Mining Significant Subspaces

Anthony J.T. Lee
Department of Information Management, National Taiwan University

Ming-Chih Lin
Department of Information Management, National Taiwan University

Yun-Ru Wang
Department of Information Management, National Taiwan University

Kuo-Tay Chen
Department of Accounting, National Taiwan University

Abstract

As both the number of dimensions increases, existing clustering methods in full feature space are not appropriate to cluster data in databases. Thus, the subspace clustering has attracted more and more attention recently. In this paper, we propose a novel method to mine significant subspaces from all frequent subspaces, where a subspace is frequent if it contains enough data points. The proposed method consists of three phases. First, we generate all frequent 2-dimensional subspaces. Second, we recursively combine frequent k-dimensional subspaces to generate frequent (k+1)-dimensional subspaces, $k \geq 2$. Finally, we adopt a greedy algorithm to summarize the frequent subspaces generated and select the significant ones. The experimental results show that the proposed method has better quality and coverage than DUSC, and better quality than FIRES.

Key words : subspace mining, subspace clustering, frequent subspace, data mining, greedy algorithm

重要子空間之資料探勘

李瑞庭
國立臺灣大學資訊管理學系

林明志
國立臺灣大學資訊管理學系

王韻茹
國立臺灣大學資訊管理學系

陳國泰
國立臺灣大學會計學系

摘要

隨著資料維度的增加，現有利用全部資料維度的分群方法，已經不適用於分析高維度的資料。因此，近年來子空間分群的方法愈來愈受重視。在本篇論文中，我們提出一個新的方法以探勘重要的子空間。我們所提出的方法包括三個步驟，首先，我們將所有的資料點投影到二維空間，並產生許多頻繁子空間；然後，我們利用遞迴的方式結合這些頻繁子空間，以形成更大的頻繁子空間；最後，我們採用貪婪演算法做總結，從所產生的頻繁子空間中選出重要的子空間。實驗結果顯示，我們所提出的方法在品質方面優於FIRES，在涵蓋率與品質方面，皆優於DUSC。

關鍵字：子空間探勘、子空間分群、頻繁子空間、資料探勘、貪婪演算法

1. INTRODUCTION

Clustering is an important technique to find the relationships in many applications such as customer behavior analysis, document classification, and Web log analysis (Parsons et al. 2004; Patrikainen et al. 2006). As both the number of dimensions increases, existing methods of clustering data in full feature space are not appropriate for these applications (Lu et al. 2007). Thus, the subspace clustering has attracted more and more attention recently.

The subspace clustering methods can be classified into two classes: bottom-up and top-down, depending on the search direction (Lu et al. 2007). The bottom-up methods include CLIQUE (Agrawal et al. 1998), ENCLUS (Cheng et al. 1999), MAFIA (Goil et al. 1999), SCHISM (Sequeira & Zaki 2004), IBUSCA (Glomba & Markowska-Kaczmar 2006), FIRES (Kriegel et al. 2005), and DUSC (Assent et al. 2007). CLIQUE (Agrawal et al. 1998) is a density-based method which partitions each dimension into several intervals of equal length, and then uses an Apriori-like approach to find interested subspaces. Based on CLIQUE, ENCLUS (Cheng et al. 1999) uses entropy to cope with the criterions of coverage, density and correlation, where the coverage increases as the entropy decreases. MAFIA (Goil et al. 1999) exploits histograms to decide how many grids to be created and uses an adaptive grid-based algorithm to partition the dimensions. SCHISM (Sequeira & Zaki 2004) adopts the support and Chernoff-Hoeffding bounds as the density thresholds and searches for maximal subspaces in a depth-first search manner. IBUSCA (Glomba & Markowska-Kaczmar 2006) is a grid-based algorithm which needs a parameter to determine the density threshold, where the data space is split based on histograms to form dense subspaces. Based on DBSCAN, FIRES (Kriegel et al. 2005) is a generic framework for finding the subspaces of high-dimensional data. DUSC (Assent et al. 2007) adopts different density thresholds for different subspaces and finds the S-connected clusters, where two data points in a cluster are connected to each other if the distance between them is within a user-specified threshold.

Generally speaking, bottom-up methods take advantage of the downward closure property of density to reduce the search space. The downward closure property of density means that if a (k+1)-dimensional subspace R is dense, all k-dimensional subspaces of R should be dense, $k \geq 2$. Bottom-up methods first create bins for each dimension and select those bins with densities above a given threshold. Adjacent dense subspaces (or bins) are then combined to form clusters. The methods proceed until no more dense subspaces can be found. However, one cluster may be mistakenly reported as two smaller clusters. The nature of the bottom-up methods leads to overlapping clusters, where one instance can be in zero or more clusters.

Top-down methods include PROCLUS (Aggarwal et al. 1999), ORCLUS (Aggarwal & Yu

2000), FINDIT (Woo & Lee 2002), and δ-Clusters (Yang et al. 2002). PROCLUS (Aggarwal et al. 1999) partitions the data points to axis-aligned subspaces and refines them by a hill-climbing approach. ORCLUS (Aggarwal & Yu 2000) is similar to PROCLUS but uses random sampling to improve the processing time. It uses the method of singular value decomposition to find arbitrarily-oriented clusters. FINDIT (Woo & Lee 2002) introduces the dimension-oriented distance as a distance measure, where the subspaces with higher number of dimensions are more meaningful. δ-Clusters (Yang et al. 2002) uses the Pearson correlation to measure coherence among all data points. It separates data points into several clusters, and randomly swaps the points in different clusters to improve the quality of clusters. The top-down methods start by finding an initial approximation of the clusters in full feature space with equally weighted dimensions. Next, each dimension is assigned a weight for each cluster. The updated weights are then used in the next iteration to regenerate the clusters. This approach requires multiple iterations of expensive clustering algorithms in the full set of dimensions. Many of the implementations of this strategy use a sampling technique to improve performance. Top-down algorithms divide the dataset into several partitions so that each data point is assigned to only one cluster. Parameter tuning is necessary in order to get meaningful results. Often the critical parameters for top-down algorithms are the number of clusters and the size of subspace, which are often difficult to determine ahead of time. Moreover, top-down algorithms tend to find clusters of the same or similar size.

The subspace clustering methods mentioned above cluster data points into groups by merging or splitting dimensions. The top-down methods do not allow overlapping clusters, where every data point belongs to just one cluster. But in real world, it is not appropriate to assign a data point to only one cluster (Kriegel et al. 2005). In addition, most bottom-up subspace mining methods find interested subspaces by pruning the potential subspaces according to monotonicity. That is, they only take a local view of subspaces to resolve this problem since they do not globally generate all potential subspaces to find the interested ones.

Therefore, in this paper, we propose a novel method to mine significant subspaces. The proposed method consists of three phases. First, we generate all frequent 2-dimensional subspaces. Second, we recursively combine frequent k-dimensional subspaces to generate frequent (k+1)-dimensional subspaces, $k \geq 2$. Finally, we adopt a greedy approach (Michael 1996) to summarize all frequent subspaces found and select the significant ones.

The contributions of this paper are summarized as follows: (1) We propose a novel method to mine significant subspaces. (2) We adopt a greedy approach to summarize all frequent subspaces and select the significant ones. (3) The experimental results show that the proposed method has better quality and coverage than DUSC, and better quality than FIRES.

The rest of this paper is organized as follows. Section 2 describes preliminary concepts and problem definitions. Section 3 discusses the algorithm in detail and demonstrates how it works.

Section 4 shows the performance evaluation. Finally, the conclusions and future work are made in Section 5.

2. PRELIMINARIES AND PROBLEM DEFINITIONS

Consider an input dataset D containing n data points (points for short) P_1, P_2, $\cdots$,P_n in an m-dimensional data space. Each point is represented by (x_1, x_2, $\cdots$,x_m), where $x_i \in R$ and $1 \leq i \leq m$. We first project every point onto a pair of dimensions (or dimension pair). Since there are m dimensions in the data space, we have $m*(m-1)/2$ dimension pairs. A point, (x_1, x_2, $\cdots$, x_m), projected to the projected space formed by a dimension pair (d_i, d_j) is denoted as (x_i, x_j), $1 \leq i < j \leq m$.

To cluster the projected points into several groups in a two-dimensional projected space, we partition each dimension into several bins (or intervals). That is, the projected space is divided into several cells, where each cell may contain a group of points (point group). A cell denoted by (i, j) is located at the ith interval of the first dimension and the jth interval of the second dimension. If the number of points in a cell is not less than a user-specified density threshold ϱ, the cell is *frequent*. A cell (i, j) is *adjacent* to another cell (i', j') if $|i-i'| \leq 1$ and $|j-j'| \leq 1$. The points in a frequent cell can be combined with those in the other adjacent frequent cells to form a larger point group. Then, we obtain a list of point groups, each of which forms a subspace. For example, Figure 1 shows that the points are combined into two point groups (or subspaces), where the projected space is divided into 4×4 = 16 cells. Similarly, we can obtain a list of subspaces for each dimension pair.

If a subspace is formed by l dimensions, it is called an l-subspace. To find the number of points in an l-subspace, we can intersect the point groups of the corresponding $l*(l-1)/2$ projected dimension pairs. Consequently, every subspace is associated with a point group. A subspace is *frequent* if the number of points in the subspace is not less than a user-specified minimum support threshold σ.

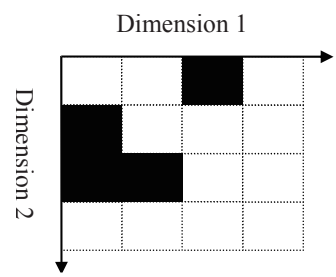

Figure 1: Partitioning the points into two groups.

For example, assume that $\sigma = 2$. Given five points in a 3-dimensional data space, we project them into dimensions (d_1, d_2) and (d_1, d_3). The points are divided into two frequent subspaces $G_{11} = \{P_2, P_5\}$ and $G_{12} = \{P_1, P_3, P_4\}$ for the projected space formed by (d_1, d_2), $G_{21} = \{P_1, P_4\}$ and $G_{22} = \{P_2, P_3, P_5\}$ for the projected space formed by (d_1, d_3) as shown in Figure 2. Since $G_{31} = G_{11} \cap G_{22} = \{P_2, P_5\}$ and $G_{32} = G_{12} \cap G_{21} = \{P_1, P_4\}$, G_{31} and G_{32} both contain two points. Thus, they are frequent 3-subspaces.

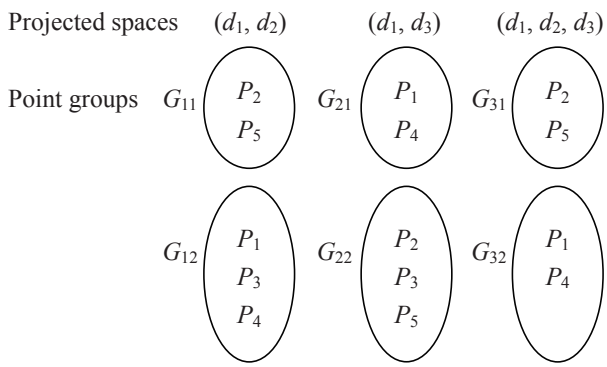

Figure 2: Generating a 3-subspace.

The objective of the proposed method is to find all frequent subspaces in a dataset with respect to the user-specified density and minimum support thresholds, and then select the significant ones from the frequent subspaces found.

3. THE PROPOSED METHOD

The proposed method consists of three phases: finding frequent 2-subspaces, finding frequent k-subspaces, and subspace summarization. These phases will be described in detail in the following subsections.

3.1 Finding frequent 2-subspaces

To find all frequent 2-subspaces, for each dimension pair, we project all points in the dataset D onto the corresponding 2-dimensional space which is partitioned into $p \times p$ cells. Next, we check if the number of points in each cell is not less than a density threshold ϱ. If this is the case, the cell is frequent. For each frequent cell, the points in the cell are combined with those in the other adjacent frequent cells to form a larger point group. This step is repeated until no more adjacent frequent cells can be combined. Finally, we obtain all frequent 2-subspaces in the 2-dimensional space. The procedure of finding frequent 2-subspaces is shown in Figure 3.

For example, let us consider a dataset containing ten points in a 4-dimensional space, where the projected points in each projected space are shown in Table 1. Assume that $\varrho = 2$ and $\sigma = 2$. To find frequent subspaces in the projected space formed by (d_1, d_2), the projected space is partitioned into 4×4 cells. The smallest and largest values of the first dimension of the points in Table 1 are 0 (P_1) and 10 (P_{10}), respectively. Thus, the first dimension is equally partitioned into four intervals: [0, 2.5), [2.5, 5), [5, 7.5), and [7.5, 10]. Similarly, the second dimension is equally partitioned into four intervals: [0, 5), [5, 10), [10, 15), and [15, 20]. In Figure 4, cell C_1 contains {P_1, P_2}, C_2 contains {P_3, P_4}, C_3 contains {P_5, P_6, P_7}, C_4 contains {P_8}, C_5 contains {P_9}, and C_6 contains {P_{10}}. Since cells C_1, C_2 and C_3 are frequent and adjacent to each other, we can combine the points in these cells together and form a point group {P_1, P_2, P_3, P_4, P_5, P_6, P_7}. There are seven points in the point group. Thus, the point group (or subspace) is frequent. Similarly, we can obtain all frequent 2-subspaces as shown in Figure 5, where $FG(d_i, d_j)$ contains all frequent 2-subspaces in the projected space formed by (d_i, d_j).

Procedure: find2FS
Input: all points in the dataset which is partitioned into $p \times p$ cells, a density threshold ϱ.
Output:a list of point groups.

```
(1)   for each dimension pair do
(2)     Sort the projected points and project them onto the cells;
(3)     for each cell do
(4)       if the number of data points in the cell ≥ ϱ then
(5)         The cell is frequent;
(6)       endif;
(7)     endfor;
(8)     for each frequent cell do
(9)       Combine the points in this cell with those in other adjacent frequent cells
            to form a larger point group until no more adjacent frequent cells can be
            combined;
(10)    endfor;
(11)  endfor;
```

Figure 3: The find2FS procedure.

Lemma 1. The time complexity of the find2FS procedure is bounded by $O(m^2*n*\log(n))$, where m is the number of dimensions in the space and n is the number of points in the dataset.

Proof: For each dimension pair, we can project all the points onto the space formed by the dimension pair. Then, we sort the projected points by the first dimension and then by the second dimension. The time complexity of sorting these projected points is bounded by $O(n*\log(n))$.

The time complexity of projecting the projected points onto the cells is bounded by O(n). Thus, the time complexity of step (2) is bounded by $O(n*\log(n))$. The time complexity of steps (3)-(7) is bounded by $O(p^2)=O(1)$, where p is a user-specified constant. Since the points on each cell is sorted, the time complexity of merging the points on two adjacent frequent cells together is bounded by $O(n_1+n_2)$, where n_1 and n_2 are the number of points in the first and second cells, respectively. Thus, the time complexity of steps (8)-(10) is bounded by O(n). Since the number of dimensional pairs is bounded by O(m^2), the time complexity of the find2FS procedure is bounded by $O(m^2*(n*\log(n)+p^2+n))=O(m^2*n*\log(n))$.

Table 1: Projecting the points onto projected spaces.

ID	Data point	(d_1, d_2)	(d_1, d_3)	(d_1, d_4)	(d_2, d_3)	(d_2, d_4)	(d_3, d_4)
P_1	(0, 0, 0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
P_2	(2.4, 3, 3.5, 2)	(2.4, 3)	(2.4, 3.5)	(2.4, 2)	(3, 3.5)	(3, 2)	(3.5, 2)
P_3	(1.5, 6, 11, 2)	(1.5, 6)	(1.5, 11)	(1.5, 2)	(6, 11)	(6, 2)	(11, 2)
P_4	(2.3, 9, 9, 7)	(2.3, 9)	(2.3, 9)	(2.3, 7)	(9, 9)	(9, 7)	(9, 7)
P_5	(3, 7, 2, 10)	(3, 7)	(3, 2)	(3, 10)	(7, 2)	(7, 10)	(2, 10)
P_6	(4, 6, 6, 11)	(4, 6)	(4, 6)	(4, 11)	(6, 6)	(6, 11)	(6, 11)
P_7	(3.5, 9.5, 10, 11)	(3.5, 9.5)	(3.5, 10)	(3.5, 11)	(9.5, 10)	(9.5, 11)	(10, 11)
P_8	(3.5, 18, 16, 12)	(3.5, 18)	(3.5, 16)	(3.5, 12)	(18, 16)	(18, 12)	(16, 12)
P_9	(6, 20, 9, 2)	(6, 20)	(6, 9)	(6, 2)	(20, 9)	(20, 2)	(9, 2)
P_{10}	(10, 13, 5, 12)	(10, 13)	(10, 5)	(10, 12)	(13, 5)	(13, 12)	(5, 12)

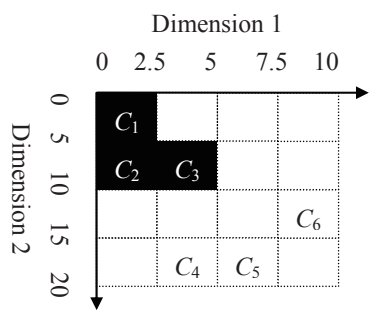

Figure 4: An example of getting point groups in cells.

3.2 Finding frequent k-subspaces

Let Dim(α) denote the dimensions spanned by a frequent subspace α. Two frequent k-subspaces α and β are *joinable* if Dim(α) and Dim(β) have k-1 dimensions in common, where $k\geq 2$. Next, let us consider how to use two joinable frequent k-subspaces α and β to generate a (k+1)-subspace γ, where Dim(γ) is Dim(α)$\cup$Dim(β). To find the points contained by γ, we can intersect the point group of α with that of β. If the number of points in γ is not less than σ, γ is a frequent (k+1)-subspace.

For example, Figure 6 shows how to use two joinable frequent k-subspaces α and β to generate a frequent (k+1)-subspace, where $\sigma = 2$. The point group of α contains P_1, P_2, and P_3, while the point group of β contains P_1, P_3, and P_4. We intersect both point groups and obtain the resultant point group which contains two points, P_1 and P_3. Since $\sigma = 2$, γ is a frequent (k+1)-subspace.

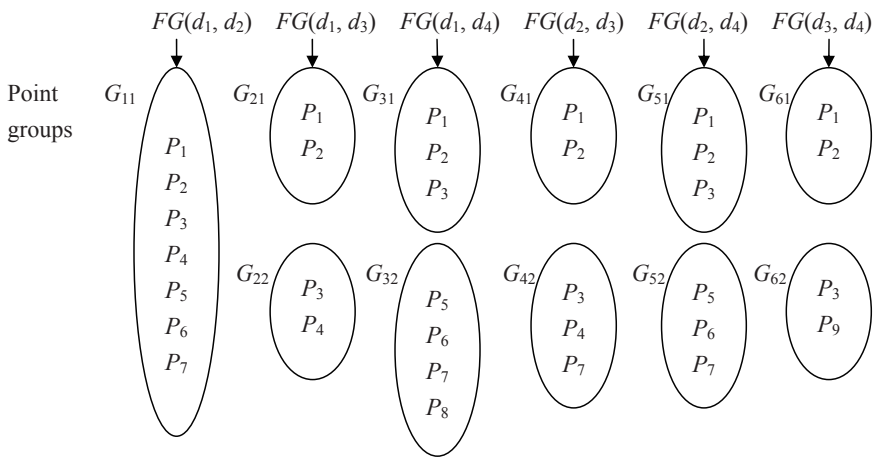

Figure 5: An example of finding frequent 2-subspaces.

We can join the joinable frequent subspaces to generate larger frequent subspaces in a depth-first search (DFS) manner: (1) For each frequent 2-subspace, we join it to the other joinable frequent 2-subspace to generate a 3-subspace and check if the 3-subspace is frequent. Thus, we find all frequent 3-subspaces, each of which is associated with a point group. (2) For each frequent k-subspace (k> 2), we join it to its joinable k-subspaces to generate frequent (k+1)-subspaces in a DFS manner. (3) Let $k = k$+1, we repeat steps 2-3 until no more frequent subspace can be generated. The procedure of finding frequent k-subspaces is shown in Figure 7.

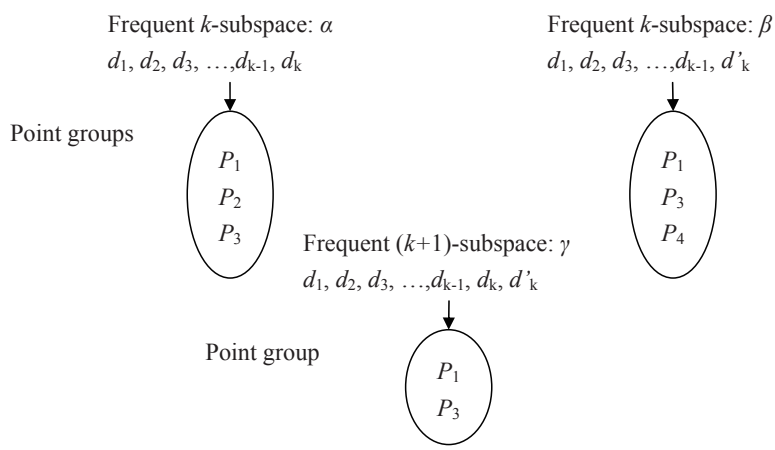

Figure 6: An example of generating frequent (k+1)-subspace.

Procedure: findKFS
Input: a joinable class containing frequent k-subspaces S, and a minimum support threshold σ.
Output:all frequent subspaces FS.

(1) **for each** k-subspace α in S do
(2) Let F_{k+1} be $\varnothing$;
(3) **for each** k-subspace β in S, which is joinable to α, $\beta>\alpha$ **do**
(4) Let Dim(α) be $d_1, d_2, \cdots, d_k$;
(5) Let Dim(β) be $d_1, d_2, \cdots, d'_k$;
(6) Let γ be the frequent (k+1)-subspace generated by joining α and β; That is, Dim(γ) contains $d_1, d_2, \cdots, d_k$, and d'_k;
(7) Intersect the point group of α with the point group of β;
(8) If the number of points in the point group obtained in step (7) is not less than σ, γ is frequent. Collect it into F_{k+1} and FS;
(9) **endfor;**
(10) Call findKFS(F_{k+1}, σ, FS);
(11) **endfor;**

Figure 7: The findKFS procedure.

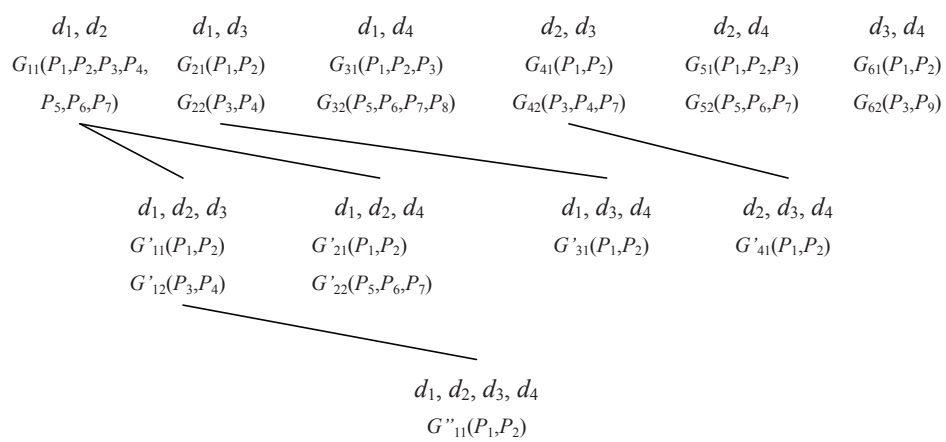

Figure 8. The frequent subspaces mined from the dataset shown in Table 1.

Let's consider the frequent subspaces shown in Figure 5. The frequent 2-subspaces G_{21} and G_{22} are joinable to G_{11}. By intersecting the point groups of G_{21} and G_{22} with the point group of G_{11}, we obtain two point groups $\{P_1, P_2\}$ and $\{P_3, P_4\}$, respectively. Since the minimum support threshold is 2, these point groups intersected are frequent. That is, we find two frequent 3-subspaces in dimensions (d_1, d_2, d_3). Similarly, we can find all frequent 3-subspaces and

frequent 4-subspaces as shown in Figure 8, where the edge between two frequent subspaces means that the child subspace is derived from the parent subspace.

Lemma 2. The time complexity of the findKFS procedure is bounded by $O(N*v*n)$, where n is the number of points in the dataset, v is the average number of joinable subspaces for each frequent subspace, and N is the number of frequent subspaces found.

Proof: The time complexity of steps (4)-(6) and (8) is bounded by $O(1)$. Since the points in subspaces α and β are sorted, the time complexity of merging two point groups together in step (7) is bounded by O(n). Thus, the time complexity of joining two frequent k-subspaces is bounded by $O(n)$. There are N frequent subspaces in total, each of which has v joinable subspaces on average. Therefore, the time complexity of the findKFS procedure is bounded by $O(N*v*n)$.

3.3 Subspace summarization

After finding all frequent subspaces, we use a greedy pattern summarization approach to summarize the subspaces. By doing so, we can reduce the number of frequent subspaces and select the significant ones.

We introduce ω as a weight and assign each frequent subspace a score by the greedy pattern summarization approach. The score is computed by the equation $Score(S) = \omega \times Q(S) + (1-\omega) \times V(S)$, where $Q(S)$ is the quality of the subspace S and $V(S)$ is the coverage of the subspace S. The quality is defined as $1-H/\log(c)$, where H is the entropy of S, and c is the number of different classes in the dataset. Dividing H by $\log(c)$ is used to normalize the entropy between 0 and 1. The coverage is defined as the number of points in the selected subspaces divided by the total number of points in the dataset.

First, we select the frequent subspace with the highest score. Next, each subspace is updated by eliminating the points contained by the selected subspaces so that the remaining points in a subspace do not overlap with those in the selected subspaces. Then, the score of each subspace is modified by using the points in the original subspace to compute the quality and using the remaining points in the updated subspace to compute the coverage. After the score of each subspace is modified, we continue to select the subspace with the highest score. The above steps are repeated until the score of every frequent space is less than a user-specified threshold τ. The procedure of selecting the significant subspaces is shown in Figure 9.

Let us consider the points shown in Table 1 again, where the class of each point is shown in Table 2. The frequent subspaces found are shown in Figure 8. Let $\omega = 0.5$. We first compute the score for each frequent subspace as shown in Table 3. Since the score of subspace G_{32} is the highest, G_{32} is added to *SFS*. Then, the scores of the other subspaces are updated as shown in Table 4, where the score of G_{31} is the highest. Thus, G_{31} is added to *SFS*. Then, the scores of

the other subspaces are updated as shown in Table 5. G_{11} is added to *SFS* and the scores of the remaining subspaces are updated. Let the score threshold τ be 0.1. Since the updated scores of the non-empty frequent subspaces are all less than τ, no more subspaces can be chosen. Finally, we obtain 3 subspaces as shown in Table 6. That is, we select 3 significant subspaces from 18 frequent ones.

Procedure: subspace summarization
Input: all frequent subspaces *FS*, a weight ω, and a score threshold τ.
Output:the selected frequent subspaces *SFS*.
(1) **for each** frequent subspace *S* in *FS* **do**
(2) Compute its score $Score(S)=\omega\times Q(S)+(1-\omega)\times C(S)$;
(3) **endfor;**
(4) Select the subspace with the highest score to *SFS*;
(5) **while** the score of any frequent space in *FS* is not less than τ **do**
(6) **for each** frequent subspace *S'* in *FS* do
(7) The subspace *S'* is updated by eliminating the data points which are contained by the subspaces in *SFS*;
(8) **if** the subspace does not contain any point **then**
(9) $Score(S')=0$;
(10) **else**
(11) $Score(S')=\omega\times Q(S')+(1-\omega)\times C(S')$, where *Q* is computed in the original subspace and *C* is computed in the updated subspace;
(12) **endif**;
(13) **endfor;**
(14) Select the subspace with the highest score to *SFS*;
(15) **endwhile;**

Figure 9: The subspace summarization procedure.

Lemma 3. The time complexity of the subspace summarization procedure is bounded by $O(M*(N+n))$, where *n* is the number of points in the dataset, *N* is the number of frequent subspaces found, and *M* is the number of significant subspaces selected.

Proof: The time complexity of steps (1)-(3) is bounded by $O(N)$. The time complexity of selecting the highest score in step (4) is bounded by $O(N)$, too. The time complexity of updating *S'* by eliminating the data points which are contained by the subspaces in *SFS* is bounded by $O(n_1+n_2)=O(n)$, where n_1 and n_2 are the number of points in *S'* and *SFS*, respectively. The time complexity of steps (8)-(12) is bounded by $O(N)$. Since the while-loop from step (5) to step

(15) is executed M-1 times, the time complexity of the subspace summarization procedure is bounded by $O(N+M*(N+n))= O(M*(N+n))$.

Theorem 1. The time complexity of the proposed method is bounded by $O(m^2*n*log(n)+N*v*n)$, where m is the number of dimensions in the space, n is the number of points in the dataset, N is the number of frequent subspaces found, and v is the average number of joinable subspaces for each frequent subspace.

Proof: Since the proposed method consists of three phases: finding frequent 2-subspaces, finding frequent k-subspaces, and subspace summarization, its time complexity is bounded by $O(m^2*n*\log(n)+N*v*n+M*(N+n))$, where M is the number of significant subspaces selected. However, $M<<N$ and $M<<n$. Therefore, the time complexity of the proposed method is bounded by $O(m^2*n*\log(n)+N*v*n)$.

Table 2: The classes of the points.

Data point	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Class	1	1	1	1	2	2	2	2	2	1

Table 3: The score of each frequent subspace in FS.

FS	Quality	Coverage	Score
$G_{11}\{P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$	0.015	0.7	0.357
$G_{21}\{P_1, P_2\}$	1	0.2	0.6
$G_{22}\{P_3, P_4\}$	1	0.2	0.6
$G_{31}\{P_1, P_2, P_3\}$	1	0.3	0.65
$G_{32}\{P_5, P_6, P_7, P_8\}$ (add to SFS)	**1**	**0.4**	**0.7**
$G_{41}\{P_1, P_2\}$	1	0.2	0.6
$G_{42}\{P_3, P_4, P_7\}$	0.082	0.3	0.191
$G_{51}\{P_1, P_2, P_3\}$	1	0.3	0.65
$G_{52}\{P_5, P_6, P_7\}$	1	0.3	0.65
$G_{61}\{P_1, P_2\}$	1	0.2	0.6
$G_{62}\{P_3, P_9\}$	0	0.2	0.1
$G'_{11}\{P_1, P_2\}$	1	0.2	0.6
$G'_{12}\{P_3, P_4\}$	1	0.2	0.6
$G'_{21}\{P_1, P_2, P_3\}$	1	0.3	0.65
$G'_{22}\{P_5, P_6, P_7\}$	1	0.3	0.65
$G'_{31}\{P_1, P_2\}$	1	0.2	0.6
$G'_{41}\{P_1, P_2\}$	1	0.2	0.6
$G''_{11}\{P_1, P_2\}$	1	0.2	0.6

Table 4: Updated scores of the frequent subspaces in FS.

Updated FS'	Updated quality	Updated coverage	Updated score
$G_{11}\{P_1, P_2, P_3, P_4\}$	0.015	0.4	0.207
$G_{21}\{P_1, P_2\}$	1	0.2	0.6
$G_{22}\{P_3, P_2\}$	1	0.2	0.6
$G_{31}\{P_1, P_2, P_3\}$ (add to *SFS*)	**1**	**0.3**	**0.65**
$G_{32}\{P_5, P_6, P_7, P_8\}$ (in *SFS*)	0	0	0
$G_{41}\{P_1, P_2\}$	1	0.2	0.6
$G_{42}\{P_3, P_4\}$	0.082	0.2	0.141
$G_{51}\{P_1, P_2, P_3\}$	1	0.3	0.65
$G_{52}\varnothing$	0	0	0
$G_{61}\{P_1, P_2\}$	1	0.2	0.6
$G_{62}\{P_3, P_9\}$	0	0.2	0.1
$G'_{11}\{P_1, P_2\}$	1	0.2	0.6
$G'_{12}\{P_3, P_4\}$	1	0.2	0.6
$G'_{21}\{P_1, P_2, P_3\}$	1	0.3	0.65
$G'_{22}\varnothing$	0	0	0
$G'_{31}\{P_1, P_2\}$	1	0.2	0.6
$G'_{41}\{P_1, P_2\}$	1	0.2	0.6
$G''_{11}\{P_1, P_2\}$	1	0.2	0.6

Table 5: Updated scores of the subspaces with a non-empty point group.

Updated FS'	Updated quality	Updated coverage	Updated score
$G_{11}\{P_4\}$ (add to *SFS*)	**1**	**0.1**	**0.55**
$G_{22}\{P_4\}$	1	0.1	0.55
$G_{31}\{P_1, P_2, P_3\}$ (in *SFS*)	0	0	0
$G_{32}\{P_5, P_6, P_7, P_8\}$ (in *SFS*)	0	0	0
$G_{42}\{P_4\}$	1	0.1	0.55
$G_{62}\{P_9\}$	0	0.1	0.05
$G'_{12}\{P_4\}$	1	0.1	0.55

Table 6: Selected frequent subspaces.

Selecting order	*SFS*	Quality	Coverage
1	**$G_{32}\{P_5, P_6, P_7, P_8\}$**	100%	40%
2	**$G_{31}\{P_1, P_2, P_3\}$**	100%	70%
3	**$G_{11}\{P_4\}$**	100%	80%

4. PERFORMANCE EVALUATION

In this section, we compared the proposed method with FIRES (Kriegel et al. 2005) and DUSC (Assent et al. 2007) by both synthetic and real datasets. All the algorithms were implemented using Microsoft Visual C++ 2005. All of experiments were performed on an IBM Compatible PC with Intel Core 2 Quad CPU Q6600 @ 2.40GHz, 2.0GB main memory, running on Windows XP Professional.

To evaluate the performance of the proposed method, we first analyzed the time complexity of the comparing methods. Based on DBSCAN, FIRES (Kriegel et al. 2005) is a generic framework for finding the subspaces of high-dimensional data. It first generates all 1-dimensional clusters using DBSCAN. The time complexity of this step is bounded by $O(m*n*\log(n))$, where m is the number of dimensions in the space and n is the number of points in the dataset. Then, it generates subspace clusters by grouping the best-merge-clusters together, where the time complexity of this step is bounded by $O(N_F^2)$, and N_F is the number of base clusters. Finally, it refines the subspace clusters generated by pruning the candidates and using DBSCAN to cluster subspaces again, where the time complexity is bounded by $O(N_F^2+N_F*n*\log(n))$. Therefore, the time complexity of FIRES is bounded by $O(m*n*\log(n)+N_F^2+N_F^2+N_F*n*\log(n))=O((m+N_F)*n*\log(n)+N_F^2)$.

DUSC (Assent et al. 2007) adopts different density thresholds for different subspaces and finds the S-connected clusters, where two data points in a cluster are connected to each other if the distance between them is within a user-specified threshold. The time complexity of finding the S-connected clusters is bounded by $O(N_D*m^2*n^2)$, where N_D is the number of frequent subspaces found. A cluster forms a subspace if its density is greater than a user-specified threshold and the number of data points in the cluster is larger than the minimum cluster size. After generating the subspace clusters, it prunes the redundant ones, where the time complexity of this step is bounded by $O(N_D^2)$. Therefore, the time complexity of DUSC is bounded by $O(N_D*m^2*n^2+N_D^2)$. The time complexities of the proposed method, FIRES, and DUSC are summarized in Table 7.

Table 7: The time complexities.

Method	Time complexity
Our method	$O(m^2*n*\log(n)+N*v*n)$
FIRES	$O((m+N_F)*n*\log(n)+N_F^2)$
DUSC	$O(N_D*m^2*n^2+N_D^2)$

4.1 Synthetic dataset

We use a synthetic dataset to test the scalability and efficiency of the proposed method. The synthetic data is generated by the methods similar to those used in (Agrawal et al. 1998; Cheng et al. 1999; Kriegel et al. 2005). The default settings of the parameters used in the data generator are shown in Table 8.

Table 8: The default settings.

Parameters	Default value
Number of points	50000
Number of dimensions	10
Number of cells	40×40
Density threshold	0.01
Minimum support	0.1

Figure 10 illustrates the execution time versus the number of dimensions, where the number of points is 50000, the number of cells is 40×40, the density threshold is 0.01, and the minimum support threshold is 0.1. When the number of dimensions increases, the execution times of both methods increase. However, the proposed method runs faster than FIRES and DUSC. This is because FIRES uses DBSCAN to cluster the neighboring points and DBSCAN spends much time in finding an appropriate radius of a cluster. On the other hand, it is quite time-consuming for DUSC to find the S-connected clusters. However, the proposed method uses the grid-based clustering method to find the frequent 2-subspaces and then uses the joinable classes and point groups to find the frequent k-subspaces in a DFS manner, $k > 2$. By using the joinable subspaces, the proposed method can localize the search space in a small number of point groups. Thus, the proposed method is more efficient than FIRES and DUSC.

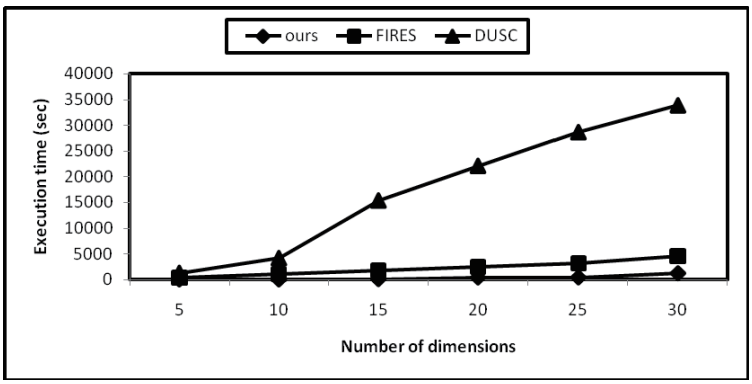

Figure 10: Execution time versus number of dimensions.

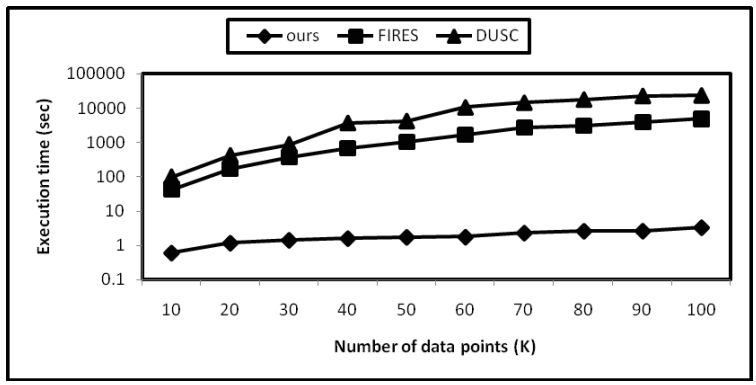

Figure 11: Execution time versus number of data points.

Figure 11 shows the execution time versus the number of points, where the number of dimensions is 10, the number of cells is 40×40, the density threshold is 0.01, and the minimum support threshold is 0.1. When the number of points increases, the execution times of these three methods increase linearly. However, the proposed method runs faster than FIRES and DUSC.

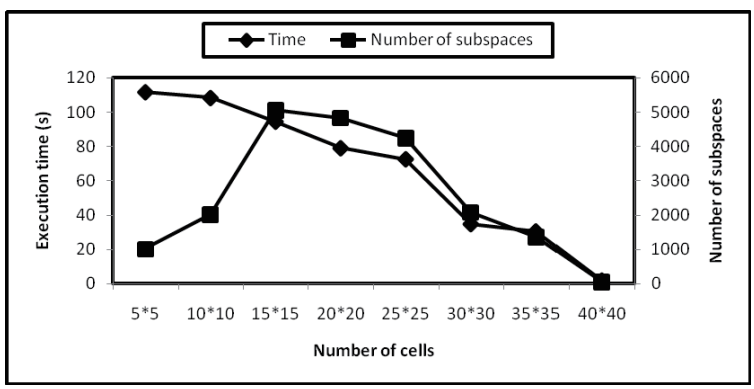

Figure 12: Execution time and number of subspaces versus number of cells.

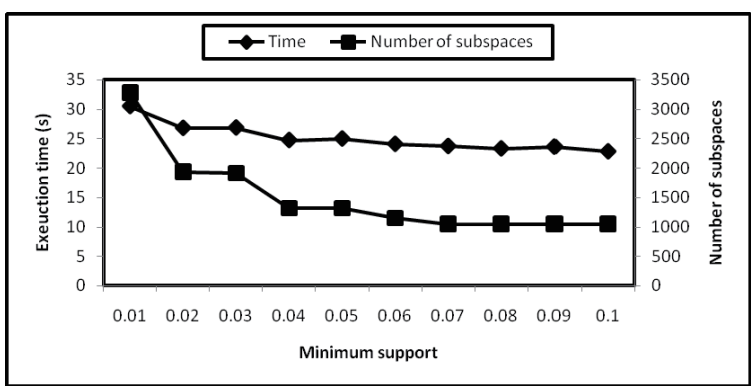

Figure 13: Execution time and number of subspaces versus minimum support.

Figure 12 shows the execution time and the number of subspaces versus the number of cells, where the number of cells is increased from 5×5 to 40×40. When number of cells increases, the execution time decreases. However, the number of frequent subspaces increases when the number of cells equals to 5×5, 10×10, and 15×15. This is because there are less candidate subspaces to be generated for the small number of cells. The number of frequent subspaces decreases when the number of cells is not less than 15×15. As the size of each cell decreases, many subspaces become infrequent under the given density and minimum support thresholds. The number of cells can be adjusted to generate the subspaces in different granularity.

Figure 13 illustrates the execution time and number of subspaces versus the minimum support threshold. Figure 14 shows the execution time and number of subspaces versus the density threshold. Both figures have similar tendency. That is, when the density or minimum support threshold increases, the number of subspaces and the execution time decrease. This is because both thresholds are used to decide whether a subspace is frequent or not. The larger the threshold is, the fewer frequent subspaces are.

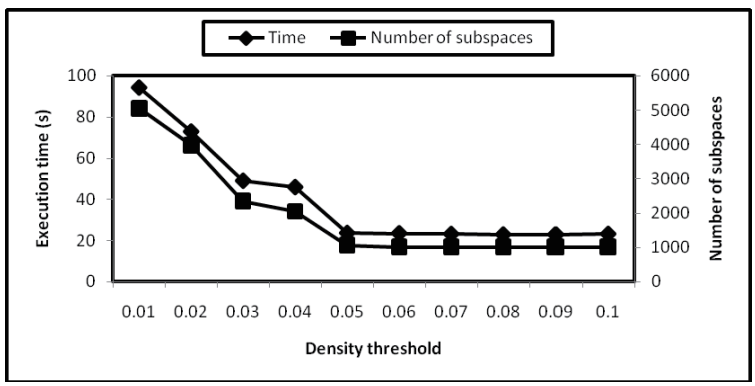

Figure 14: Execution time and number of subspaces versus density threshold.

In summary, the proposed method uses the grid-based clustering method to find the frequent 2-subspaces and then utilizes the joinable subspaces to find the frequent *k*-subspaces in a DFS manner, $k > 2$. By using the joinable subspaces, the proposed method can localize the search space in a small number of points groups. Thus, the proposed method is more efficient than FIRES and DUSC.

4.2 Real datasets

We compared the proposed method with FIRES (Kriegel et al. 2005) and DUSC (Assent et al. 2007) by six real datasets (Newman et al. 1998), namely, pendigits, glass, vowel, wine,

handwritten digit, and letter recognition. Since DUSC uses the quality and coverage to evaluate the quality of subspaces selected, we adopt the same measures to evaluate the proposed algorithm and the comparing algorithms.

The pendigits dataset has 16 dimensions and 7494 data instances, where all the instances in the dataset are classified into 10 classes. The glass dataset has 9 dimensions and 214 data instances, where all the instances in the dataset are classified into 6 classes. The vowel dataset has 10 dimensions and 990 data instances, where all the instances in the dataset are classified into 11 classes. The wine dataset has 13 dimensions and 178 data instances, where all the instances in the dataset are classified into 3 classes. The handwritten digit dataset has 256 dimensions and 1593 data instances, where all the instances in the dataset are classified into 10 classes. The letter recognition dataset has 16 dimensions and 20000 data instances, where all the instances in the dataset are classified into 26 classes.

Table 9: The results of the six datasets.

	Our method		DUSC		FIRES	
	Quality	Coverage	Quality	Coverage	Quality	Coverage
Pendigits	99%	75%	86%	74%	55%	100%
Glass	99%	95%	60%	87%	23%	98%
Vowel	92%	94%	82%	70%	0.003%	98%
Wine	100%	99%	68%	97%	2.9%	100%
Handwritten digit	96%	100%	33%	86%	5.7%	100%
Letter recognition	88%	83%	32%	81%	7.6%	100%

Table 10: Number of subspaces selected.

	Number of subspaces		
	Our method	DUSC	FIRES
Pendigits	132	14	378
Glass	7	7	2
Vowel	69	140	1
Wine	10	47	18
Handwritten digit	61	74	7
Letter recognition	98	96	46

Table 9 shows the results of the six datasets obtained by the proposed method, FIRES and DUSC. Table 10 illustrates the number of subspaces selected by these three methods. FIRES has less quality because it uses DBSCAN to cluster the neighboring data instances together. The subspaces obtained may contain many points of different classes. Thus, FIRES has bad quality

but good coverage. Generally speaking, the higher coverage will lead to the lower quality. This is because we need to include more data points in the selected subspaces in order to increase the coverage. Consequently, the possibility of a subspace containing the data points with different classes will increase. Thus, as the coverage is getting higher, the quality will become lower. For example, for the vowel and glass datasets, FIRES combines many data points with different classes into a subspace. The coverage of both datasets is high and the quality is comparatively low for FIRES. DUSC combines fewer data points with different classes into a subspace. Its coverage is less than that of FIRES; however, its quality is better than that of FIRES. On the other hand, the proposed method uses the greedy summarization approach to select the significant subspaces by balancing both quality and coverage simultaneously. Therefore, the proposed method has better quality and coverage than DUSC, and better quality than FIRES.

Table 11: Confusion matrix for the wine dataset.

Class	SFS_1	SFS_2	SFS_3	SFS_4	SFS_5	SFS_6	SFS_7	SFS_8	SFS_9	SFS_{10}
1	42	0	0	0	8	0	10	0	0	0
2	0	0	36	19	0	7	0	0	5	0
3	0	38	0	0	0	0	0	6	0	4

Table 11 shows the subspaces selected by the proposed method for the wine dataset, where the classes are the wines derived from three different cultivars. The first subspace SFS_1 contains 42 instances of class 1. The second subspace SFS_2 contains 38 instances of class 3, and so on.

Table 12: The wines in the first three subspaces.

Dimension	All wines	SFS_1	SFS_2	SFS_3
alcohol	11.03～14.83	13.39～14.83	-	-
magnesium	70～162	-	-	84～94
total phenols	0.98～3.88	2.35～3.88	-	-
flavonoids	0.34～5.08	-	0.47～1.28	-
nonflavonoid phenols	0.13～0.66	0.17～0.5	-	0.19～0.66
color intensity	1.28～13	3.52～8.9	3.85～11.75	1.28～4.8
hue	0.48～1.71	-	0.54～0.96	-
OD280/OD315 of diluted wines	1.27～4	-	-	1.82～3.57
proline	278～1680	-	-	278～714

Table 12 illustrates the attributes of the wines in the first three subspaces, where the values in the cells represent the characteristics of the wines in the subspaces and "-" represents the attribute not included in the subspace. For example, SFS_1 contains the wines with high alcohol, high total phenols, middle nonflavonoid phenols, and middle color intensity. SFS_2 consists of the wines with low flavonoids, middle color intensity, and low hue. SFS_3 is comprised of the wines

with low magnesium, middle nonflavonoid phenols, low color intensity, middle OD280/OD315 of diluted wines, and low proline.

In summary, since the proposed method first finds all frequent subspaces and then uses the subspace summarization approach to select the significant subspaces, it can balance both quality and coverage simultaneously by considering all frequent subspaces. Therefore, the proposed method has better quality and coverage than DUSC, and better quality than FIRES.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a novel subspace mining method. The proposed method consists of three phases. First, we generate all frequent 2-subspaces. Second, for each frequent k-subspace ($k \geq 2$), we join it to each joinable k-subspace and generate its frequent super (k+1)-subspace. Let $k = k+1$, the steps in phase two are performed recursively in a depth-first search manner until no more frequent subspaces can be found. Finally, we adopt a greedy algorithm to summarize all frequent subspaces found and select the significant ones.

Since the proposed method uses the joinable subspaces to find the frequent subspaces in a DFS manner, the proposed method can localize the search space in a small number of point groups. Thus, the proposed method is more efficient than FIRES and DUSC. Moreover, since the proposed method first finds all frequent subspaces and then uses the subspace summarization approach to select the significant ones. It can balance both quality and coverage simultaneously by considering all frequent subspaces. The experimental results show that the proposed method is efficient and scalable, has better quality and coverage than DUSC, and has better quality than FIRES in the real datasets.

In addition to the applications of the real datasets shown in Section 4.2, the proposed method can be also applied to the following potential areas such as customer segmentation, customer profiling, stock portfolio selection, trend analysis, spam mail filtering, text-mining, and bioinformatics. However, the proposed method still has some limitations. First, we use a simple score function to compute the score for each frequent subspace. Thus, it is worth further study on how to provide a better score function for a complex system. Finally, it is worth developing an algorithm which embeds the constraint of score computation in the process of mining frequent subspaces.

6. ACKNOWLEDGEMENTS

The authors are grateful to the anonymous referees for their helpful comments and suggestions. This research was supported in part by the National Science Council of Republic of China under Grant No. NSC 97-2410-H-002-126-MY3.

REFERENCE

1. Aggarwal, C.C., Wolf, J.L., Yu, P.S., Procopiuc, C., and Park, J.S. "Fast algorithms for projected clustering," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1999, pp. 61-72.
2. Aggarwal, C.C., and Yu, P.S. "Finding generalized projected clusters in highdimensional spaces," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000, pp. 70-81.
3. Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. "Automatic subspace clustering of high dimensional data," In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1998, pp. 94-105.
4. Assent, I., Krieger, R., Müller, E., and Seidl, T. "DUSC: Dimensionality unbiased subspace clustering," In *Proceedings of the Seventh IEEE International Conference on Data Mining*, 2007, pp. 409-414.
5. Cheng, C.H., Fu, A.W.C., and Zhang, Y. "Entropy-based subspace clustering for mining numerical data," In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 84-93.
6. Glomba, M., and Markowska-Kaczmar, U. "IBUSCA: A grid-based bottom-up subspace clustering algorithm," In *Proceedings of the Sixth International Conference on Intelligent Systems Design and Application*, 2006, pp. 671-676.
7. Goil, S., Nagesh, H., and Choudhary, A. *Mafia: Efficient and scalable subspace clustering for very large data sets*, Technical Report CPDC-TR-9906-010, Northwestern University, 1999.
8. Kriegel, H.P., Kröger, P., Renz, M., and Wurst, S. "A genetic framework for efficient subspace clustering of high-dimensional data," In *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005, pp. 250-257.
9. Michael, S. *Introduction to the Theory of Computation*, PWS Publishing Company, 1996.
10. Newman, D., Hettich, S., Blake, C., and Merz, C. *UCI repository of MLDBs*, 1998 (available online at http://archive.ics.uci.edu/ml/).
11. Lu, Y., Tian, Q., Liu, F., Sanchez, M., and Wang, Y. "Interactive semisupervised learning for microarray analysis," *IEEE/ACM Transactions on Computational biology and bioinformatics* (4:2), 2007, pp. 190-203.
12. Parsons, L., Haque, E., Liu, H. "Subspace clustering for high dimensional data: A review," *ACM SIGKDD Explorations Newsletter* (6:1), 2004, pp. 90-105.
13. Patrikainen, A., and Meila, M. "Comparing subspace clusterings," *IEEE Transactions on Knowledge and Data Engineering* (18:7), July 2006, pp. 902-916.

14. Sequeira, K., and Zaki, M.SCHISM: A new approach for interesting subspace mining," In *Proceedings of the Fourth IEEE International Conference on Data Mining*, 2004, pp. 186-193.
15. Woo, K.G., and Lee, J.H. *FINDIT: A fast and intelligent subspace clustering algorithm using dimension voting*, PhD thesis, Korea Advanced Institute of Science and Technology, Taejon, Korea, 2002.
16. Yang, J., Wang, W., Wang, H., and Yu, P. δ-clusters: Capturing subspace correlation in a large data set," In *Proceedings of eighteenth International Conference on Data Engineering,* 2002, pp. 517-528.