Centroid-based Actionable 3D Subspace Clustering

Kelvin Sim, Ghim-Eng Yap, David R. Hardoon, Vivekanand Gopalkrishnan, Gao Cong, and Suryani Lukman

Abstract—Actionable 3D subspace clustering from real-world continuous-valued 3D (i.e. *object-attribute-context*) data promises tangible benefits such as discovery of biologically significant protein residues and profitable stocks, but existing algorithms are inadequate in solving this clustering problem; most of them are not actionable (ability to suggest profitable or beneficial actions to users), do not allow incorporation of domain knowledge and are parameter sensitive, i.e. the wrong threshold setting reduces the cluster quality. Moreover, its 3D structure complicates this clustering problem. We propose a centroid-based actionable 3D subspace clustering framework, named CATSeeker, which allows incorporation of domain knowledge, and achieves parameter insensitivity and excellent performance through a unique combination of singular value decomposition, numerical optimization and 3D frequent itemset mining. Experimental results on synthetic, protein structural and financial data show that CATSeeker significantly outperforms all the competing methods in terms of efficiency, parameter insensitivity, and cluster usefulness.

Index Terms—3D subspace clustering, Singular Vector Decomposition, numerical optimization, protein structural and dynamics analysis, financial data mining.

1 Introduction

Clustering aims to find groups of similar objects and due to its usefulness, it is popular in a large variety of domains, such as geology, marketing, etc. Over the years, the increasingly effective data gathering has produced many high dimensional datasets in these domains. As a consequence, the distance (difference) between *any* two objects becomes similar in high dimensional data, thus diluting the meaning of cluster [1]. A way to handle this issue is by clustering in subspaces of the data, so that objects in a group need only to be similar on a subset of attributes (subspace), instead of being similar across the entire set of attributes (full-space) [23].

The high-dimensional datasets in these domains also potentially change over time. We define such datasets as three-dimensional (3D) datasets, which can be generally expressed in the form of *object-attribute-time*, e.g., the *stock-ratio-year* data in the finance domain, and the *residues-position-time* protein structural data in the biology domain, among others.

In such datasets, finding subspace clusters per timestamp may produce a lot of spurious and arbitrary clusters, hence it is desirable to find clusters that persist in the database over a given period.

- *K. Sim and G.-E. Yap are with the Institute for Infocomm Research, A*STAR, Singapore. Email: {shsim, geyap}@i2r.a-star.edu.sg*
- *D. H. Hardoon is with the SAS, Singapore. Email: davidrh@me.com*
- *V. Gopalkrishnan is with the Nanyang Technological University, Singapore. Email: asvivek@ntu.edu.sg*
- *G. Cong is with the Nanyang Technological University, Singapore. Email: gaocong@ntu.edu.sg*
- *S. Lukman is with the University of Cambridge, United Kingdom. Email: sl471@cam.ac.uk*

1.1 Problem Motivation

The problems of *usefulness* and *usability* of subspace clusters are very important issues in subspace clustering [23]. The usefulness of subspace clusters, and in general of any mined patterns, lies in their ability to suggest concrete actions. Such patterns are called *actionable* patterns [21], and they are normally associated with the amount of profits or benefits that their suggested actions bring [21], [45], [46].

The usability of subspace clusters can be increased by allowing users to incorporate their domain knowledge in the clusters [22]. To achieve usability, we allow users to select their preferred objects as centroids, and we cluster objects that are similar to the centroids.

In this paper, we identify real-world problems, which motivate the need to infuse subspace clustering with actionability and users' domain knowledge via centroids.

Financial Example Value investors scrutinize fundamentals or financial ratios of companies, in the belief that they are crucial indicators of their future stock price movements [4], [14]. For example, if investors know which particular financial ratio values will lead to rising stock price, they can buy stocks having these values of financial ratio to generate profits. Experts like Benjamin Graham [14] have recommended certain financial ratios and their respective values. For example, Graham prefers stocks whose *Price-Earnings* ratio (measures the price of the stock in relative to the earnings of the stock) is not more than 7. However, there is no concrete evidence to prove their accuracy, and the selection of the right financial ratios and their values has remained subjective.

On the other hand, investors usually know a (limited) number of profitable stocks and these stocks can be used as centroids to find other stocks that are fundamentally similar

Digital Object Indentifier 10.1109/TKDE.2012.37

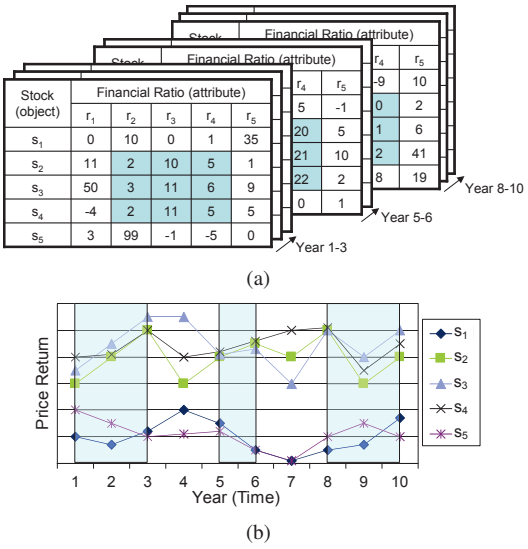


Fig. 1. (a) Example of a 3D financial dataset defined by stocks, financial ratios and years. The shaded region is a cluster of stocks s_2, s_3, s_4 that have similar financial fundamentals reflected in financial ratios r_2, r_3, r_4, for year 1-3, 5-6, 8-10. (b) The price return of the stocks. Stocks s_2, s_3, s_4 have high price returns.

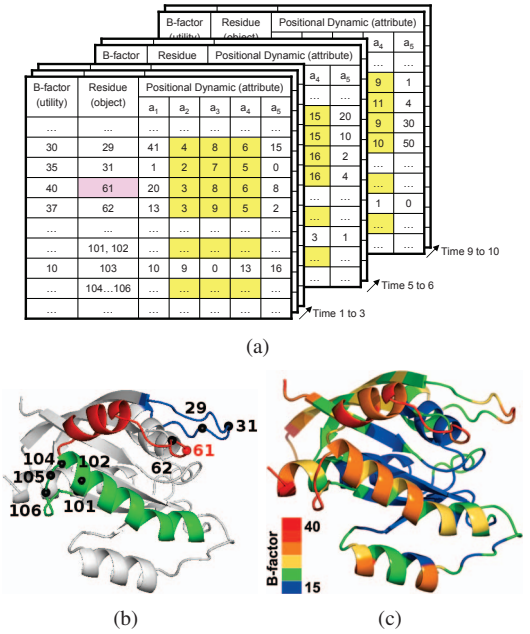


Fig. 2. (a) An actionable 3D subspace cluster: Residues shown have similar dynamics (homogeneous) in subspace defined by positional dynamics a_2, a_3, a_4 and time 1-3, 5-6, 9-10, and they have high B-factor (actionable). (b) The clusters of K-Ras residues (black spheres) discovered by CATSeeker with residue 61 as centroid (red sphere). Our results (in Section 4.4.1) suggest that the catalytic residues (blue and red) are regulated by the distant regulating residues (green) through similar dynamics, in agreement with a recent experimental study [3]. (c) These residues are actionable as indicated by their relatively high B-factor.

(in the context of financial ratios) to the centroids, and at the same time, are profitable. Through this way, investors can understand which values of financial ratios are related to high price returns.

Figure 1(a) shows a 3D *stock-financial ratio-year* financial data. Let us assume that an investor uses s_2 (that is worth investing, e.g. Apple) as a centroid. The shaded region shows a cluster of stocks with centroid s_2, and they are homogeneous in the subspace defined by financial ratios r_2, r_3, r_4 and year 1-3, 5, 6, 8-10. This cluster of stocks are actionable (can generate profits) as shown by their high and correlated price returns ((sold price of stock - purchased price of stock) / purchased price of stock) (see Figure 1(b)).

Biological Example Figure 2(a) shows a 3D *residue-positional dynamics-time* K-Ras protein structural data, with their corresponding B-factor. The structure of the K-Ras protein is shown in Figure 2(b).

Biologists are interested in finding *regulating residues* that can regulate *catalytic residue(s)* [41], and these regulating residues have the following two properties:

- *Actionable* The regulating residues are highly *flexible*
- *Homogeneous* The regulating residues have similar *dynamics* with the catalytic residue

Flexibility and dynamics are properties of biological molecules, e.g. proteins [30]. The flexibility of the residues are indicated by their B-factor, and the dynamics of the residues are indicated by their positional dynamics across time.

The catalytic residues can be used as centroids, to find regulating residues that have similar dynamics with the centroids and are as flexible as their centroids. For example, a biologist knows that residue 61 in Figure 2(a) is a catalytic residue and uses it as a centroid to find regulating residues. The shaded region in Figure 2(a) shows residues 29, 31, 61-62, 101, 102, 104-106, which are homogeneous (similar in dynamics) to residue 61, in the subspace defined by positional dynamics a_2, a_3, a_4 and timestamps 1-3, 5-6, 9-10. These residues are actionable, as indicated by their high flexibility (represented by their high B-factor, shown in Figure 2(c)), and they are potential regulating residues.

These two examples highlight the needs to find actionable clusters of objects that suggest profits (stocks with high returns in the financial example) or benefits (regulating residues that are flexible in the biological example), and to substantiate their actionability, these clusters should be homogeneous and correlated across time. In addition, users should be allowed to incorporate their domain knowledge, by selecting their preferred objects as centroids of the actionable subspace clusters.

We denote such clusters as *centroid-based, actionable 3D subspace clusters (CATSs)*, and we also denote *utility* as a function measuring the profits or benefits of the objects. A CAT should have the following properties:

1) its objects have high and correlated utilities in a set of timestamps T, so that the action suggested by the cluster is profitable or beneficial to users.

TABLE 1
Properties of continuous-valued 3D subspace clustering algorithms

	MASC [39]	TRICLUSTER [48]	MIC [37]	GS-search [19]
Domain knowledge incorporation	✓			
Generates 3D subspaces		✓	✓	
Parameter Insensitive	✓		✓	
Actionable	✓			

2) its objects exhibit a significant degree of homogeneity in the subspace defined by a set of attributes A, across the set of timestamps T. This ensures that the high utilities of its objects do not co-occur by chance.

1.2 Limitations of Existing Approaches

Existing 3D subspace clustering algorithms are inadequate in mining actionable 3D subspace clusters (see Table 1).

Domain knowledge incorporation In protein structural data, biologists need to know what residues potentially regulate the specified residue(s), and in stock data, investors want to find stocks which are similar in profit to the preferred stock of the investor. Hence, users' domain knowledge can increase the usability of the clusters [22]. In addition, users should be allowed to select the utility function suited for the clustering problem.

3D subspace generation In protein structural data, the residues do not always have the same dynamics across time [30]. In stock data, stocks are homogeneous only in certain periods of time [25]. Hence, a true 3D subspace cluster should be in a subset of attributes and a subset of timestamps. Algorithm GS-search [19] and MASC [39] do not generate true 3D subspace clusters but 2D subspace clusters that occur in every timestamps.

Parameter insensitivity The algorithm should not rely on users to set the tuning parameters [22], or the results should be insensitive to the tuning parameters. Algorithm GS-search [19] and Tricluster [48] require users to tune parameters which strongly influence the results.

Actionable Actionability, that was first proposed in frequent patterns [21] and in subspace clusters [39], is the ability to generate benefits / profits.

1.3 Proposed Solution

We propose mining **C**entroid-based, **A**ctionable **3**D **S**ubspace clusters (CATSs) with respect to a set of centroids, to solve the above issues.

CATS allows *incorporation of users' domain knowledge*, as it allows users to select their preferred objects as centroids, and preferred utility function to measure the actionability of the clusters. *3D subspace generation* is allowed, as CATS is in subsets of all three dimensions of the data.

Mining CATSs from continuous-valued 3D data is non-trivial, and it is necessary to breakdown this complex problem into sub-problems: (1) pruning of the search space, (2) finding subspaces where the objects are homogeneous and have high and correlated utilities, with respect to the centroids, and (3) mining CATSs from these subspaces.

We propose a novel algorithm, CATSeeker, to mine CATSs via solving the three sub-problems:

(1) CATSeeker uses SVD to prune the search space, which can efficiently prune the uninteresting regions, and this approach is *parameter-free*.

(2) CATSeeker uses augmented Lagrangian multiplier method to score the objects in subspaces where they are homogeneous and have high and correlated utilities, with respect to the centroids. This approach is shown to be *parameter insensitive* [33], [39].

(3) CATSeeker uses the state of the art 3D frequent itemset mining algorithm [5], [13], [18] to efficiently mine CATSs, based on the score of the objects in the subspaces.

1.4 Our Contributions

This paper is a substantial extension of an earlier work [39], and the differences are explained in Section 5. The following summarizes our contributions:

- We identify the need to mine centroid based, actionable 3D subspace clusters (CATSs), which are clusters of objects that suggest profits or benefits to users, and users are allowed to incorporate their domain knowledge, by selecting their preferred objects as centroids of the clusters.
- We propose algorithm CATSeeker, which uses a hybrid of SVD, optimization algorithm and 3D frequent itemset mining algorithm to mine CATSs in an efficient and parameter insensitive way.
- We conduct a comprehensive list of experiments to verify the effectiveness of CATSeeker and to demonstrate its strengths over existing approaches:
  - **Robustness** Correct clusters are found using CATSeeker, even with 20% perturbation in the data.
  - **Parameter-insensitivity** Correct clusters are found across diverse settings of CATSeeker's tuning parameters.
  - **Effectiveness** CATSeeker has on average 180% higher accuracy in recovering embedded clusters than current subspace clustering algorithms.
  - **Efficiency** CATSeeker is at least 2 orders of magnitude faster than the other centroid-based subspace clustering algorithm MASC.
  - **Applications on real-world data** We show that CATSeeker has 82% higher profit/risk ratio than the next best approach in financial data, and is able to discover biologically significant clusters where other approaches have not succeeded.

2 Problem Formulation

We deal with a continuous-valued, 3D data $\mathcal{D}$, with its dimensions defined by objects $\mathcal{O}$, attributes $\mathcal{A}$ and timestamps $\mathcal{T}$. Let the *value* of object o on attribute a and in timestamp t be denoted v_{oat}. We denote *feature* (a, t) as a pair of attribute a and timestamp t. Let c be an object selected as the centroid. We denote $h_c(v_{oat})$ as a *homogeneous*

function to measure the homogeneity between object o and centroid c, on attribute a and in timestamp t.

We denote *homogeneous value* s_{oat} as the output of the homogeneous function $h_c(v_{oat})$, i.e. $h_c(v_{oat}) = s_{oat}$. We allow users to define the homogeneous function, but the homogeneous values must be normalized to $[0, 1]$, such that $s_{oat} = 1$ indicates that the value v_{oat} is "perfectly" homogeneous to the value of the centroid v_{cat}, while $s_{oat} = 0$ indicates otherwise.

We use the Gaussian function as the homogeneous function, as it normalizes the similarity between object o and centroid c on feature (a, t), to $[0, 1]$. The homogeneous function is given as:

$$h_c(v_{oat}) = exp\left(-\frac{|v_{cat} - v_{oat}|}{2\sigma_c^2}\right) \quad (1)$$

σ_c is a parameter which controls the width of the Gaussian function, centered at centroid c. Note that our similarity function is not symmetric, i.e., $h_c(v_{oat}) \neq h_o(v_{cat})$, since it is calculated based on the distribution of objects centered at the former object.

The width of the Gaussian function is estimated using k-nearest neighbors heuristic [32], which is defined as:

$$\sigma_c = \frac{1}{k} \sum_{n \in Neigh_{cat}} dist_a(c, n) \quad (2)$$

$Neigh_{cat}$ is the set of k-nearest neighbors of object o on feature (a, t), and $k = \rho|\mathcal{O}|$, given that ρ is the neighborhood parameter defined by users. The k-nearest neighbors is obtained by using Equation 1.

The k-nearest neighbors heuristic adapts the width of the Gaussian function accordingly to the distribution of the objects projected in the data space of attribute a, thus it is more robust than setting σ_c to a constant value. In our experiments, we show that the results are insensitive to a considerable range of ρ. We denote the homogeneous function as $h(v_{oat})$ if centroid c is known.

To improve the quality of clusters, we define u_{ot} as the *utility* of object o at time t. The higher the utility, the higher the quality of an object is.

Definition 1: (CENTROID-BASED, ACTIONABLE 3D SUBSPACE CLUSTER (CATS)) Let C be a cuboid defined by $O \times A \times T$, where $O \subseteq \mathcal{O}$, $A \subseteq \mathcal{A}$ and $T \subseteq \mathcal{T}$. Given a centroid c, C is a CATS, if $\forall o \in O$: o has

- high homogeneity, i.e., $\forall (a, t) \in A \times T : h(v_{oat}) = s_{oat}$ is high
- high and correlated utilities, i.e., $\forall t \in T : u_{ot}$ is high.

We do not set thresholds to explicitly define the goodness of the objects' homogeneity, utility and correlation for CATSs, as setting the right thresholds is generally a 'guessing game'. Instead, our algorithm optimizes an objective function (Equation 4) that factors in the objects' homogeneity, utility and correlation to obtain the optimal set of CATSs.

We only mine *maximal* CATSs to remove redundancies in the results. A CATS $O \times A \times T$ is maximal if there is no other CATS $O' \times A' \times T'$ such that $O \subseteq O'$, $A \subseteq A'$ and $T \subseteq T'$. Since the mined CATSs are maximal, we omit the term 'maximal' for conciseness.

Definition 2: (CATS MINING PROBLEM) Given a continuous-valued 3D dataset $\mathcal{D}$ and a set of centroids $\{c_1, \ldots, c_n\}$, we set out to find all CATSs $O \times A \times T$ with respect to the given centroids.

Selection of centroids The centroids are selected by users and centroid selection is different from parameters setting. Centroid selection allows incorporation of users' preference or domain knowledge into the clustering results, e.g., in the stocks data, users can select high quality stocks such as Apple as the centroids. If domain knowledge is not available, users can use certain functions on the utility to select the centroids. For example, we can select centroids whose average utilities $\overline{u}_c = \frac{1}{T} \sum_{t \in \mathcal{T}} u_{ot}$ are higher than a threshold u_{min}.

3 ALGORITHM CATSEEKER

The framework of CATSeeker is illustrated in Figure 3, which consists of three main modules:

1. Calculating and pruning the homogeneous tensor using SVD: Given a centroid c, we define a homogeneous tensor $S \in [0, 1]^{|\mathcal{O}| \times |\mathcal{A}| \times |\mathcal{T}|}$, which contains the homogeneity values s_{oat} with respect to centroid c. The first dataset of Figure 3 shows a 3D continuous-valued dataset with centroid o_5, and the second dataset shows its homogeneous tensor.

Mining CATSs from the high-dimensional and continuous-valued tensor S is a difficult and time consuming process. Hence, it is vital to first remove regions that do not contain CATSs. A simple solution is by removing values s_{oat} that are less than a threshold, but it is impossible to know the right threshold. Hence, we propose to efficiently prune tensor S in a parameter-free way, by using the variance of the data to identify regions of high homogeneity values s_{oat}.

2. Calculating the probabilities of the values using the augmented Lagrangian Multiplier Method: We use the homogeneous tensor S with the utilities of the objects to calculate the probability of each value v_{oat} of the data to be clustered with the centroid c, as shown in the fourth dataset of Figure 3. We map this problem to an objective function, and use the *augmented Lagrangian Multiplier Method* to maximize this function. This approach is robust to perturbations in data and less sensitive to the input parameters [33].

3. Mining CATSs using 3D closed pattern mining: After calculating the probabilities of the values, we binarize the values that have high probabilities to '1', as shown in the fifth dataset of Figure 3. We then use efficient 3D closed pattern mining algorithms [5], [18] to efficiently mine subcuboids of '1', which correspond to the CATSs. An example of a CATS is shown in the last dataset of Figure 3.

3.1 Pruning the Homogeneous Tensor Using SVD

The pruning is done using Algorithm SVDpruning (shown in Algorithm 1).

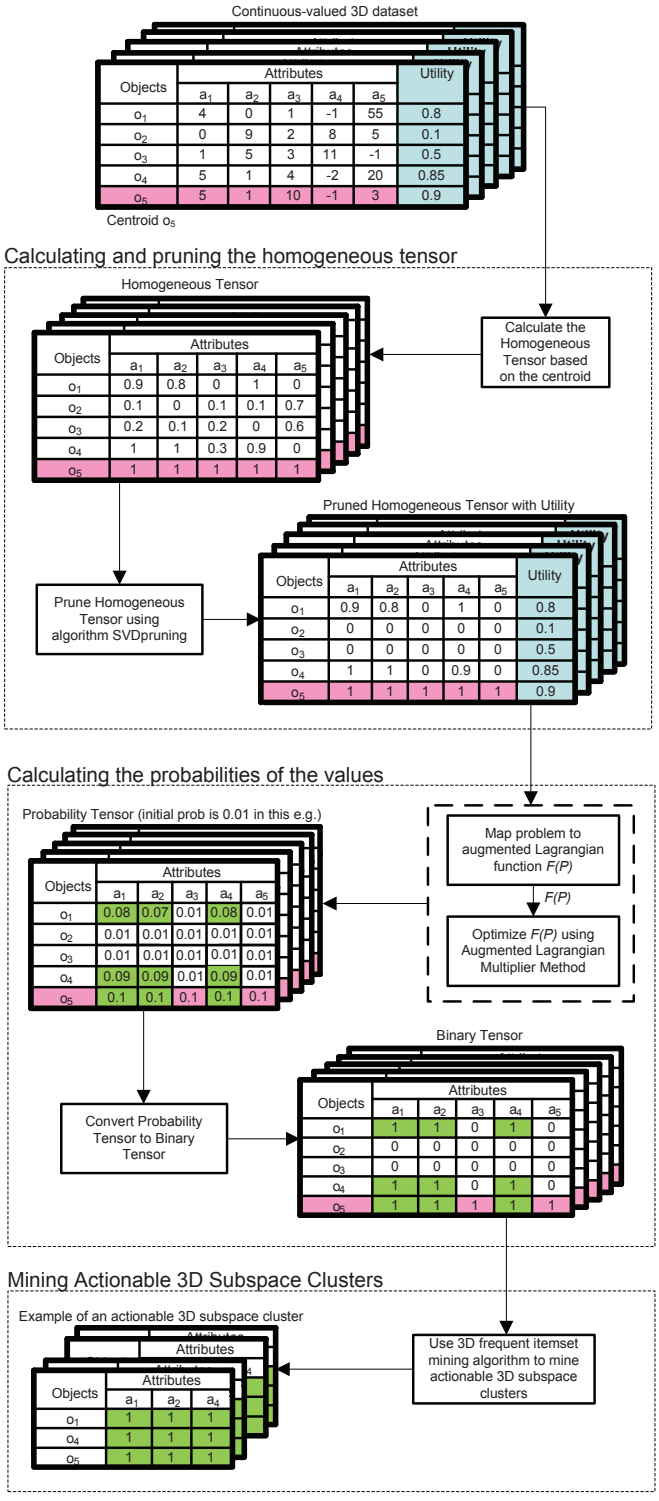


Fig. 3. Framework of algorithm CATSeeker

Algorithm 1 SVDpruning

Input:
$|\mathcal{O}| \times |\mathcal{A}| \times |\mathcal{T}|$ homogeneous tensor S
Output:
Pruned homogeneous tensor S
Description:
1: $M = unfold(S)$;
2: add dummy row and column to M;
3: **while** true **do**
4: $N \leftarrow zero\_mean\_normalization(M)$
5: $U\Sigma V' \leftarrow N$ //SVD decomposition of N
6: $\mathbf{u} \leftarrow principalComp(U)$; $\mathbf{v} \leftarrow principalComp(V')$
7: calculate thresholds τ_u, τ_v
8: prune row i of M if $|\mathbf{u}(i)| < \tau_u$, $1 \leq i \leq m$
9: prune column j of M if $|\mathbf{v}(j)| < \tau_v$, $1 \leq j \leq n$
10: **if** there is no pruning **then break**
11: **end while**
12: remove dummy row and column from M;
13: $S = fold(M)$;

Object	Feature a_1, t_1	Feature a_2, t_1	Feature dummy
o_1	0.85	0.1	0
o_2	0.9	0.5	0
o_3	1	0.1	0
o_4	0.01	0.3	0
o_5	0.05	0.2	0
dummy	0	0	0

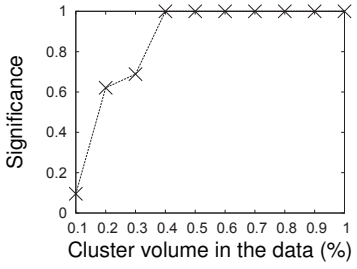
(a)

Object	u
o_1	-0.513
o_2	-0.6076
o_3	-0.6007
o_4	-0.0549
o_5	-0.0619
dummy	0

(b)

Feature	v
a_1, t_1	-0.963
a_2, t_1	-0.2694
dummy	0

(c)



(d)

Fig. 4. (a) A homogeneous matrix M which the shaded region contains high homogeneous values. (b), (c) The principal components of M where their elements indicate the objects' or features' contribution to the variance of the new basis. (d) Effectiveness of SVDpruning with respect to the volume of the CATS in the dataset $\mathcal{D}$.

Unfolding of homogeneous tensor Algorithm SVDpruning uses SVD to handle matrices. Hence, we need to unfold the homogeneous tensor S into a matrix M (Line 1), using the matrix unfolding function *unfold* : $\mathbb{R}^{|\mathcal{O}|\times|\mathcal{A}|\times|\mathcal{T}|} \rightarrow \mathbb{R}^{|\mathcal{O}|\times(|\mathcal{A}|\times|\mathcal{T}|)}$ [40]. For example, a $\{o_1, o_2, o_3, o_4, o_5\} \times \{a_1, a_2\} \times \{t_1\}$ homogeneous tensor S is unfolded into a $\{o_1, o_2, o_3, o_4, o_5\} \times \{a_1t_1, a_2t_1\}$ homogeneous matrix M. SVD involves calculating the covariance matrix of M, and combining the attribute and time dimensions in function *unfold*, instead of other dimensions, results in less memory usage.

Figure 4(a) shows an example of homogeneous matrix M. The rows of M correspond to objects o while the columns correspond to features (a, t). Each entry of M is the homogeneous value of object o to centroid c, at feature (a, t). We add a dummy row and a dummy column containing all '0' to M, to *stretch* the variance of the values of the matrix M.

Gist of the pruning The gist of this algorithm is in using the variance of the homogeneity values to guide the pruning process. By using SVD on the matrix M, we can

calculate the variance of the homogeneity values of each row(object) or column(feature) of M. A row or column that contains high homogeneity values has high variance, as its values are away from the dummy '0' values. Therefore, we keep those rows or columns that have high variance, and discard the rest. In the homogeneous matrix M of Figure 4(a), we keep rows o_1, o_2, o_3 and column (a_1, t_1) as they have high variances, and prune the rest.

Instead of matrix SVD, we could use tensor SVD [10], which does not unfold the tensor to matrix. However, tensor SVD is too aggressive in its pruning, as removing an object, attribute or time means removing a matrix of the tensor.

Usage of principal components to find rows and columns of high variance We shall explain the steps of algorithm SVDpruning. We first perform zero mean normalization on matrix M to obtain the zero mean normalized matrix N (Line 4), which will later be used to calculate the covariance matrices. Let m, n be the number of rows and columns of M respectively. Zero mean normalization is done by calculating each column mean avg_j of M, i.e. $\forall j \in \{1, \ldots, n\} : avg_j = \frac{1}{m} \sum_{i=1}^{m} M(i, j)$, and then subtracting each entry of M with its corresponding column mean, i.e. $\forall j \in \{1, \ldots, n\} : N(i, j) = M(i, j) - avg_j, i \in \{1, \ldots, m\}$.

Next, we calculate NN' and $N'N$, the covariance matrices of the homogeneous values in the object space and feature space respectively (N' is the conjugate transpose of matrix N). We use SVD to decompose both covariance matrices, as follows:

$$\begin{aligned} NN' &= U\Sigma^2 U' \\ N'N &= V\Sigma^2 V' \end{aligned} \tag{3}$$

U is a $m \times m$ orthonormal matrix (its columns are the eigenvectors of NN'), Σ^2 is a $m \times n$ diagonal matrix with the eigenvalues on the diagonal, and V is a $n \times n$ orthonormal matrix (its columns are the eigenvectors of $N'N$).

The eigenvectors show the principle directions of the homogeneous values and the eigenvalues indicate the variance of the homogeneous values in the principle directions. The eigenvector with the largest eigenvalue is known as the *principal component*. We denote **u** and **v** as the principal components of NN' and $N'N$ respectively (Line 6), i.e., **u** and **v** are the respective principal components of homogeneous matrix M in object and feature spaces.

The principal components can project the homogeneous values in new bases which show the *maximum* spread of the homogeneous values. That is, an object o is projected into the new basis using $N(o, 1)\mathbf{v}(1) + N(o, 2)\mathbf{v}(2) + \ldots + N(o, n)\mathbf{v}(n)$, and a feature i is projected into the new basis using $N(1, i)\mathbf{u}(1) + N(2, i)\mathbf{u}(2) + \ldots + N(m, i)\mathbf{u}(m)$.

Interestingly, the elements of the principal component **u** or **v** indicate the objects' or features' contribution to the variance of the homogeneous values in the new basis; objects (features) that have high homogeneous values have high magnitude in their corresponding elements of their principal component **u** (**v**).

Therefore, we can prune features or objects whose magnitude in their corresponding elements of their principal components are small (Line 8 and 9). We proposed a heuristic but parameter-free method to determine the threshold τ_u for pruning objects. We sort the absolute value of the elements of principle component **u**, find the two adjacent absolute values with the largest difference, and take the larger absolute value to be the threshold. Thus, there is a clear divide between the large and small absolute values. We use the same approach to determine threshold τ_v. For pruned rows (objects) and columns (features) of matrix M, we set their homogeneous values to '0'.

The process of calculating SVD and pruning the matrix M is repeated until there is no more pruning.

Figure 4(a) shows a homogeneous matrix M where the unshaded regions are pruned due to their low homogeneous values. The principal component **u** of M is shown in Figure 4(b), where objects o_4, o_5 are pruned due to the small magnitude of their corresponding elements in **u**. Likewise, the principal component **v** of M is shown in Figure 4(c), where feature (a_2, t_1) is pruned due to the small magnitude of its corresponding elements in **v**. After M is pruned, we fold it back into the homogeneous tensor S using the fold function $fold : \mathbb{R}^{|\mathcal{O}| \times (|\mathcal{A}| \times |\mathcal{T}|)} \rightarrow \mathbb{R}^{|\mathcal{O}| \times |\mathcal{A}| \times |\mathcal{T}|}$ (Line 13).

Effectiveness of SVDpruning in detecting high homogeneous values that are rare For high homogeneous values that occur infrequently in the dataset, they may be deemed as outliers and may be pruned. We conducted an experiment to investigate the effectiveness of SVDpruning in detecting them. We created a CATS having exact values, 2 attributes, 2 timestamps and 10 objects. We embedded this CATS into a 3D synthetic dataset $\mathcal{D}$ of 10 attributes, 10 timestamps with values from 0 to 1, and we varied its objects from 400 to 4000, to vary the volume of the CATS in the dataset. Figure 4(d) shows the correctness (denoted as significance, see Section 4.2) of obtaining the CATS using algorithm CATSeeker. SVDpruning does not prune the objects or features of CATS even if the volume of the cluster is only 0.4% of the data. Hence this approach is effective even for extremely small but significant clusters.

3.2 Calculating the Probabilities of the Values

Let $p_{oat} \in \mathbb{R}$ be the probability of object o to be clustered with centroid c on attribute a and at time t. Let $P \in \mathbb{R}^{|\mathcal{O}| \times |\mathcal{A}| \times |\mathcal{T}|}$ be the probability tensor, such that p_{oat} is an element of it, given the respective indices o, a, t. We maximize the following objective function to calculate our probabilities

$$f(P) = \sum_{o \in \mathcal{O}} \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} p_{oat}\ h(v_{oat})\ util(u_{ot}) \tag{4}$$

under constraint

$$g(P) = \sum_{o \in \mathcal{O}} \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} p_{oat} - 1 = 0 \tag{5}$$

From this objective function $f(P)$, we can see that the value v_{oat} will have high probability if (1) it is highly homogeneous to the value of the centroid v_{cat}, (2) the utility of its object o is high. As the probabilities are calculated in

Algorithm 2 BCLM

Input:
Initial probability distribution P_a
Output:
The optimal probability distribution P_a^*
Description:
1: initialize $P_a^0, \delta \leftarrow 0.001, \lambda \leftarrow 0.1, \tau \leftarrow 1, \mu \leftarrow 10, i \leftarrow 1$
2: **while** true **do**
3: Use $L\text{-}BFGS(P^{i-1}, \mu, \lambda)$ to find an approximate solution P_a^i of $F(P_a)$ such that $||\nabla F(P_a^{i-1})|| \leq \tau$
4: **if** $|P_a^i - P_a^{i-1}| < \delta$ **then return** P_a^i
5: **if** $|g(P_a^i)| < |g(P_a^{i-1})|$ **then**
6: $\tau \leftarrow \tau \cdot 0.1$ `//strictly tighten` τ
7: **else**
8: $\tau \leftarrow \tau \cdot 0.9$ `//loosely tighten` τ
9: $\mu \leftarrow \mu \cdot 10$ `//increase penalty violation`
10: **end if**
11: $\lambda^i \leftarrow \lambda^i - \mu \cdot g(P_a^i)$
12: $i \leftarrow i + 1$
13: **end while**

each timestamp, the correlation of the utilities of the objects are implicitly calculated.

We multiply our objective indices $h(v_{oat})$ and $util(u_{ot})$ in $f(P)$, instead of summing them, for two reasons. First, the domains of $h(v_{oat})$ and $util(u_{ot})$ are different; $h(v_{oat})$ is normalized to $[0, 1]$, and $util(u_{ot})$ is a real number. Summing them will result in the objective function to be bias towards $util(u_{ot})$, and multiplication will remove this bias. Second, it is undesirable to have negative $util(u_{ot})$ and summing the objective indices will unintentionally mitigate negative $util(u_{ot})$, if $h(v_{oat})$ is high.

Optimizing $f(P)$ under constraint $g(P)$ is a linear programming problem, as $f(P)$ and $g(P)$ are linear functions of the design variable P. Hence, constrained optimization methods can be used to solve this problem [33].

We use the *augmented Lagrangian multiplier method* to maximize the objective function $f(P)$, as this method is shown to work efficiently and robustly [33], [39]. The objective function is modeled as the following augmented Lagrangian function $F(P)$:

$$F(P) = -f(P) - \lambda g(P) + \frac{\mu}{2} g(P)^2 \tag{6}$$

The augmented Lagrangian multiplier method seeks to minimize $F(P)$, so we negate $f(P)$ in Equation 6. λ is the Lagrange multiplier, and μ specifies the severity of the penalty on $F(P)$ when the constraint $g(P)$ is violated. The augmented Lagrangian multiplier method requires both $f(P)$ and $g(P)$ to be smooth, which is satisfied in our case.

The constraint function $g(P)$ is a summation of probabilities, and it is possible that only the probabilities involving the centroid are non-zeros and the rest are zeros. One remedy is to use a multiplication of probabilities for the constraint function [7], to ensure all probabilities are non-zeros. However, we do not force clusters to be created, as it is possible that a centroid is highly dissimilar to other objects. In Section 4, we show that clusters can be mined in both synthetic and real world datasets, which means that our approach does not give trivial solutions.

We use the augmented Lagrangian multiplier method, known as the Bound-Constrained Lagrangian Method (BCLM) [33], to optimize $F(P)$ (see Algorithm 2). BCLM exploits the smoothness of both $f(P)$ and $g(P)$ and replaces our constrained optimization problem with iterations of unconstrained optimization subproblems, and the iterations continue until the solution converges.

Our 3D problem space is large, which requires solving large-scale constrained optimization problems, so we need to keep both the storage and computational costs low to be efficient. Newton methods [33] require explicit construction of the Hessian matrix and involve factorization of the matrix, which often result in prohibitive storage and computational costs in large-scale problems. In contrast, the Hessian approximations generated by the limited-memory quasi-Newton methods (e.g. L-BFGS) can be stored compactly, making them much more efficient than Newton methods for large problems. Hence, we choose BCLM, which uses the L-BFGS algorithm to solve the unconstrained optimization subproblem in each iteration.

In each iteration (Line 3), we minimize $F(P^{i-1})$ and generate an approximate solution P^i using L-BFGS, such that the Euclidean norm of the gradient of $F(P^{i-1})$, $||\nabla F(P^{i-1})||$, is not greater than the parameter τ. Parameter τ controls the tolerance level of violation to the constraint function $g(P)$. The gradient of $F(P)$ is $\nabla F(P) = \{\frac{\partial F(P)}{\partial p_{oat}} | o \in \mathcal{O}, a \in \mathcal{A}, t \in \mathcal{T}\}$, with

$$\frac{\partial F(P)}{\partial p_{oat}} = -\overline{h}(v_{oat})\, util(u_{ot}) - \lambda + \mu\Big(\sum_{o\in\mathcal{O}} \sum_{a\in\mathcal{A}} \sum_{t\in\mathcal{T}} p_{oat} - 1\Big) \tag{7}$$

Algorithm BCLM requires four parameters, δ, λ, τ, and μ. In most cases, the results are not sensitive to these parameters, and hence they can be set to their default values. Parameter δ specifies the closeness of the result to the optimal solution. Thus, δ provides the usual trade-off between accuracy and efficiency, i.e., smaller δ implies longer computation time but better result. Parameter τ controls the tolerance level of violation to the constraint $g(P)$. Parameter μ specifies the severity of the penalty on $F(P)$, when the constraint is violated. Lastly, parameter λ is the Lagrange multiplier, and details of λ is in [33].

We initialize the probability distribution P^1 by allocating equal probability to each object, i.e. $\frac{1}{|\mathcal{O}|\times|\mathcal{A}|\times|\mathcal{T}|}$. Parameters τ and μ are set to 1 and 10 respectively, as recommended in [33]. Parameter λ is set to 0.1 and parameter δ is set to 0.001. In our experiments, we show that the default settings work well in practice, and the results are not sensitive to various settings of the parameters.

L-BFGS is globally convergent on uniformly convex problems, with a linear rate of convergence [27]. Through the iterative invocation of L-BFGS, BCLM's convergence is assured as the penalty parameter is increased [33]; by selecting a sufficiently large value of the penalty parameter, BCLM ensures that the Lagrange muliplier becomes increasingly accurate to lead to the optimized probability distribution P_a^*.

3.3 Mining CATSs

After obtaining the resulting probability tensor P, values with probabilities greater than the initial probability $\frac{1}{|\mathcal{O}|\times|\mathcal{A}|\times|\mathcal{T}|}$ are clustered with the centroid. We first binarize P by assigning '1' to the values whose probabilities are greater than the initial probability, and the rest as '0'. The binarized P becomes $P \in \{0,1\}^{|\mathcal{O}|\times|\mathcal{A}|\times|\mathcal{T}|}$.

We then use efficient 3D closed frequent itemset (CFI) mining algorithm [5], [18] to mine sub-cuboids, which correspond to CATSs. We set the minimum supports of the algorithm to 1, to mine all CATSs. We define a sub-cuboid as $O \times A \times T$, where $O \subseteq \mathcal{O}$, $A \subseteq \mathcal{A}$ and $T \subseteq \mathcal{T}$, and the cells of the sub-cuboid are '1's.

A CFI corresponds to a maximal biclique subgraph [26], and this correspondence can be extended to a 3D CFI and a maximal cross-graph biclique subgraph [38]. Since the set of adjacency matrices of a maximal cross-graph biclique subgraph can be represented as a sub-cuboid $O \times A \times T$, this means that the sub-cuboid is also maximal. This inturn infers that the corresponding CATS is maximal.

3.4 Time Complexity Analysis of CATSeeker

CATSeeker consists of three sub algorithms: SVDpruning, BCLM and 3D CFI mining algorithm, and we analyze the time complexity of each of them, per centroid. As SVDpruning and BCLM are iterative algorithms, we can restrict the number of iterations in them.

Algorithm SVDpruning uses SVD, which consists of two computations: reducing the matrix M to a bidiagonal matrix, and diagonalizing the bidiagonal matrix. The time complexity of the first computation is $O(|\mathcal{O}|(|\mathcal{A}||\mathcal{T}|)^2)$, and the second computation is $O((|\mathcal{A}||\mathcal{T}|)^2)$ [43]. Let n_S be the number of iterations of SVDpruning, and the time complexity of SVDpruning is $O(n_S|\mathcal{O}|(|\mathcal{A}||\mathcal{T}|)^2)$.

Algorithm BCLM uses algorithm L-BFGS, which is also an iterative algorithm. Let n_B and n_L be the number of iterations of algorithms BCLM and L-BFGS respectively. In an iteration of L-BFGS, functions $F(P)$ and $\nabla F(P)$ are calculated, and both of their time complexity are $O(|\mathcal{O}||\mathcal{A}||\mathcal{T}|)$. Hence the time complexity of algorithm BCLM is $O(n_B n_L|\mathcal{O}||\mathcal{A}||\mathcal{T}|)$.

3D CFI mining algorithms [5], [18] use the set enumeration tree approach to mine sub-cuboids from the binarized probability tensor P, and the worst case time complexity of this approach is exponential to the number of clusters generated [42]. However, by using SVDpruning to prune P and BCLM to identify the cluster regions of P, only true CATSs, and not spurious clusters, are generated. Let N be the number of CATSs generated. The worst case time complexity of generating CATSs is $O(dN)$, where d is the overhead cost, which depends on the 3D CFI mining algorithm used.

In summary, the time complexity of CATSeeker per centroid is $O(n_S|\mathcal{O}|(|\mathcal{A}||\mathcal{T}|)^2 + n_B n_L|\mathcal{O}||\mathcal{A}||\mathcal{T}| + dN)$.

4 Experimentation Results

Our experiments consist of five parts: (1) efficiency analysis and (2) parameter sensitivity analysis of CATSeeker, (3) quality analysis of the clusters mined by different algorithms, (4) application on stock market, and (5) application on protein structure.

Algorithms Used for Comparison We took five algorithms that broadly represent the different classes of subspace clustering as comparison; actionable subspace clustering MASC [39], parameter sensitive 3D and 2D subspace clustering TRICLUSTER [48] and STATPC [31] respectively, parameter insensitive 3D and 2D subspace clustering MIC [37] and STATPC [31] respectively. For 2D subspace clustering MaxnCluster and STATPC, we need to mine subspace clusters from each timestamp, and intersect all combinations of the subspace clusters to form 3D subspace clusters. To reduce their redundancies, we only output 3D subspace clusters that are maximal.

All algorithms were coded in C++, and codes or programs for competing algorithms were kindly provided by their respective authors. All experiments were performed using computers with Intel Core 2 Quad 3.0 GHz CPU, 8Gb RAM. They were performed in Windows 7 environment, except for experiments involving TRICLUSTER, which were performed in Ubuntu 10.10 environment. We used the code from ALGLIB [2] for algorithm L-BFGS and SVD.

For CATSeeker, we used the default settings for the parameters of algorithm BCLM given in [33], [39], τ=1, μ=10, δ=0.001, λ=0.1. Unless otherwise stated, we set ρ=0.3 and select centroids by setting the minimum average utility threshold u_{min}=0.5.

Datasets Used We used both synthetic and real world datasets in our experiments, for a comprehensive study.

Synthetic Datasets In our synthetic dataset $\mathcal{D}$, the values of the objects are randomly valued from 0 to 1, and the utilities of the objects are randomly valued from -1 to 1. We randomly embedded 10 CATSs, each having 15-25 objects, 5-10 attributes and 5-10 timestamps in $\mathcal{D}$. In order to ensure homogeneity in each cluster, we set a maximum difference allowance, *diff*, on its objects' values on each feature (a,t) of the cluster. Unless otherwise stated, the synthetic dataset $\mathcal{D}$ contains 1000 objects, 10 attributes, 10 timestamps, the utility of the objects in the cluster is at least 0.5 and the *diff* is at most 0.1. Hence, the volumes of the clusters are from 0.375 to 2.5% of the dataset $\mathcal{D}$, and mining such small clusters is a non-trival problem.

Protein Structural Dataset We extracted the Cα atom coordinate of each residue from the molecular dynamics structural ensemble of K-Ras protein [29] to define a dataset of 167 residues (objects), 4 positional dynamics (attributes) and 182 timestamps. The crystallographic B-factor of each residue o is used as the utility u_o, and is constant across time. The positional dynamics are calculated with respect to 4 reference structures, each obtained at every interval of 10 nanoseconds. The value v_{oat} is the Euclidean distance between residue o's positional Cartesian (x,y,z) coordinate at time t and at the reference structure indexed by a.

Stock Market Dataset We downloaded 28 years (1980 to 2006) of financial figures and price data of all North American stocks from Compustat [9]. We converted these financial figures into 30 financial ratios, based on the

formulae from Investopedia [17] and Graham's ten criteria [34]. We removed microcap stocks (whose prices are less than USD$5) from the data as these stocks have a high risk of being manipulated and their financial figures are less transparent [44]. Hence, we have a continuous-valued 3D dataset of 3200 stocks $\times$ 30 financial ratios $\times$ 28 years, and 20% of the dataset are missing values.

4.1 Efficiency Analysis of CATSeeker

We investigated how the number of objects and attributes of continuous-valued 3D data $\mathcal{D}$ affect the computation costs of CATSeeker. We did not conduct experiment on the time dimension, since it is handled similarly with the attribute dimension. We compared the efficiency of CATSeeker with MASC, as it is the only other centroid-based 3D subspace clustering algorithm. We also compared CATSeeker with STATPC and MIC, as these two algorithms are parameter insensitive and default parameter setting can be used. We did not compare with the other algorithms, as their efficiency depends on their parameter settings.

Given that both CATSeeker and MASC are run per centroid, we ran 10 centroids for each dataset $\mathcal{D}$ and averaged their running time to obtain the results.

First, we varied the number of objects from 10,000-100,000 and fixed the number of timestamps and number of attributes at 10, resulting in a medium-sized 3D dataset with 1-10 million values. The results in Figure 5(a) show that the running time of CATSeeker is a polynomial function of the number of objects. Both STATPC and MIC could not finish running after 6 hours.

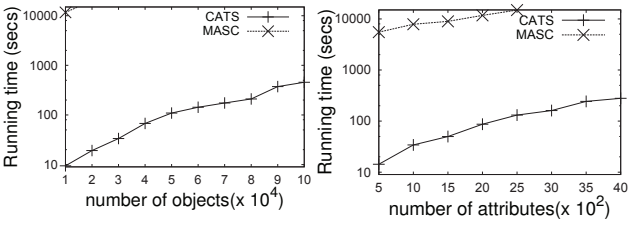
Second, we varied the number of attributes from 500-4000 and fixed the number of objects at 1,000 and number of timestamps at 10, resulting in a medium-sized 3D dataset with 5-40 million values. Figure 5(b) presents the results, which also shows that the running time of CATSeeker is a polynomial function of the number of attributes. Again, STATPC and MIC could not finish running after 6 hours.

Both experiments show that CATSeeker took at most 500 seconds to finish the mining task, which is reasonable for real-world applications containing medium-sized datasets. For MASC, we can see that it is orders of magnitude slower than CATSeeker. This is due to two reasons; first, for each centroid, MASC needs to run its optimization algorithm $|\mathcal{A}|$ times, while CATSeeker only needs to run its optimization algorithm once. Second, MASC does not prune the dataset, while CATSeeker does. STATPC and MIC are slower than CATSeeker as they mine the complete set of subspace clusters, while CATSeeker is run per centroid.

4.2 Parameter Sensitivity Analysis of CATSeeker

4.2.1 Tuning Parameters

We evaluated how the neighborhood parameter ρ, Lagrange multiplier λ, accuracy parameter δ, tolerance parameter τ and penalty parameter μ (see Section 2 and Section 3.2 for details of the parameters) of algorithm CATSeeker affect the mining of the embedded clusters, in a wide range of synthetic datasets with embedded clusters. Each parameter



(a) (b)

Fig. 5. Average running time of algorithms when varying the (a) number of objects and (b) number of timestamps or attributes in 3D synthetic datasets.

is tested on a total of 110 datasets $\mathcal{D}$; we varied *diff* from 0-0.2 (with an interval of 0.02), to obtain 11 settings of *diff*, and for each *diff* setting, we created 10 datasets $\mathcal{D}$.

Let $\mathcal{C}^*$ be the set of embedded CATSs, and $C^* = O^* \times A^* \times T^*$ be one such embedded cluster. Similarly, let $\mathcal{C}$ be the set of mined CATSs, and $C = O \times A \times T$ be one such mined CATS. We use the following metrics to measure the closeness of $\mathcal{C}$ to $\mathcal{C}^*$ [16].

Recoverability measures the ability of C to recover C^*

$$re(C^*) = \max_{C \in \mathcal{C}} \sum_{t \in T^*} \frac{r(S^*, S)}{|S^*|}$$

where $r(S^*, S) = |S^* \cap S|$ such that $S^* = O^* \times A^* \times \{t\} \subset C^*, S = O \times A \times \{t\} \subset C$.

Spuriousness is used to measure how many spurious or redundant values are in the actionable subspace cluster C.

$$sp(C) = \min_{C^* \in \mathcal{C}^*} \sum_{t \in T} \frac{s(S, S^*)}{|S|}$$

where $s(S, S^*) = |S| - |S^* \cap S|$ such that $S^* = O^* \times A^* \times \{t\} \subset C^*, S = O \times A \times \{t\} \subset C$.

Significance is a measure to find the best trade-off between *recoverability* and *spuriousness*. High *significance* indicates high similarity between the mined and the embedded clusters.

$$\textit{Significance} = \frac{2Re(1 - Sp)}{Re + (1 - Sp)}$$

where $Re = \sum_{C^* \in \mathcal{C}^*} re(C^*)$ and $Sp = \sum_{C \in \mathcal{C}} sp(C)$.

Figure 6(a-e) present the average significance of the mined CATSs from the synthetic datasets, across the varying settings of the parameters. Parameters $\rho, \delta, \lambda, \tau$ and μ are varied in the intervals $[0.1, 1], [10^{-7}, 0.1], [10^{-4}, 100], [10^{-4}, 100]$ and $[10^{-4}$,100] respectively. For δ, λ, τ and μ, the significance of their results is high across a wide range of their settings, meaning that the clustering results are insensitive to the parameters. For ρ, the significance of its results is high from 0.2 to 0.5, which is still a considerable range.

4.2.2 Centroid Selection Parameter u_{min}

We investigated the sensitivity of using u_{min} in obtaining the true centroids in a wide range of synthetic datasets. To this end, we checked if the selected centroids using u_{min}

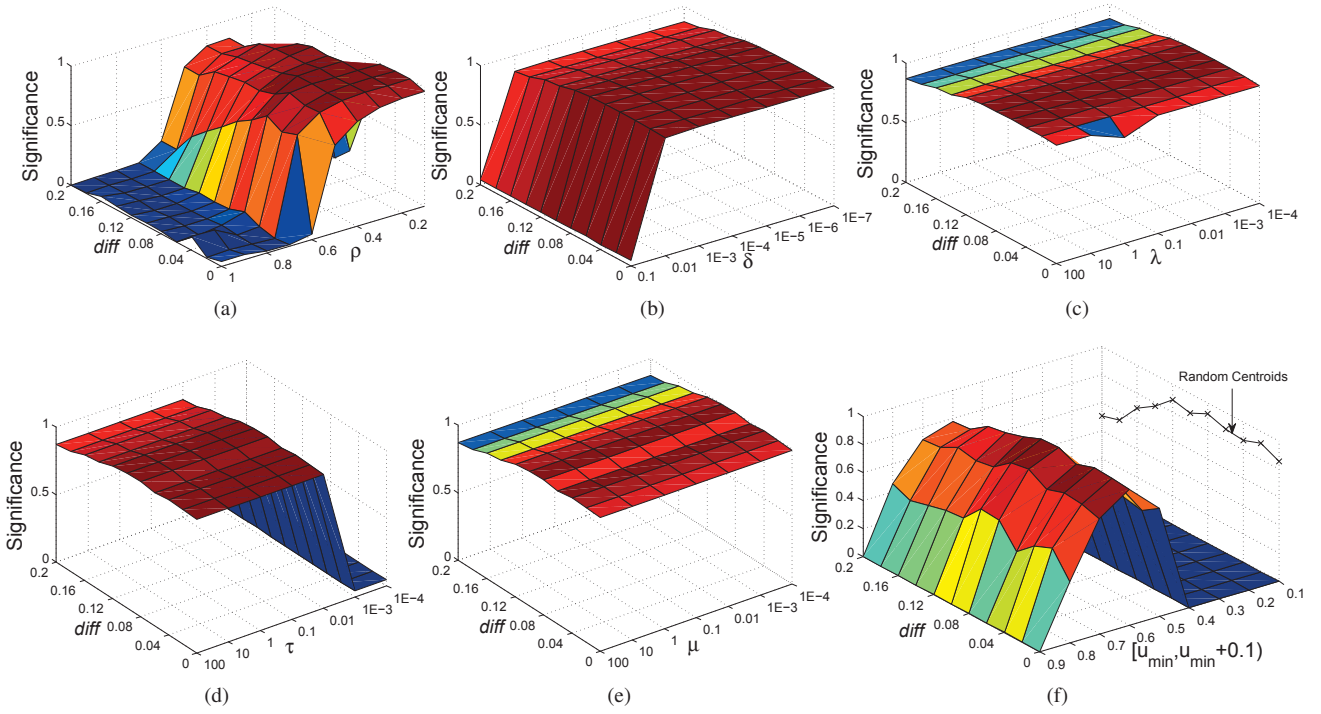


Fig. 6. (a),(b),(c),(d),(e) Significance of the CATSs mined by varying the parameters $\rho, \delta, \lambda, \tau, \mu$ across different settings of *diff*. (f) Significance of the CATSs mined by varying u_{min} and by random selection of centroids, across different settings of *diff*.

can be used to mine the embedded clusters in the synthetic datasets. We varied u_{min} from 0.1-0.9, and for each u_{min}, we select centroids whose average utilities $\overline{u}_c$ are within $[u_{min}, u_{min}+0.1)$. Similar to previous experimental setup, we created 110 datasets $\mathcal{D}$, by varying *diff* from 0-0.2. The objects' utilities in the embedded clusters are at least 0.5.

Figure 6(f) presents the results, which shows that we are able to mine the embedded clusters using centroids whose average utilities are from 0.5-0.8, as shown by the high significance of the results in this range. The significance of the results in other settings of $[u_{min}, u_{min}+0.1)$ is low as good centroids are filtered out. Hence, if the objects' average utilities in the clusters are at least u_{min}, then objects whose average utilities that are in $[u_{min}, u_{min}+0.3]$ can be chosen as centroids.

We also randomly selected 10% of the objects as centroids in each dataset, and evaluated the quality of the clusters mined using them. The line in Figure 6(f) presents the result. Surprisingly, the significance of the clusters are high for *diff* from 0-0.12, although the centroids are randomly selected. This result shows that it is still possible to mine good quality clusters, even without selection of centroids by users.

4.3 Evaluation of the Quality of the Clusters

We created synthetic datasets with embedded clusters, and used the embedded clusters as the 'ground truth' to evaluate the quality of the clusters mined by the different algorithms. We also studied the effectiveness of the SVDpruning of CATSeeker by comparing it with (1) CATSeeker without SVDpruning and (2) CATSeeker with simple pruning. In CATSeeker with simple pruning, values below a threshold in the homogeneous tensor are pruned.

TRICLUSTER and MaxnCluster have 7 and 3 parameters respectively, and it is hard to enumerate all possible settings. Hence, we give them an unfair advantage by setting their minimum object (min_o), attribute (min_a), timestamps (min_t) sizes in accordance to those of the embedded clusters. For TRICLUSTER, we varied its similarity parameters ϵ=0.01, 0.05, 0.1 and δ^x=0.01, 0.1, 1, and for MaxnCluster, we varied its similarity parameter δ=0, 0.5, 0.9. For STATPC, MIC and MASC, we used their default settings, unless otherwise stated.

4.3.1 *Varying diff*

For a thorough comparison, this experiment was conducted on 110 datasets $\mathcal{D}$; we varied *diff* from 0-0.2 (with an interval of 0.02), to obtain 11 settings of *diff*, and for each *diff* setting, we created 10 datasets $\mathcal{D}$. Figure 7(a) presents the average significance of the clusters mined by the different algorithms. We show the best results of TRICLUSTER (TRI) and MaxnCluster (MNC), among their varying parameter settings. We denote CATSeeker without SVDpruning as Prune 0, and CATSeeker with simple pruning at threshold 0.5 and 0.9 as Prune 0.5 and Prune 0.9 respectively. For MIC, we set its p-value to 10^{-20}, as it could not finish running after 24 hours at its default setting.

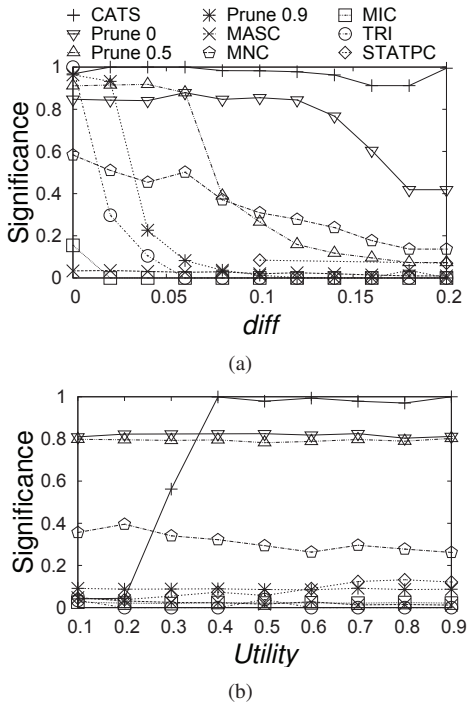


Fig. 7. Quality of the 3D subspace clusters mined by different algorithms across 110 synthetic datasets, by varying the (a) *diff* and (b) utility of the objects in the embedded clusters.

In Figure 7(a), the significance of CATSeeker is the highest across all *diff* and even at a large *diff*, the significance is maintained at more than 0.9. This result demonstrates the robustness of CATSeeker in mining clusters, compared to the other algorithms.

For TRICLUSTER and MaxnCluster, both performances drop as *diff* increases. Although STATPC and MIC are parameter insensitive, they performed poorly in this experiment. MASC performed poorly even when *diff* is small, due to the stringent requirement of the cluster to be present in each timestamp of the dataset.

On the pruning approach of CATSeeker, pruning 0 has high significance when *diff* is low, but its performance drops as *diff* increases. Thus, there is a need to prune the homogeneous tensor. However, a simple pruning approach is ineffective (e.g. the low significance of Prune 0.5 and Prune 0.9 in Figure 7(a)) due to its inability to recognize the correct regions to be pruned.

4.3.2 Varying utility

We created 110 datasets $\mathcal{D}$ for this experiment; we varied the utility of the objects in the embedded clusters of the datasets $\mathcal{D}$ from 0.1-0.9 (with an interval of 0.1), to obtain 9 settings of utility, and for each utility setting, we created 10 datasets $\mathcal{D}$. For CATSeeker, we set u_{min} to be the same as the utility of the objects in the embedded clusters.

Figure 7(b) presents the average significance of the clusters mined by the different algorithms. The top performer is CATSeeker for utility ≥ 0.4. The effectiveness of CATSeeker increases as the utility increases. Similar to the previous experiment, the competitors performed poorly, which highlight two points: utility helps to improve the clustering and for the other algorithms, it is hard to set their correct parameter settings, to get the actual clusters.

4.4 Applications on the Protein Structural Dynamics and the Stock Market

Real world data normally do not have the 'ground truth' of the synthetic dataset that we used in Section 4.2 and 4.3, to evaluate the actionability of the results. Thus, in our experiments on the protein structural dynamics and the stock market, we evaluated the actionability of the clusters, by their identification of potential drug binding residues and the profits that they generate from financial data respectively.

4.4.1 Clusters of residues from protein structural dynamics

Biologists are interested in the catalytic or binding site of a protein structure (which consists of amino acid residues), where drug molecules can bind to stop the aberrant function of the protein. A drug that binds to the conserved catalytic site of a target (disease) protein, can simultaneously bind to the catalytic site of other proteins which are required for normal functions, hence leading to unwanted side effects. Instead of targeting the conserved catalytic site, it is desirable to seek an alternative site, named *allosteric site* (formed by regulating residues), where drug molecules can bind selectively only to the disease protein but not other proteins in the family.

In this experiment, we attempted to find biologically significant clusters that contain catalytic residue 61 and regulating residues from the K-Ras protein, as they are potential drug binding sites. K-Ras is selected because of its highest prevalence in cancer among the Ras proteins, and mutation of residue 61 is associated with a large number of cancers [35]. A cluster is biologically significant if it is *small* and *functional*; small clusters are desirable because they render drug design efforts manageable, and a cluster is functional if it contains residues located at catalytic and allosteric sites, which regulate the protein function(s). For the K-Ras protein, a cluster is functional if it contains catalytic sites *loop 2* (residues 26-36) and *loop 4* (residues 59-65), and allosteric site *helix 3-loop 7* (residues 98-108).

It is challenging to identify the allosteric site because it is less conserved than the catalytic site. Fortunately, the residues of the allosteric site can be mined based on (1) their highly flexible nature and (2) the similarity of their dynamics to the residues of the catalytic site (loop). To measure the flexibility of the residues, we use B-factor that accounts for the thermal motions of residues, can be used. The information on the residues' dynamics can be obtained from molecular dynamics simulation producing an ensemble of structures at varying timepoints [30].

For CATSeeker, MASC and STATPC, we used their default settings, and we used residue 61 as the centroid for CATSeeker and MASC. For TRICLUSTER and MaxnCluster, we used the same similarity parameter settings as

TABLE 2
Summary of the clusters of residues found in K-Ras protein structural dynamics

Algorithm	No. of Clusters	Max cluster size	No. of Clusters with residue 61	No. of biologically significant clusters
CATSeeker	164	7	164 (100%)	141 (86%)
MASC	0	N/A	N/A	N/A
MnC δ=0	0	N/A	N/A	N/A
δ=0.5	864	121	12 (1.4%)	0
δ=0.9	557	161	557 (100%)	0
MIC p=0.0001	9	14	0	0
p=0.001	136	28	8(5.8%)	0
STATPC	93	115	0	0
TRI $\delta^x = 0.01$	4	2	0	0
$\delta^x = 0.1$	4	2	0	0

in Section 4.3, and we set min_o=min_a=min_t=2 for TRICLUSTER, and min_o=min_t=10, min_a=2 for MaxnCluster. Setting parameters for parameter-sensitive algorithm is not easy, as the minimum sizes and similarity parameters must be balanced to prevent an exponential number of clusters, or no clusters, to be generated.

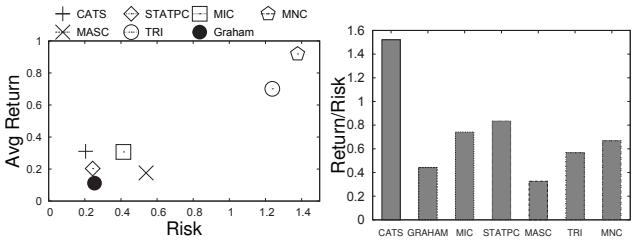
Table 2 presents the summary of the clusters mined by the different algorithms. CATSeeker is the only algorithm that succeeds in finding biologically significant clusters. MASC is the other centroid-based subspace clustering algorithm, but it does not find any cluster because it strictly requires a cluster to occur in every timestamp. MaxnCluster (δ=0.5) and MIC (P=0.001) find clusters with residues 61 but they are not biologically significant because non-catalytic or non-allosteric residues are also clustered together with residue 61. STATPC and MaxnCluster (δ=0.9) find large-size clusters which are undesirable.

CATSeeker is able to cluster K-Ras catalytic residue 61 with physically closed catalytic residues 29, 31, and 62, and interestingly with physically far allosteric residues 101-102, 104-106 (Figure 2). This shows that CATSeeker is able to capture their homogeneous dynamics in subspaces of time (i.e. not across all time). Moreover, these residues are relatively flexible (high utility, i.e. B-factor) as compared to the other residues of K-Ras (Figure 2). Overall, CATSeeker is able to identify a novel allosteric site which is essential for discovering selective drug molecules that interfere with K-Ras oncogenic functions [3], [15].

4.4.2 Clusters of profitable stocks

Value investment is a type of investment strategy where investors select stocks based on their fundamentals (financial ratios). The founder of value investment, Benjamin Graham, has formulated one of the most successful value investment strategies [34], which is based on a set of rules. Graham's strategy consists of a buy and a sell phase. The buy phase requires the analysis of the stock data for the past 10 years and if a stock fulfills Graham's rules, the stock will be bought [34]. In the sell phase, the stock is sold if its price appreciates to at least 50% within the next two years. Otherwise, it is sold after two years.

We have 17 ten-year stock data (training data) 1980-1989, 1981-1990, ..., 1997-2006, resulting in 17 buy and sell phases. In a stock data, a selected stock is bought based on its closing price of the last day of its last fiscal year, and it is sold based on its closing price of the last day of its later fiscal years. Using Graham's strategy, we calculated the average return of the selected stock purchases on the 17 stock data, which is a considerable average return of 11.2%, with a risk of 0.25 (standard deviation of the return).



(a) (b)

Fig. 8. (a) Average return and risk and (b) average return/risk ratio of the different investment strategies.

We studied the effectiveness of replacing Graham's rule based buy phase with clustering based buy phase. CATS is suitable for this value-investment problem; to identify potential good stocks, CATSeeker clusters stocks which have similarly good fundamentals (homogeneous subset of financial ratios), and historical high price returns (utilities).

In the buy phase of the clustering based strategy, we mined CATSs in a stock dataset and bought the stocks that are in the clusters. We only buy stocks which are in clusters that contain the latest year, e.g. if the stock data is 1980-1989, the clusters must contain year 1989, assuming that investments are made on the latest information. We take the price return of a stock o at year t as the utility u_{ot}, and the average price return of o as the average utility $\overline{u}_o$ over the ten years of the stock data. We set the threshold $u_{min} = 0.8$ to select centroids which have high $\overline{u}_o$.

For TRICLUSTER and MaxnCluster, we used the same similarity parameter settings as in Section 4.3. For TRICLUSTER, we set min_o=min_a=min_t=2, and for MaxnCluster, we set min_o=10, min_a=min_t=5 when $\delta = 0$, and min_o=min_a=20, min_t=5 when δ=0.5, 0.9. For the other algorithms, we used their default parameter settings.

Figure 8(a) presents the average return and risk of the stocks bought based on the different algorithms and Graham's strategy. The most desired result is in the top-left corner of the figure, which corresponds to high return with low risk. We can see that using CATSeeker leads to an average return of 30% with a low risk of 0.2. Both TRICLUSTER and MaxnCluster have high return but high risk, which are undesirable. Figure 8(b) shows the return/risk ratio, and high ratio is desired, as it implies high return with low risk. CATSeeker has the highest return/risk ratio, and is 82% better than the next better competitor. This shows the usefulness of CATSeeker in financial application.

5 Related Work

Majority of the subspace clustering algorithms handle 2D data [8], [11], [12], [24], [28], [31], i.e. data having two

dimensions, namely *object* and *attribute*. More recently, algorithms have been proposed to handle 3D data [5], [13], [18], [19], [37], [39], [47], [48], i.e. data having an additional *context* dimension (typically *time* or *location*). The solutions in [5], [13], [18] mine subspace clusters in 3D binary data, thus they are not suitable for the more complicated 3D continuous-valued data. Xu et al. [47] mine 3D subspace clusters that are non-axis-parallel, so it is not within our scope. Only algorithms GS-search [19], TRICLUSTER [48], MASC [39] and MIC [37] mine subspace clusters in 3D continuous-valued data.

GS-search [19] and MASC [39] 'flatten' the continuous-valued 3D dataset into a dataset with a single timestamp. They require the clusters to occur in every timestamp, and it is hard to find clusters in dataset that has a large number of timestamps. CATSeeker, TRICLUSTER [48] and MIC [37] have the concept of subspace in all three dimensions, i.e. they mine 3D subspace clusters that are subsets of attributes and subsets of timestamps.

TRICLUSTER [48] is the pioneer work on mining 3D subspace clusters with the concept of subspace in all three dimensions. Its clusters are highly flexible as users can use different homogeneity functions such as distance, shifting and scaling functions. Users are required to set thresholds on the parameters of these homogeneity functions and clusters that satisfy these thresholds are mined.

TRICLUSTER, along with most of the subspace clustering algorithms, are parameter-based (clusters that satisfied the parameters are mined), and their results are sensitive to the parameters. In general, it is difficult to set the correct parameters, as they are not semantically meaningful to users. For example, the distance threshold [28], [48] is a parameter that is difficult to set; at any distance threshold setting, different users can perceive its degree of homogeneity differently. Moreover, at certain settings, it is possible that a large number of clusters will be mined.

Algorithm MIC [37] proposed mining significant 3D subspace clusters in a parameter insensitive way. Significant clusters are intrinsically prominent in the data, and they are usually small in numbers. There are also works that use the concept of significance, but they focus on mining interesting subspaces [20], [36] or significant subspaces [6], and not on the mining of subspace clusters.

Both TRICLUSTER and MIC do not allow incorporation of domain knowledge into their clusters, and their clusters are not actionable. Only CATSeeker and MASC [39] can achieved these. However, CATSeeker is better than MASC, in the handling of subspace clusters in 3D data and in terms of efficiency and scalability.

1) CATSeeker mines CATSs, which are subspace clusters in 3D subspaces, while MASC mines subspace clusters, which must occur in every timestamp of the dataset.
2) CATSeeker uses a SVD-based algorithm to effectively prune the search space, while MASC does not prune the search space.
3) CATSeeker is guaranteed to be $|\mathcal{A}|$ times faster than MASC, where $|\mathcal{A}|$ is the number of attributes. For each centroid, MASC needs to run the optimization algorithm $|\mathcal{A}|$ times, whereas CATSeeker only needs to run it once.

There is constraint subspace clustering [11], and constraint is similar to actionability, as both dictate the clustering in a semi-supervised manner. However, constraints are indicators if objects should be clustered together, while utilities (that represent actionability) are continuous values indicating the quality of the objects.

In summary, there lacks a centroid-based, actionable 3D subspace clustering algorithm that is parameter insensitive and efficient. CATSeeker can effectively achieve all these.

6 Conclusion

Mining actionable 3D subspace clusters from continuous-valued 3D (*object-attribute-time*) data is useful in domains ranging from finance to biology. But this problem is non-trivial as it requires input of users' domain knowledge, clusters in 3D subspaces, and parameter insensitive and efficient algorithm. We developed a novel algorithm CATSeeker to mine centroid-based actionable 3D subspace clusters (CATS), which concurrently handles the multi-facets of this problem. In our experiments, we verified the effectiveness of CATSeeker in synthetic and real world data. In protein application, we show that CATSeeker is able to discover biologically significant clusters (particularly residues that form potential drug binding site) while other approaches have not succeeded. In financial application, we show that CATSeeker is 82% better than the next best competitor in the return/risk (maximizing profits over risk) ratio. For future work, we plan to develop an algorithm where the optimal centroids are mined during the clustering process, instead of using fixed centroids.

References

[1] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *ICDT*, pages 217–235, 1999.
[2] S. Bochkanov and V. Bystritsky. ALGLIB 2.0.1 L-BFGS algorithm for multivariate optimization. http://www.alglib.net/optimization/lbfgs.php, 2009.
[3] G. Buhrman et al. Allosteric modulation of Ras positions Q61 for a direct role in catalysis. *Proc Natl Acad Sci U S A*, 107(11):4931–4936, 2010.
[4] J. Y. Campbell and R. J. Shiller. Valuation ratios and the long run stock market outlook: An update. In *Advances in Behavioral Finance II*. Princeton University Press, 2005.
[5] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Data peeler: Constraint-based closed pattern mining in n-ary relations. In *SDM*, pages 37–48, 2008.
[6] C. H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *KDD*, pages 84–93, 1999.
[7] H. Cheng, K. A. Hua, and K. Vu. Constrained locally weighted clustering. *Proc VLDB Endow*, 1(1):90–101, 2008.
[8] Y. Cheng and G. M. Church. Biclustering of expression data. In *ISMB*, pages 93–103, 2000.
[9] Compustat. http://www.compustat.com [Last accessed 2009].
[10] L. De Lathauwer et al. A multilinear singular value decomposition. *SIAM J Matrix Anal A*, 21(4):1253–1278, 2000.
[11] É. Fromont, A. Prado, and C. Robardet. Constraint-based subspace clustering. In *SDM*, pages 26–37, 2009.
[12] Q. Fu and A. Banerjee. Bayesian overlapping subspace clustering. In *ICDM*, pages 776–781, 2009.
[13] E. Georgii, K. Tsuda, and B. Schölkopf. Multi-way set enumeration in weight tensors. *Mach Learn*, 2010.

[14] B. Graham. *The Intelligent Investor: A Book of Practical Counsel*. Harper Collins Publishers, 1986.
[15] B. J. Grant et al. Novel allosteric sites on ras for lead generation. *PLoS One*, 6(10):e25711, 2011.
[16] R. Gupta et al. Quantitative evaluation of approximate frequent pattern mining algorithms. In *KDD*, pages 301–309, 2008.
[17] Investopedia. http://www.investopedia.com/university/ratios/ [Last accessed 2009].
[18] L. Ji, K.-L. Tan, and A. K. H. Tung. Mining frequent closed cubes in 3D datasets. In *VLDB*, pages 811–822, 2006.
[19] D. Jiang, J. Pei, M. Ramanathan, C. Tang, and A. Zhang. Mining coherent gene clusters from gene-sample-time microarray data. In *KDD*, pages 430–439, 2004.
[20] K. Kailing, H.-P. Kriegel, P. Kröger, and S. Wanka. Ranking interesting subspaces for clustering high dimensional data. In *PKDD*, pages 241–252, 2003.
[21] J. Kleinberg, C. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. *Data Min Knowl Disc*, 2(4):311–324, 1998.
[22] H.-P. Kriegel et al. Future trends in data mining. *Data Min Knowl Disc*, 15(1):87–97, 2007.
[23] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans Knowl Disc Data*, 3(1):1–58, 2009.
[24] P. Kröger, H.-P. Kriegel, and K. Kailing. Density-connected subspace clustering for high-dimensional data. In *SDM*, pages 246–257, 2004.
[25] A. S. Kyle and W. Xiong. Contagion as a wealth effect. *J of Finance*, 56(4):1401–1440, 2001.
[26] J. Li et al. A correspondence between maximal complete bipartite subgraphs and closed patterns. In *PKDD '05*, pages 146–156, 2005.
[27] D. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math Program*, 45(1):503–528, 1989.
[28] G. Liu, K. Sim, J. Li, and L. Wong. Efficient mining of distance-based subspace clusters. *Stat Anal Data Min*, 2(5-6):427–444, 2009.
[29] S. Lukman et al. The distinct conformational dynamics of K-Ras and H-Ras A59G. *PLoS Comput Biol*, 6(9):e1000922, 2010.
[30] J. A. McCammon and S. C. Harvey. *Dynamics of Proteins and Nucleic Acids*. Cambridge University Press, 1987.
[31] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: A novel approach to projected and subspace clustering. In *KDD*, pages 533–541, 2008.
[32] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Comput.*, 1(2):281–294, 1989.
[33] J. Nocedal and S. J. Wright. *Numerical Optimization*, pages 497–528. Springer, 2006.
[34] H. R. Oppenheimer. A test of Ben Graham's stock selection criteria. *Finan Anal J.*, 40(5):68–74, 1984.
[35] S. Schubbert, K. Shannon, and G. Bollag. Hyperactive Ras in developmental disorders and cancer. *Nat Rev Cancer*, 7(4):295–0308, 2007.
[36] K. Sequeira and M. J. Zaki. SCHISM: A new approach for interesting subspace mining. In *ICDM*, pages 186–193, 2004.
[37] K. Sim, Z. Aung, and V. Gopakrishnan. Discovering correlated subspace clusters in 3D continuous-valued data. In *ICDM*, pages 471–480, 2010.
[38] K. Sim, G. Liu, V. Gopalkrishnan, and J. Li. A case study on financial ratios via cross-graph quasi-bicliques. *Inf. Sci.*, 181(1):201–216, 2011.
[39] K. Sim, A. K. Poernomo, and V. Gopalkrishnan. Mining actionable subspace clusters in sequential data. In *SDM*, pages 442–453, 2010.
[40] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD*, pages 374–383, 2006.
[41] J. F. Swain and L. M. Gierasch. The changing landscape of protein allostery. *Curr Opin Struct Biol*, 16(1):102–108, 2006.
[42] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques. In *COCOON*, pages 161–170, 2004.
[43] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.
[44] U.S. Securities and Exchange Commission. Microcap stock: A guide for investors. http://www.sec.gov/investor/pubs/microcapstock.htm, 2009.
[45] K. Wang, S. Zhou, and J. Han. Profit mining: From patterns to actions. In *EDBT*, pages 70–87, 2002.
[46] K. Wang, S. Zhou, Q. Yang, and J. M. S. Yeung. Mining customer value: From association rules to direct marketing. *Data Min Knowl Disc*, 11(1):57–79, 2005.
[47] X. Xu, Y. Lu, K.-L. Tan, and A. K. H. Tung. Finding time-lagged 3D clusters. In *ICDE*, pages 445–456, 2009.
[48] L. Zhao and M. J. Zaki. TRICLUSTER: An effective algorithm for mining coherent clusters in 3D microarray data. In *SIGMOD*, pages 694–705, 2005.

Kelvin Sim is a senior research engineer at Data Mining Department, Institute for Infocomm Research, A*STAR. He is currently pursuing a part-time PhD in Computer Engineering from Nanyang Technological University, Singapore. His research interests include subspace clustering, graph mining, co-clustering, financial data mining, and ADLs (activities of daily living) recognition.

Ghim-Eng Yap is a Scientist in the Agency for Science, Technology and Research (A*STAR). His research expertise is in Recommender Systems and Bayesian Statistics, and his current interests include Privacy-Preserving Data Mining, Business Analytics and Social IPTV. Ghim-Eng received his PhD in Computer Engineering from the Nanyang Technological University, Singapore.

David R. Hardoon is the Principal of Analytics at SAS Singapore. He is the in-house Analytics subject matter expert and business Analytics evangelist. He has established expertise in developing and applying computational analytical models for business knowledge discovery and analysis. David received his PhD in Computer Science in the field of Machine Learning from the University of Southampton.

Vivekanand Gopalkrishnan received his PhD from the City University of Hong Kong. He has worked extensively in OLAP and data warehousing, encompassing various aspects of the integrated schema management, storage architectures, indexing, query optimization and materialized views. His current interests are in querying and mining biological data (genomic, proteomic, microarray) and streaming data (sensor network, time series, multimedia, spatio-temporal).

Gao Cong received his PhD degree from the National University of Singapore, and is an assistant professor at Nanyang Technological University, Singapore. Before that, he worked at Aalborg University, Microsoft Research Asia, and the University of Edinburgh. His current research interests include geospatial keyword queries and mining social media.

Suryani Lukman received her BSc degree in biological sciences from Nanyang Technological University, Singapore and her PhD degree in chemistry from University of Cambridge, United Kingdom. She is currently a post-doctoral research fellow with the Bioinformatics Institute, Singapore. Her research interests include molecular dynamics simulation and multivariate statistical analysis.