between two circles reflects their relevance. The intuition of redundancy-aware top-k patterns is illustrated in Figure 1(b) as opposed to the traditional top-k patterns in Figure 1 (c) and the k summarized patterns in Figure 1(d). Redundancy-aware top-k patterns make a trade-off between significance and redundancy. The three patterns pointed by arrow in Figure 1(b) have high significance and low redundancy. On the other hand, the traditional top-k approach picks patterns based on significance solely and a pattern summarization approach picks patterns based on relevance solely.



(a) a set of patterns (b) redundancy-aware top-k

(c) traditional top-k (d) summarization

Figure 1: Redundancy-aware Top-k, Traditional Top-k, and Summarization

In this paper, we formulate the redundancy-aware top-k pattern extraction problem through a general ranking model which integrates two measures, *significance* and *redundancy*, into one objective function. We first examine the evaluation functions for measuring the combined significance of a pattern set and propose the *MMS* (Maximal Marginal Significance) as the problem formulation. The problem is known as NP-hard. We further present a greedy algorithm which approximates the optimal solution with performance bound $O(\log k)$, where k is the number of reported patterns.

Although our work focuses on pattern extraction, the methodology developed in this paper can also be applied to many top-k query applications [2] to help users explore query results more effectively. More specifically, since similar results are often ranked closely, the top-k query results may not provide enough diversified information to users. Our method can be used to get the redundancy-aware top-k ranking.

The rest of the paper is organized as follows. Section 2 introduces the concept of redundancy-aware top-k pattern extraction and its problem formulation. A comparison of the alternative objective functions is made in Section 3. We propose an improved algorithm for the *MMS* problem in Section 4. Section 5 presents two case studies of document theme extraction and correlation-directed prefecth. The related work is presented in Section 6 and we conclude our study in Section 7.

2. PROBLEM FORMULATION

In this section, we first discuss measures for pattern significance and pattern redundancy, and then propose the formal problem formulation.

2.1 Significance and Redundancy

Here we define significance and redundancy in the context of this paper.

Definition 1. *(Pattern Significance) A significance measure S is a function mapping a pattern $p \in \mathcal{P}$ to a real value such that $S(p)$ is the degree of interestingness (or usefulness) of the pattern p.*

There are several previous studies on the significance (or interestingness) measure of patterns, which include [11] on rule interestingness, and [22, 24, 12] on interesting measure of frequent item-set or association patterns. According to [22], the significance measure can be divided into *objective* measures and *subjective* measures. Commonly used objective measures include support, confidence, lift, coherence, and *tf-idf* for text patterns and attribute values for database tuples. Subjective measure is usually a relative score compared with some prior knowledge or background model. It measures the unexpectedness of a pattern by computing its divergence from the background model. [11, 12] are examples that use subjective measures.

We further extend the expression S to *combined significance* and *relative significance*. Let $S(p, q)$ be the combined significance of patterns p and q, and $S(p|q) = S(p, q) - S(q)$ be the relative significance of p given q. Note that the combined significance $S(p, q)$ means the collective significance of two individual patterns p and q, not the significance of a single super pattern $p \cup q$.

Given significance measures, we can define the redundancy between two patterns.

Definition 2. *(Pattern Redundancy) Given the significance measure S, the redundancy R between two patterns p and q is defined as $R(p, q) = S(p)+S(q)-S(p, q)$. Subsequently, we have $S(p|q) = S(p) - R(p, q)$.*

In this paper, we make the assumption that the combined significance of two patterns is no less than the significance of any individual pattern (since it is a collective significance of two patterns) and does not exceed the sum of two individual significance (since there exists redundancy). This simply says that the redundancy between two patterns should satisfy

$$0 \leq R(p,q) \leq \min(S(p), S(q)). \quad (1)$$

The ideal redundancy measure $R(p,q)$ is usually hard to obtain. In this paper, we approximate redundancy using distance between patterns.

DEFINITION 3. *(Pattern Distance) A distance measure $D : \mathcal{P} \times \mathcal{P} \rightarrow [0,1]$ is a function mapping two patterns $p, q \in \mathcal{P}$ to a value in $[0,1]$, where 0 means p, q are completely relevant and 1 means p, q are totally independent.*

The distance can be calculated based on the pattern structure, *e.g.*, the edit distance between two DNA sequences; or based on the underlying data used in the discovery process, *e.g.*, the Jaccard distance used in [13]; or based on the distribution of the patterns, *e.g.*, Kullback-Leibler Divergence. If a distance is a metric measure, *i.e.*, it has properties of isolation, symmetry, and triangle inequality, it will bring many desirable properties. In the above example, both string edit distance and the Jaccard distance are metrics.

More generally, the distance $D(p,q)$ can be weighted to reflect users' preference on penalizing redundancy. Since distance is the complementary of redundancy, we use the following equation to approximate R:

$$R(p,q) = (1 - D(p,q)) \times \min(S(p), S(q)). \quad (2)$$

The above function indicates that the value of $R(p,q)$ is bounded by $[0, \min(S(p), S(q))]$ (see Eqn. (1)).

2.2 Evaluating k Patterns

We extend our formulation to a set of k patterns. Let G be an evaluation function measuring the significance of a set of k patterns $\mathcal{P}^k = \{p_1, p_2, \ldots, p_k\}$. If we assume patterns in $\mathcal{P}^k$ are all independent, we have:

$$G_{ind}(\mathcal{P}^k) = \sum_{i=1}^{k} S(p_i),$$

where S is the significance measure.

In general, there are redundancies between patterns. Let L be a function returning redundancies among $\mathcal{P}^k$:

$$G_{gen}(\mathcal{P}^k) = \sum_{i=1}^{k} S(p_i) - L(\mathcal{P}^k).$$

In many cases, L is very hard to formulate. We propose two heuristic evaluation functions G_{as} (average significance) and G_{ms} (marginal significance), which sacrifice some generality but are more practical for computation and search. We first define our computation model based on a new concept, *redundancy graph*.

DEFINITION 4. *(Redundancy Graph) Given a significance measure S and redundancy measure R, a redundancy graph of a set of patterns $\mathcal{P}$ is a weighted graph $G(\mathcal{P})$ where each node i corresponds to a pattern p_i. The weight on node i is pattern significance $S(p_i)$ and the weight on an edge (i,j) is the redundancy $R(p_i, p_j)$.*

Let the redundancy subgraph induced by the set of k patterns be $G(\mathcal{P}^k)$. The natural formulation of L is to consider all pair-wise redundancy by summing the edge weights of $G(\mathcal{P}^k)$. Since there are k patterns and $\frac{k(k-1)}{2}$ edges, we further normalize it by taking average weights on edges. Typically, the average weights associated with a pattern p_i are:

$$\overline{w}(p_i) = \frac{1}{k-1} \sum_{j=1, j\neq i}^{k} R(p_i, p_j).$$

The evaluation function G_{as} is defined as below:

$$G_{as}(\mathcal{P}^k) = \sum_{i=1}^{k} S(p_i) - \frac{1}{2} \sum_{i=1}^{k} \overline{w}(p_i), \quad (3)$$

where $\frac{1}{2}$ is introduced because every redundancy $R(p_i, p_j$ is counted twice by both p_i and p_j. Substitute $\overline{w}(p_i)$ in Eqn. (3):

$$G_{as}(\mathcal{P}^k) = \sum_{i=1}^{k} S(p_i) - \frac{1}{k-1} \sum_{i=1}^{k} \sum_{j=1}^{i-1} R(p_i, p_j) \quad (4)$$

We refer this formulation as *average significance*.

An alternative formulation for L is to compute the maximum spanning tree of $G(\mathcal{P})$. Let the sum of edge weights on the maximum spanning tree be $w(MST_{\mathcal{P}})$. The evaluation function G_{ms} is defined as below:

$$G_{ms}(\mathcal{P}^k) = \sum_{i=1}^{k} S(p_i) - w(MST_{\mathcal{P}}). \quad (5)$$

Note that the G_{ms} formulation is a generalization of maximal marginal relevance (*MMR*) heuristic in information retrieval [5], where a document has high marginal relevance if it is both relevant to the query and contains minimal marginal similarity to previously selected documents. The marginal similarity is computed by choosing the most relevant selected document. Different from G_{ms}, this definition gives a *procedural* way to evaluate a set of documents. If we use this concept to compute the score of a set of patterns $\mathcal{P}^k$ (by adding patterns $p_1, p_2, \ldots, p_k$ incrementally), we have

$$MMR(\mathcal{P}^k) = S(p_1) + \sum_{i=2}^{k} (\min_{j=1}^{i-1} S(p_i | p_j)).$$

Combining the definition of relative significance, one can easily verify that *MMR* approximates L by computing a spanning tree on $G(\mathcal{P}^k)$. However, the score of *MMR* depends on the order on which patterns are selected. G_{ms} is the minimum score over all possible *MMR* scores. We refer G_{ms} formulation as *marginal significance*.

Correspondingly, the problems of finding redundancy-aware top-k patterns are as follows:

DEFINITION 5. *(Maximal Average Significance) Given a set of pattern collection $\mathcal{P}$, the problem of* Maximal Average Significance *(MAS) is to find k-pattern set $\mathcal{P}^k$ such that $G_{as}(\mathcal{P}^k)$ is maximized.*

DEFINITION 6. *(Maximal Marginal Significance) Given a set of pattern collection $\mathcal{P}$, the problem of* Maximal Marginal Significance *(MMS) is to find k-pattern set $\mathcal{P}^k$ such that $G_{ms}(\mathcal{P}^k)$ is maximized.*

3. COMPARING MAS AND MMS

In this section, we examine the two proposed evaluation functions. We show that both *MAS* and *MMS* problems are NP-hard, and adopt a well-known greedy algorithm to compare their performance.

3.1 The Greedy Algorithm

We consider a special case of the redundancy graph where all patterns have the same significance score, and thus only the weights on edges take effect. The problem of *MAS* is thus to find a k-pattern set where the sum of edge weights are minimized. This problem is equivalent to k-dense subgraph problem, which is known to be NP-hard [7]. The problem of *MMS* is to find a k-maximum spanning tree whose overall weights are minimized. Holldorsson *et al.* [9] show that this problem is NP-hard.

Since it is difficult to find the optimal solutions, we adopt a well-known greedy algorithm to examine the performance of *MAS* and *MMS*. The algorithm incrementally selects patterns from $\mathcal{P}$ with an estimated gain g. A pattern is selected if it has the maximum gain among the remaining patterns. Given a set of selected patterns $\mathcal{P}^k$, the gain of a pattern $p \in \mathcal{P} - \mathcal{P}^k$ is:

$$g(p) = \begin{cases} S(p) - \frac{1}{|\mathcal{P}^k|}\sum_{q \in \mathcal{P}^k} R(p,q), & for\ MAS, \\ S(p) - \max_{q \in \mathcal{P}^k} R(p,q), & for\ MMS. \end{cases}$$

At beginning, the result set $\mathcal{P}^k$ is empty. The algorithm picks the most significant pattern and inserts it to $\mathcal{P}^k$. When $|\mathcal{P}^k| < k$, we will compute gain $g(p)$ for every remaining pattern $p \in \mathcal{P} - \mathcal{P}^k$, and select the pattern with the maximum gain. After a pattern is inserted into $\mathcal{P}^k$, it remains in $\mathcal{P}^k$.

The naive implementation of the above algorithm takes time $O(k^2n)$. The alternative approach with time complexity $O(kn)$ can be implemented as follows. For each remaining pattern, we can remember the previous gain and compute the new gain by updating the redundancy with the last pattern added to $\mathcal{P}^k$. As an example, assume at the i^{th} iteration, the pattern p_i is selected, and for each pattern $p \in \mathcal{P} - \mathcal{P}^k$, $g^i(p)$ was computed with respect to $\mathcal{P}^k - \{p_i\}$. To search for next candidate pattern, we need to update $g(p)$ by incorporating the newly selected pattern p_i. One can verify the following update formulas for *MAS* and *MMS*:

$$g^{i+1}(p) = \begin{cases} S(p) - \frac{1}{i}\left((i-1)(S(p) - g^i(p)) + R(p,p_i)\right), \\ S(p) - \max\left((S(p) - g^i(p)), R(p,p_i)\right). \end{cases}$$

The execution of update functions takes constant time. The algorithm is described in Algorithm 1. Finding the most significant pattern takes time $O(n)$. At each iteration, we need to compute gain $g(p)$ for each pattern $p \in \mathcal{P} - \mathcal{P}^k$, and select the one with the maximum value. Using the update functions, each iteration also takes time $O(n)$. The total time complexity of the greedy algorithm is $O(kn)$.

Algorithm 1 The Greedy Algorithm

```
Input: A set of n patterns, $\mathcal{P}$
       Number of output patterns, $k$
       Significance Measure, $S$
       Divergence Measure, $D$
Output: $k$-pattern set, $\mathcal{P}^k$

1: Let $p$ be the most significant pattern;
2: $\mathcal{P}^k = \{p\}$;
3: while ($|\mathcal{P}^k| < k$)
4:     Find a pattern $p$ such that the gain $g(p)$ is the
5:         maximum among the set of patterns in $\mathcal{P} - \mathcal{P}^k$;
6:     $\mathcal{P}^k = \mathcal{P}^k \cup \{p\}$;
7: return
```

3.2 Comparing MAS and MMS

We examine both formulations using the same greedy algorithm. The experiments are conducted on two real applications: disk block prefetch and document theme extraction. For clear presentation, the results are organized in Section 5. We observe that *MMS* performs much better in both experiments. There are two possible reasons that may explain the results. First, *the unified greedy algorithm may favor MMS*; and second, *the formulation of MMS is more reasonable.* We discuss these two issues one by one.

Since both problems are NP-hard and the greedy algorithm reports approximate solutions. We study the performance bound of the greedy solutions with respect to the optimal solutions. The following theorem shows that Algorithm 1 has performance bound 2 for *MAS*. Due to limited space, we omit the proof.

THEOREM 1. *Let the k-pattern set returned by Algorithm 1 (with MAS gain) be $\mathcal{P}^k$, and the optimal pattern set be $\mathcal{O}^k$. We have:*

$$G_{as}(\mathcal{O}^k) \leq 2G_{as}(\mathcal{P}^k).$$

To our best knowledge, the algorithm does not have performance bound for *MMS*. In fact, a counter example in Section 4.2 shows that the worst case performance bound on *MMS* could be much worse than that of *MAS*. This analysis indicates that Algorithm 1 does not favor *MMS* and the worse performance of *MAS* may be caused by the limitation of its formulation.

We further examine the top-k patterns returned by both algorithms in our experiments. The patterns returned by *MAS* clearly contain more redundancy. This is because the redundancy penalty in *MAS* formulation is averaged by the number of patterns k, and each pattern usually has redundancy with a few other patterns. The larger the value of k, the smaller the redundancy penalties. One may suggest to remove the denominator (*i.e.*, $k-1$) in Eqn. (4). However, this may lead to over penalizing in the objective function since the number of redundancy penalties is the order of square of the number of patterns. On the other hand, the *MMS* formulation is not sensitive to the value of k.

In summary, the *MMS* formulation is quite reasonable. One possible extension to *MMS* formulation is to allow weighted combination of the significance and redundancy penalty. This actually is implicitly handled by our definition of distance measure because we can always incorporate the user-defined weights into the distance definitions. In the rest of the paper, we mainly focus on the *MMS* problem.

4. AN IMPROVED METHOD FOR MMS

Here we discuss an improved method to the *MMS* problem. We assume that the distance measure satisfies triangle inequality. Our method is not restricted to this constraint. However, if this condition holds, our solution has a guaranteed performance bound.

4.1 The Computational Model

We first introduce a variant computation model based on *redundancy graph*. As defined in Section 2, the redundancy graph is an edge-weighted and node-weighted undirected graph. We transform it to the *directed redundancy graph* as follows: for each pair of patterns p_i and p_j, we create a directed edge from p_i to p_j, and the associated edge weight is the *relative significance* $S(p_j|p_i)$. The weight on each node p_i is still the pattern significance $S(p_i)$. An example of this transformation is shown in Fig. 2 (Not all directed edges are shown in the transformed redundancy graph).

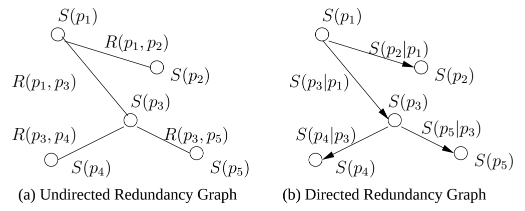


Figure 2: Directed redundancy graph

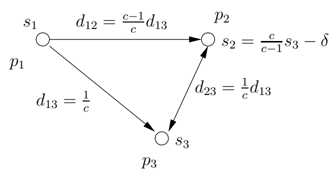
In *MMS* problem, $G_{ms}(\mathcal{P}^k)$ is evaluated by computing the *maximum spanning tree* on the sub redundancy graph $G(\mathcal{P}^k)$. There are k node weights and $k-1$ edge weights in the tree. We particularly select the most significant pattern as the root of the maximum spanning tree, and combine the other $k-1$ node weights and the $k-1$ edge weights. Example 1 shows this procedure.

EXAMPLE 1. *In Fig. 2, suppose the set of pattern $\mathcal{P}^k = \{p_1, p_2, p_3, p_4, p_5\}$ is evaluated by the spanning tree shown in Fig. 2 (a), and p_1 is the most significant pattern. Originally, $G_{ms}(\mathcal{P}^k) = \sum_{i=1}^{5} S(p_i) - R(p_1,p_2) - R(p_1,p_3) - R(p_3,p_4) - R(p_3,p_5)$. It is equivalent to $G_{ms}(\mathcal{P}^k) = S(p_1) + S(p_2|p_1) + S(p_3|p_1) + S(p_4|p_3) + S(p_5|p_3)$, as shown in Fig. 2 (b).*

Since we transform the negative redundancy penalties to positive relative significance, the original *maximum spanning tree* on the undirected redundancy graph corresponds to the *minimum spanning tree* on the directed redundancy graph. The *MMS* problem is equivalent to searching a *constrained rooted minimum spanning tree* on the directed redundancy graph such that the overall weights on the root node and on the edges in the tree are maximized. The constraint specifies that the root must be the most significant pattern in the tree.

4.2 Performance Study of Algorithm 1

We study the worst case performance of *MMS* by Algorithm 1, under the assumption that the distance measure satisfies triangle inequality. The following example shows that this greedy approach may lead to a serious problem in some case. We rewrite the computation equation of $S(p|q)$ here for easy understanding of the example: $S(p|q) = S(p) - (1 - D(p,q))\min(S(p), S(q))$.



s_1
p_1
$d_{12} = \frac{c-1}{c} d_{13}$
p_2
$s_2 = \frac{c}{c-1} s_3 - \delta$
$d_{13} = \frac{1}{c}$
$d_{23} = \frac{1}{c} d_{13}$
s_3
p_3

Figure 3: A directed redundancy graph with 3 patterns

EXAMPLE 2. *Consider a graph with three patterns p_1, p_2, and p_3 (Fig. 3). For simplicity, we use s_i and d_{ij} to denote $S(p_i)$ and $D(p_i, p_j)$, respectively. Let $s_1 \geq s_2 \geq s_3$, and $s_2 = \frac{c}{c-1} s_3 - \delta$ (where $\delta > 0$ is a small perturbation). Let $d_{12} = \frac{c-1}{c} d_{13}, d_{23} = \frac{1}{c} d_{13}$, and $d_{13} = \frac{1}{c}$. One can verify that d_{12}, d_{13}, and d_{23} satisfy triangle inequality. The greedy algorithm will first select pattern p_1. Since $S(p_3|p_1) = d_{13}s_3 > d_{12}s_2 = S(p_2|p_1)$, the algorithm will pick p_3 as the next. The estimated gain on the objective function is $r = S(p_3|p_1)$. The algorithm continues to look for the next pattern p_2. The estimated gain for adding p_3 and p_2 is:*

$$S(p_3|p_1) + \min(S(p_2|p_1), S(p_2|p_3)) \approx 2r.$$

However, the real objective function of MMS is evaluated by the spanning tree $p_1 \to p_2 \to p_3$, with the gain $S(p_2|p_1) + S(p_3|p_2) \approx r + \frac{r}{c}$, where c can be chosen arbitrarily large. This over-estimation can be accumulated quite large when the number of patterns increases.

The reason that the greedy approach has the over-estimation problem is that the relative significance is not symmetric. Given patterns p and q, we have $S(q|p) \geq S(p|q)$ if $S(q) > S(p)$. If we select the less significant pattern p first, there will be an over-estimation. To avoid this problem, we should try to incrementally add patterns according to significance decreasing order. This motivates our alternative approximation algorithm.

4.3 An Alternative Approach

We first outline the main ideas. The algorithm searches for a specific value r, with which, the algorithm first finds the most significant pattern (as p_1), and removes all patterns p such that $S(p|p_1) \leq r$; then finds the most significant pattern in the remaining patterns (as p_2), and removes all patterns p such that $S(p|p_2) \leq r$, and so on. We finally get k_r patterns. Ideally, we want to find the perfect r value such that $k_r = k$.

The first intuition is that when r value is small, we may have $k_r > k$, and when r value is large, $k_r < k$. If the k_r value is monotonic to r, then we can run a binary search on the domain of r. Unfortunately, k_r is not monotonic to r. Fig. 4 shows a counter example that a larger r value leads to a larger k_r.

EXAMPLE 3. *Suppose $S(p_1) \geq S(p_2) \geq \ldots \geq S(p_5)$. We only display the edges whose weights are less than* 1.5. *When $r = 1.0$, we get two patterns p_1 and p_3. When $r = 1.4$, we get three patterns p_1, p_4 and p_5.* ■

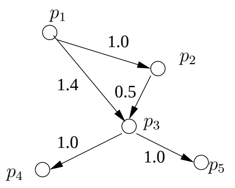


Figure 4: A Counter Example

Instead of searching for the perfect r value, we search for a pair of trial values t and T ($t < T$), such that T leads to $k_T \le k$ and t leads to $k_t \ge k$. If the difference $T - t = \epsilon$ is sufficiently small, we can pick k patterns from the k_t patterns with some desired property (*i.e.*, Lemma 1).

We introduce the ϵ-normalization on edge weights. For each pattern pair p_i and p_j, the edge weight $S(p_i|p_j) = S(p_i) - R(p_i, p_j) \le S(p_i)$. Suppose p_1 is the most significant pattern, we have $S(p_i|p_j) \le S(p_1)$. That is, every edge weight is upper bounded by $S(p_1)$. We partition $[0, S(p_1)]$ into B equi-width intervals, and each interval has width $\epsilon = \frac{S(p_1)}{B}$. $S(p_i|p_j)$ is normalized to $\overline{S(p_i|p_j)} = \lfloor \frac{B \times S(p_i|p_j)}{S(p_1)} \rfloor \times \epsilon$. With this normalization, we run a binary search on the normalized edge weights whose search space is 0 to $S(p_1)$ (*i.e.*, B intervals). Initially, $k_T = 1 \le k$ by $T = S(p_1)$, and $k_t = |\mathcal{P}| \ge k$ by $t = 0$. If $k_{(T+t)/2} \ge k$, we update $t = (T+t)/2$. Otherwise, we update $T = (T+t)/2$. After $\log B$ times binary search, we have $T - t = \epsilon$ and $k_T \le k \le k_t$.

We discuss how to select k patterns from k_t patterns when $T - t = \epsilon$. Our goal is to find k patterns such that (1) the directed-edge weight between them is lower bounded by a positive value d, and (2) for any other pattern q, there exists one pattern p in the selected k patterns such that the edge weight $S(q|p)$ is upper bounded by a constant factor of d.

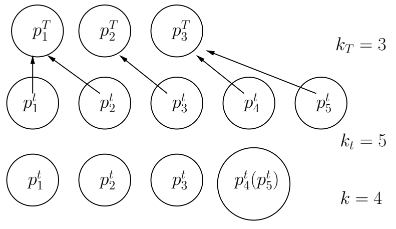


Figure 5: Find k patterns from (u, l)-pair

The selecting strategy is demonstrated by Fig. 5. Let $p_1^t, p_2^t, \ldots, p_{k_t}^t$ be the selected k_t patterns (assume $S(p_1^t) \ge S(p_2^t) \ge \ldots \ge S(p_{k_t}^t)$), and $p_1^T, p_2^T, \ldots, p_{k_T}^T$ be the selected k_T patterns. Each pattern is around by a circle which indicates a set of patterns removed due to the selection of this pattern. Every pattern p_i^t must belong to one circle in $p_j^T (j = 1, 2, \ldots, k_T)$. We select k patterns from the k_t patterns by the following rules:

1. The most significant pattern p_i^t in each p_j^T circle is first selected. In our example, patterns p_1^t, p_3^t and p_4^t are selected;
2. While the number of selected patterns is less than k, we select the most significant p_i^t patterns in the remaining patterns (*i.e.*, we select pattern p_2^t). After k patterns are selected, the remaining p_i^t will find a selected pattern which belongs to the same circle p_j^T with p_i^t. In our example, p_5^t is a remaining pattern, and it belongs to circle p_3^T with a selected pattern p_4^t. We further merge p_5^t as well as all the patterns in circle p_5^t to circle p_4^t.

The complete procedure is summarized as Algorithm 2, which is self-explanatory. Each iteration takes time $O(kn)$, and the complexity to find the values of T and t ($T - t = \epsilon$) is $O(kn \log B)$. Generally, we use $k \le B \le n$. The complexity of selecting k patterns from k_t patterns relies on the generation of k_t patterns, whose complexity is $O(k_t n)$. In most cases, k_t is comparable to k.

Algorithm 2 Greedy Algorithm for MMS

```
Input: A set of n patterns, $\mathcal{P}$
       Number of output patterns, $k$
       Significance measure, $S$
       Divergence measure, $D$
       Weight normalization, $B$
Output: $k$-pattern set, $\mathcal{P}^k$

1:  $\epsilon = \frac{S(p_1)}{B}$, $t = 0$, $T = S(P_1)$;
2:  Run the binary search with $(t, T)$ in space $[0, S(p_1)]$;
3:  $selected[i] = false$ $(i = 1, \ldots, n)$;
4:  $removed[i] = false$ $(i = 1, \ldots, n)$;
5:  for $i = 1$ to $k$
6:    if there is no pattern left //$k_{\frac{T+t}{2}} < k$, decrease $T$
7:      $T = \frac{T+t}{2}$, goto line 2;
8:    Let $p_s$ be the most significant pattern s.t.
        $selected[s] \equiv false$ and $removed[s] \equiv false$;
9:    Assign $selected[s] = true, removed[s] = true$;
10:     for $j = 1$ to $n$
11:       if (!$removed[j]$ and !$selected[j]$))
12:         if ($\overline{S(p_j|p_s)} \le \frac{T+t}{2}$)
13:           $removed[j] = true$;
14:   if there are patterns left //$k_{\frac{T+t}{2}} > k$, increase $t$
15:     $t = \frac{T+t}{2}$, goto line 2;
16: Generate $k_t$ patterns;
17: Select $k$ patterns from $k_t$ patterns;
18: return;
```

The desired property as we claimed earlier is summarized in Lemma 1.

Lemma 1. *Let $d = t$ and the selected k patterns be $p_1, p_2, \ldots, p_k$ (significance decreasing order). If the distance satisfies triangle inequality, then for each p_i and p_j, $S(p_i|p_j) \ge d$ and $S(p_j|p_i) \ge d$; and for each pattern q within the circle of pattern p_i, $S(q|p_i) \le 3d + 5\epsilon$.*

Sketch of Proof. See Appendix. ■

The following theorem shows that Algorithm 2 has a performance guarantee for the *MMS* problem.

Theorem 2. *Let the k-pattern set returned by Algorithm 2 be $\mathcal{P}^k$, and the optimal pattern set be $\mathcal{O}^k$. If the distance measure satisfies triangle inequality, we have:*

$$G_{ms}(\mathcal{O}^k) \le (6 + \frac{10k}{B} + \log k) G_{ms}(\mathcal{P}^k).$$

Sketch of Proof. See Appendix. ∎

By setting $B \geq k$, the performance bound of algorithm 2 for *MMS* problem is $O(\log k)$, while the additional factor on complexity (*i.e.*, $\log B$) does not introduce heavy computational cost. In fact, as we will show in the experiments, the running time of Algorithm 2 is similar to that of Algorithm 1.

5. EXPERIMENTAL RESULTS

To test the performance of the proposed algorithms, we design two sets of experiments. The first examines the quality of extracted top-k patterns, and the second measures the computational performance of the proposed methods. For simplicity, we refer Algorithm 1 for *maximal average significance* as *MAS*, Algorithm 1 for *maximal marginal significance* as *MMS*, and Algorithm 2 (with bound) for *maximal marginal significance* as *MMSb*. We use *SIG* to refer to the method extracting top-k patterns completely based on significance (without considering redundancy). In all experiments, the number of intervals for the binary search in *MMSb* is set as $B = k$.

5.1 Quality of Top-K Patterns

Here we demonstrate two case studies that use our proposed methods: (1) *document theme extraction*, and (2) *correlation-directed disk block prefetch*. For each case study, we discuss pattern generation, significance measure, distance measure, and quality evaluation.

5.1.1 Document Theme Extraction

Theme discovery uses knowledge about the meaning of words in a text to identify broad topics covered in a document [3, 15]. One way to find themes from text document is to extract the frequent patterns of term occurrence. For example, a frequent pattern of "database management" indicates that the document might be related to a collection of database papers, whereas a frequent pattern like "red cross" might identify the topic of the documents as aid and relief. In this case study, we show how to apply our methods to discovering redundancy-aware top-k term occurrence patterns.

Pattern Generation: A document collection is constructed by a mixture of documents of four topics: 386 news articles about Tsunami, 367 research papers about data mining, 350 research papers about bioinformatics, and 347 blog articles about iPod Nano. A document is broken into sentences as transactions. We mine sequential patterns [27] with a minimum support of 0.02%, and $8,718$ patterns are generated.

Significance and Distance Measure: A pattern's significance is modeled by a *tf-idf* scoring function similar to the Pivoted Normalization weighting based document score [23]. Specifically, given a theme pattern $p = w_1 ... w_t$, the significance is defined by

$$S(p) = \sum_{i=1}^{t} \frac{1 + ln(1 + ln(tf_i))}{(1-s) + s\frac{dl}{avdl}} \cdot ln\frac{N+1}{df_i},$$

where tf_i equals the support of the pattern p, df_i is the inverse sentence frequency of word w_i in the whole transaction set, dl is the average sentence length associated with P, $avdl$ is the overall average sentence length and s is a parameter. Given two patterns, p_1 and p_2, we use the Jaccard distance measure [13]:

$$D(p_1, p_2) = 1 - \frac{|TS(p_1) \cap TS(p_2)|}{|TS(p_1) \cup TS(p_2)|},$$

where $TS(p_1)$ is the set of transactions containing pattern p_1.

Quality Evaluation: We run *SIG*, *MAS*, *MMS* and *MMSb* on the original collection of $8,718$ themes to extract top-10 results, which are displayed in Table 1. Without considering redundancy, the top-10 results returned by *SIG* only consist of two valuable themes (themes 1 and 4), and all the others are redundant. *MMS* and *MMSb* report the identical results, where all 10 themes have high significance score and are different from each other. There are two redundant themes in *MAS*. This suggests that the redundancy penalty by *MAS* formulation is not enough, and some theme patterns whose high significance scores compensate the redundancy penalties can still survive.

5.1.2 Correlation Directed Prefetch

Block correlations are common semantic patterns in storage systems [14]. Correlated blocks tend to be accessed relatively close to each other in an access stream. Exploring these correlations is very useful for improving the effectiveness of storage caching, pre-fetching, and data layout. Particularly, at each access, a storage system can pre-fetch correlated blocks into its storage cache so that subsequent accesses to these blocks do not need to access disks, which is several orders of magnitude slower than accessing directly from a storage cache. A *correlation pattern* is a rule in the form of "$b_{35}b_{100} \rightarrow b_{9039}$" implying that if disk block b_{35} and b_{100} are accessed sequentially, then disk block b_{9039} will be pre-fetched (note there is always only one block-id at the right-hand side of a rule). Since the computer resources are limited, our task is to extract top-k important rules for prefetch purposes.

Pattern Generation: We use the rules provided by [14]. The experiment uses a set of real system traces, Cello-92, collected at the Hewlett-Packard Laboratories [20]. It captured all low-level disk I/Os performed on Cello, which is a timesharing system used by a group of researchers at the HP Labs to do simulation, compilation, editing, and e-mail. The traces include the accesses to 8 disks. Long trace sequences are broken into fixed-size short sequential transactions (in our experiment, the window size is 50). We mine sequential patterns from the transformed transaction database and $276,054$ rules are generated.

Significance and Distance Measure: The significance of a rule should be measured by the performance gain with its existence. The model of cost-benefit of prefetching could be very complicated. Here we adopt a simplified yet effective measure [14]. Given a rule $l \rightarrow r$, the significance of this rule is $|TS(l, r)|$, where $TS(l, r)$ is the set of transactions having l followed by r. Given two rules, "$rule_1 : l_1 \rightarrow r_1$" and "$rule_2 : l_2 \rightarrow r_2$", the

Table 1: Top-10 Document Themes

Top-k	*SIG*	*MAS*	*MMS*	*MMSb*
1	permission make digital copy personal grant without fee distribute commercial full citation	permission make digital copy personal grant without fee distribute commercial full citation	permission make digital copy personal grant without fee distribute commercial full citation	permission make digital copy personal grant without fee distribute commercial full citation
2	permission make digital copy personal distribute commercial full citation	database manage database application mine algorithm keyword	database manage database application mine algorithm keyword	database manage database application mine algorithm keyword
3	permission make digital copy personal copy fee	pattern recognition design method classify evaluate	pattern recognition design method classify evaluate	pattern recognition design method classify evaluate
4	database manage database application mine algorithm keyword	information retrieval storage information search keyword	information retrieval storage information search keyword	information retrieval storage information search keyword
5	database manage database mine algorithm keyword	permission make digital copy personal distribute commercial full citation	artificial intelligence learn general term algorithm experimentation	artificial intelligence learn general term algorithm experimentation
6	database manage database application mine algorithm	artificial intelligence learn general term algorithm experimentation	international federate red cross red crescent	international federate red cross red crescent
7	database manage database application mine keyword	international federate red cross red crescent	australia prime minister john howard australia	australia prime minister john howard australia
8	database manage database mine term algorithm	australia prime minister john howard australia	indonesia president susilo bambang yudhoyono	indonesia president susilo bambang yudhoyono
9	database manage database mine term keyword	database manage database application mine keyword	deputy defense secretary paul wolfowitz	deputy defense secretary paul wolfowitz
10	database manage database application mine term	indonesia president susilo bambang yudhoyono	thailand prime minister thaksin shinawatra	thailand prime minister thaksin shinawatra

distance measure is defined as follows:

$$D(rule_1, rule_2) = \begin{cases} 1 & , r_1 \neq r_2, \\ 1 - \dfrac{|TS(l_1, r_1) \cap TS(l_2, r_2)|}{|TS(l_1, r_1) \cup TS(l_2, r_2)|}, & r_1 = r_2. \end{cases}$$

If two rules have different block-ids at the right-hand side, then they are not related to each other. Otherwise, these two rules trigger the same pre-fetching target. We compare the support sets of these two rules. If the overlap is significant, then the relative significance of one rule with respect to the other is small.

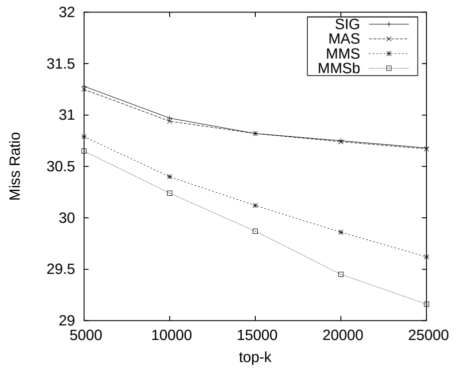


Figure 6: Miss Ratio w.r.t top-k

Quality Evaluation: We run *SIG*, *MAS*, *MMS*, and *MMSb* on the original collection of 276,054 rules to extract top-k rules, which are further fed into a simulation system [14]. The performance is evaluated by *miss ratio* (Fig. 6) and *response time* (Fig. 7). We observe (1) both *MMS* and *MMSb* perform much better than *SIG*, indicating that the redundancy-aware top-k patterns contain more valuable information; (2) the *MMSb* method is better than *MMS*, which is consistent with our claim that *MMSb* is more robust; and (3) *MAS* is almost identical to *SIG*. This is because in this experiment, k is relative large, whereas redundancy only exists among very limited number of patterns (*i.e.*, only the rules that have the same right-hand side are possibly redundant to each other). Averaging by a very large number of k makes the redundancy penalty negligible.

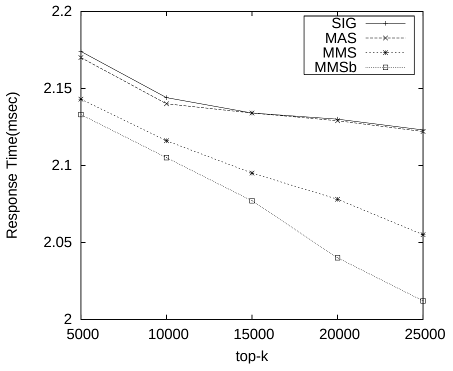


Figure 7: Response Time w.r.t top-k

5.2 Computational Performance

Here we examine the computational performance of the two proposed greedy algorithms for *MMS*. We run the experiments on the document theme data set. The computation times w.r.t. different top-k values are shown in Fig. 8. Given a collection of patterns, both algorithms scale well with respect to k. Although *MMSb* has higher complexity in the worst case, its running time is comparable to *MMS*. This is because (1) it generally stops early in each trial r where we try to find k patterns, thus the complexity of each iteration is less than $O(kn)$; and (2) a pattern does not participate in further computation as soon as it is removed (while in *MMS* each pattern will be compared with all the selected k patterns).

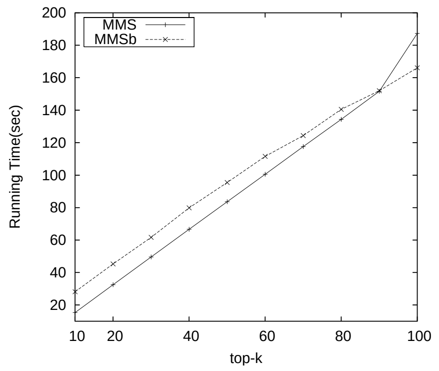


Figure 8: Computation Time w.r.t top-k

6. RELATED WORK

In Section 1, we have discussed the connection of our work with previous pattern compression (summarization) approaches and database top-k query processing. A closely related work is the pattern ordering problem studied in [16], where the authors also compute top-k patterns. Their criterion of the top-k pattern set is to provide best frequency estimation of those patterns that are not selected. Thus the objective function to evaluate the k pattern set is well defined. Our problem definition is more general since we do not assume any specific application. The greedy algorithm used in [16] is similar to Algorithm 1.

Our work is also related to document retrieving and ranking problem in Information Retrieval [5, 21]. The formulation of *MMS* is a generalization of *maximal marginal relevance* heuristic [5]. Different from techniques in IR where results are generally evaluated by user study, we propose explicit objective functions and develop an approximate algorithm with the near optimal solution.

The problem of *MAS* is identical to the maximum dispersion problem in graph algorithm. Ravi *et al.* [19] show that the bound of performance guarantee of any polynomial approximation is at least 2 and Algorithm 1 achieves this. The problem of *MMS* is related to finding a minimum spanning tree in a subgraph. Finding subset *maximizing* the minimum weight of a combinatorial structure was first proposed by Halldorsson *et al.* [9]. They give approximation algorithms in the metric undirected graph, where only edge weights exist. Our problem is different because patterns form a node-weighted as well as the edge-weighted graph.

7. CONCLUSIONS

To extract redundancy-aware top-k patterns, we examined two problem formulations: *MAS* and *MMS*. We studied a unified greedy approach to compare these two functions and show that *MMS* is a reasonable formulation to our problem. We further present an improved algorithm for *MMS* and show that the performance is bounded by $O(\log k)$. We present two case studies to examine the performance of our proposed approaches. Both *MMS* algorithms are able to find high-significant and low-redundant top-k patterns. Particularly, in block correlation experiments, we observe that our improved algorithm performs better.

This study opens a new direction on finding both diverse and significant top-k answers to querying, searching, and mining, which may lead to promising further studies. One further issue is the formal study of the evaluation functions for a pattern set. Direct mining of top-k patterns from data is another promising direction.

8. APPENDIX

Sketch of Proof for Lemma 1.

The first result is true because all p_i patterns are selected from k_t patterns. If $i > j$, we also have $S(p_i|p_j) \geq S(p_j|p_i) \geq d$. To prove the second result, we first show two related claims. For simplicity, we use d_{12} and s_1 to denote $D(p_1, p_2)$ and $S(p_1)$, respectively.

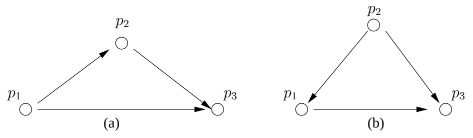


Figure 9: Two directed triangles

If the distance measure satisfies triangle inequality, then given a directed triangle as shown in Fig. 9(a), $S(p_2|p_1) + S(p_3|p_2) \geq S(p_3|p_1)$ (*Claim 1*); and given a directed triangle as shown in Fig. 9(b), $S(p_1|p_2) + S(p_3|p_2) \geq S(p_3|p_1)$, where $s_1 \geq s_3$ (*Claim 2*).

The proof of these two claims are similar. We show one case for claim 1. If $s_1 \geq s_2 \geq s_3$, we have $S(p_2|p_1) + S(p_3|p_2) = d_{12}s_2 + d_{23}s_3 \geq d_{12}s_3 + d_{23}s_3 \geq d_{13}s_3 = S(p_3|p_1)$.

For each pattern q in the circle of p_i, assume q originally belongs to circle p_j, and both p_i and p_j belong to circle p_v^T. We have:

$$\begin{aligned}
S(q|p_i) &\leq S(q|p_j) + S(p_j|p_i) && (Claim 1)\\
&\leq S(q|p_j) + S(p_j|p_v^T) + S(p_i|p_v^T) && (Claim 2)\\
&\leq \overline{S(q|p_j)} + \overline{S(p_j|p_v^T)} + \overline{S(p_i|p_v^T)} + 3\epsilon\\
&\leq t + T + T + 3\epsilon \leq 3t + 5\epsilon. && \blacksquare
\end{aligned}$$

Sketch of Proof for Theorem 2.

Let us call the patterns in $\mathcal{P}^k$ greedy patterns and the patterns in $\mathcal{O}^k$ optimal patterns. The algorithm partitions all patterns in $\mathcal{P}$ into k groups. In each group, the most significant pattern is reported (let the pattern reported from group i be p_i). The edge weight between any p_i and p_j $(i, j \in \{1, 2, \ldots, k\})$ is at least d. We have $G_{ms}(\mathcal{P}^k) \geq S(p_1) + (k-1)d$, where p_1 is the most significant pattern.

Assume the k optimal patterns in $\mathcal{O}^k = \{q_1, q_2, \ldots, q_k\}$ are distributed in $k' \leq k$ groups. We create a spanning tree for $\mathcal{O}^k$ based on the following two rules. First, if there are multiple optimal patterns $q_1^i, q_2^i, \ldots, q_{ki}^i$ within group i, we locate the most significant pattern q_1^i and include edges $S(q_j^i|q_1^i)$ for all other patterns. According to Lemma 1, $S(q_j^i|q_1^i) \leq S(q_j^i|p^i) + S(q_1^i|p^i) \leq 6d + 10\epsilon$. The overall sum of weights inside k' groups is $(k - k')(6d + 10\epsilon)$.

Second, we further include edges between optimal patterns q_1^i to make a spanning tree on $\mathcal{O}^k$. This is achieved by an iterative procedure. Let the spanning tree corresponding to $G_{ms}(\mathcal{P}^k)$ be MST_p. We can decompose MST_p into $\lceil \frac{k'}{2} \rceil$ paths such that the two end nodes of each path are patterns p_i, whose group contains an optimal pattern q_1^i. We group k' optimal patterns into $\lceil \frac{k'}{2} \rceil$ pairs. In each pair (a, b), we include the edge $S(a|b)$ (or $S(b|a)$) if $S(b) \geq S(a)$ (otherwise). There are at most $\lceil \frac{k'}{2} \rceil$ edges that will be included. The sum of weights of the included edges is: $w(k') \leq w(MST_p) + k'(6d + 10\epsilon)$, where $w(MST_p)$ is the sum of edge weights on MST_p. In each pair (a, b), we remove the pattern whose significance value is smaller, and the larger one stays for the next iteration. Since we remove half number of patterns at each iteration, there will be at most $\log(k')$ iterations. When there is only one pattern left, a spanning tree over $\mathcal{O}^k$ is constructed. The overall sum of edge weights included in this procedure is: $w(k') + w(\frac{k'}{2}) + \ldots + w(2) \leq \log(k')w(MST_p) + k'(6d + 10\epsilon)$.

Since $G_{ms}(\mathcal{O}^k)$ is the minimum score of all spanning trees on $\mathcal{O}^k$, we have $G_{ms}(\mathcal{O}^k) \leq G'_{ms}(\mathcal{O}^k)$. Because p_1 is the globally most significant pattern, $max_{i=1}^k S(q_i) \leq S(p_1)$. Furthermore, $G_{ms}(\mathcal{P}^k) = S(p_1) + w(MST_p) \geq S(p_1) + (k-1)d$, we have $d \leq \frac{1}{k-1}(G_{ms}(\mathcal{P}^k) - S(p_1)) \leq \frac{1}{k}G_{ms}(\mathcal{P}^k)$. Finally, fr

$B\epsilon = S(p_1)$, we have $k\epsilon = \frac{k}{B}B\epsilon = \frac{k}{B}S(p_1) \leq \frac{k}{B}G_{ms}(\mathcal{P}^k)$. Combining all of the above, we have:

$$
\begin{aligned}
& G_{ms}(\mathcal{O}^k) \leq G'_{ms}(\mathcal{O}^k) \\
& \leq \max_{i=1}^{k} S(q_1^i) + k(6d + 10\epsilon) + \log(k')w(MST_p) \\
& \leq S(p_1) + 6kd + 10k\epsilon + \log(k')(G_{ms}(\mathcal{P}^k) - S(p_1)) \\
& \leq S(p_1) + (6 + \frac{10k}{B} + \log k)G_{ms}(\mathcal{P}^k) - \log k S(p_1) \\
& \leq (6 + \frac{10k}{B} + \log k)G_{ms}(\mathcal{P}^k). \qquad \blacksquare
\end{aligned}
$$

9. REFERENCES

[1] F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. *KDD'04*.

[2] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated Ranking of Database Query Results. *CIDR'03*.

[3] P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[4] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. *PKDD'02*.

[5] J. Carbonell and J. Coldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. *SIGIR'98*.

[6] S. Chaudhuri and L. Gravano Evaluating Top-k Selection Queries. *VLDB'99*.

[7] E. Epkut, T. Baptie, and B. Hohenbalken. The discrete p-maxian localtion problem. *Comput. & Opns. Res.*, 17:51–61, 1990.

[8] R. Hassin, S. Rubinstein, and A. Tamir. Approximation algorithms for maximum dispersion. *Operations Research Let.*, 21:133–137, 1997.

[9] M. Holldorsson, K. Iwano, N. Katoh, and T. Tokuyama. Finding subsets maximizing minimum structures. *SODA'95*.

[10] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *ICDM'02*.

[11] S. Jaroszewicz and D.A. Simovici. A general measure of rule interestingness. *PKDD'01*.

[12] S. Jaroszewicz and D.A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. *KDD'04*.

[13] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[14] Z. Li, Z. Chen, S. Srinivasan, and Y. Zhou. Mining block correlations in storage systems. *USENIX FAST'04*.

[15] Q. Mei and C. Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. *KDD'05*.

[16] T. Mielikäinen and H. Mannila. The pattern ordering problem. *PKDD'03*.

[17] D. Mount. Bioinformatics: Sequence and genome analysis. *Cold Spring Harbor Lab.*, 2001.

[18] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. *ICDT'99*.

[19] S. Ravi, D. Rosenkrantz, and G. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42:299–310, 1994.

[20] C. Ruemmler and J. Wilkes. Unix disk access patterns. *USENIX'93*.

[21] X. Shen and C. Zhai. Active feedback in ad-hoc information retrieval. *SIGIR'05*.

[22] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. Knowledge & Data Engineering*, 8:970–974, 1996.

[23] A. Singhal. Modern information retrieval: A brief overview. *Bull. IEEE CS Tech. Comm. Data Eng.*, 24(4):35–43, 2001.

[24] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. *KDD'02*.

[25] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. *VLDB'05*.

[26] X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: A profile-based approach. *KDD'05*.

[27] X. Yan, J. Han and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. *SDM'03*.