

UNIVERSITY OF CINCINNATI

Date: 20-May-2010

I, Faris Alqadah ,

hereby submit this original work as part of the requirements for the degree of:

Doctor of Philosophy

in Computer Science & Engineering

It is entitled:

Clustering of Multi-Domain Information Networks

Student Signature: Faris Alqadah

This work and its defense approved by:

Committee Chair: Raj Bhatnagar, PhD
Raj Bhatnagar, PhD

Karen Davis, PhD
Karen Davis, PhD

Anil Jegga, DVM, MRes
Anil Jegga, DVM, MRes

Ali Minai, PhD
Ali Minai, PhD

John Schlipf, PhD
John Schlipf, PhD

Yizong Cheng, PhD
Yizong Cheng, PhD

Clustering of Multi-Domain Information Networks

A dissertation submitted to the

Graduate School

of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

in the Department of Computer Science

of the College of Engineering and Applied Science

by

Faris Alqadah

B.S. University of Cincinnati

June 2005

Committee Chair: Raj Bhatnagar, Ph.D.

Abstract

Clustering is one of the most basic mental activities used by humans to handle the huge amount of information they receive every day. As such, clustering has been extensively studied in different disciplines including: statistics, pattern recognition, machine learning and data mining. Nevertheless, the body of knowledge concerning clustering has focused on objects represented as feature vectors stored in a single dataset. Clustering in this setting aims at grouping objects of a single type in a single table into clusters using the feature vectors. On the other hand, modern real-world applications are composed of multiple, large interrelated datasets comprising distinct attribute sets and containing objects from many domains; typically such data is stored in an information network. The types of patterns and knowledge desired in these applications goes far beyond grouping similar homogenous objects, but rather involves unveiling dependency structures in the data in addition to pinpointing hidden associations across objects in multiple datasets and domains. For example consider an information network that contains the domains of authors, papers and conferences. Two authors a_1 and a_2 may work in the same research field but never publish in the same conference. Hence clustering only the domains of authors and conference would fail to place a_1 and a_2 in the same cluster; however considering the entire information network would reveal a hidden link via the papers domain, placing a_1 and a_2 in the same cluster. Notice, that knowledge discovery in the preceding example was derived by clustering objects based on their relation to other objects, as opposed to grouping objects based on their attributes. This form of relational clustering is essential for knowledge discovery in several applications: bioinformatics: exploring the clusters among the domains of genes, diseases, drugs, and patients; social networking: segmenting customers based on friendship relations, social groups, and demographics; and recommender systems: leveraging user ratings, product ratings, product functionality, and blog entries to cluster customers and products simultaneously.

Information-network clustering advances knowledge discovery in two manners. First, hidden associations amongst objects from differing domains are unveiled, leading to a better understanding of the hidden structure of the entire network. Second, local clusters of the objects within a domain

are sharpened and put into greater context, leading to more accurate local clustering. In order to extract knowledge of this form, information network clustering algorithms must consider 1) overlapping clusters and 2) a clustering structure that relates the clusters. In this dissertation we develop a framework and several algorithms for information network clustering by leveraging the above two fundamental aspects that facilitate knowledge discovery.

Current state-of-the-art information network clustering algorithms have had relative success in addressing the computational challenge of high dimensional data; however, the majority of these approaches have not addressed the fundamental aspects of overlapping clusters and clustering structure. In this dissertation, we address the information-network clustering problem from a fresh perspective and introduce a novel framework based on Formal Concept Analysis (FCA). Based on mathematical order theory, in particular, the theory of complete lattices, FCA provides a rich theoretical basis for investigating and structuring overlapping relational clustering in a single dataset. Shortcomings of previous methods were overcome by extending FCA to information networks, yielding effective and efficient information network clustering algorithms. Several empirical evaluations performed on a large variety of real-world information networks reveal that the FCA-based algorithms work more effectively and efficiently than the current state of the art.

Additionally, the dissertation addresses two drawbacks of FCA in single-edge information network (bi-clustering). One drawback is that the set of bi-clusters tends to be quite large, which makes reasoning about the bi-clusters quite difficult. We address this problem by introducing the idea of significant distinguishing sets, and an algorithm to efficiently enumerate these sets. The second shortcoming of the FCA framework is the strict definition of a bi-cluster. FCA specifies that a bi-cluster is a maximal sub-matrix of 1s in the dataset, however, in many knowledge discovery tasks the bi-clusters of interest are those subspaces of objects and attributes that exhibit a banded structure. We show, theoretically, the correspondence between bandedness and FCA based clustering. Moreover, we present an algorithm to uncover banded structures via intelligent search of the bi-cluster lattice.

Acknowledgements

I would like to thank my advisor, Dr. Raj Bhatnagar for his support and encouragement throughout my graduate career. This dissertation would not have been possible without him. Not only did he provide me the freedom to pursue novel and interesting problems, but he also guided me at every stage in all aspects of the work. Working with him has been a great learning experience in both research and life in general. I am also thankful to the other members of my dissertation committee, Dr. Ali Minai, Dr. Yizong Cheng, Dr. Karen Davis, Dr. Anil Jegga and Dr. John Schlipf for taking the time to read and evaluate my dissertation as well as offer useful suggestions to improve the quality of the dissertation.

I am grateful to the Department of Computer Science, University of Cincinnati for providing me with a great opportunity to pursue my degree as well as financial support to pursue my research. My friends at UC and in the Machine Intelligence Lab have constantly provided me with a great avenue for discussing new ideas, giving useful input and blowing off steam; in particular I extend my gratitude to Dr. Haiyun Bian, Dr. Amit Sinha, Dr. Eric Matson, Amer Ghanem, Hassan Al Atat, Osama Abu-Ali, Kevin Schmidt, Zhen Hu, Hema Subramanian, Aditya Sinha, Chao Wu, Nidhi Gupta, and Jacob Schlather.

I was very fortunate to have had both my brother and sister, Hatim and Amel, jointly pursuing similar degrees at UC and hence always around to discuss, brainstorm, formulate and constructively debate every aspect of my research. I will never forget the patience with which they listened to me describe my work over and over again without ever complaining. My parents have provided me with unconditional love and support, in addition to teaching me everything I know about determination, hard work and perseverance; without the support, encouragement, and assistance of my family I would not be where I am today. My inlaws have also been a constant source of promotion and cheer in my life for which I'm most grateful and appreciative.

Last, but certainly not least I would like to thank my wife, Manar Maghathe, for her boundless love, care, and companionship throughout this journey. She has learned much more than she ever wanted to about Data Mining, Computer Science, and the Linux operating system as a result of her

endless support and willingness to help. She has also slept far fewer nights than she would like in her unwavering effort to help me through demanding times, tough problems and paper deadlines. I have been truly blessed by her love and patronage and would like to express my sincere gratitude to her.

Contents

Abstract	i
Acknowledgements	iv
1 Overview	1
1.1 Introduction and Motivation	1
1.2 Subspace Clustering	2
1.3 Multi-Domain Data Collections	4
1.4 Contributions of Dissertation	6
1.5 Dissertation Outline	7
2 Related Work and Preliminary Concepts	8
2.1 Basic Definitions	8
2.2 Overview of Full Dimensional Clustering	8
2.2.1 Defining Clusters	8
2.2.2 Proximity Measures	9
2.2.3 Full Dimensional Clustering Algorithms	10
2.3 Subspace Clustering	11
2.3.1 Bottom up approaches	12
2.3.2 Top Down Approaches	15
2.3.3 Categorical Data	17
2.3.4 Bi-clustering Algorithms	18
2.3.5 Closed Patterns, Maximal Bi-cliques and Formal Concepts	22
2.4 Information Network Clustering	23
2.4.1 Multi-way Clustering	23
2.4.2 Relational Clustering	26
2.4.3 Information Network Clustering	27

2.5	Detecting Group Differences	28
2.6	Banded Matrices	29
3	FCA: Theoretical Foundation	30
3.1	Formal Concept Analysis	30
3.1.1	Concept Lattice	33
3.2	Enumerating Formal Concepts	34
3.2.1	Closed Itemset Mining	35
3.2.2	Computing the Concept Lattice	35
3.2.3	Incremental Approaches	36
3.3	Information Networks	37
3.4	Conclusion	38
4	<i>n</i>-Concepts in Star-Shaped Networks	39
4.1	<i>n</i> -Concepts	39
4.1.1	Problem Formulation	41
4.1.2	Quality Measure	44
4.2	Algorithms for Enumerating <i>n</i> -concepts	47
4.2.1	Preprocessing and Defining Search Space	47
4.2.2	Identifying Candidate <i>n</i> -concepts	52
4.2.3	Forming <i>n</i> -concepts	57
4.2.4	Asserting High Quality	58
4.2.5	NClu Algorithm	59
4.3	Experimental Results	62
4.3.1	Performance Tests	62
4.3.2	Effect of <i>k</i> and β	63
4.3.3	Clustering Results	65
4.4	Conclusion	74

5	Information-Tree Clustering	76
5.1	Problem Formulation	77
5.1.1	Connectivity and Self Connectivity in a Single Context	77
5.1.2	Connectivity in an Information Tree	83
5.1.3	Cluster Definition	85
5.2	TreeClu Algorithm	88
5.2.1	Initial Candidate Clusters	88
5.2.2	Local Refinement	90
5.2.3	Network Refinement	94
5.2.4	Analysis	97
5.3	Experimental Results	100
5.3.1	Real world datasets	100
5.3.2	Parameter Study and Sample clusters	106
5.4	Conclusion	109
6	Information-Network Clustering and Game Theory	110
6.1	Game Theory	111
6.2	Bi-Clustering as a Game	113
6.2.1	Satisfaction Reward Function	114
6.2.2	Connectivity Based Reward Functions	117
6.3	NashClu Algorithm	117
6.3.1	Formal Concepts as Candidates	118
6.4	Information-network Clustering as a Game	121
6.5	NetNashClu Algorithm	122
6.5.1	TreeClu as a Special Case and Analysis	124
6.6	Preliminary Experimental Results	124
6.7	Conclusion	126

7	Distinguishing Sets	127
7.1	Problem Model	128
7.1.1	Motivating Application	129
7.1.2	Quantifying Distinction in Θ	130
7.2	MIDS Algorithm	133
7.2.1	Adapting Prim's algorithm	133
7.2.2	Computing Upper Neighbors	136
7.2.3	Optimized neighbors	137
7.2.4	Optimizing upper neighbors check	139
7.3	Experimental Results	143
7.3.1	Application to synthetic data	144
7.3.2	Experiments in Real Data	147
7.4	Conclusion	149
8	Maximally Banded Sub-Matrices	150
8.1	Problem Definition	151
8.2	Banded Structures and FCA	152
8.2.1	Banded sub-matrices and paths in the concept lattice	155
8.3	MMBS Algorithm	159
8.3.1	Search Space	160
8.3.2	Determining top bands	160
8.4	Pseudocode and complexity	161
8.4.1	Speeding up the algorithm	163
8.5	Experimental Results	163
8.5.1	Synthetic datasets	164
8.5.2	Real world datasets	167
8.5.3	Performance Tests	171
8.6	Conclusion	171

9	Conclusion and Future Work	173
9.1	Future Work	175
9.1.1	N-Concept Lattices	175
9.1.2	Information Network Clustering	176
9.1.3	Distinguishing Sets	176
9.1.4	Banded Matrices	176
A	Computing Expected Self-Connectivity	177

List of Figures

1	Full dimensional clustering	2
2	Subspace clustering	3
3	An information network	4
4	Dissimilarity and similarity measures	10
5	Bi-cluster types	18
6	Bi-clustering structures	19
7	Sample context and concept lattice	34
8	List of symbols	38
9	A star-shaped information network	40
10	Direct sum of contexts and their concepts	41
11	Depiction of a 3-concept as 2 maximally filled, axis aligned rectangles in 3 dimensional Euclidean space connected by the "Genes" dimension	44
12	Objects and their associated attributesets	48
13	Prefix based search space for n -concepts	51
14	Performance Study Results on NClu	64
15	Parameter Study on NClu	65
16	Information networks utilized for clustering experiments	67
17	NClu Clustering Results	69
18	Hard clusters produced by NClu	70
19	Sample Clusters mined by NClu	71
20	Effect of K and β on clustering results	72
21	Information networks utilized for clustering experiments	72
22	Sample n -concepts in real world information networks	73
23	An information network where n -concepts fail	76
24	Sample context and concept lattice	82
25	Example contexts, concept lattices and information tree	87

26	Real-world information networks utilized in empirical evaluation of TreeClu	101
27	Clustering results with TreeClu algorithm	102
28	Parameter study on TreeClu performance	104
29	Parameter study on TreeClu clustering quality	105
30	Clusters mined by TreeClu from MER,PCR, FOURAREAS, and GENE informa- tion networks	108
31	Sample information network	111
32	Sample game displayed in normal form, with Nash equilibrium points highlighted .	113
33	Bi-clustering as the party planning game	114
34	Clustering results with NetNashClu algorithm	125
35	Sample data set and it's bi-cluster lattice	129
36	Viewing concepts as maximal rectangles	130
37	Sample data set and attempt to enumerate upper neighbors with prefix tree	137
38	Data Set and lattice characteristics	144
39	Performance Study Results	145
40	Bitmaps of experimental results on Syn1, Syn2, Syn3	146
41	Experimental results utilizing real world data sets	147
42	A fully banded matrix	151
43	Bi-clusters and maximal rectangles	153
44	Constructing banded matrices from paths in $\Theta(\mathbb{K})$	156
45	Band quality in synthetic data	165
46	Synthetic data and experimental results for banded matrices	166
47	Band quality comparison on real-world data	167
48	Real world datasets and mined sub-matrix bands	168
49	Natural interpretation of band as overlapping communities	169
50	Performance Study on MMBS	170

1 Overview

1.1 Introduction and Motivation

Clustering is a fundamental mental activity used by humans to handle the enormous amount of information they receive every day. Humans categorize entities such as people, places, events, and animals into different clusters, because processing every piece of information as a single entity would be impossible. Not only do humans categorize different entities, but they also typify a cluster by the common attributes of the entities contained in the cluster. For example, most humans identify the cluster “dog”. If a person comes across a dog, they will infer that this dog barks, even though the person may have never heard this specific dog bark before [77].

Clustering forms the foundation of unsupervised learning and knowledge discovery. Unlike supervised learning, labeled data and training patterns are not available; thus unsupervised learning techniques must discover patterns and organize these patterns into “rational” clusters. This idea is found in many different fields, but may carry different names such as numerical taxonomy (biology, ecology), typology (social sciences), customer segmentation (marketing) and partitioning (graph theory).

Clustering methodologies are applied to objects where each object is represented in terms of *features* or *attributes*. Mathematically, each object is then represented as an n -dimensional feature vector. Given this model, the basic steps to performing a clustering task can be summarized as follows [77]:

1. **Feature Selection:** Properly selecting features to represent the objects. The goal is to encode as much information as possible while limiting the information redundancy to a minimum.
2. **Proximity Measure:** Selection of a measure to quantify how “similar” or “dissimilar” two feature vectors are. Typically, these measures should ensure that each feature contributes meaningfully.
3. **Clustering criterion:** This is the criterion for selecting a cluster based on the type of clusters

	f_1	f_2	f_3	f_4
o_1	2.5	3.1	7.2	6.5
o_2	2.8	3.9	7.3	6.7
o_3	2.5	3.2	7.7	6.2
o_4	2.9	3.6	7.9	6.3
o_5	9.4	7.2	1.5	2.3
o_6	9.2	7.8	2.5	3.3
o_7	9.3	7.9	2.6	1.3

Figure 1: Full dimensional clustering

that are expected to appear in the data. For example in some data sets elongated clusters may make sense, while in others spherical clusters are better suited.

4. **Clustering Algorithms:** Once the proximity measure and clustering criterion have been selected an algorithm is necessary to unfold the clustering structure of the data.
5. **Interpretation and validation of the Results:** The results of the clustering algorithm should be validated and interpreted. Validation can be carried out with appropriate tests, while interpretation sometimes requires an expert in the application field.

Consider the dataset in figure 1, containing 7 objects and 4 features. Utilizing the Euclidean distance as a proximity measure a fairly intuitive clustering of the dataset would be to form two clusters with objects $\{o_1, o_2, o_3, o_4\}$ (green) and $\{o_5, o_6, o_7\}$ (red).

1.2 Subspace Clustering

Traditionally, clustering algorithms utilize the entire feature space, and the process is referred to as *full-dimensional* clustering. Full dimensional clustering algorithms have shown satisfactory performance on lower dimensional datasets; however, their performance has been shown to degrade as the dimensionality of datasets increases [55, 63, 77, 47]. This occurs due to the curse of dimensionality which entails proximity measures becoming increasingly meaningless and influenced by

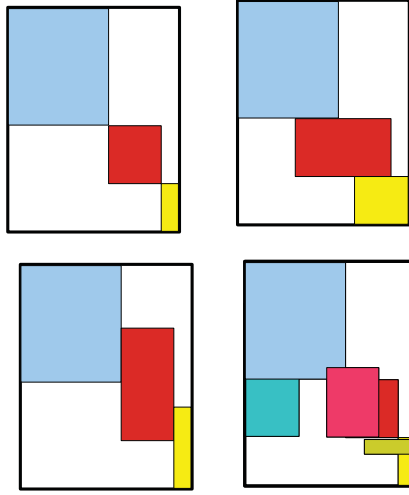


Figure 2: Subspace clustering

noise. Also, in high dimensional data many of the features are irrelevant or only relevant to a fraction of the objects. This makes it difficult for clustering algorithms to discover clusters concealed behind the noisy or irrelevant features [63]. Subspace clustering techniques attempt to overcome the shortcomings of full dimensional clustering in high dimensional data by searching for clusters within various subspaces of the feature space. These algorithms not only reveal clusters of objects, but also the subspaces of features within which they live. In other words, subspace clustering methodologies incorporate feature selection into the clustering algorithm. Subspace clusters are also referred to as *bi-clusters*, *co-clusters* and *projective clusters*. Figure 2 illustrates different subspace clustering structures. As can be seen, unlike full dimensional clustering, a subspace cluster entails clustering both the features and the objects, hence it is referred to as two-way clustering or bi-clustering.

Real world applications such as social networks, e-commerce, and bioinformatics all contain high dimensional data in which subspace clustering techniques promise enhanced knowledge discovery.

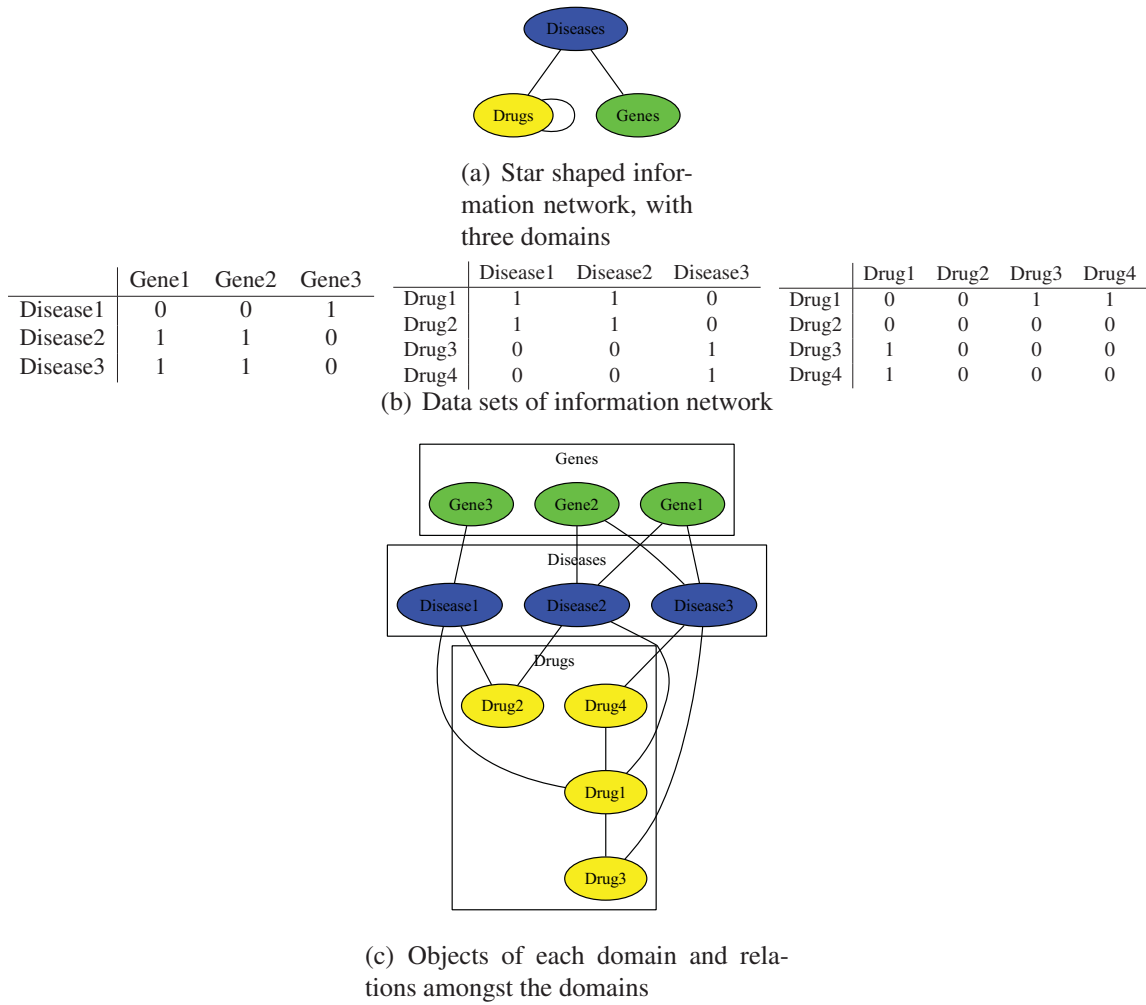


Figure 3: An information network

1.3 Multi-Domain Data Collections

Most of the above applications listed above also comprise of multiple, interrelated datasets involving several distinct attribute sets. For example, bibliographic data may contain a set of authors, keywords, research papers, and conferences, all of which are different domains. Additionally, any single dataset may define a relation between any two of these domains - one domain labeling the rows and the other the columns. Extracting useful knowledge from such multi-domain data collections requires taking the subspace clustering algorithms much farther. A multi-domain data collection may be viewed as an **information network** in which the nodes of the network are the domains and the edges are the relations between the domains; hence every edge of the information network represents a dataset. For example, in the bioinformatics field we may have the domains of *genes*, *diseases*, and *drugs* that form the nodes of the information network. Datasets relating *dis-*

eases to *genes*, *diseases* to *drugs*, and *drugs* to *drugs* form a meaningful collection for integrated analysis, and the resulting information network is shown in figure 3(a)). Notice, that the information network no longer entails viewing data as objects defined by feature vectors, but rather data objects from one dataset related to data objects in another dataset through shared domains. Hence, subspace clustering in an information network entails clustering related objects across all datasets, and defining them in terms of the subsets of attributes selected from multiple domains. In this dissertation we consider information networks in which datasets are defined as binary relations. Clustering across an information network is also referred to as **multi-way** and **relational** clustering.

Multi-way clustering across information networks advances knowledge discovery capabilities in two manners. First, hidden associations amongst objects from different domains are unveiled, leading to a better understanding of the structures hidden across the the entire information network. Considering the sample information network in figure 3, hidden associations between drugs and genes are revealed through a multi-way clustering. Intuitively, $\{(Disease2, Disease3), (Gene1, Gene2)\}$ form a subspace cluster as these objects are fully connected to each other. On the other hand, *Disease3* is associated with both *Drug3* and *Drug4* which are both associated with *Drug1* which in turn is associated with *Disease2*. Hence we may infer that $\{(Disease2, Disease3), (Drug1, Drug3, Drug4)\}$ form a subspace cluster. Aggregating these clusters we obtain the following information network cluster: $\{(Disease2, Disease3), (Gene1, Gene2), (Drug1, Drug3, Drug4)\}$. We see that this information network cluster reveals associations between a set of genes and a set drugs as mediated by a set of diseases, even though, no explicit relational data is available linking genes and drugs.

In addition to revealing hidden associations, multi-way clustering sharpens and improves the local clustering of objects within a single domain by incorporating additional information via the information network. Once again, consider the example in figure 3; a subspace clustering of Genes and Diseases unveils two distinct clusters of diseases, $\{Disease1\}$ and $\{Disease2, Disease3\}$. On the other hand, considering the relation between diseases and drugs we may revise this strict clustering as *Disease1* and *Disease2* are both only associated with the same set of drugs, and hence

form a subspace cluster in this relation.

1.4 Contributions of Dissertation

Clustering of multi-domain information networks has been addressed in the literature as multi-way [54, 53, 6, 14], information-network [73, 74], and relational clustering [17, 83, 80]. However, most of the proposed algorithms have focused on non-overlapping (hard) clusters and/or are limited in the type of network they operate on. Applications such as bioinformatics and text mining demand soft clustering (i.e. clusters may overlap); consider the simple case of a news document that belongs to several categories such as “Politics” and “Religion”, or a gene that may encompass several functional groups. Moreover, multi-way clustering algorithms do not explicitly unveil associations among the domains, rather they focus solely on clustering the objects of each domain by utilizing information in the network. On the other hand, existing information network clustering approaches have only addressed clustering in single edge and star-shaped information networks.

In this dissertation we address these shortcomings by modeling the information network clustering problem in terms of Formal Concept Analysis (FCA). FCA has served as the theoretical basis of association rule analysis, closed itemset mining [61], and bi-clustering [51][5]. Rooted in order and lattice theory, the application of FCA yields overlapping, arbitrarily positioned, hierarchical bi-clusters that form a lattice. We develop a framework for information network clustering by significantly extending the FCA model to multiple relations and constructing effective and efficient clustering algorithms within the context of this framework.

Additionally, we address two drawbacks of FCA in single-edge information network clustering (bi-clustering). One drawback of FCA-based modeling is that the set of bi-clusters tends to be quite large, which makes reasoning about the bi-clusters quite difficult. We address this problem by introducing the idea of significant distinguishing sets, and an algorithm to efficiently enumerate these sets. Significant distinguishing sets are those sets of objects and / or attributes that induce the most change among the incremental bi-clusters of a dataset. Enumerating such sets reveals the key attributes and/or objects that truly partition the data into bi-clusters. A second shortcoming of

FCA, is the strict definition of a bi-cluster. FCA specifies that a bi-cluster as a maximal sub-matrix of 1's in the dataset; however, in many knowledge discovery tasks the bi-clusters of interest are those subspaces of objects and attributes that exhibit a banded structure. We show, theoretically, how such banded bi-clusters correspond to sequences of FCA bi-clusters ordered by the bi-cluster lattice. In addition to the development of a theoretical framework, we present an algorithm to discover such banded bi-clusters.

1.5 Dissertation Outline

In chapter 2 we survey and discuss related work in subspace clustering, bi-clustering and information network clustering. Chapter 3 introduces preliminary ideas from FCA and institutes the notation that will be used throughout the dissertation. Chapters 4, 5, and 6 present information network clustering algorithms for different network structures. Initially, we present the `NClu` algorithm for mining clusters in a star shaped network. Chapter 5 expands the definition of clusters in information networks, and utilizes `NClu` as the basis of developing the `TreeClu` to mine clusters in tree-based information networks. In chapter 6 a general framework, based on game theory, is proposed for defining and enumerating clusters in information networks with differing topologies and clustering criterion. Chapters 7 and 8 focus on bi-clustering: Chapter 7 establishes the conception of distinguishing sets and proposes the `MIDS` algorithm to to enumerate such sets, while chapter 8 illustrates the `MMBS` algorithm for discovering banded bi-clusters. Finally, chapter 9 concludes the dissertation and explores future directions of research.

2 Related Work and Preliminary Concepts

In this chapter a review of the existing full dimensional, subspace, and multi-way clustering methodologies that are related to our research in mining of information networks is outlined. The existing research is also compared and contrasted with our proposed solutions.

2.1 Basic Definitions

A *database*, *data table*, or *data set* \mathcal{D} is an $m \times n$ matrix where each row represents an *object*. Each column is labeled by a *feature*, also referred to as a *dimension*, *attribute*, or *item*. Each row of the matrix is referred to as *data point* or *n-dimensional feature vector*.

Varying definitions have been proposed for the definition of a cluster. However, most of these definitions are vague and circular in nature [32]. In [32] data points are viewed as points in the n -dimensional space and clusters are described as “continuous regions of this space containing a relatively high density of points, separated from other high density regions by regions of relatively low density of points.”

2.2 Overview of Full Dimensional Clustering

2.2.1 Defining Clusters

In [77] full dimensional clustering is defined in terms of a *hard* clustering, where each data point only belongs to one cluster. Let \mathcal{D} be an $m \times n$ data set and $X = \{x_1, \dots, x_m\}$ be the data points of \mathcal{D} . Then [77] defines a hard clustering of \mathcal{D} , C , as a partitioning of X into l sets, C_1, \dots, C_l such that the following conditions are met:

1. $C_i \neq \emptyset, i = 1, \dots, l$
2. $\bigcup_{i=1}^l C_i = X$
3. $C_i \cap C_j = \emptyset, i \neq j, i, j = 1, \dots, l$

In addition, the data points contained in each C_i are more “similar” to each other and “less similar” to the data points of the other clusters. In order to quantify the degree of similarity or dissimilarity between data points the notion of a proximity measure is required.

A **soft clustering** of the data differs from hard clustering in that a data point may belong to more than one cluster.

2.2.2 Proximity Measures

Full dimensional clustering algorithms define similarity and dissimilarity between two feature vectors using *dissimilarity measures* and *similarity measures*. A dissimilarity measure d on a dataset \mathcal{D} is a function:

$$d : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{R} \quad (1)$$

where \mathcal{R} is the set of real numbers, such that

$$\exists d_0 \in \mathcal{R} : -\infty < d_0 \leq d(x, y) < +\infty, \forall x, y \in \mathcal{D} \quad (2)$$

$$d(x, x) = d_0, \forall x \in \mathcal{D} \quad (3)$$

$$d(x, y) = d(y, x), \forall x, y \in \mathcal{D} \quad (4)$$

A similarity measure s on \mathcal{D} is defined as :

$$s : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{R} \quad (5)$$

such that

$$\exists s_0 \in \mathcal{R} : -\infty < s(x, y) \leq s_0 < +\infty, \forall x, y \in \mathcal{D} \quad (6)$$

$$s(x, x) = s_0, \forall x \in \mathcal{D} \quad (7)$$

$$s(x, y) = s(y, x), \forall x, y \in \mathcal{D} \quad (8)$$

Euclidean distance	$\sqrt{\sum_{i=1}^l (x_i - y_i)^2}$	Real valued vector, dissimilarity measure
Chebyshev distance	$\max_i(x_i - y_i)$	Real valued vector, dissimilarity measure
Manhattan distance	$\sum_{i=1}^l x_i - y_i $	Real valued vector, dissimilarity measure
Hamming distance (binary valued)	$\sum_{i=1}^l (x_i - y_i)^2$	Discrete valued vector, dissimilarity measure
Tanimoto coefficient	$\frac{X \cdot Y}{\ X\ ^2 + \ Y\ ^2 - X \cdot Y}$	Binary valued vector, similarity measure
(a) Real valued vectors		

Figure 4: Dissimilarity and similarity measures

Given two feature vectors x, y of length l , figure 4 shows some popular dissimilarity measures and similarity measures.

2.2.3 Full Dimensional Clustering Algorithms

Full dimensional clustering algorithms may be divided into the following broad categories:

1. **Sequential Algorithms:** Algorithms that produce a single hard clustering. These methods are fast and simple to implement. However the final result may depend on the order in which the data vectors are presented to the algorithm. Examples of these algorithms are the BSAS algorithm [8] and the MBSAS scheme. Both of these clustering methodologies are sensitive to the order in which data vectors are presented to the algorithm. In addition the user must specify the maximum number of clusters to be enumerated.
2. **Hierarchical Algorithms:** These differ in philosophy from sequential algorithms. Instead of producing a single clustering of the data they produce a hierarchy of clusters. These are further divided into *agglomerative* algorithms that produce a decreasing number of clusters at each step while *divisive* algorithms work in the opposite direction [77]. Due to the high computational cost of hierarchial algorithms special types of these algorithms have been developed specifically for high dimensional data sets that typically occur in bioinformatics, web mining, and market basket analysis. These include the CURE algorithm [38] and ROCK [39].

3. Cost function optimization: These algorithms produce a clustering based on optimizing a cost function J . Usually in these the number of clusters is kept fixed and differential calculus concepts are used to produce successive clusterings that optimize J . The best known algorithm of this category is the K -means algorithm [10, 72].

2.3 Subspace Clustering

In high dimensional data it is often the case that many dimensions are irrelevant for clustering. Moreover noisy features may in fact hinder full dimensional clustering algorithms yielding poor results. For this reason methods are needed that uncover clusters in subspaces of the original feature space. In addition, subspace clusters are needed and make sense in several application fields, such as biology and text mining as they provide deeper insight into the structure of data and a finer grained approach than traditional clustering algorithms. Subspace clustering has been referred to by many different names in many different fields [55, 21]. Names such as co-clustering, bi-clustering, and bidimensional clustering all refer to the same problem formulation as subspace clustering. In full dimensional techniques clustering is applied to either the rows or columns of the data table. Subspace clustering on the other hand performs clustering on both the rows and columns simultaneously. Thus subspace clustering algorithms localize the search for relevant dimensions, allowing them to find clusters in multiple, possibly overlapping subspaces [63]. Therefore a subspace cluster is viewed as a pair $(\mathcal{C}, \mathcal{D})$, where \mathcal{C} is a subset of the data points and \mathcal{D} is a subset of the attributes. Two major branches of subspace clustering techniques exist based on their strategies: top-down approaches and bottom up approaches. The top down approaches first find clusters in the full dimensional space, then evaluate the subspaces of each cluster, iteratively improving the result. Bottom up approaches on the other hand find candidate regions in lower dimension subspaces and combine them to form the larger clusters.

Moreover, several bi-clustering and co-clustering algorithms have been developed for specific applications such as text clustering and bio-informatics. These bi-clustering and co-clustering methods extract sub-matrices of the original dataset such that an error measure, such as mean

square error, for values in the sub-matrix is below some threshold [55, 21].

2.3.1 Bottom up approaches

The `CLIQUE` [67] algorithm is a classical grid-based bottom up subspace clustering algorithm. The feature space is partitioned by applying an l -dimensional grid of edge size ξ . Each unit in the grid is denoted by $u_{t_1} \times \dots \times u_{t_k} (t_1 < \dots < t_k, k \leq l)$, where $u_{t_i} = [a_{t_i}, b_{t_i})$ is a right-open interval in the partitioning of the t_i -th dimension of the feature space. A point x is *contained* in a k -dimensional unit $u = u_{t_1} \times \dots \times u_{t_k}$ if $a_{t_i} \leq x_{t_i} < b_{t_i}$ for all t_i . A unit u is called *dense* if the fraction of the total number of data points contained in u is above some user-defined threshold. Two k -dimensional units $u = u_{t_1} \times \dots \times u_{t_k}$ and $u' = u'_{t_1} \times \dots \times u'_{t_k}$ share a *face* if there are $k - 1$ dimensions such that $u_{t_j} = u'_{t_j}, j = 1, 2, \dots, k - 1$ and either $a_{t_k} = b'_{t_k}$ or $b_{t_k} = a'_{t_k}$. Two k -dimensional units u_1 and u_2 are said to be *directly connected* if they have in common a $(k - 1)$ dimensional face. In addition, two k -dimensional units u_1 and u_2 are said to be *connected* if there exists a sequence of k -dimensional units v_1, \dots, v_k with $v_1 = u_1$ and $v_k = u_2$ such that each pair (v_i, v_{i+1}) of units is directly connected. Finally under this formulation a cluster is defined as a maximal set of connected units in k dimensions. Utilizing this definition of a cluster `CLIQUE` proceeds with 3 main stages. First the subspaces that contain clusters are identified. The key *downward closure property* is applied in this stage. This property states that: “if there is a dense unit u in a k -dimensional space, there are also dense units in the projections of u in all $(k - 1)$ dimensional subspaces of the k -dimensional space”. Following the identification stage `CLIQUE` computes the actual clusters, and each cluster is formed in each subspace separately. This leads to the third stage in which a minimal cluster description is derived by expressing the clusters as the minimum possible union of hyper-rectangular regions. As can be expected the choice of grid size and density threshold greatly affect the quality of clusters mined by `CLIQUE`.

`ENCLUS` [24] is based on `CLIQUE`, however the method by which subspaces with clusters are identified differs. `ENCLUS` seeks subspaces with (a) *high coverage* (high percentage of points covered by all dense units of the subspace), (b) *high density*, and *high correlation* among the dimen-

sions of the subspace. All of these requirements imply that a subspace should contain a non-random structure, which can be expressed by utilizing the notion of entropy. In order to measure entropy $H(X^k)$ of a k -dimensional subspace $X^k (k \leq l)$ of X , a k dimensional grid is applied on it and the percentage p_i of points that fall in each unit of the grid is calculated. $H(X^k)$ is then defined as:

$$H(X^k) = - \sum_{i=1}^n p_i \log_2 p_i \quad (9)$$

where n is the total number of units. In addition, ENCLUS quantifies the degree of correlation among the dimensions using another entropy based measure called *interest*, defined as:

$$\text{interest}(X^k) = \sum_{j=1}^k H(x_{t_j}^k) - H(X^k) \quad (10)$$

where $t_1 < \dots < t_k (k \leq l)$ with $x_{t_j}^k$ denoting the j -th dimension of the X^k subspace. The higher the interest the stronger the correlation among the dimensions of X^k is. ENCLUS identifies subspaces that have “low entropy” and “high interest” (user defined parameters). Like CLIQUE , ENCLUS is insensitive to the order in which the data is presented and is able to unravel arbitrarily shaped clusters that overlap. However, ENCLUS still suffers from the fact that it is heavily dependent upon the choice of edge size and the parameter choices for high interest and low entropy.

The MAFIA algorithm [41] applies an adaptively adjusted grid to match the distribution of data. Each dimension x_i of the feature space is divided into a window of small size d . Then the algorithm projects the points of the data set on each dimension; determines the projections that lie in each window, and sets the value of the corresponding window equal to the maximum among the projections lying in the window. Following this, two adjacent windows are merged if their values happen to be within a user defined threshold β . Following this the algorithm proceeds exactly as CLIQUE . MAFIA is still sensitive to parameter choices such as β , however the authors offer parallel versions of the algorithm which allow the algorithm to perform much faster than CLIQUE . However the computational time still grows exponentially as dimensionality increases [63].

Density-based Optimal projective clustering (DOC) algorithm [22] takes a monte-carlo approach

for iteratively computing sub-space clusters. A mathematical formulation is presented to capture the notion of an optimal sub-space cluster. A subspace cluster is defined using the following notation. Let $p = (p_1, \dots, p_d)$ be a point in R^d . Then it is required for subspace cluster (C, \mathcal{D}) for $|C|$ to be sufficiently large and the projection of C on the subspace spanned by \mathcal{D} must be contained in a hyper-cube of given edge-length w . It is further pointed out in [22] that the intuitive objective of many subspace clustering techniques is to maximize the number of points in a cluster; while the dimensionality of the cluster should also be maximized, since more bounded dimensions encode more amounts of correlation. However it is pointed out that these two objectives are at odds [22]. In other words, the more points a cluster contains the smaller the dimensionality is bound to be and vice versa. Therefore, the solution is to specify a trade-off between the number of points and dimensionality of a cluster. Taking these into account the DOC algorithm defines a quality measure for a subspace cluster (C, \mathcal{D}) as follows:

1. The quality of (C, \mathcal{D}) is defined as $\mu(|C|, |\mathcal{D}|)$
2. For any $0 \leq \beta < 1$ it is said that μ is β -balanced if $\mu(a, b) = \mu(\beta a, b + 1)$ for all $a > 0, b \geq 1$

The subspace clustering problem is now to compute a cluster that maximizes μ given a value of β . Intuitively, β represents the percentage of points the user is willing to drop for each added dimension. DOC creates a small subset \mathcal{X} of data points called the discriminating set by random sampling. The discriminating sets are used to discriminate between relevant and irrelevant dimensions for a cluster. For a given cluster pair (C, \mathcal{D}) , instances $p \in C$, and instances $q \in \mathcal{X}$ the following should hold true: for each dimension $i \in \mathcal{D} |q(i) - p(i)| \leq w$. The algorithm is repeated with the best result being reported. The results are heavily dependent on parameter settings. The value of w determines the maximum length of one side of a cluster, although a heuristic is presented to choose an optimal value for w , a parameter specifying the minimum density of a subspace is also required. The main attractions of this approach is that it presents a framework for creating a quality measure based on very intuitive ideas.

Sequeira et. al [47] proposed the SCHISIM algorithm to mine subspace clusters. While their

approach does not guarantee mining all interesting subspaces, the algorithm performs in a more reasonable time than `CLIQUE` , `MAFIA` , and `ENCLUS` . A non-linear monotonically decreasing threshold function is proposed to speed up the previous algorithms. To measure the level of interestingness of clusters the Chernoff-Hoeffding bound is utilized. The actual algorithm proceeds by discretizing the dataset and performing a depth-first search with backtracking to discover maximal interesting subspaces.

In [40, 18] the idea of building a lattice of subspace clusters is introduced. Moreover many interesting properties, besides minimum density, are introduced to help the user in discovering interesting subspace clusters. These properties include: succinctness, max, min, average properties, and distance measurement properties. Furthermore, by organizing the subspace clusters into lattice structures semantic insights can be gained by viewing the generalizations and specializations of a subspace cluster. The algorithms presented in [40, 18] were shown to outperform `CLIQUE` in terms of computation cost. However, the methodologies did not address the problems of 1) mining subspace clustering and simultaneously building a lattice 2) mining the extremely large size lattices resulting when minimum density parameters are set to lower values.

2.3.2 Top Down Approaches

The `PROCLUS` algorithm [1] was the first top-down subspace clustering algorithm. It borrows concepts from the full dimensional k -medoids algorithm; it samples the data, selects k medoids and iteratively improves the clustering. Given as inputs to the algorithm are the number of clusters, k , and the *average dimensionality* s of the subspaces where the clusters lie. The algorithm consists of three main steps: *initialization*, *iterative stage*, and *refinement stage*. During initialization a greedy approach is used to select a set of medoids that are far apart from each other, while attempting to ensure that each cluster is represented by at least one instance in the selected set. During the iteration phase a random set of k medoids from the reduced dataset replaces bad medoids and determines if the clustering has improved. For each medoid a set of dimensions is chosen whose average distances are small compared to statistical expectations. The total number of dimensions

associated with medoids must be $k * s$. Once subspaces have been determined for each medoid, points are assigned to medoids in order to form clusters using the average Manhattan segmental distance. In the refinement phase new dimensions for each medoid are computed based on the clusters formed; outliers are also removed. Like many of the top-down methods, PROCLUS is biased towards clusters that are hyper-spherical in shape. Moreover, all subspaces must be of similar size and non-overlapping. Due to sampling the computational performance of PROCLUS is faster than CLIQUE for large datasets.

ORCLUS is an extension of PROCLUS [2] that seeks non-axis aligned parallel subspaces. It was observed that many datasets contain inter-attribute correlations. Once again the algorithm can be divided into three steps: *assign clusters, subspace determination, and merge*. During the assign phase, the algorithm iteratively assigns data points to the nearest cluster centers. The distance between two points is defined in a subspace S , where S is a set of orthonormal vectors in some d -dimensional subspace. The covariance matrix of each cluster is computed, and the subspace S is redefined by selecting the orthonormal vectors with the smallest eigenvalues. Clusters that are near each other and have similar directions of least spread are merged during the merge phase. Although the number clusters and the size of the subspace dimensions must be specified, the authors provide a general scheme for selecting suitable values. The algorithm is computationally intensive due to the computation of covariance matrices; therefore, random sampling is used to improve speed. ORCLUS is also biased towards hyperspherical clusters because the mean of a cluster is used as a representative.

A unique distance measure is introduced in [79], along with an algorithm similar in structure to PROCLUS. The distance measure *Dimension Oriented Distance (DOD)* tallies the number of dimensions on which two instances are within a threshold distance ϵ of each other. An underlying assumption is that in high dimensional data it is more meaningful for two instances to be close in several dimensions rather than in a few. This algorithm, FINDIT, also consists of three phases: *sampling, cluster formation, and data assignment*. The sampling phase selects two small sets of randomly sampled data. These are used to determine the initial representative medoids of the clus-

ters. In the cluster formation phase correlated dimensions are found using the *DOD* measure for each medoid. The algorithm then increments ϵ and repeats the previous step until the cluster quality stabilizes. Finally, all instances are assigned to medoids. The two input parameters required by *FINDIT* are the minimum number of instances in a cluster and the minimum distance between two clusters. Subspaces of varying size are found, unlike *PROCLUS*, and moreover the ϵ value on which *DOD* depends on is determined by the algorithm in an iterative fashion. However, determining the best threshold adds significantly to the running time of the algorithm; due to this fact *FINDIT* utilizes sampling.

2.3.3 Categorical Data

All the subspace clustering algorithms reviewed in the preceding section were designed for numeric data and not categorical data. Adapting these methodologies for categorical data introduces several new problems.

DOC, *PROCLUS*, *ENCLUS*, *ORCLUS* and bi-clustering methods all require numerical computations for determining means and variances. On the other hand algorithms such as *CLIQUE*, *MAFIA*, and *SCHISM* all depend on dense regions to be ordered. However no order exists for categorical data. *CACTUS* [36] mines subspace clusters in categorical data utilizing summary information from the dataset, and a k -subspace cluster is defined as *maximal, dense, and strongly connected* subspace of the original dataset **D**. Cluster projections are first taken over the individual attributes. In order to reduce the complexity of this step, the authors assume the existence of a *distinguishing number* (k), the minimum size of an attribute-value set that uniquely occur within only one cluster. These sets (called distinguishing sets) are computed on each attribute and then extended to find cluster candidates over multiple attributes. These candidates have to be validated against the original dataset. *CACTUS* suffers from three main pitfalls: the distinguishing set assumption is unnatural, no extension step follows the projection step, and finally only ordered subspaces of attributes can be mined.

Zaki et. al [59] introduced *CLICKS* to overcome the shortcomings of *CACTUS*. While the two

2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	3.0	1.0	5.0	S1	S1	S1	S1
2.0	2.0	2.0	2.0	3.0	3.0	3.0	3.0	3.0	4.0	2.0	6.0	S1	S1	S1	S1
2.0	2.0	2.0	2.0	1.0	1.0	1.0	1.0	1.0	5.0	6.0	4.0	8.0	S1	S1	S1
2.0	2.0	2.0	2.0	7.0	7.0	7.0	7.0	7.0	1.0	2.0	0.0	4.0	S1	S1	S1

(a) Bi-cluster with constant values (b) Bi-cluster with constant values on rows (c) Bi-cluster with coherent values (d) Bi-cluster with coherent evolution

Figure 5: Bi-cluster types

approaches share the same definition of a subspace cluster, the authors of [59] introduce the novel notion of modeling the dataset as a k -partite graph. Under this model the subspace clusters correspond to maximal k -partite cliques in the graph. Furthermore, a postprocessing step is introduced to merge the k -partite cliques together in order to form clusters. Under this formulation `CLICKS` is able to mine arbitrary subspaces of attributes and does not rely on the distinguishing set assumption.

2.3.4 Bi-clustering Algorithms

Bi-clustering algorithms were developed in parallel with subspace clustering algorithms, but have mainly focused on biological data analysis. Consider a data set that contains n samples and m features organized as a matrix $A = (a_{ij})_{m \times n}$. Let $\mathbf{S}_1, \dots, \mathbf{S}_r$ be a one-way clustering of the samples and $\mathbf{F}_1, \dots, \mathbf{F}_r$ be a one-way clustering of the features. Then a bi-clustering \mathbf{B} of A is $\mathbf{B} = ((\mathbf{S}_1, \mathbf{F}_1), \dots, (\mathbf{S}_r, \mathbf{F}_r))$. We may therefore think of the bi-clusters as sub-matrices of A that exhibit some homogeneity criterion. In [55] four major classes of bi-clusters were identified (see figure 5):

1. Bi-clusters with constant values.
2. Bi-clusters with constant values on rows or columns.
3. Bi-clusters with coherent values.
4. Bi-clusters with coherent evolutions.

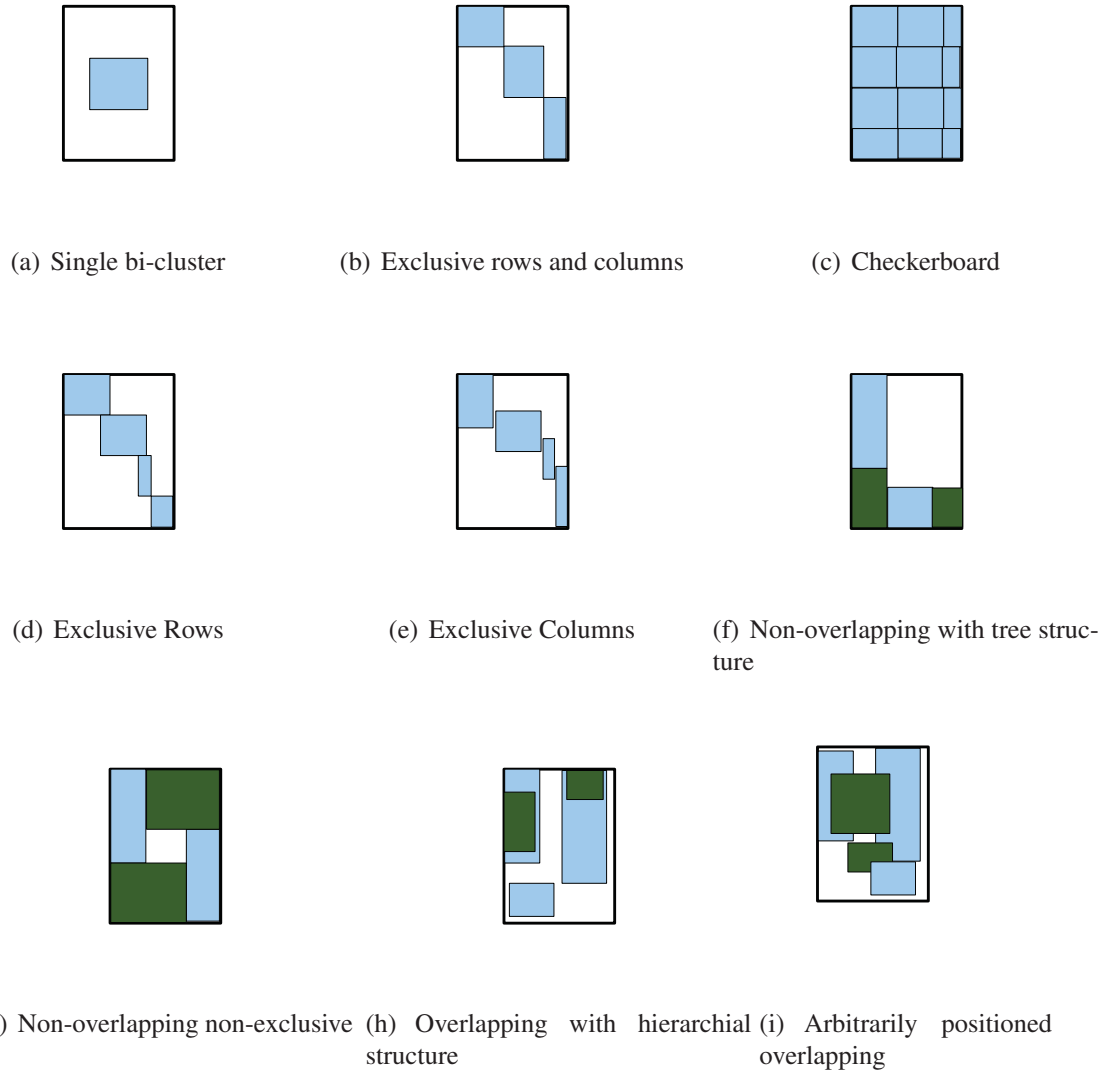


Figure 6: Bi-clustering structures

Most bi-clustering methodologies seek only one bi-cluster or k bi-clusters where k is defined apriori. When the bi-clustering algorithm assumes the existence of several bi-clusters in the data matrix several bi-clustering structures can be obtained (figure 6):

1. Exclusive row and column bi-clusters.
2. Non-overlapping bi-clusters with checkerboard structure.
3. Exclusive-row bi-clusters.

4. Exclusive-columns bi-clusters.
5. Non-overlapping bi-clusters with tree structure.
6. Non-overlapping nonexclusive bi-clusters.
7. Overlapping bi-clusters with hierarchial structure.
8. Arbitrarily positioned overlapping bi-clusters.

Bi-clustering algorithms that aim to find bi-clusters with constant values (figure 5(a)) attempt to find subsets of rows and columns which have constant values, under suitable permutation of the data matrix. In other words the rows and columns can be re-ordered in order to discover such bi-clusters. These approaches produce good results in non-noisy datasets. Since most datasets tend to be noisy, approaches, such as Hartigan's direct clustering algorithm [43] perform an analysis of variance on the values, in which noise may mask bi-clusters with constant values. The goal is to discover sub-matrices that minimize the variance, however, under this definition single-row or single-column matrices may be considered ideal. To address this problem, Hartigan fixes the number of bi-clusters a-priori and minimizes the objective function:

$$\sum_{k=1}^r \sum_{i \in S_k} \sum_{j \in F_k} (a_{ij} - \mu_k)^2 \quad (11)$$

where μ_k is the average number in the bi-cluster:

$$\mu_k = \frac{\sum_{i \in S_k} \sum_{j \in F_k} a_{ij}}{|S_k| |F_k|} \quad (12)$$

In his paper, Hartigan also mentioned that other objective functions that minimize the variance in rows and variance in columns may also be used; hence, he was refereing to the second type of bi-cluster.

A generalization of bi-clusters with constant values are bi-clusters that contain coherent values. Algorithms that search for such bi-clusters can be considered using an additive model to define

a perfect bi-cluster. A perfect bi-cluster with coherence values is $(\mathbf{S}_k, \mathbf{F}_k)$ whose values a_{ij} are predicted using the following expression:

$$a_{ij} = \mu_k + \alpha_i + \beta_j \quad (13)$$

where μ_k is the average value in the bi-cluster and α_i is an adjustment factor for each row, and β_j is an adjustment factor for each column. Other bi-clustering approaches assume a multiplicative model for a perfect coherent bi-cluster

$$a_{ij} = \mu_k * \alpha_i * \beta_j \quad (14)$$

Cheng and Church [23] produced one of the earliest algorithms for mining bi-clusters with coherent values. The node-detection algorithm seeks bi-clusters that minimize the mean squared residue H_k defined as

$$H_k = \sum_{\mathbf{S}_k} \sum_{\mathbf{F}_k} (r_{ij})^2 \quad (15)$$

where r_{ij} is the residue of an element defined as

$$r_{ij} = a_{ij} - \mu_{ik}^r - \mu_{jk}^c + \mu_k \quad (16)$$

where μ_{ik}^r and μ_{jk}^c are mean values of the i^{th} row and j^{th} column in the k^{th} bi-cluster. In a perfect bi-cluster, ($H = 0$) each row/column or both rows and columns can be generated by shifting the values of other rows or columns. The authors aim to discover the largest single maximal bi-cluster that has $H < \delta$, $\delta > 0$, where δ is specified by the user. In [23] it was shown that this problem is NP-complete, therefore a greedy algorithm was given, that requires $O((m+n)mn)$ time. Bryan et. al improved this algorithm by employing simulated annealing [19]. Yang et. al [81] generalized the node-detection algorithm to include missing data. The authors developed a heuristic algorithm to discover r clusters with low average residue, where the bi-clusters could be arbitrarily positioned and overlapping.

Numerous other bi-clustering algorithms have been developed [44, 11, 28, 76, 20, 50, 15]. We will further review some of these in the sequel as many information-network clustering algorithms are direct extensions of bi-clustering algorithms. A comprehensive review of all algorithms can be found in [55] and [21].

2.3.5 Closed Patterns, Maximal Bi-cliques and Formal Concepts

In this dissertation we consider bi-clusters in binary-valued data, and the extension of the concept to information networks. In [55] the correspondence between maximal bi-cliques and bi-clusters was illustrated. Furthermore, in [51] the one-to-one correspondence between closed patterns and maximal bi-clique subgraphs was proven. These results indicate that the subspace clustering problem we are tackling is *NP*-complete, as it was shown in [64] that the maximal bi-clique problem is *NP*-complete. However, many efficient closed pattern mining algorithms have been developed, specifically for mining itemsets in order to form association rules [42, 60, 58, 75]. While these algorithms are quite efficient, the main pitfall of utilizing these methodologies from a subspace clustering point of view is that only closed itemsets (attributes) are output. However when performing subspace clustering it is essential that both objects and attributes are computed. This issue was addressed in [51] and [33]. The resulting algorithms `LCM-BMC` and `mineIMBC` were shown to outperform the closed pattern algorithms when enumerating maximal bi-clique subgraphs.

Formal Concept Analysis (FCA) is a principled method of deriving a data hierarchy from a collection of objects and their properties. FCA builds upon lattice and order theory. The correspondence between FCA and closed patterns was pointed out in [61]. While the authors of [51] claim that this correspondence only occurs under a restrictive model we illustrate in chapter 3 that a natural equivalence exists between subspace clustering and FCA.

2.4 Information Network Clustering

2.4.1 Multi-way Clustering

Most work in multi-way clustering has focused on extending co-clustering techniques to relational data [53, 54, 14, 6]. However the focus has been on performing a hard clustering with a few exceptions. Initial work in [14] focused on extending the information theoretic co-clustering technique introduced by Dhillon et. al [44]. We first describe the original bi-clustering approach by Dhillon. In this work the input matrix A is considered a joint probability distribution $p(X, Y)$ between two discrete random variables X and Y which take values in the set $\{x_1, \dots, x_m\}$ and $\{y_1, \dots, y_n\}$. The goal of the bi-clustering is to cluster X into at most k disjoint clusters $\hat{X} = \{\hat{x}_1, \dots, \hat{x}_k\}$ and Y into at most l disjoint clusters $\hat{Y} = \{\hat{y}_1, \dots, \hat{y}_l\}$. Dhillon considers an optimal bi-cluster as one that minimizes

$$I(X; Y) - I(\hat{X}; \hat{Y}) \quad (17)$$

where I is the mutual information. It is further shown that the loss in mutual information can be expressed as

$$I(X; Y) - I(\hat{X}; \hat{Y}) = D(p(X, Y) || q(X, Y)) \quad (18)$$

where D is the relative entropy or Kullback-Leibler (KL) divergence between two probability distributions. Hence, finding an optimal bi-clustering is equivalent to finding a distribution q which is close to p in KL divergence. For ease of formulating the problem, the author considers the joint distributions $p(X, Y, \hat{X}, \hat{Y})$ with the goal of finding an approximation of $p(X, Y, \hat{X}, \hat{Y})$ such that

$$q(x, y, \hat{x}, \hat{y}) = p(\hat{x}, \hat{y})q(x|\hat{x})p(y|\hat{y}) \quad (19)$$

The core lemma of the paper proves that the objective function can be expressed only in terms of row-clustering or column-clustering. Hence starting with some initial clustering (C_X^0, C_Y^0) new clusterings are obtained iteratively, using column clusterings to improve row clusterings and vice

versa:

$$C_X^{t+1}(x) = \arg \min_{\hat{x}} D(p(Y|x) || q^t(Y|x)) \quad (20)$$

Utilizing the above update equation it was proved that the objective is monotonically decreased, although this process may only converge to a local minimum.

Bekkerman et.al [14] extended the formulation of Dhillon to an information network. Here we have to introduce the information network formulation. Let $\mathbf{X} = \{X_i | i = 1, \dots, m\}$ be the nodes of the information network (the variables to be clustered) and $\hat{\mathbf{X}} = \{\hat{X}_i | i = 1, \dots, m\}$ be their respective clusterings. Let E represent the set of edges in the information network. Then the multi-way clustering of the network should maximize the following function

$$\sum_{e_{ij} \in E} w_{ij} I(\hat{X}_i; \hat{X}_j) \quad (21)$$

where e_{ij} is a weight on the edges of the information network that can be used to incorporate prior information. As in the bi-clustering case, the objective function is subject to constraint by the number of desired clusters. Bekkerman et. al attempt to avoid local minima by introducing three different schemes of clustering: top-down, bottom-up and flat. In addition to these schemes a schedule is selected by the user to specify the order in which each of these schemes is utilized to cluster a specific node (variable) of the network. In the top-down scheme all elements are placed in a single cluster, then each successive cluster is randomly split. Following this the improvement phase utilizes Dhillon's approach to adjust to adjust the clusters and attempt to maximize the objective. The bottom-up approach takes the opposite approach and assigns each object to an individual cluster, then performs random merges, and corrections. Experimental results indicated that combining these clustering schemes consistently lead to better results than direct application of flat clustering.

In [54] Long et. al attempted to generalized several bi-clustering approaches that are extended-able to multi-way clustering. In this paper, the authors view an information network as a k -partite graph, where each set of vertices represents a different data type. The clustering model introduces,

is that of a relation summary network (RSN). In essence, the idea of the RSN model is to add a small number of hidden nodes to the graph to make the hidden structure of the graph explicit. To ensure that the relational summary is optimal, it is desired to be “as close” as possible to the original graph. The authors introduce a formulation for the relation summary network, in which each node in the original graph can only be connected to a single hidden node (hard clustering) and hidden nodes of the same type are linked only if the nodes connected to them are linked in the original graph. Moreover, the distance between the relation graph and the original graph must be minimal. This framework allows for several distance measures to be utilized to ensure that the RSN is as close as possible to the original. Due to this, the framework generalizes bi-partite spectral clustering [28], information-theoretic clustering [44] and K -means clustering. Long et. al gives an iterative algorithm that is guaranteed to monotonically decrease the objective function, but may also get stuck at local optimums.

Banjee et. al [6] generalized all the previously mentioned algorithms by representing relations via a multi-dimensional tensor. Previous work had focused only on pairwise relations. Their multi-way clustering formulation is based on the objective of “accurately approximating the original tensor using a reconstruction determined solely by the multi-way clustering and certain summary statistics of the original tensor”. Once again the framework in [6] focuses on hard clustering.

Zhao and Zaki presented the `TriCluster` algorithm in [85]. While this algorithm was designed specifically for 3D microarray data, the problem formulation follows a multi-way clustering approach. The algorithm searches for sub-matrices in a three dimensional matrix with certain homogeneity properties. The approach is based on mining cliques from a graph representing the original data matrix. The clusters mined by [85] are arbitrarily positioned and can be overlapping. The algorithm however requires five parameters and a post processing step, that requires further parameters. An important difference between this work and our proposed work, is that we mine maximal edge-connected rectangle as opposed to solid 3-d sub-matrices in the data.

2.4.2 Relational Clustering

The field of relational clustering has developed disjointly from multi-way clustering, although both fields address the same problem. Work in this field have also mainly focused on hard clustering, with a pre-defined number of clusters. In [17] the task of multi-type entity resolution is formulated as a relational clustering problem. Subsequently, similarity measures between two relational clusters are formulated by weighting the similarity between the attributes of the two clusters, and the similarity of edges that connect the two clusters. Utilizing this similarity measure a greedy agglomerative algorithm tailored to entity resolution is developed, but no general purpose methodology is presented. X. Yin proposed the `CrossClus` and `LinkClus` methodologies in [83] and [80]. The `CrossClus` methodology that takes advantage of a user's guidance, by allowing the user specifies the pertinent features and relations that they are interested in. The same authors introduced the `LinkClus` framework [80] which focused on efficiently computing the similarity between objects in a multi-way setting. The similarity between any two objects is measured based on the similarities between the objects linked with them. Utilizing this recursive definition of similarity along with the observations that 1) a hierarchy of structures naturally exists among objects of many types and 2) there exist power law distributions among the linkages in many domains, the authors develop the `SimTree` data structure for computing and storing the similarities between objects.

As mentioned previously, closed itemsets and subspace clusters have a close correspondence. Mining relational itemsets has been tackled in the literature [30, 27]. Early approaches to relational itemset mining focused on a *key* table, which would specify which table was of specific interest to the user. Thus the patterns discovered with these approaches answer questions as to what relational information is relevant only to the key table. Multi-way clustering on the other hand tackles the problem from a symmetric or global viewpoint. In [48] the authors also tackle the problem of relational itemset mining from a global viewpoint. However the resulting algorithm fails to enumerate all relational itemsets, focusing instead of discovering small characteristic patterns that approximate the result and neglect the objects.

2.4.3 Information Network Clustering

The term information network clustering was introduced recently by Sun et. al in two papers [73][74]. First the `RankClus` algorithm was introduced which utilizes ranking measures (measures similar to PageRank and SimRank) and clustering to mutually reinforce each other. The algorithm is efficient, as it avoid computing pairwise similarity between each object; however, it is limited to a single edge relation graph (i.e. it is a bi-clustering algorithm), and only clusters a single domain of the relation graph. A follow-up paper [74] introduces the idea of a net-cluster over a star-shaped information network. Soft clustering is utilized for the objects of all the domains, except for the target domain, which constitutes the central node in the relation graph, for which a hard clustering is maintained. The `NetClus` algorithm works iteratively refining the pre-selected k clusters utilizing the assumption that each target objects is generated from a ranking-based generative model.

To summarize, the information-network clustering problem has been approached from several angles. However, most work has not considered overlapping clusters and the algorithms pre-specify the number of clusters that the data contains. Both of these pre-conditions, are very unnatural, and may even impose a clustering on the data as opposed to extracting data driven clusters. On the positive side, the reviewed algorithms tend to be quite efficient as they employ greedy or iterative approaches that may settle for local minimums. In this dissertation we address these shortcomings, specifically for unweighed information networks. We adopt FCA as our theoretical framework and illustrate how FCA maybe extended to incorporate information network clustering. On the other hand, we address two draw-backs of FCA, which include: 1) the explosive number of clusters, and 2) the strict clustering definition. To address the first issue we introduce the concept of distinguishing sets. These are the sets of objects that truly distinguish the bi-clusters from each other. Next, we show that although the definition of FCA does not allow for banded bi-clusters (sub-matrices with a banded structure), such bi-clusters are derived naturally by considering unions of FCA bi-clusters. The next two sections of this chapter address previous work in both of these directions.

2.5 Detecting Group Differences

In this dissertation we propose a completely unsupervised approach to discovering significant distinctions among the incremental bi-clusters of a dataset. The goal of this knowledge discovery task is to discover sets of objects that truly characterize the difference between bi-clusters. A seemingly related field of research is that of emerging patterns. Emerging patterns were first introduced by Dong and Li in order to capture significant changes and differences between datasets [29]. Specifically, an emerging pattern is an itemset whose ratio of support in one dataset over another dataset is larger than a given threshold. Our work differs significantly from emerging patterns in two respects. First, emerging patterns assume the availability of a class label, and therefore are able to split a single dataset D into two datasets. Thus emerging patterns can be viewed as a supervised learning task to a large degree, because without the availability of the class label the very definition of an emerging pattern would not be possible. Moreover, emerging patterns only consider the support level to enumerate a significant differences, thus only pointing out differences in the attribute space. On the other hand we consider both changes in attributes *and* objects. The task of mining contrast sets has also been proposed in the literature [12, 78, 69]. Bay and Pazzani introduced this task in [12] with the purpose of detecting “conjunctions of attributes and values that differ meaningfully in their distribution across groups”. The goal of contrast sets is to highlight the differences between two groups, for example high and low income earners. Contrast sets once again assume the availability of a class label and dataset partitioning. Furthermore it was pointed out in [78] that contrast set mining is in fact a special case of the more general rule discovery task. In fact certain frequent itemset mining algorithms find rules corresponding to all the contrast sets discovered by the STUCCO algorithm proposed in [12]. Moreover the detection of the contrast sets does not explicitly utilize the structure of the data (itemset lattice) to detect differences, rather it depends upon testing if contrast set support is independent of group membership. This is accomplished by using the chi-square test. In our approach we propose mining distinctions based solely upon the inherent structure of the data.

2.6 Banded Matrices

The properties of banded matrices and how they relate to data analysis were first studied in [37]. The authors addressed the minimum banded augmentation (MBA) problem and the maximum banded submatrix (MBS) problem. The MBA problem is given a binary matrix K , find the minimum number of 0s that need to be modified into 1s so that K becomes fully banded. The MBS problem is given K and integer n find the maximum sub-matrix K' of K s.t. it is banded after n flips. The authors assume a fixed column permutation in the proposed solutions for both problems. While this is not a very realistic assumption in many real world scenarios, heuristic methods are proposed to determine a suitable fixed column permutation. The solution for the MBS problem builds upon the fixed-column algorithm utilized for MBA, and therefore also assumes fixed column permutations; moreover, only a single maximum sub-matrix is produced. Clearly, we may think of a banded sub-matrix as a bi-cluster; however, the FCA definition of a bi-cluster does not allow for such banded structures. On the other hand, in this dissertation we show that a one to one correspondence exists between sets of FCA bi-clusters and banded bi-clusters. Moreover, our work does not make any a-priori assumptions about the permutations, and discovers multiple (possibly overlapping) banded sub-matrices in the data.

In [56] the ecological concept of nestedness in binary data was introduced. A dataset is nested if for all pairs of rows one row is either a superset or subset of the other. It was shown in [37], however, that bandedness is a generalization of this ecological concept. In [45, 70] the authors establish a hierarchy between different classes of binary matrices. They consider banded matrices, zero partitionable matrices and nested matrices. It was shown that every banded matrix is a zero partition and they characterized how to determine if a zero partition contains a banded structure. In the numerical analysis field [68, 25, 9] work has focused on minimizing the distance of non-zero entries from the main diagonal of the matrix (bandwidth). This problem differs from the problem we addressed, as we attempt to discover several sub-matrices that have an approximate banded structure, as opposed to minimizing the bandwidth of the entire matrix.

3 FCA: Theoretical Foundation

In this chapter we illustrate how subspace clustering is very closely related to the field of Formal concept analysis (FCA). The rooting of FCA in order theory allows for greater understanding and insight into how the clusters in the data are organized and relate to each other. The theory of FCA also paves the way to extending the subspace model to an information network clustering model quite naturally. Finally, the FCA model yields arbitrarily positioned, overlapping subspace clusters with a hierarchical structure; this formulation encompasses most other subspace clustering structures.

3.1 Formal Concept Analysis

Datasets in FCA are modeled by a **formal context**.

Definition 1. A *formal context* $\mathbb{K}_{ij} = (\mathbf{G}_i, \mathbf{G}_j, \mathbf{I}_{ij})$ is a tripe that consists of two sets \mathbf{G}_i , \mathbf{G}_j and a relation \mathbf{I}_{ij} between G_i and G_j . The elements of the sets G_i and G_j are called **objects**.

Without loss of generality we may enumerate the objects of \mathbf{G}_i and \mathbf{G}_j as $\{g_i^1, \dots, g_i^{|\mathbf{G}_i|}\}$ and $\{g_j^1, \dots, g_j^{|\mathbf{G}_j|}\}$. A context may be depicted as an $|\mathbf{G}_i| \times |\mathbf{G}_j|$ binary matrix, denoted as $mat(\mathbb{K}_{ij})$, where $mat(\mathbb{K}_{ij})_{mn} = 1$ if $g_i^m \mathbf{I}_{ij} g_j^n$ and 0 otherwise. Moreover, \mathbb{K}_{ij} may also be viewed as a bipartite graph, denoted $grph(\mathbb{K}_{ij})$ with vertex set $\mathbf{G}_i \cup \mathbf{G}_j$, and edge set \mathbf{I}_{ij} . Therefore $mat(\mathbb{K}_{ij})$ is the adjacency matrix of $grph(\mathbb{K}_{ij})$. For a set $G_i \subseteq \mathbf{G}_i$, called an **object-set**, we define

$$\psi^j(G_i) = \{g_j \in \mathbf{G}_j | g_j \mathbf{I}_{ij} g_i \quad \forall g_i \in G_i\} \quad (22)$$

the objects of \mathbf{G}_j common to the objects in G_i . Dually, for a set $G_j \in \mathbf{G}_j$ we also have

$$\psi^i(G_j) = \{g_i \in \mathbf{G}_i | g_i \mathbf{I}_{ij} g_j \quad \forall g_j \in G_j\} \quad (23)$$

Without loss of generality, $\psi^j(G_i)$ may be computed as

$$\bigcap_{g \in G_i} \psi^j(g) \quad (24)$$

The cardinality of $\psi^j(G_i)$ (dually $\psi^i(G_j)$) is referred to as the **support** of G_i (G_j) in \mathbb{K}_{ij} . A **subspace** of \mathbb{K}_{ij} is any pair of object-sets $(G_i \subseteq \mathbf{G}_i, G_j \subseteq \mathbf{G}_j)$.

Definition 2. A *formal concept* of the context $(\mathbf{G}_i, \mathbf{G}_j, \mathbf{I}_{ij})$ is a subspace $C_{ij} = (G_i, G_j)$ such that $\psi^j(G_i) = G_j$ and $\psi^i(G_j) = G_i$. $\mathfrak{B}(\mathbb{K}_{ij})$ denotes the set of all concepts of the context \mathbb{K}_{ij}

Utilizing the binary matrix representation of a context, a concept (G_i, G_j) can be represented by a **maximal** sub-matrix of 1s under suitable permutations of the rows and columns of $mat(\mathbb{K}_{ij})$. The term maximal indicates that no row or column can be added to the sub-matrix without the introduction at least one zero. Given a concept (G_i, G_j) , the objects of G_i and G_j are very closely connected by the relation \mathbf{I}_{ij} . The objects of G_i determine the objects G_j and vice versa. The next proposition further illuminates this interaction [34].

Proposition 1. If \mathbb{K}_{ij} is a context, and G_i^1, G_i^2, G_i^3 are object-sets, then

1. $G_i^1 \subseteq G_i^2 \Rightarrow \psi^j(G_i^2) \subseteq \psi^j(G_i^1)$
2. $G_i^1 \subseteq \psi^i(\psi^j(G_i^1))$
3. $\psi^j(G_i^1) = \psi^j(\psi^i(\psi^j(G_i^1)))$

Proof. See [34] □

An object-set G_i is **closed** if $\psi^i(\psi^j(G_i)) = G_i$. Moreover, proposition 1 yields two closure systems on \mathbf{G}_i and \mathbf{G}_j which are dually isomorphic to each other [34]. For any object-set $G_i \subseteq \mathbf{G}_i$, $\psi^j(G_i)$ forms a subspace $(\psi^i(\psi^j(G_i)), \psi^j(G_i))$ which is always a concept. The intersection of any number of closed object-sets is always a closed object-set. This is proved by the following proposition:

Proposition 2. *If T is an index set and for every $t \in T$, $G_i^t \subseteq \mathbf{G}_i$ is an object-set then*

$$\Psi^j \left(\bigcup_{t \in T} G_i^t \right) = \bigcap_{t \in T} \Psi^j(G_i^t) \quad (25)$$

Proof.

$$\begin{aligned} g_j \in \Psi^j \left(\bigcup_{t \in T} G_i^t \right) &\Leftrightarrow g_i I_{ij} g_j \forall g_i \in \bigcup_{t \in T} G_i^t \\ &\Leftrightarrow g_i I_{ij} g_j \forall g_i \in G_i^t \forall t \in T \\ &\Leftrightarrow g_j \in \Psi^j(G_i^t) \forall t \in T \\ &\Leftrightarrow g_j \in \bigcap_{t \in T} \Psi^j(G_i^t) \end{aligned}$$

□

Utilizing proposition 2 we now show that $\mathfrak{B}(\mathbb{K})$ corresponds exactly to the set of maximal bi-cliques in a bipartite graph with no self loops. This is important as it was illustrated in [55, 51] that the maximal bi-cliques of a dataset correspond exactly to the bi-clusters (subspace clusters) of a binary dataset.

Let us recall the definition of a maximal bipartite clique in a graph $J = (V, E)$, where $V(J)$ and $E(J)$ denote the vertices and edges of J respectively. Two vertices u and v are adjacent if $uv \in E$. The set of all vertices adjacent to u is the *neighborhood of u* , denoted by $\Gamma(u)$. The neighborhood $\Gamma(X)$ for a non-empty subset X of the vertices of a graph J is the set of common neighborhoods of the vertices in X . In other words:

$$\Gamma(X) = \bigcap_{x \in X} \Gamma(x) \quad (26)$$

A *subgraph H* of J is a graph such that $V(H) \subseteq V(J)$ and $E(H) \subseteq E(J)$. Given a subset U of vertices of J the subgraph $J[U]$ *induced by U* is the subgraph with vertex set U . A graph is *bipartite* if there exists a bipartition of the vertex set V into two sets V_1 and V_2 such that every edge has one

end in X and the other in Y . Given a bipartite graph J , they are usually denoted as $H = (V_1 \cup V_2, E)$. H is a *complete bipartite graph* or *bi-clique* if $E = V_1 \times V_2$.

Definition 3. Given a graph J , a bi-clique subgraph $H = (V_1 \cup V_2, E)$ of J is a **maximal bi-clique** subgraph of J if $\Gamma(V_1) = V_2$ and $\Gamma(V_2) = V_1$.

From the above definition we can see the correspondence between formal concepts and maximal bi-cliques. Notice that $\text{grph}(\mathbb{K})$ contains no self loops since \mathbf{G}_i and \mathbf{G}_j are disjoint.

Theorem 1. Given any context \mathbb{K} , the set of all concepts $\mathfrak{B}(\mathbb{K})$ corresponds directly to the set of all maximal bi-cliques of $\text{grph}(\mathbb{K})$.

Proof. Given an object-set $G_i \subseteq \mathbf{G}_i$ let \bar{G}_i refer to the corresponding vertices in $\text{grph}(\mathbb{K})$. Clearly $\psi^j(G_i) = \Gamma(\bar{G}_i)$ by proposition 2 and equation 26. The same holds for object-sets $G_j \subseteq \mathbf{G}_j$. Moreover, any concept (G_i, G_j) corresponds exactly to a bi-clique subgraph of $\text{grph}(\mathbb{K})$ since $\bar{G}_i \in V_1$ and $\bar{G}_j \in V_2$ and all vertices are connected. Furthermore this bi-clique is maximal since $\Gamma(\bar{G}_i) = \Gamma(\bar{G}_j)$ by the definition of a concept. \square

3.1.1 Concept Lattice

Theorem 1 allows us to model subspace clustering through FCA. One advantage of the FCA model, is that we may define an ordering over the entire set of subspace clusters of a context.

Definition 4. If (A_i^1, A_j^1) and (A_i^2, A_j^2) are concepts of a context, \mathbb{K}_{ij} , (A_i^1, A_j^1) is called a **subconcept** of (A_i^2, A_j^2) , provided that $(A_i^1 \subseteq A_i^2)$ (which is equivalent to $A_j^2 \subseteq A_j^1$). In this case, (A_i^2, A_j^2) is a **superconcept** of (A_i^1, A_j^1) , and we write $(A_i^1, A_j^1) \leq (A_i^2, A_j^2)$. The relation \leq is called the **hierarchial order** of the concepts. A concept (A_i^2, A_j^2) is called an **upper neighbor** of (A_i^1, A_j^1) if $(A_i^1, A_j^1) \leq (A_i^2, A_j^2)$ and there is no concept (A_i^3, A_j^3) in \mathbb{K}_{ij} fulfilling $(A_i^2, A_j^2) \leq (A_i^3, A_j^3) \leq (A_i^1, A_j^1)$, this is denoted by $(A_i^2, A_j^2) \succ (A_i^1, A_j^1)$. The set of all concepts of $\mathfrak{B}(\mathbb{K}_{ij})$ ordered by the hierarchial order is denoted as $\overline{\mathfrak{B}(\mathbb{K}_{ij})}$ and is called the **concept lattice** of the context \mathbb{K}_{ij} .

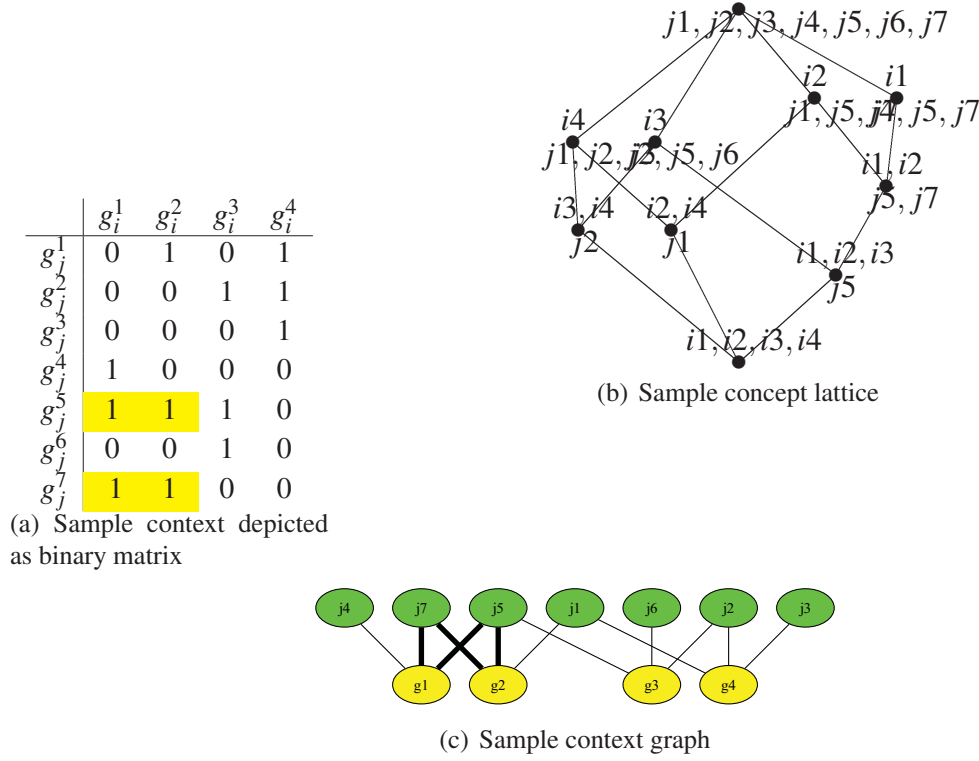


Figure 7: Sample context and concept lattice

Example 1. The context from figure 7(a) contains 10 concepts. Figure 7(b) depicts the concept lattice, while figure 7(c) depicts the context graph of this context. The concept $\{(j5, j7), (g1, g2)\}$ is depicted as both a maximal sub-matrix of $\text{mat}(\mathbb{K}_{ij})$ and as a maximal bi-clique of $\text{grph}(\mathbb{K}_{ij})$.

The Basic Theorem on Concept Lattices [34] states that the concept lattice $\overline{\mathfrak{B}(\mathbb{K})}$ is a complete lattice in which the infimum and supremum are given by [34]:

$$\bigwedge_{t \in T} (G_i^t, G_j^t) = \left(\bigcap_{t \in T} G_i^t, \Psi^j \left(\Psi^i \left(\bigcup_{t \in T} G_j^t \right) \right) \right) \quad (27)$$

$$\bigvee_{t \in T} (G_i^t, G_j^t) = \left(\Psi^i \left(\Psi^j \left(\bigcup_{t \in T} G_i^t \right) \right), \bigcap_{t \in T} G_j^t \right) \quad (28)$$

3.2 Enumerating Formal Concepts

The task of enumerating all concepts in a context has been studied extensively in the FCA and data mining communities [49]. The task is at least *NP*-complete as enumerating the largest maxi-

mal bi-clique in a graph is known to be *NP*-complete [64]. Hence to make the problem tractable, data mining approaches typically mine a subset of the concepts which satisfy user-defined constraints. These constraints typically encompass constraints on the supports of each object-set of each concept.

3.2.1 Closed Itemset Mining

Association rule analysis is a fundamental aspect of data mining [3]. In order to identify association rules it is first necessary to identify the frequent itemsets of a dataset. As mentioned previously, the concepts of a context correspond precisely to the closed itemsets of a transactional dataset. Hence, closed itemset mining algorithms may be viewed as constraint-based concept enumeration algorithms. Numerous algorithms exist to perform this task [42, 60, 51, 3, 33, 75]. However, none of these algorithms produce the concept lattice in conjunction with the concepts. The most efficient of these algorithms, LCM [75], was shown to mine the concepts in $O(|\mathfrak{B}(\mathbb{K}_{ij})| * \min(|\mathbf{G}_i|, |\mathbf{G}_j|)^2)$ time, which is linear with respect to the number of concepts in the context. In the worse case however, $|\mathfrak{B}(\mathbb{K}_{ij})|$ may be exponential with respect to \mathbf{G}_i or \mathbf{G}_j .

3.2.2 Computing the Concept Lattice

Computing the concept lattice of a context has also been widely studied in the FCA literature, and to a lesser degree data mining literature [34, 52, 40, 49, 58]. These algorithms can be either incremental or non-incremental. Non-incremental algorithms such as those described in [58, 49] do not attain the full and correct lattice structure until the termination of the algorithm. On the other hand, incremental algorithms, such as those given in [52, 16, 13, 49] compute the concept lattice one concept at a time, by determining the upper and lower neighbors of any given concept. Hence, at any given point during execution, the correct lattice structure of all discovered concepts is attained using incremental approaches. These approaches also have the added advantage of directly computing the upper and lower neighbors of a given concept locally, without storing the entire concept lattice in memory.

3.2.3 Incremental Approaches

Incremental approaches depend on the ability to compute the upper and lower neighbors of a given concept (G_i, G_j) locally. Here we review two different approaches given by Lindig [52] and Berry et. al [16]. Clearly, generating upper and lower neighbors are duals, hence we focus on generating the upper neighbors of a concept without loss of generality. In [52], the authors make use of the monotonic property of closure.

Theorem 2. *Let (G_i, G_j) be a concept of \mathbb{K}_{ij} and (G_i, G_j) is not the supremum or infimum. Then $\Psi^i(\Psi^j((G_i \cup \{g_i\})))$ where $g_i \in \mathbf{G}_i \setminus G_i$ is an object-set of an upper neighbor of (G_i, G_j) if and only if for all $y \in \Psi^i(\Psi^j((G_i \cup \{g_i\}))) \setminus G_i$ the following holds: $\Psi^i(\Psi^j((G_i \cup \{y\}))) = \Psi^i(\Psi^j((G_i \cup \{g_i\})))$.*

Utilizing theorem 2 the upper neighbors of any given concept may be generated by selecting different $g \in \mathbf{G}_i \setminus G_i$ and testing if they form an upper neighbor. The entire concept lattice may be generated by performing an implicit depth-first or breadth first traversal and enumerating the upper neighbors of every concept encountered. The complexity of generating upper neighbors with direct application of theorem 2 is $O(|G_i|^2 * |G_j|)$. To generate the entire lattice, utilizing a traversal the complexity then becomes $O(|\mathbf{G}_i|^2 * |\mathbf{G}_j| * |\mathfrak{B}(\mathbb{K}_{ij})|)$.

Berry et. al improved the running time of incremental algorithms to $O(|G_i| * |G_j|)$ per generated concept plus $O(|\mathbf{G}_i|^2 * |\mathbf{G}_j|)$ time per maximal branch the recursive tree induced by a depth first search of the concept lattice. The key to this approach is utilizing an equivalence class of objects called a *maxmod*.

Definition 5. *Let \mathbb{K}_{ij} be a context, and $x, y \in G_i$. We will say that x **dominates** y if $\Psi^j(x) \subseteq \Psi^j(y)$.*

Definition 6. *Given a context \mathbb{K}_{ij} a **maxmod** of \mathbf{I}_{ij} is an object-set G_i such that $\forall x, y \in G_i, \Psi^j(x) = \Psi^j(y)$ and $\forall z \in \mathbf{G}_i \setminus G_i, \Psi^j(z) \neq \Psi^j(x)$. In this case, $\Psi^j(G_i) = \Psi^j(x)$, where x is an arbitrary element of G_i .*

Definition 7. *Given two maxmods G_i^1 and G_i^2 , we say that G_i^1 **dominates** G_i^2 if $\Psi^j(G_i^1) \subset \Psi^j(G_i^2)$. A*

maxmod is said to be **dominating** if there is a maxmod G_i^2 which it dominates, and **non-dominating** if there is no other maxmod which it dominates.

Theorem 3. Given a concept (G_i, G_j) , $G_i \cup X$ is the object-set of a concept, \hat{C}_{ij} belonging belonging to the upper neighbors of (G_i, G_j) iff X is a non-dominating maxmod of $\mathbf{I}_{ij}(\mathbf{G}_i \setminus G_i, \mathbf{G}_j)$. In this case $\hat{C}_{ij} = (G_i \cup X, \psi^j(G_i) \cap G_j)$.

Following theorem 3 in order to identify upper neighbors of a concept efficiently, an efficient algorithm for determining the non-dominating maxmods of a relation is required. In addition to presenting an efficient method of determining the non-dominating maxmods of a context, the authors further present methods of speeding up this computation along paths of the lattice, by utilizing monotonicity arguments. For more details see [16].

3.3 Information Networks

We now generalize FCA to information networks, as opposed to a single relation. Let $\mathbf{G}_1, \dots, \mathbf{G}_n$ be a set of **domains** where each domain \mathbf{G}_i contains a set of objects $g_i^1, \dots, g_i^{|\mathbf{G}_i|}$. Without loss of generality we assume that $\mathbf{G}_i \cap \mathbf{G}_j = \emptyset$ for any $\mathbf{G}_i, \mathbf{G}_j$.

Definition 8. Let $V = (\mathbf{G}_1, \dots, \mathbf{G}_n)$, then an **information network** $\mathcal{T} = (V, E)$ is a graph where the vertices correspond to the set of domains and the edge set E is defined as

$$E = \{(\mathbf{G}_i, \mathbf{G}_j) \mid \exists \mathbb{K}_{ij}\}$$

The information network \mathcal{T} represents the relationships among the set of domains, whereas each edge in \mathcal{T} corresponds to a context. We will use $\Gamma(\mathbf{G}_i)$ to denote the set of neighboring domains of \mathbf{G}_i (i.e. the set of domains that \mathbf{G}_i shares a context with). The ψ operator can now be generalized as

$$\psi^j(G_i) = \begin{cases} \{g_j^m \in \mathbf{G}_j \mid g_j^m \mathbf{I}_{ij} g_i^n \quad \forall g_i^n \in G_i\} & \text{if } (\mathbf{G}_i, \mathbf{G}_j) \in E \\ \emptyset & \text{otherwise} \end{cases} \quad (29)$$

Symbol	Definition
\mathbf{G}_i	Domain
\mathbf{I}_{ij}	Binary relation between domains \mathbf{G}_i and \mathbf{G}_j
\mathbb{K}_{ij}	Context containing domains $\mathbf{G}_i, \mathbf{G}_j$, and \mathbf{I}_{ij}
G_i	Object-set of domain \mathbf{G}_i
g_i	Object of domain \mathbf{G}_i
C_{ij}	Concept in context \mathbb{K}_{ij}
$\mathfrak{B}(\mathbb{K}_{ij})$	Set of all concepts in \mathbb{K}_{ij}
$\overline{\mathfrak{B}}(\mathbb{K}_{ij})$	Concept lattice of \mathbb{K}_{ij}
$\mathcal{T} = (V, E)$	Information network
V	Set of domains $(\mathbf{G}_1, \dots, \mathbf{G}_n)$ in an information network
E	Edge set in information network $\{(\mathbf{G}_i, \mathbf{G}_j) \mid \exists \mathbb{K}_{ij}\}$
$\psi^j(G_i)$	$\{g_j^m \in \mathbf{G}_j \mid g_j^m \mathbf{I}_{ij} g_i^n \quad \forall g_i^n \in G_i\}$

Figure 8: List of symbols

3.4 Conclusion

We have given a brief introduction to the broad field of FCA. The field of FCA is grounded in theory of lattices, while algorithms concerning the enumeration of concepts have been well studied. In addition to serving as model for subspace clustering in binary valued datasets, FCA notation was extended to incorporate information networks. In chapters 4-6 we will present different definitions of clusters in information networks with differing topologies. In those chapters we leverage the many mathematical and algorithmic properties of concept lattices discussed here to facilitate the discovery of information network clusters. Chapters 7-8 make use of the concept lattice in a single context to enumerate distinguishing sets and maximally banded sub-matrices. The table in figure 8 summarizes the symbols and notation that will be utilized throughout the rest of the dissertation.

4 n -Concepts in Star-Shaped Networks

Making use of the FCA theoretical background we construct algorithms for enumerating information-network clusters in star shaped information networks in this chapter. Specifically, the definition of a concept in a single context is generalized to a star shaped network. Theoretically, we show that such an extension is always possible for a star-shaped network, and that the generalized concepts, n -concepts, also form a complete lattice with respect to the hierarchical order. Viewing the n -concepts as maximal rectangles, an intuitive quality criterion function based on the user's preferred trade-off between height and width is developed to rank the top n -concepts in the data. The `NClu` algorithm is proposed to efficiently enumerate n -concepts in star-shaped networks. It is shown that the computational cost of `NClu` is no more than that of enumerating concepts in a single context, while experimental results indicate that in practice `NClu` is orders of magnitude more efficient than baseline approaches. Experiments with real-world information networks indicate that n -concepts form very accurate clusters and reveal significant associations across contexts in the network. On the other hand, our experiments also reveal that the strict definition of n -concepts is a limiting factor for modeling information network clusters as n -concepts. Nonetheless, n -concepts form the basis of building more general types of clusters in information networks, as demonstrated in subsequent chapters.

4.1 n -Concepts

A running example, from the bioinformatics domain, will be utilized throughout this chapter for illustration purposes. In the example, the information network contains the domains of *Genes*, *Gene Ontology (GO) terms* and *Diseases*. The goal of an information network clustering is to unveil the underlying interactions between these three domains, while grouping similar objects in each domain together. Hence, clusters should retain the knowledge captured by local clusters of a context, while enhancing this with the added information from other contexts. Building upon the success of FCA in a single context, one strategy to achieve the goals of an information-network

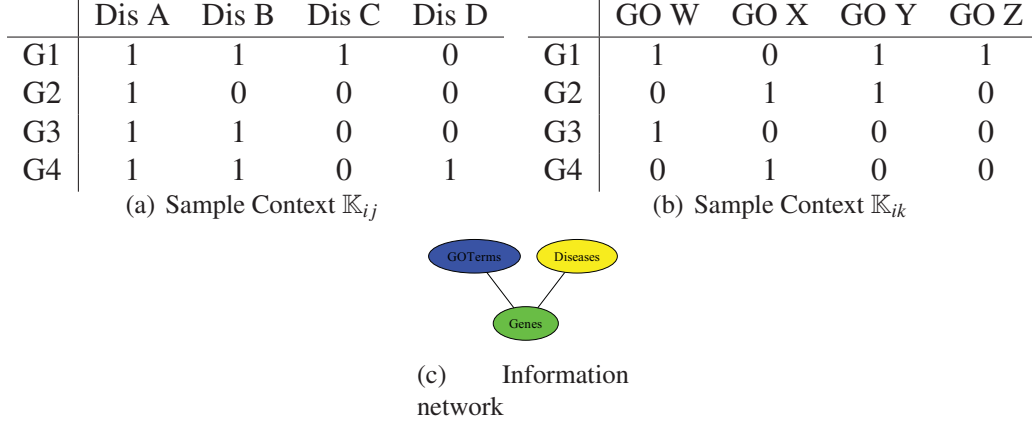


Figure 9: A star-shaped information network

clustering is to join \mathbb{K}_{ij} and \mathbb{K}_{ik} into a new context and search for formal concepts. The join of the contexts corresponds to the **direct sum** of contexts which is defined in [34] as

$$\mathbb{K}_{ij} + \mathbb{K}_{ik} = (\mathbf{G_i}, \mathbf{G_j} \cup \mathbf{G_k}, \mathbf{I_{ij}} \cup \mathbf{I_{ik}} \cup (\mathbf{G_i} \times \mathbf{G_j}) \cup (\mathbf{G_i} \times \mathbf{G_k})) \quad (30)$$

Moreover the concept lattice of the a sum of contexts is isomorphic to the product of its concept lattices. In the case of two contexts we obtain

$$\mathfrak{B}(\mathbb{K}_{ij} + \mathbb{K}_{ik}) \cong \mathfrak{B}(\mathbb{K}_{ij}) \times \mathfrak{B}(\mathbb{K}_{ik}) \quad (31)$$

Following this naive approach is not adequate for two major reasons:

1. The computational cost of performing a join and then enumerating concepts is too high. The complexity of an algorithm such as `Lattice` [52] is already expensive at $O(|\mathfrak{B}(\mathbb{K}_{ij})| * |\mathbf{G_i}|^2 * |\mathbf{G_j}|)$. Assuming $O(|\mathbf{G_j}|) = O(|\mathbf{G_k}|)$ then this jumps to at least $O(|\mathfrak{B}(\mathbb{K}_{ij})|^2 * |\mathbf{G_i}|^2 * |\mathbf{G_j}|^2)$, which is too expensive for most practical applications. This will be verified in our experimental section.
2. Many concepts in the join of \mathbb{K}_{ij} and \mathbb{K}_{ik} do not reveal associations between the concepts of \mathbb{K}_{ij} and \mathbb{K}_{ik} . For example consider the concepts in figure 10(d). The concepts $(\{A\}, \{G1, G2, G3, G4\})$ and $(\{A, B\}, \{G1, G3, G4\})$ only correspond to concepts found within

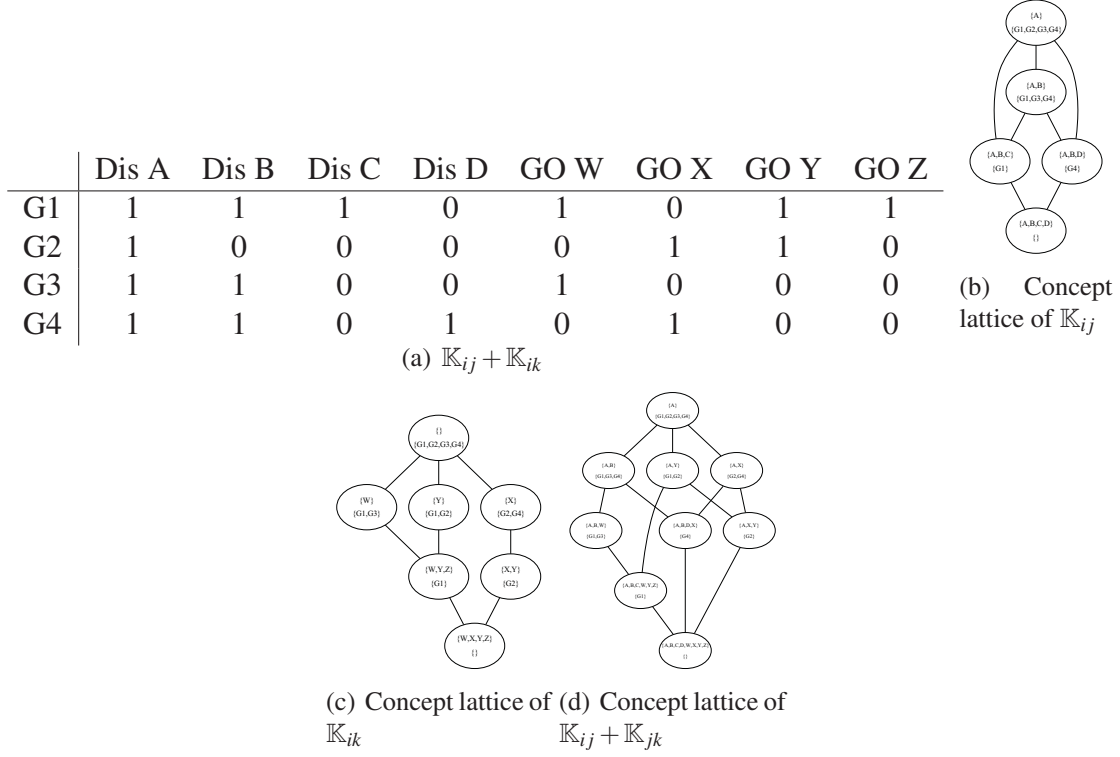


Figure 10: Direct sum of contexts and their concepts

\mathbb{K}_{ij} and thus do not reveal any association between gene-ontology terms and diseases. The next five concepts $(\{A, B, W\}, \{G1, G3\})$, $(\{A, Y\}, \{G1, G2\})$, $(\{A, X\}, \{2, 4\})$, $(\{A, B, D, X\}, \{G4\})$, $(\{A, X, Y\}, \{G2\})$ do reveal associations, however, if we partition each concept, the partitions do not correspond to local concepts in the individual contexts. In other words the knowledge extracted from a local clustering of the contexts is lost. For example consider $(\{A, Y\}, \{1, 2\})$, the partition $(\{A\}, \{1, 2\})$ is not a concept in \mathbb{K}_{ij} .

Even in this small example we see that only one of the nine concepts in the join of \mathbb{K}_{ij} and \mathbb{K}_{ik} revealed associations and retained complete local information.

4.1.1 Problem Formulation

As stated earlier, an information-network cluster should retain the knowledge extracted from local clusters of contexts, while enhancing that clustering with the added information from other contexts. To this end we propose modeling such clusters as a “matching” of formal concepts across

all contexts. In this manner, all local knowledge is completely retained while global knowledge is incorporated by disregarding those local concepts that do not contain a matching in all other domains. Such a matching naturally extends the definition of a concept to a star-shaped information network due to the topology of the network.

Definition 9. Given an information network $\mathcal{T} = (V, E)$, if $\exists \mathbf{G}_c \in V$ such that $\forall \mathbb{K} \in E$, $\mathbb{K} = (\mathbf{G}_c, \mathbf{G}_i, \mathbf{I}_{ci})$ then \mathcal{T} is a *star shaped* network. We refer to the contexts $\mathbb{K}_{ci} \in E$ as *vertically partitioned*.

Definition 10. Given a star shaped information network \mathcal{T} , an *n-concept* of \mathcal{T} is a subspace $C = (G_c, G_1, \dots, G_{n-1})$, s.t. $\Psi^i(G_c) = G_i$ and $\Psi^c(G_i) = G_c$ for all $1 \leq i \leq n-1$.

Clearly an *n-concept* is a generalization of a formal concept. A formal concept (G_i, G_j) of the single context \mathbb{K}_{ij} is simply the 2-concept (G_i, G_j) . Furthermore, every pair (G_c, G_k) $1 \leq k \leq n-1$ is a concept in \mathbb{K}_{ck} . Utilizing this fact it is straightforward to show that any *n-concept* in a star shaped information network can be represented by a formal concept of the direct sum of the all the contexts. This results leads to the fact that the set of all *n-concepts* forms a complete lattice. Thus, in addition to *n-concepts* representing information network clusters across vertically partitioned data, we can also reason about the relationships among *n-clusters*, and the data hierarchy existent within vertically partitioned data. Moreover, from an algorithmic perspective this fact allows us to transform the enumeration of the *n-concepts* into a task similar to enumerating formal concepts or frequent closed itemsets.

Proposition 3. Every *n-concept*, of a star-shaped information network $\mathcal{T} = (V, E)$ can be represented by a formal concept of the context $\mathbb{K}_{IJ} = \sum_{\mathbb{K}_{ci} \in E} \mathbb{K}_{ci}$.

Proof. Let $C = (G_c, G_1, \dots, G_{n-1})$ be an *n-concept*. We will show that $C = (X, Y) = \left(G_c, \bigcup_{i=1}^{n-1} G_i\right)$ is always a formal concept in \mathbb{K}_{IJ} . By definition of a *n-concept* $\Psi^i(G_c) = G_i$ for all k , $1 \leq k \leq n-1$. However the objects of \mathbb{K}_{IJ} are precisely those objects of any \mathbb{K}_{ci} . Thus $\Psi^J(X) = \bigcup_{i=1}^n G_i = Y$. On

the other hand by proposition 2 we have

$$\begin{aligned}
\psi^I(Y) &= \psi^I\left(\left(\bigcup_{i=1}^n G_i\right)\right) \\
&= \bigcap_{i=1}^n \psi^c(G_i) \\
&= G_c \bigcap G_c \cdots \bigcap G_c \\
&= X
\end{aligned}
\tag{32}$$

Therefore, by definition (X, Y) is a concept. □

From the above proposition it is clear that the set of n -concepts is a subset of all the formal concepts of the direct sum of the contexts. At the same time we derive the fact that the set of all n -concepts, ordered by the hierarchical order, forms a complete sublattice of the concept lattice of $\sum_{\mathbb{K}_{ci} \in E} \mathbb{K}_{ci}$. This can be shown with the idea of a **closed relation** from FCA.

Definition 11. A relation $\mathbf{J}_{ij} \subseteq \mathbf{I}_{ij}$ is called a **closed relation** of the context $(\mathbf{G}_i, \mathbf{G}_j, \mathbf{I}_{ij})$ if every concept of the context $(\mathbf{G}_i, \mathbf{G}_j, \mathbf{J}_{ij})$ is also a concept of $(\mathbf{G}_i, \mathbf{G}_j, \mathbf{I}_{ij})$.

Theorem 4. For every set $T \subset \mathfrak{B}(\mathbb{K}_{ij})$ of concepts, there is a smallest closed relation \mathbf{J}_{ij} of $(\mathbf{G}_i, \mathbf{G}_j, \mathbf{I}_{ij})$ containing all sets $G_i \times G_j$ with $(G_i, G_j) \in T$. $\overline{\mathfrak{B}}(\mathbf{G}_i, \mathbf{G}_j, \mathbf{J}_{ij})$ is the complete sublattice of $\overline{\mathfrak{B}}(\mathbf{G}_i, \mathbf{G}_j, \mathbf{I}_{ij})$ generated by T .

Proof. See [34] □

As a direct result of the above theorem we conclude that the set of n -concepts ordered by the hierarchal order forms a complete lattice. We will refer to this lattice as the **n -concept Lattice**.

As mentioned in the previous chapter, formal concepts may be thought of as maximal submatrices of 1s under suitable permutation in $mat(\mathbb{K})$. n -concepts directly correspond to concepts in the direct sum of contexts, thus we may also represent them in this manner; however, this

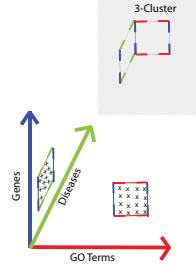


Figure 11: Depiction of a 3-concept as 2 maximally filled, axis aligned rectangles in 3 dimensional Euclidean space connected by the "Genes" dimension

representation loses the intuition behind an n -concept. On the other hand, each (G_c, G_i) of the n -concept represents a concept in the context \mathbb{K}_{ci} , hence an n -concept can be represented as $n - 1$ maximally filled, axis aligned, rectangles in n -dimensional Euclidean space. Moreover all the rectangles are connected by one dimension (the common set of objects G_c). This depiction is illustrated in figure 11. The number of 1s contained in an n -concept then corresponds directly to the combined area of the connected rectangles.

Definition 12. The *connectivity* or *area* of an n -concept, $C = (G_c, G_1, \dots, G_{n-1})$ in a star shaped information network is given by

$$\phi(C) = |G_c| \cdot \sum_{i=1}^{n-1} |G_i| \quad (33)$$

For convenience we will refer to $\sum_{i=1}^{n-1} |G_i|$ as the *width* of C

$$W(C) = \sum_{i=1}^{n-1} |G_i| \quad (34)$$

4.1.2 Quality Measure

In the worst case, a star-shaped information network \mathcal{T} , may contain $2^{|G_c|}$ n -concepts. Although this is rarely the case in real-world data, it prompts the key question: *which n -concepts are most*

informative to the user? Intuitively, we would like our result to maximize the number 1s in an n -concept, while also maximizing the number of objects from each domain. However, we recognize the trade-off between the number of objects from the central domain, \mathbf{G}_c , and the number of objects from all other domains, $\mathbf{G}_1, \dots, \mathbf{G}_{n-1}$. The more objects an n -concept contains from the central node, the fewer objects it is bound to include from all other domains, and vice-versa. Thus, the objective of maximizing the number of objects from all domains in an n -concept are at odds. Building on the trade-off ideas of [22], we first develop a quality measure for local formal concepts, and then use the same intuition to construct a quality measure for n -concepts.

Let $C = (G_c, G_i)$ denote a 2-concept in context \mathbb{K}_{ci} . Simply utilizing the connectivity of a 2-concept as a quality measure has two major pitfalls. Thinking of the 2-concept as a maximal rectangle, we first notice that the individual contributions of width and height to the total number of ones is indistinguishable when using the connectivity measure. Second, the connectivity of a 2-concept does not take into account the fact that as height increases, the width of a concept must decrease. This raises the question: *how many objects of \mathbf{G}_i are we willing to trade for each additional object from \mathbf{G}_c ?* In order to overcome these pitfalls, we introduce a parameter β which represents a trade-off value for the percentage of objects from \mathbf{G}_i we are willing to drop for each additional object added from \mathbf{G}_c . Equivalently, β represents how many units of width we are willing to drop for each additional unit of height. We now construct a quality measure, Ω centered around β . Formally, given C and β ($0 < \beta < 1$), let $\Omega(C) = \mu(|G_c|, |G_i|)$ denote the quality of C in \mathbb{K}_{ci} . Then Ω should:

$$\begin{aligned} & \text{maximize} \quad \phi(C) \\ & \text{subject to} \quad \mu(|G_c|, |G_i|) = \mu(\beta * |G_i|, |G_c| + 1) \end{aligned}$$

The above equations imply that the function $\mu(a, b)$ must satisfy the following two conditions [22]:

1. $\mu(a, b)$ should be monotonically increasing in both a and b .

2. $\mu(a, b)$ should be a β -balanced. Mathematically this implies that

$$\mu(a, b) = \mu(\beta^n a, b + n), 0 < \beta < 1, n \in \mathbb{N}^+ \quad (35)$$

One such function that satisfies conditions 1 and 2 is:

$$\mu(a, b) = a \left(\frac{1}{\beta} \right)^b \quad (36)$$

This results in the following definition for the quality of a concept.

Definition 13. Given β and a 2-concept $C = (G_c, G_i)$ of \mathbb{K}_{ci} its quality $\Omega(C)$ is given by

$$\Omega(C) = \mu(|G_c|, |G_i|) = |G_i| \left(\frac{1}{\beta} \right)^{|G_c|} \quad (37)$$

The above definition follows directly from equation 36 and the intuition discussed above. Utilizing the properties of Ω and the definition of a concept we can show that for any context \mathbb{K}_{ci} , the 2-concepts of \mathbb{K}_{ci} are the subspaces that maximize Ω .

Theorem 5. Let \mathbb{K}_{ci} be a context and $C = (G_c, G_i)$ be a subspace of \mathbb{K}_{ci} such that $\psi^i(G_c) = G_i$ or $\psi^c(G_i) = G_c$. If (G_c, G_i) is not a concept, then there exists a subspace $\hat{C} = (\hat{G}_c, \hat{G}_i)$ such that $\Omega(\hat{C}) > \Omega(C)$ and \hat{C} is a concept.

Proof. Without loss of generality assume $\psi^i(G_c) = G_i$, but $\psi^c(G_i) \neq G_c$. Then by proposition 1 there exists a set \hat{G}_c such that $\psi^c(G_i) = \hat{G}_c$ and $\hat{G}_c \supset G_c$. Let, $\hat{C} = (\hat{G}_c, G_i)$, then by the monotonicity property of Ω we have $\Omega(\hat{C}) > \Omega(C)$, and by definition \hat{C} is a concept. \square

We now develop a quality measure for n -concepts utilizing the same intuition that was used for 2-concepts. Recall that the connectivity or area of an n -concept $C = (G_c, G_1, \dots, G_{n-1})$ is given by $|G_c| \cdot W(C)$. Therefore, just as the case was with 2-concept, as the height of an n -concept increases, its width also must decrease. Utilizing this fact, and properties 1 and 2 and we may now derive

$\Omega(C)$:

$$\Omega(C) = \mu \left(|G_c|, \sum_{i=1}^{n-1} |G_i| \right) \quad (38)$$

$$= \left(\sum_{i=1}^{n-1} |G_i| \right) * \left(\frac{1}{\beta} \right)^{|G_c|} \quad (39)$$

$$= |G_1| \left(\frac{1}{\beta} \right)^{|G_c|} + \dots + |G_{n-1}| \left(\frac{1}{\beta} \right)^{|G_c|} \quad (40)$$

$$= \Omega((G_c, G_1)) + \dots + \Omega((G_c, G_{n-1})) \quad (41)$$

Equation 41 is clearly β -balanced, and can be easily computed since it is the sum of the quality of local formal concepts.

Definition 14. Given β and a N -concept C in a star shaped information network, \mathcal{T} , then its quality $\Omega(C)$ is given by

$$\Omega(C) = \sum_{i=1}^{n-1} \Omega(G_c, G_i) \quad (42)$$

4.2 Algorithms for Enumerating n -concepts

Enumerating n -concepts in a star-shaped information network can be accomplished in a similar manner to enumerating concepts in a single context. The fact that an n -concept consists of a matching among several contexts presents more opportunities for pruning the search space. Moreover, most users are only interested in viewing the top K n -concepts, hence the search space may be further condensed by taking advantage of the properties of Ω . In the sequel we develop the `NClu` algorithm by exploiting the opportunities discussed above.

4.2.1 Preprocessing and Defining Search Space

Each context in the information network can be viewed as a data matrix, however, in practice these matrices are often sparse. Therefore each context is represented by a list of object-sets OS_i , by associating each individual object within a domain \mathbf{G}_i with its corresponding object-set from the

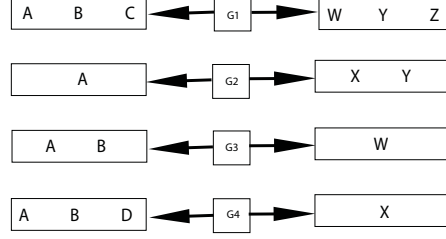


Figure 12: Objects and their associated attributesets

domain \mathbf{G}_j , within context \mathbb{K}_{ij} . Without loss of generality we assume that each object may be represented by a positive integer. Representing the context via object-sets cuts down on the need for complex data structures, and we can simply resort to set operations such as intersection and difference for the major computational operations. In other words, computing Ψ only requires computing intersections among elements of OS_i respectively. For each context \mathbb{K}_{ij} a simple linear scan is utilized to obtain OS_i . Given the large main memories available today our experiments indicate that these lists can be housed in main memory for many real-world information networks.

For any star-shaped network, \mathcal{T} , with $V = (\mathbf{G}_c, \mathbf{G}_1, \dots, \mathbf{G}_{n-1})$, an n -concept may be formed in any subspace of \mathbf{G}_c and any subspace of all \mathbf{G}_i . Therefore, at a minimum each OS_i may constitute a suitable search space for enumerating candidate concepts in each \mathbb{K}_{ci} . Moreover, for each space the number of states to be searched is potentially exponential. However, utilizing the structure of the information network and a proper state space formulation we can manage the daunting task of enumerating n -concepts. Clearly, each context \mathbb{K}_{ci} shares a common set of objects \mathbf{G}_c ; therefore, it makes sense to formulate the search over this space. In order to distinguish the object-sets of \mathbf{G}_c from the object sets of all \mathbf{G}_i , we will refer to the object-sets of \mathbf{G}_i as “attribute-sets”. The object-set search tree shown in figure 13 can be used to define the search space. This tree utilizes the idea of prefix-based equivalence classes in order to break the original search space into independent sub-problems [60]. A *prefix tree* is an ordered data structure used for sorting strings. Viewing the object-sets as strings that conform to some ordering, two object-sets are in the same prefix class if they share a common k -length prefix P . Given two objects $g_c^i, g_c^j \in \mathbf{G}_c$ we denote $g_c^i \ll g_c^j$

if g_c^j is larger than g_c^i according to the selected ordering. Given two object-sets G_c^1 and G_c^2 we denote $G_c^1 \ll G_c^2$ if $\forall g_c^1, g_c^2, g_c^1 \in G_c^1, g_c^2 \in G_c^2 \quad g_c^1 \leq g_c^2$. The *tail* of an object-set P_c , denoted as $\text{tail}(P_c) = (g_c^1, g_c^2, \dots, g_c^x)$ consists of individual objects such that $\forall g_c^i \in \text{tail}(P_c) \quad P_c \ll g_c^i$. That is each g_c^i can be viewed as an object having P_c as a prefix. Given P_c and $\text{tail}(P_c)$ we define the sub-search space of P_c , denoted by $\langle P_c \rangle$, as :

$$\langle P_c \rangle = \left\{ B^1, \dots, B^{|\text{tail}(P_c)|} \right\}$$

where

$$\begin{aligned} B^i &= (\hat{P}_c, G_1^i, \dots, G_{n-1}^i) \\ &= (\{P_c \cup g_c^i\}, \{\psi^1(P_c \cup g_c^i)\}, \dots, \{\psi^{n-1}(P_c \cup g_c^i)\}) \end{aligned}$$

$\langle P_c \rangle$ contains $|\text{tail}(P_c)|$ nodes. Each node consists of the object-set $P_c \cup g_c^i$ and $n-1$ attribute-sets from each context in the information network. The attribute-set of each node is formed by first taking the union of P_c and each $g_c^i \in \text{tail}(P_c)$. Next this object-set is translated to an attribute-set as $\psi^k(P_c \cup g_c^i)$ for $1 \leq k \leq n-1$. The nodes of a sub-search space P_c are ordered by the prefix ordering of their objects. Beginning with the root object-set, \emptyset , along with $\text{tail}(\emptyset)$ and $\langle \emptyset \rangle$ every possible node in which an n -concept may occur can be enumerated utilizing the `Enumerate` procedure displayed below.

The `Enumerate` procedure mimics a depth first search in the prefix tree, however there is no need to explicitly maintain the prefix tree in memory, as the tree is dynamically generated as the search proceeds. Each sub-search space of P , $\langle \hat{P}^i \rangle$ is generated by the second `for` loop (lines 3-9) along with each tail $\text{tail}(\hat{P}^i)$. The tails are computed by simply adding in every g^j (line 5). Each sub-search space should clearly contain $|\text{tail}(P)| - i$ nodes, and each of these nodes are generated by lines 6-8. The attribute-sets in each node are computed for each context by intersecting the current attribute-set, G_k^i with every successor G_k^j in $\langle P \rangle$ (line 8). In other words, $\psi^k(\hat{P}^i)$ is being computed for each domain k , via set intersections as proposition 2 prescribes. Finally the procedure recurses with the child object-set \hat{P}^i along with its tail and sub-search space. Notice that

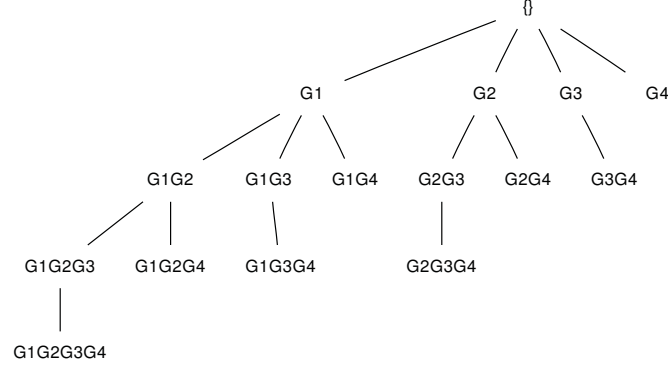
	Input: Prefix node P , $tail(P) = \{g^1, \dots, g^x\}$, $\langle P \rangle$
	Data: Star-shaped information network \mathcal{T}
1	begin
2	foreach $B^i = (\hat{P}^i, G_1^i, \dots, B_{n-1}^i) \in \langle P \rangle$ do
3	foreach $B^j = (\hat{P}^j, G_1^j, \dots, G_{n-1}^j) \in \langle P \rangle$, $j > i$ do
4	$\hat{P}^{ij} \leftarrow \hat{P}^i \cup g^j$;
5	$tail(\hat{P}^i) \leftarrow tail(\hat{P}^i) \cup g^j$;
6	$\hat{B}^{ij} \leftarrow (\hat{P}^{ij}, G_1^{ij}, \dots, G_{n-1}^{ij})$;
7	for $k \leftarrow 1$ to $n-1$ do
8	$G_k^{ij} \leftarrow G_k^i \cap G_k^j$;
9	$\langle \hat{P}^i \rangle \leftarrow \langle \hat{P}^i \rangle \cup \hat{B}^{ij}$;
10	Enumerate($\hat{P}^i, tail(\hat{P}^i), \langle \hat{P}^i \rangle$);
11	end

Procedure Enumerate

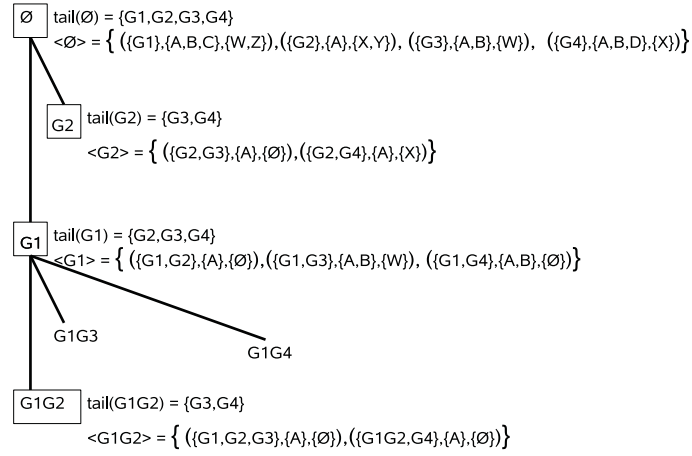
this algorithm in effect saves the results of all intermediary set intersections and passes them on to the next level of the search. This reduces the time complexity of the intersection operation as the search proceeds. Due to the fact that Enumerate is proceeding as a depth first search, saving the intermediate set intersections is not memory intensive as the memory claimed by each node along a branch can be reclaimed once the search backtracks to the parent node. Moreover, no duplicate set intersections ever occur since each attribute-set is only intersected with members of $\langle P \rangle$ that are “greater” than it. Similar procedures have been utilized to enumerate frequent itemsets in a single context [75, 42, 59]. This strategy will work for enumerating n -concepts, however several optimizations are introduced in subsequent sections that greatly improve the efficiency of the procedure.

Example 2. Utilizing our running sample contexts we have

$$\begin{aligned}
 P &= \{\emptyset\} \\
 tail(P) &= \{G1, G2, G3, G4\} \\
 \langle P \rangle &= \left\{ \left(\{G1\}, \{A, B, C\}, \{W, Y, Z\} \right), \dots \right\}
 \end{aligned}$$



(a) Object-set search tree



(b) Nodes with associated sub-search space

Figure 13: Prefix based search space for n -concepts

Running through the first iteration of `Enumerate` we obtain

$$\text{tail}(\hat{P}^1) = \{G2, G3, G4\}$$

Next $\langle \hat{P}^1 \rangle$ will be computed as

$$\langle \hat{P}^1 \rangle = \left\{ \hat{B}^{12}, \hat{B}^{13}, \hat{B}^{14} \right\}$$

Wherein \hat{B}^{12} , for example, will be computed as follows:

$$\begin{aligned}\hat{P}^{12} &= \{G1\} \\ G_1^{12} &= \{A, B, C\} \cap \{A\} = \{A\} \\ G_2^{12} &= \{W, Y, Z\} \cap \{X, Y\} = \{Y\} \\ \hat{B}^{12} &= \left(\{G1, G2\}, \{A\}, \{Y\} \right)\end{aligned}$$

4.2.2 Identifying Candidate n -concepts

<p>Input: Current nodes and search space: $B = (G_i, G_j), tail(G_i), < G_i >$ Input: Integer: $k, 1 \leq k \leq tail(G_i)$ Input: Next tail and sub-search space: $tail(\hat{G}_i), < \hat{G}_i >$ Result: Compute sub-search space $< \hat{G}_i >$ and $tail(\hat{G}_i)$, Returns closure of G_i over $tail(G_i)$</p> <pre> 1 $tail(\hat{G}_i) \leftarrow \emptyset$; 2 $\hat{G}_i \leftarrow \emptyset$; 3 $< \hat{G}_i > \leftarrow \emptyset$; 4 for $m \leftarrow k + 1$ to $tail(G_i)$ do 5 $B^m \leftarrow (G_i^m, G_j^m) \leftarrow B^m \in < G_i >$; 6 $D \leftarrow G_j^m \cap G_i$; 7 if $D \neq \emptyset$ then 8 if $G_j = G_j^m$ then 9 $G_i \leftarrow G_i \cup g_i^m$; 10 $tail(G_i) \leftarrow tail(G_i) - g_i^k$; 11 else if $G_j \subset G_j^k$ then 12 $G_i \leftarrow G_i \cup g_i^m$; 13 else if $G_j \supset G_j^k$ then 14 $tail(G_i) \leftarrow tail(G_i) - g_i^m$; 15 $tail(\hat{G}_i) \leftarrow tail(\hat{G}_i) \cup g_i^m$; 16 else 17 $tail(\hat{G}_i) \leftarrow tail(\hat{G}_i) \cup g_i^m$; 18 Recompute $< G_i >$ according to $tail(G_i)$; 19 return \hat{G}_i ; </pre>

Function Optimize

The `Enumerate` procedure successfully enumerates every possible sub-space in which n -concepts may be discovered. However, this procedure may be vastly improved in terms of computational efficiency by introducing additional pruning steps. Moreover, the procedure provides no mechanism to ensure that object-set and attribute-set pairs are closed. Finally, `Enumerate` does not take advantage of the properties of the quality measure, Ω , to help further constrict the search space. Zaki et. al [58] introduced the following theorem that stipulates how to guarantee closure of object and attribute-set pairs, while condensing the search space for concepts in a single context.

Theorem 6. *Let $B^1 = (G_i^1, G_j^1)$ and $B^2 = (G_i^2, G_j^2)$ be any two nodes in the sub-search space $\langle P \rangle$. Then the following properties hold:*

1. *If $\psi^j(G_i^1) = \psi^j(G_i^2)$ then $\psi^i(\psi^j(G_i^1)) = \psi^i(\psi^j(G_i^2))$. Thus every occurrence of G_i^1 maybe replaced with $G_i^1 \cup G_i^2$ and B^2 can be removed from $\langle P \rangle$.*
2. *If $\psi^j(G_i^1) \subset \psi^j(G_i^2)$ then $\psi^i(\psi^j(G_i^1)) \neq \psi^i(\psi^j(G_i^2))$, but $\psi^i(\psi^j(G_i^1)) = \psi^i(\psi^j((G_i^1 \cup G_i^2)))$. Thus every occurrence of G_i^1 maybe replaced with $G_i^1 \cup G_i^2$.*
3. *If $\psi^j(G_i^1) \supset \psi^j(G_i^2)$ then $\psi^i(\psi^j(G_i^1)) \neq \psi^i(\psi^j(G_i^2))$, but $\psi^i(\psi^j(G_i^1)) = \psi^i(\psi^j(G_i^1 \cup G_i^2))$. Thus B^2 may be removed from $\langle P \rangle$, but \hat{B}^2 must be added to $\langle \hat{P} \rangle$.*
4. *If $\psi^j(G_i^1) \neq \psi^j(G_i^2)$ then $\psi^i(\psi^j(G_i^1)) \neq \psi^i(\psi^j(G_i^2)) \neq \psi^i(\psi^j((G_i^1 \cup G_i^2)))$. Thus no condensation of the search space is possible and \hat{B}^2 must be added to $\langle \hat{P} \rangle$.*

Proof. See [58] □

The above theorem can be applied to replace lines 3-9 of the `Enumerate` procedure, to ensure three outcomes. First, it will compute the closure of \hat{P}^i with respect to $tail(P)$. Second, the current search space, $\langle P \rangle$ will be condensed, and finally the next sub-search space $\langle \hat{P}^i \rangle$ will be computed. One implementation of theorem 6 (when searching only for 2-concepts) is given by the `Optimize` function displayed above.

Combining `Enumerate` and `Optimize` was the driving idea behind the state-of-the-art `CHARM` algorithm for discovering closed itemsets [58]. However, incorporating theorem 6 into the search

for n -concepts is not straightforward, as the information network adds additional complications. The problem arises from the fact that the theorem considers only one context; however, in the case of vertically partitioned contexts we must consider different attribute-sets and relations, which leads to seemingly contradictory pruning steps. Recall the computations in example 2, when computing \hat{B}^{12} we have $G_1^{12} = \{A, B, C\} \cap \{A\}$ and $G_2^{12} = \{W, Y, Z\} \cap \{X, Y\}$. After performing the first intersection theorem 6 stipulates removing B^2 from $\langle \emptyset \rangle$. However, upon performing the second intersection, property 4 imposes that no condensation is possible. Clearly, these are two contradicting steps.

An intuitive solution is to split the tail $tail(P)$ and sub-search space $\langle P \rangle$ into $n - 1$ tails $tail_1(P), \dots, tail_{n-1}(P)$ and sub-search spaces $\langle P_1 \rangle, \dots, \langle P_{n-1} \rangle$. The nodes of each $\langle P_i \rangle$ only contain the attribute-sets of the i^{th} context. That is each $\langle P_i \rangle$ is defined as follows:

$$\langle P_i \rangle = \left\{ B_i^1, \dots, B_i^{|tail(P)|} \right\}$$

where

$$\begin{aligned} B_i^j &= (\hat{P}, \hat{B}_i) \\ &= \left(\{P \cup g_c^j\}, \{\Psi^i(P \cup g_c^j)\} \right) \end{aligned}$$

`Optimize` can be called for each $tail_i(P)$ and sub-search space $\langle P_i \rangle$. If a sub-search space undergoes a pruning (cases 1 and 3) then all other sub-search spaces should be updated to reflect the same pruning steps. By definition each n -concept must be a concept in all contexts, therefore if a subspace is eliminated in one context, then there is no point in exploring that subspace in any other context. Following the application of `Optimize`, $n - 1$ new sub-search spaces are created along with $n - 1$ closures of $\hat{P} : \hat{P}_1, \dots, \hat{P}_{n-1}$. Two cases now arise:

1. All $\hat{P}_i \in \langle \hat{P}_i \rangle$ are equivalent. By definition then the subspace $(\hat{P}_i, G_1, \dots, G_{n-1})$ is a candidate n -concept.
2. All $\hat{P}_i \in \langle \hat{P}_i \rangle$ are not equivalent. In this case the algorithm must decide if it is still possible to form a n -concept in the newly generated sub-search spaces $\langle \hat{P}_1 \rangle, \dots, \langle \hat{P}_{n-1} \rangle$.

The following example illustrates the two cases described above:

Example 3. Utilizing our running sample contexts we have

<i>Input</i>	<i>After Applying Optimize</i>
$P = \{\emptyset\}$	
$tail(P_1) = \{G1, G2, G3, G4\}$	$tail(P_1) = \{G1, G4\}$
$tail(P_2) = \{G1, G2, G3, G4\}$	$tail(P_2) = \{G1, G4\}$
$\langle P_1 \rangle = \{B_1^1, B_1^2, B_1^3, B_1^4\}$	$\langle P_1 \rangle = \{B_1^1, B_1^4\}$
$\langle P_2 \rangle = \{B_2^1, B_2^2, B_2^3, B_2^4\}$	$\langle P_2 \rangle = \{B_2^1, B_2^4\}$
$\hat{B}_1^1 = (\hat{P}_1^1 = \{G1\}, B_1^1 = \{A, B, C\})$	$\hat{B}_1^1 = (\hat{P}_1^1 = \{G1\}, B_1^1 = \{A, B, C\})$
$B_2^1 = (\hat{P}_2^1 = \{G1\}, \{W, Y, Z\})$	$B_2^1 = (\hat{P}_2^1 = \{G1\}, \{W, Y, Z\})$

In this case $\hat{P}_1 = \hat{P}_2 = \{G1\}$ thus $(\{G1\}, \{A, B, C\}, \{W, Y, Z\})$ is a candidate n -concept. On the other hand consider the following:

<i>Input</i>	<i>After Applying Optimize</i>
$P = \{G1\}$	
$tail(P_1) = \{G2, G3\}$	$tail(P_1) = \{G2\}$
$tail(P_2) = \{G2, G3\}$	$tail(P_2) = \{G2, G3\}$
$\langle P_1 \rangle = \{B_1^1, B_1^2\}$	$\langle P_1 \rangle = \{B_1^1\}$
$\langle P_2 \rangle = \{B_2^1, B_2^2\}$	$\langle P_2 \rangle = \{B_2^1, B_2^2\}$
$B_1^1 = (\hat{P}_1^1 = \{G1, G2\}, \{A\})$	$\hat{B}_1^1 = (\hat{P}_1^1 = \{G1, G2, G3\}, \{A\})$
$B_2^1 = (\hat{P}_2^1 = \{G1, G2\}, \{Y\})$	$\hat{B}_2^1 = (\hat{P}_2^1 = \{G1, G2\}, \{Y\})$

Since $\hat{P}_1 = \{G1, G2, G3\} \neq \{G1, G2\}$, then a candidate n -concept has not been discovered.

After recognizing that a candidate n -concept has not been discovered the search space may be pruned if it is determined that a n -concept cannot be found in subsequent search spaces.

Proposition 4. Given nodes $B_1 = (\hat{P}_1, G_1), \dots, B_{n-1} = (\hat{P}_{n-1}, G_{n-1})$ such that $\hat{P}_i \neq \hat{P}_j, 1 \leq i, j \leq n-1, i \neq j$, let $\mathbf{B} = \bigcup_{i=1}^{n-1} \hat{P}_i$. If an n -concept $C = (G_c, G_1, \dots, G_{n-1})$ exists in the sub-search spaces $\langle \hat{P}_1 \rangle, \dots, \langle \hat{P}_{n-1} \rangle$ then $\mathbf{B} \setminus \hat{P}_i \subseteq \text{tail}(\hat{P}_i)$.

Proof. Assume $C = (G_c, G_1, \dots, G_{n-1})$ exists in the sub-search spaces $\langle \hat{P}_1 \rangle, \dots, \langle \hat{P}_{n-1} \rangle$. Then clearly $G_c \supseteq \mathbf{B}$. Thus in order for the 2-concept (G_c, G_i) to exist in each sub-search space $\langle \hat{P}_i \rangle$, $\text{tail}(\hat{P}_i) \cup \hat{P}_i \supseteq \mathbf{B}$. Equivalently, $\mathbf{B} \setminus \hat{P}_i \subseteq \text{tail}(\hat{P}_i)$. \square

Proposition 4 allows for an easy check to determine if it is possible to locate a candidate n -concept in the sub-search spaces. If this is not possible the entire search branch may be pruned. On the other hand, for any n -concept $(G_c, G_1, \dots, G_{n-1})$ that may exist in the sub-search spaces, then $G_c \supseteq \mathbf{B}$; thus a candidate n -concept may be formed as follows :

Conceptually, the above function attempts to construct a candidate n -concept by removing every

<p>Input: Nodes: $B_1 = (\hat{P}_1, G_1), \dots, B_{n-1} = (\hat{P}_{n-1}, G_{n-1})$ Data: $\text{tail}(\hat{P}_1), \dots, \text{tail}(\hat{P}_{n-1}), \langle \hat{P}_1 \rangle, \dots, \langle \hat{P}_{n-1} \rangle$ Result: Determines if candidate can be formed in the search space and returns it</p> <pre> 1 $\mathbf{B} \leftarrow \bigcup_{i=1}^{n-1} \hat{P}_i;$ 2 for $i \leftarrow 1$ to $n-1$ do 3 foreach $g \in \mathbf{B} - \hat{P}_i$ do 4 $(G_c, \hat{G}_i) \leftarrow$ the node in $\langle \hat{P}_i \rangle$ that corresponds to g ; 5 $G_i \leftarrow G_i \cap \hat{G}_i;$ 6 if $G_i \neq \emptyset$ then 7 $\hat{P}_i \leftarrow \hat{P}_i \cup g$; 8 $\text{tail}(\hat{P}_i) \leftarrow \text{tail}(\hat{P}_i) - g;$ 9 $\langle \hat{P}_i \rangle \leftarrow \langle \hat{P}_i \rangle - (G_c, \hat{G}_i)$; 10 if all \hat{P}_i equivalent then 11 Re-compute all $\langle \hat{P}_i \rangle$; 12 return $(\hat{P}_i, G_1, \dots, G_{n-1})$; 13 else 14 return \emptyset ; </pre>
--

Function IdCandidate

object g of \mathbf{B} from every sub-search space and computing $\psi^i(\hat{P}_i \cup g)$. If this computation results

in an empty set, then by definition a n -concept cannot be formed. In the case that an empty set is never encountered, then all \hat{P}_i will be equivalent and a candidate has been identified. The search spaces $\langle \hat{P}_i \rangle$ must be recomputed since all \hat{P}_i include all the objects of \mathbf{P} .

4.2.3 Forming n -concepts

Once a candidate n -concept $(\hat{P}_i, G_1, \dots, G_{n-1})$ is identified an n -concept may be formed. However there is no guarantee that \hat{P}_i is closed in each context. Theorem 6 guarantees that \hat{P}_i is closed with respect to $tail(P)$ (through properties 1 and 2), however there may exist a $g \notin tail(P)$ that was not included in \hat{P}_i . Checking for closure in each context consists of computing all $g \notin tail(P)$ and checking if $\psi^i(\hat{P}_i \cup g) = G_i$. If this is true for any g and any i , $1 \leq i \leq n-1$, then by definition (\hat{P}_i, G_i) is not closed, thus $(\hat{P}_i, G_1, \dots, G_{n-1})$ is not an n -concept. In this case, the entire sub-search space may be pruned because the anti-monotonic property of set intersections guarantees that all subsequent candidate n -concepts will not be closed. Conversely, if a candidate is identified as an n -cluster then the search must delve deeper into the sub-search space. At this point the $n-1$ sub-search spaces and tails may be joined back together by performing the intersection of each $tail(\hat{P}_i)$ and computing the resulting combined sub-search space $\langle \hat{P} \rangle$.

Example 4. Recall the first case in example 3 where a candidate n -concept was identified.

<i>Candidate Identified</i>	<i>Join new sub-search spaces</i>
$P = \{\emptyset\}$	$\hat{P} = \hat{P}_1 = \hat{P}_2 = \{G1\}$
$tail(P_1) = \{G1, G4\}$	$tail(\hat{P}_1) \cap tail(\hat{P}_2) = \{G2, G3\}$
$tail(P_2) = \{G1, G4\}$	$\langle \hat{P} \rangle = \{\hat{B}^1, \hat{B}^2\}$
$\langle P_1 \rangle = \{B_1^1, B_1^4\}$	$\hat{B}_1 = (\{G1, G2\}, \{A\}, \{Y\})$
$\langle P_2 \rangle = \{B_2^1, B_2^4\}$	$\hat{B}_2 = (\{G1, G3\}, \{A, B\}, \{W\})$
$\hat{B}_1^1 = (\{G1\}, \{A, B, C\})$	
$\hat{B}_2^1 = B_2^1 = (\{G1\}, \{W, Y, Z\})$	
$tail(\hat{P}_1) = \{G2, G3, G4\}$	
$tail(\hat{P}_2) = \{G2, G3\}$	

The intersection of the tails is computed to avoid searching branches that will never lead to n -concepts. In this example, the search node associated with $G4$ was pruned.

4.2.4 Asserting High Quality

One key question remains: given an n -concept $(G_c, G_1, \dots, G_{n-1})$, is it possible to form a higher quality n -concept in the sub-search space $\langle G_c \rangle$? If this is not the case then the sub-search space $\langle G_c \rangle$ should be pruned. The properties of Ω allows us to perform such pruning, despite the fact that Ω is neither monotonic nor anti-monotonic in the search space. Consider a concept $C = (G_c, G_1, \dots, G_{n-1})$ formed at prefix node G_c , then for any concept $C^1 = (G_c^1, G_1^1, \dots, G_{n-1}^1)$ formed at a lower level in the search space, the following must be true:

$$\Omega(C^1) \geq \Omega(C) \leftrightarrow \frac{W(C^1)}{W(C)} \geq \beta^{|G_c^1| - |G_c|} \quad (43)$$

Equation 43 stipulates that C^1 is of higher quality than C only if the trade-off criterion imposed by β is met. On the other hand, if $\Omega(C) > \Omega(C^1)$ we may not automatically prune the search branch, because a concept C^2 may exist at a lower level that does satisfy the trade-off criterion. In other words, that fact that

$$\frac{W(C^1)}{W(C)} \geq \beta^{|G_c^1| - |G_c|}$$

by no means implies that

$$\frac{W(C^2)}{W(C)} \geq \beta^{|G_c^2| - |G_c|}$$

for $W(C^2) < W(C^1)$ and $|G_c^2| > |G_c^1|$. However, if indeed $\Omega(C) > \Omega(C^1)$, then we may compute the least number of objects, x , that must be added to C^1 so that it meets the trade-off criterion. If the sub-search space contains fewer than x objects then clearly the entire branch of the search tree may be pruned.

Proposition 5. *Given n -concepts $C = (G_c, G_1, \dots, G_{n-1})$ and $C^1 = (G_c^1, G_1^1, \dots, G_{n-1}^1)$ formed at*

prefix nodes G_c and G_c^1 , if

$$\Omega(C^1) < \Omega(C) \quad \wedge \quad |tail(G_c^1)| < \frac{\log W(C^1) - \log W(C)}{\log \beta} - |G_c^1|$$

then there does not exist a concept C^2 that may be formed with $tail(G_c^1)$ s.t. $\Omega(C^2) > \Omega(C)$; therefore the branch may be pruned.

Proof. Given that $\Omega(C^1) < \Omega(C)$, then let x be the least number of objects that must be added to any concept C^2 formed with $tail(G_c^1)$ s.t. $\Omega(C^2) > \Omega(C)$. Utilizing equation 43 we have

$$\begin{aligned} \beta^{|G_c^1|+x} &\geq \frac{W(C^1)}{W(C)} \\ (x + |G_c^1|) \log \beta &\geq \log W(C^1) - \log W(C) \\ x &\geq \frac{\log W(C^1) - \log W(C)}{\log \beta} - |G_c^1| \end{aligned}$$

Clearly if $|tail(G_c^1)| < x$ then no higher quality concept may be formed. □

Proposition 5 yields a simple pruning rule that can be formulated by simply checking the length of $tail(G_c)$. On the other hand, this pruning rule makes a very optimistic assumption. It assumes that the width of clusters in the sub-search space of G_c will be equal to $W(C)$. Plainly, this is not true as the anti-monotonic property of set intersection guarantees that width of every n -concept in the sub-search space will decrease. Hence ; making the above pruning rule may fail to prune several branches that otherwise should have been pruned. Paradoxically, computing the width of potential n -concept in the sub-search space of is equivalent to exploring the space. One possibility, is to only enumerate the top k , n -concepts and utilize the β -pruning by comparing every potential n -concept to the current k^{th} ranked n -concept.

4.2.5 NClu Algorithm

The four major steps of our algorithmic approach have been fully described in the preceding four sections. Aggregating all three the pseudo code of NClu algorithm is presented as algorithm 4.

```

Input: Prefix node  $P, \text{tail}(P) = \{g^1, \dots, g^x\}, \langle P \rangle, K$ 
Data:  $\mathcal{T}$ 
1 begin
2   Split  $\text{tail}(P)$  into  $\text{tail}(P_1), \dots, \text{tail}(P_{n-1})$ ;
3   Split  $\langle P \rangle$  into  $\langle P_1 \rangle, \dots, \langle P_{n-1} \rangle$ ;
4   for  $k \leftarrow 1$  to  $|\text{tail}(P)|$  do
5     for  $i \leftarrow 1$  to  $n-1$  do
6        $B_i^k \leftarrow (\hat{P}_i^k, G_i^k) \leftarrow \mathbf{B}_k^i \in \langle P_i \rangle$ ;
7        $\hat{B}_i^k \leftarrow \text{Optimize}(\hat{B}_i^k, \text{tail}(P_i), \langle P_i \rangle, k, \text{tail}(\hat{P}_i^k), \langle \hat{P}_i^k \rangle)$ ;
8       Prune all  $\text{tail}(P_j), \langle P_j \rangle, j \neq i$  based on  $\text{tail}(P_i), \langle P_i \rangle$ ;
9      $\text{foundCandidate} \leftarrow \text{false}$ ;
10    if all  $\hat{P}_i^k$  equivalent then
11       $\text{foundCandidate} \leftarrow \text{true}$ ;
12    else
13       $C \leftarrow \text{IdCandidate}(\hat{B}_1^k, \dots, \hat{B}_{n-1}^k)$ ;
14      if  $C \neq \emptyset$  then
15         $\text{foundCandidate} \leftarrow \text{true}$ ;
16    if  $\text{foundCandidate} = \text{true} \wedge$  all  $\hat{P}_i^k$  are closed then
17      Found  $n$ -concept,  $C \leftarrow (\hat{P}_i, \hat{B}_i^k, \dots, \hat{B}_{n-1}^k)$ ;
18       $\text{tail}(\hat{P}^k) \leftarrow \bigcap_{l=1}^{n-1} \text{tail}(\hat{P}_l^k)$ ;
19      Compute  $\langle \hat{P}^k \rangle$  from  $\text{tail}(\hat{P}^k)$ ;
20       $\text{prune} \leftarrow \text{BetaPrune}(C, \langle P^k \rangle, K)$ ;
21      if NOT prune then
22         $\text{NClu}(\hat{P}^k, \text{tail}(\hat{P}^k), \langle \hat{P}^k \rangle)$ ;
23 end

```

Algorithm 4: NClu

Candidate clusters are identified in lines 4-8 utilizing the `Optimize` function. If the algorithm fails to identify a candidate then the `IdCandidate` function is called to determine the availability of a candidate in the sub-search space (line 13). The correctness of these functions is guaranteed by theorem 6 and proposition 4. If a candidate concept is found and all the object-sets are closed in the individual contexts then an n -concept is identified (line 17). The next sub-search space is computed by intersecting all tails, and the algorithm checks if it is possible to form a higher quality n -concept in the new sub-search space (line 20). If this is possible then the search recurses to the

next level. The `BetaPrune` function is a direct implementation of proposition 5 (pseudo code not shown).

The correctness of `NClu` is guaranteed by the correctness of the `Optimize`, `IdCandidate` procedures and proposition 5.

Theorem 7. *Given a star-shaped information network \mathcal{T} and integer K , the `NClu` algorithm correctly identifies the top K n -concepts with respect to the Ω quality measure.*

The number of n -concepts in \mathcal{T} is bounded by the minimum number of concepts in a single context $\mathbb{K}_{ci} \in E$; we will denote this by $X = \min\{|\mathcal{B}(\mathbb{K}_{ci})| \in E\}$. The basic operation of the algorithm is set intersection, therefore we analyze the computational complexity of `NClu` in terms of X and set intersections. The initial scanning of object-sets and attribute-sets is performed in $O(|G_i| * |V|)$ and $O(|G|)$ operations. Computing \hat{P}_i^k and its associated sub-search space consists of at most $|G_i|$ set intersections and this is performed $n - 1$ times. The checks in `Optimize` can be performed by maintaining some additional information after each set intersection, therefore they come for free. Checking all \hat{P}_i^k are equal and determining common objects requires at most $|G_i|$ equality checks. On the other hand, the `IdCandidate` will perform at most $|G_i|$ set-intersections. Clearly `NClu` will be called at most X times. Hence each call of `NClu` requires $O(|G_i| * |V| + |G_i|)$ operations and the total cost is $O(X) * O(|G_i| * |V| + |G_i|) = O(X * |G_i|)$.

Theorem 8. *The worst case complexity of `NClu` is $O(X * |G_i|)$.*

In the worst case `NClu` is linear in X and encompasses the same computational cost as enumerating 2-concepts (bi-clusters, closed itemsets) in a single context. This result is significant in terms of computation cost, as it was illustrated earlier that a naive approach to mining n -concepts would grow quadratically in the number of contexts. In the subsequent section, several performance tests illustrate the computational advantage that `NClu` offers, and the utility of n -concepts in real-world datasets.

4.3 Experimental Results

4.3.1 Performance Tests

A naive method of enumerating n -concepts in a star-shaped information network is to form the sum of all contexts, search for closed itemsets, and post process the result. Our claim is that `NClu` is a far more efficient solution. In order to verify this claim, several performance tests were conducted utilizing real world and benchmark data sets. The characteristics of the data sets are presented in figure 14(a). The first three datasets constitute real world biological datasets that all contain common genes (objects) and gene ontology terms, phenotypes, and microRna information respectively. The Chess, Connect, Kosarak, pumsb*, and Mushrooms are all benchmark datasets in the data mining community; these were randomly partitioned to simulate vertically partitioned data. All experiments were performed on a Pentium D 2.8 GHz, 3.68 GB of RAM, operating SUSE Linux 10, and all implementations were programmed in C++. We compared the performance of `NClu` against the state of the art closed mining algorithms `LCM`, which is currently the fastest known closed set mining algorithm and implementation [75]. `LCM` requires a *min_sup* parameter which instructs the algorithm to only mine closed itemsets that have support of at least *min_sup* percent of all the attributes. In our implementation of `NClu` we also allowed for this parameter to be optionally set. The procedure we followed for comparison was

1. `NClu` was executed with out β -pruning to ensure a fair comparison.
2. The `if` statement on line 2 of the `Optimize` algorithm was modified to check if the *min_sup* criterion was met.
3. *min_sup* was varied for every dataset
4. `LCM` was executed on the summed context of each dataset, and the resulting itemsets were stored to disk. Following this, the itemsets were post-processed to determine the n -concepts. The reported CPU time is the total CPU time of running `LCM` and post processing.
5. Each performance test was executed 20 times.

The results of the performance tests are displayed in figure 14. `NClu` outperforms `LCM` in every experiment, and in most cases by orders of magnitude, despite the fact that β -pruning was disabled. The driving force behind the gigantic difference in performance is the fact that `NClu` simultaneously prunes the search space and enumerates n -concepts, while `LCM` enumerates orders of magnitude more closed itemsets, most of which do not end up forming n -concepts.

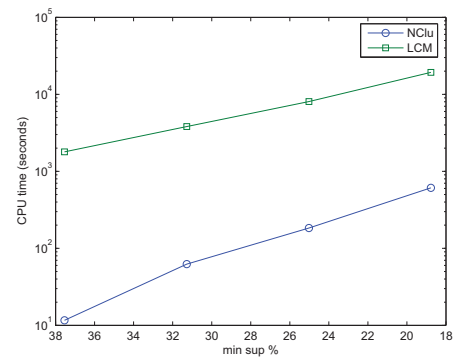
4.3.2 Effect of k and β

Next, the effect of the parameters K and β have upon the computational cost of `NClu` were investigated. The same datasets from the preceding section were utilized. These results are displayed in figure 15. For each data point in the CPU time plots, the experiment was repeated 20 times, with the average being displayed. Since set intersection is the basic operation of the algorithm the number of set intersections was also investigated. The β experiments were executed with K being held constant at 100. Moreover *min_sup* was set to 1000 for Kosarak12 and Chess12. For all other datasets *min_sup* was set to 1. The plots illustrate that in most datasets as β approaches extreme values (0.01,0.99) computational savings are realized. In the case of $\beta = 0.01$, then the quality measure prefers n -concepts with more height. As these clusters are discovered early on in the search, more pruning can take place. As β is varied towards middle values, this trend no longer holds. However once β is set to the other extreme, 0.99, then n -concepts with large widths are preferred. In this case the number of search levels is considerably reduced, and thus the computational savings. This trend holds for all the datasets except GPM (GO, Pheno, MicroRna). One possible explanation for this is that the GPM information network is considerably sparser than all other information networks. In turn, even though β is set at 0.99, the search must still traverse many levels of the search, as is common with sparse data.

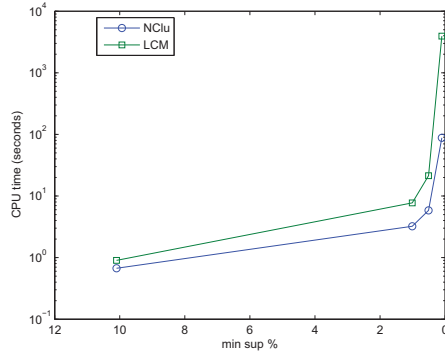
The value of K proved to be a bigger factor in terms of computation cost. This is expected as the bigger K is the less pruning at early stages of the algorithm can occur. Generally, setting k between 0 and 500 scaled linearly. However as K approaches 1,000 and beyond the computation cost increases significantly. From a practical point of view this is tolerable, since investigating the

Data Set Name	#Objects (rows)	#Attributes (columns)
GOTerms	1,910	3,385
PhenoTypes	1,910	3,965
MircroRna	1,910	318
Chess1	75	1,621
Chess2	76	1,599
Connect1	130	33,712
Connect2	130	33,856
Kosarak1	41,270	495,462
Kosarak2	41,270	494,541
Mushrooms1	8,124	119
Mushrooms2	8,124	119

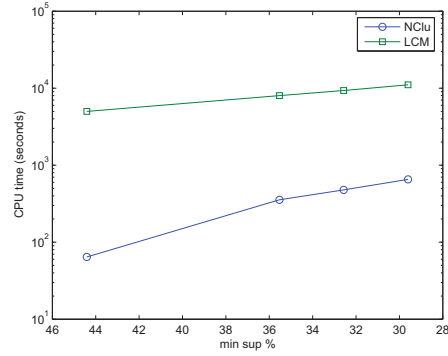
(a) Dataset characteristics



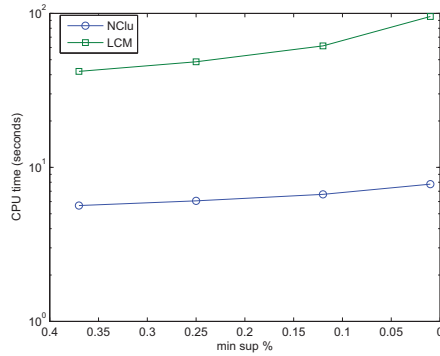
(b) Chess1 and Chess2



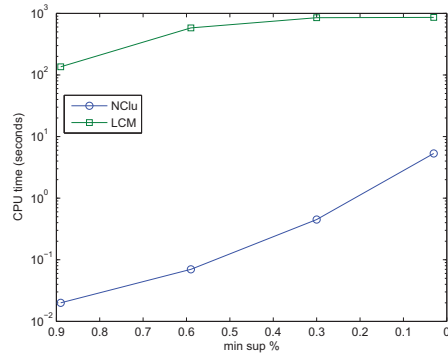
(c) Kosarak1 and Kosarak2



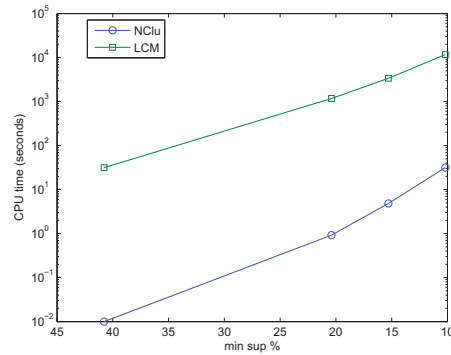
(d) Connect1 and Connect2



(e) Mushrooms1 and Mushrooms2



(f) GOTerms, PhenoTypes, and MicroRna



(g) Pumbs*1 and Pumbs*2

Figure 14: Performance Study Results on NClu

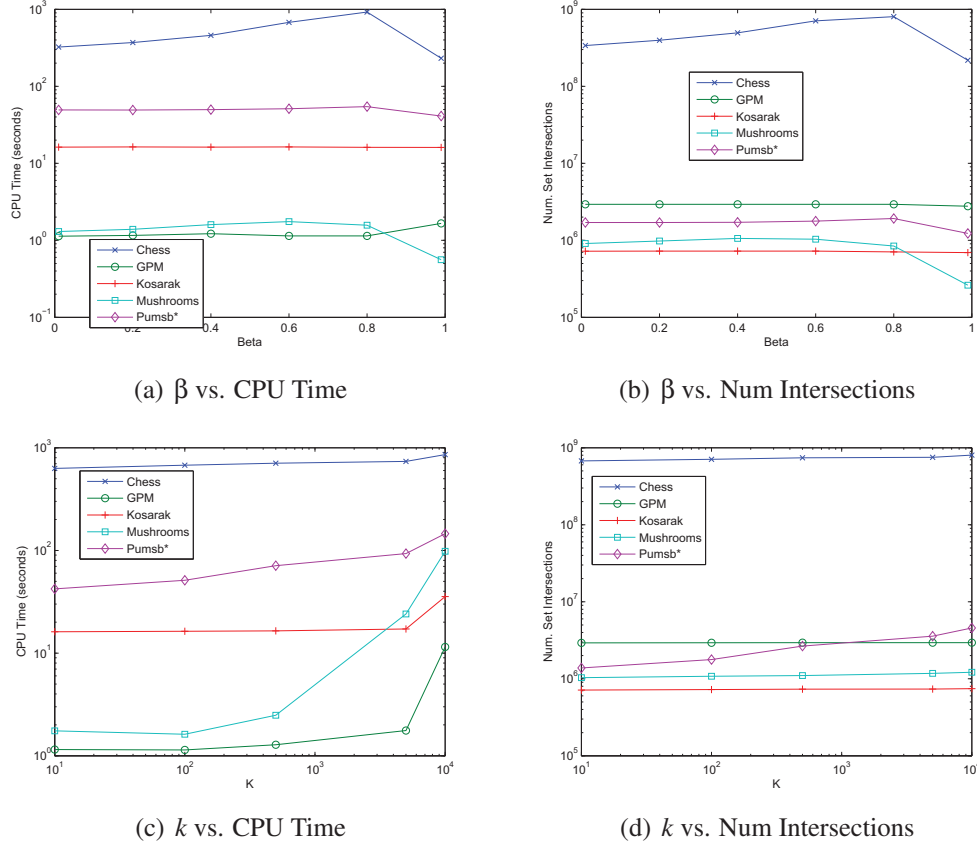


Figure 15: Parameter Study on NClu

results of 10,000 clusters is a data mining task in of itself. In the subsequent experimental sections we investigate the effect of K on the quality of the clustering.

In conclusion, through performance testing, we have shown that NClu outperforms closed set mining approach by orders of magnitude. Furthermore, the selection of K and β impact the computational cost of NClu, however not in a significant manner.

4.3.3 Clustering Results

Evaluating and validating the authenticity of clusters is a difficult task. This is further complicated by the fact that n -concepts are arbitrarily positioned, overlapping, and reveal associations across partitioned data. Therefore, a valid clustering may truly lay in the eye of the beholder or a domain expert. However, the two fundamental properties that n -concepts should retain are 1) preserving the

global clustering structure, and 2) reveal valid relations among all sets in the information network. Therefore we employed two basic methods to ensure that these properties were upheld.

Global Clustering Structure Labeled datasets were obtained and randomly partitioned to simulate vertically partitioned data. Due to the availability of the class label an “ideal clustering” is already defined. In such information networks, our aim is to show that the global clustering structure of the larger context is retained by n -concepts. This property was a fundamental consideration when formulating the information network clustering model. Much work has been devoted to formulating an extrinsic clustering quality measure [26][31]. Most quality measures are not adequate for handling overlapping clusters. However, recently the authors of [31] presented an extended version of the BCubed clustering measure for overlapping clusters, and illustrated that this measure retains the desirable properties of cluster homogeneity, cluster completeness, rag bag, and cluster size vs quantity. Formally, let \mathcal{T} be a star-shaped information network. Then for any object e of G_c or $\{G_i, \dots, G_{n-1}\}$ or both there exists a set of ideal categories (class labels), denoted by $L(e)$ to which e belongs. Also let $C(e)$ denote the set n -concepts that e belongs to. Given this we may define the multiplicity precision and multiplicity recall between any two item e and e' as follows:

$$MultPrec(e, e') = \frac{\min(|C(e) \cap C(e')|, |L(e) \cap L(e')|)}{|C(e) \cap C(e')|} \quad (44)$$

$$MultRcl(e, e') = \frac{\min(|C(e) \cap C(e')|, |L(e) \cap L(e')|)}{|L(e) \cap L(e')|} \quad (45)$$

Note that equation 44 is only defined when e and e' share a cluster and equation 45 is only defined when e and e' share a category. Intuitively, $MultPrec$ grows if there is a matching category for each clusters where two items co-occur; $MultRcl$ grows when we add a shared cluster for each category shared by two items. Thus if we have less shared clusters than needed, we loose recall; if we have less categories than clusters we lose precision. From these measure the extended BCubed

Relation Graph Name	Domain	Num Classes	Contexts
<i>All_PC</i>	Newsgroup Documents	446,overlapping	$G_c = \text{words } G_1=\text{documents}, G_c = 21,463, G_1 = 1621$ $G_c = \text{words } G_2=\text{documents}, G_c = 21,463, G_2 = 1691$ $G_c = \text{words } M_3=\text{documents}, G_c = 21,463, M_3 = 1689$
<i>Mideast_Christian</i>	Newsgroup Documents	135,overlapping	$G_c = \text{words } G_1=\text{documents}, G_c = 21,889, G_1 = 1019$ $G_c = \text{words } G_2=\text{documents}, G_c = 21,889, G_2 = 982$
<i>Politics_Religion</i>	Newsgroup Documents	143,overlapping	$G_c = \text{words } G_1=\text{documents}, G_c = 21,463, G_1 = 970$ $G_c = \text{words } G_2=\text{documents}, G_c = 21,463, G_2 = 1028$
<i>All_PC_Subjects</i>	Newsgroup Documents	446,overlapping	$G_c = \text{words } G_1=\text{subject lines}, G_c = 2,805, G_1 = 1722$ $G_c = \text{words } G_2=\text{subject lines}, G_c = 2,805, G_2 = 1,659$ $G_c = \text{words } G_3=\text{subject lines}, G_c = 2,805, G_3 = 1,620$
<i>Mideast_Christian_Subjects</i>	Newsgroup Documents	135,overlapping	$G_c = \text{words } G_1=\text{subject lines}, G_c = 890, G_1 = 1,011$ $G_c = \text{words } G_2=\text{subject lines}, G_c = 890, G_2 = 990$
<i>Politics_Religion_Subjects</i>	Newsgroup Documents	143,overlapping	$G_c = \text{words } G_1=\text{subject lines}, G_c = 1,025, G_1 = 1036$ $G_c = \text{words } G_2=\text{documents}, G_c = 1,025, G_2 = 962$
<i>Congress</i>	Political	2,non-overlapping	$G_c = \text{voting records } G_1=\text{congress people}, G_c = 50, G_1 = 222$ $G_c = \text{voting records } G_2=\text{congress people}, G_c = 50, G_2 = 214$
<i>Mushrooms</i>	Botany	2, non-overlapping	$G_c = \text{mushroom attributes } G_1=\text{mushrooms}, G_c = 119, G_1 = 4,044$ $G_c = \text{mushroom attribute } G_2=\text{mushrooms}, G_c = 119, G_2 = 4,083$

Figure 16: Information networks utilized for clustering experiments

measures are derived as:

$$BCubed \text{ Precision} = Avg_e [Avg_{e', C(e) \cap C(e') \neq \emptyset} [MultPrec(e, e')]] \quad (46)$$

$$BCubed \text{ Recall} = Avg_e [Avg_{e', L(e) \cap L(e') \neq \emptyset} [MultRcl(e, e')]] \quad (47)$$

It is important to note, that when clusters do not overlap the *BCubed* evaluation metrics described above behave exactly as the original *BCubed* metrics for non-overlapping clusters. Combining *BCubed Precision* and *BCubed Recall* can be achieved by using the *F* measure, which is the harmonic mean of the two values. The $F_{0.5}$ and F_2 measures can also be used to weigh precision twice as much as recall and recall twice as much as precision, respectively.

Datasets Used All datasets used in the experiments were real-world datasets, derived from varied domains. The table in figure 16 summarizes each dataset and its characteristics.

The first 3 information networks were obtained from the large 20 NewsGroups dataset [7]. This dataset contains usenet documents from 20 different newsgroups. The newsgroup a document belongs to can be considered its class label. Many of the documents in these information networks belong to several to different newsgroups, and thus have several class labels. In such applications, overlapping clusters are essential to capture the ideal grouping of documents. The contexts in these

information networks were constructed by first removing all stop words from the documents, then simply placing a cross between a document and the word if the word appears in the document. The next 3 information networks were obtained from the same documents however only the words that appear in the subject lines were utilized to form the contexts. We expect the clustering of these information networks to be a difficult task given the number of class labels and the noisy nature of the text datasets.

Congress and *Mushrooms* were also obtained from the UCI repository. The *Congress* dataset contains the voting records of 435 congress people, while *Mushrooms* lists 8,124 mushrooms in terms of 22 categorical attributes. Congress people are labeled as democrats or republicans, and mushrooms as poisonous or edible. Both datasets were randomly partitioned to simulate vertically partitioned data. In these information networks, the class labels are non-overlapping and thus test the ability of NClu to detect these forms of global clusters.

Results The BCubed precision, BCubed recall, F_1 , $F_{0.5}$, and F_2 scores were obtained with NClu was compared to the results of the MDC algorithm [14]. MDC , like most current multi-way clustering algorithms does not reveal associations across the sets of the relations, but aims to improve the global clustering by incorporating the additional information present in the information network; thus this algorithm is a good basis for comparison. MDC has the option of executing as a hierarchical or flat clustering, so results were compared to both modes of operation. Moreover MDC is a probabilistic algorithm, therefore the reported results are the average over 10 runs. Each algorithm was executed with K varied at 10, 100, 1000. In figure 17 we report the scores for each information network for each value of k . As the results indicate this was clearly a difficult clustering task. However, NClu outperforms MDC_{Flat} and MDC_{Agg} in most cases, with the exception of a few instances. The driving force behind this superior performance is the soft clustering ability of NClu . Hard clustering algorithms cannot achieve maximum recall since items that share multiple categories will never share multiple clusters. This was especially true in the full information networks (not subject lines). The precision suffered in the full information networks due to the amount of noise present in the documents. Interestingly, precision improved dramatically in the subject line information

Relation Graph	k	Algorithm	BCube Prec	BCube Rcl	F_1 Score	$F_{0.5}$ Score	F_2 Score
<i>All_PC</i>	10	NClu	0.15851	0.21085	0.18097	0.16679	0.19779
		MDCFlat	0.15845	0.06816	0.09531	0.12526	0.07692
		MDCAgg	0.58931	0.00351	0.00699	0.01716	0.00439
	100	NClu	0.14989	0.41961	0.22088	0.17200	0.30856
		MDCFlat	0.20541	0.00929	0.01777	0.03932	0.01148
		MDCAgg	0.59136	0.00348	0.00692	0.01700	0.00434
	1000	NClu	0.13502	0.49562	0.21223	0.15801	0.32306
		MDCFlat	0.56753	0.00278	0.00553	0.01363	0.00347
		MDCAgg	0.59945	0.00336	0.00668	0.01643	0.00419
<i>Mideast_Christian</i>	10	NClu	0.26931	0.49724	0.34939	0.29649	0.42525
		MDCFlat	0.31305	.09139	0.14148	0.21080	0.10647
		MDCAgg	0.80627	0.00498	0.00991	0.02431	0.00622
	100	NClu	0.22495	0.63803	0.33262	0.25841	0.46664
		MDCFlat	0.43214	0.01230	0.02392	0.05521	0.01527
		MDCAgg	0.79564	0.00476	0.00947	0.02327	0.00595
	1000	NClu	0.18457	0.70852	0.29285	0.21660	0.45193
		MDCFlat	0.85776	0.00225	0.00450	0.01116	0.00282
		MDCAgg	0.79564	0.00476	0.00947	0.02327	0.00595
<i>Politics_Religion</i>	10	NClu	0.33269	0.53746	0.41098	0.36013	0.47855
		MDCFlat	0.38446	0.09929	0.15782	0.24419	0.11658
		MDCAgg	0.82345	0.00444	0.00884	0.02175	0.00555
	100	NClu	0.27166	0.66933	0.38646	0.30829	0.51774
		MDCFlat	0.47211	0.01203	0.02346	0.05457	0.01494
		MDCAgg	0.80155	0.00431	0.00857	0.02108	0.00538
	1000	NClu	0.22607	0.73996	0.34633	0.26253	0.50869
		MDCFlat	0.51715	0.05382	0.09750	0.19002	0.06557
		MDCAgg	0.81465	0.00421	0.00837	0.02060	0.00525
<i>All_PC_Subject</i>	10	NClu	0.34575	0.01612	0.03081	0.06794	0.01992
		MDCFlat	0.15784	0.06414	0.09121	0.12215	0.07278
		MDCAgg	0.63754	0.00420	0.00834	0.02046	0.00524
	100	NClu	0.42987	0.03495	0.06465	0.13187	0.04282
		MDCFlat	0.22393	0.00981	0.01880	0.04174	0.01213
		MDCAgg	0.63500	0.00410	0.00816	0.02000	0.00512
	1000	NClu	0.44439	0.03785	0.06977	0.14117	0.04633
		MDCFlat	0.63174	0.00391	0.00777	0.01908	0.00488
		MDCAgg	0.64021	0.00421	0.00836	0.02049	0.00525
<i>Mideast_Christian_Subject</i>	10	NClu	0.80105	0.05460	0.10223	0.21451	0.06710
		MDCFlat	0.34682	0.09736	0.15205	0.22932	0.11372
		MDCAgg	0.90069	0.00611	0.01213	0.02972	0.00762
	100	NClu	0.75506	0.06638	0.12203	0.24555	0.08119
		MDCFlat	0.59107	0.02282	0.04394	0.09882	0.02825
		MDCAgg	0.90770	0.00601	0.01195	0.02929	0.00750
	1000	NClu	0.76018	0.06669	0.12262	0.24683	0.08157
		MDCFlat	0.96229	0.00305	0.00608	0.01505	0.00381
		MDCAgg	0.89543	0.00593	0.01178	0.02887	0.00740
<i>Politics_Religion_Subject</i>	10	NClu	0.77142	0.01968	0.03839	0.08930	0.02445
		MDCFlat	0.39131	0.09527	0.15324	0.24134	0.11226
		MDCAgg	0.90092	0.00565	0.01122	0.02754	0.00705
	100	NClu	0.75647	0.03160	0.06067	0.13539	0.03910
		MDCFlat	0.56901	0.01442	0.02812	0.06546	0.01791
		MDCAgg	0.89576	0.00498	0.00990	0.02434	0.00621
	1000	NClu	0.76664	0.03213	0.06168	0.13759	0.03975
		MDCFlat	0.95942	0.00282	0.00563	0.01395	0.00353
		MDCAgg	0.89408	0.00490	0.00974	0.02397	0.00611
<i>Congress</i>	10	NClu	0.15910	0.98262	0.27385	0.19113	0.48280
		MDCFlat	0.75075	0.16963	0.27673	0.44551	0.20070
		MDCAgg	0.92662	0.02096	0.04100	0.09612	0.02606
	100	NClu	0.03947	0.99376	0.07593	0.04886	0.17031
		MDCFlat	0.93019	0.02483	0.04836	0.11215	0.03083
		MDCAgg	0.92489	0.02085	0.04078	0.09563	0.02592
	1000	NClu	0.02241	0.99383	0.04384	0.02786	0.10279
		MDCFlat	1.00000	0.00464	0.00924	0.02279	0.00580
		MDCAgg	0.92489	0.02085	0.04078	0.09563	0.02592
<i>Mushrooms</i>	10	NClu	0.27257	0.81449	0.40845	0.31441	0.58276
		MDCFlat	0.55079	0.11377	0.18859	0.31149	0.13523
		MDCAgg	1.00000	0.00103	0.00206	0.00513	0.00129
	100	NClu	0.16366	0.86880	0.27543	0.19537	0.46666
		MDCFlat	0.99620	0.02515	0.04907	0.11424	0.03125
		MDCAgg	1.00000	0.00103	0.00206	0.00514	0.00129
	1000	NClu	0.02149	0.99757	0.04207	0.02672	0.09892
		MDCFlat	1.00000	0.00216	0.00431	0.01071	0.00270
		MDCAgg	1.00000	0.00103	0.00206	0.00513	0.00129

Figure 17: NClu Clustering Results

networks, because the amount of noise was drastically reduced. On the other hand, the recall suffered tremendously. Upon investigating the reason for this, we discovered that the percentage of

Category	C1	C2	C3	C4	C5
Democrat	0.06	0	1.0	0	0.2
Republican	0.94	0	1.0	1.0	0.8

(a) Congress 62 % items clustered

Category	C6	C7	C8	C9	C10
Poisonous	0	0	0	1.0	1.0
Edible	1.0	1.0	1.0	0	0

Category	C11	C12	C13	C14	C15
Poisonous	0	0	1.0	1.0	0
Edible	1.0	1.0	0	0	1.0

(b) Mushrooms 86 % items clustered

Figure 18: Hard clusters produced by NClu

items actually clustered was quite low, and this can be directly attributed to the strict definition of an n -concept, which was required to maintain full local information. This may be viewed as one short coming of our approach and an avenue for future investigation. However, among items that were clustered, the precision was quite high. To further investigate this, a different procedure was followed with the *Mushrooms* and *Congress* information networks. After running NClu a post processing step was added that assigned each item only to the first cluster it appeared in, effectively creating a hard clustering. Clearly, we do not expect every item in the information network to be clustered (due to the strict definition), but we expect items in the same "hard" cluster to more or less belong to the same category. This is especially true with *Congress* and *Mushrooms* because they only contain 2 categories, and those categories do not overlap. The results of this experiment are illustrated in figure 18. Each matrix cell in the matrix represents a cluster, and the percent of items that belong to each class. As the figure shows, the clusters are for the most part, "pure" with respect to the class label, and thus represent valid global clusters. The results in figure 18 may seem to contradict those in figure 17, however, this is not the case. The BCubed precision of NClu suffers due to the fine grain approach that our model specifies; as more clusters are formed the BCubed precision decreases. However, this does not imply that the clusters formed are not pure with respect to their category, as was shown. While the BCubed precision metric penalizes the fine grain philosophy of NClu, this approach is necessary in domains with several categories.

Finally, figure 19 depicts some sample clusters obtained from the newsgroup information net-

works with $NClu$. The same post processing step described above was applied. In the *Mideast_Christian* sample clusters, the word clusters are clearly distinguishable between middleast newsgroups and Christian newsgroups. This same trend can be seen with the sample word clusters mined from the *Politics_Religion* information network.

C1	C2	C3	C4	C5
turks turkish armenia genocide army	theistic genesis theory god heavens	arab arabs israel answer	children christians faith jesus	southern flag christ president

(a) Mideast Christian

1	2	3	4	5
digital compatible 256 processing tiff	runtime language instructions collection	compressing compressed uncompressed basis standards	quantization edges gif	integraion mathematical fortran plotting graphical

(b) AllIPC

C1	C2	C3	C4	C5
republican republicans democrat libertarian policies	government police legal guns dictators	economic america civil justice setting	jesus god question guess problem	christ christians religious

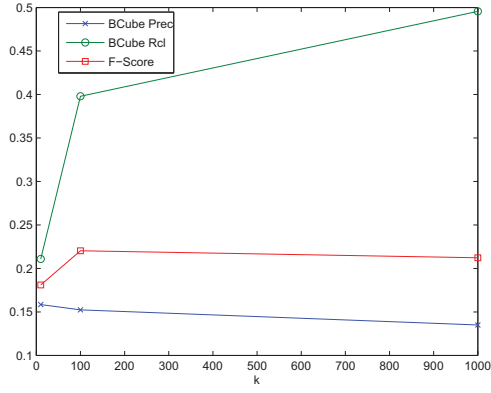
(c) Politics Religion

Figure 19: Sample Clusters mined by $NClu$

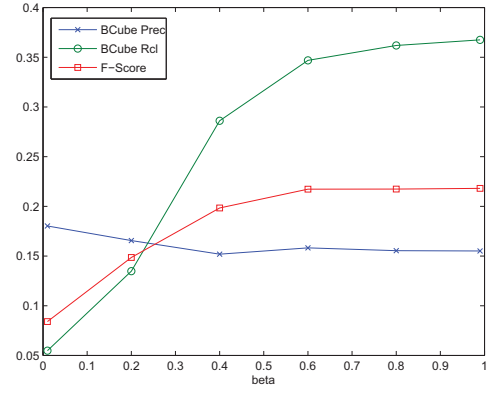
The experimental results conclude that $NClu$ successfully mines the global clustering structure. Moreover, $NClu$ outperforms previous methodologies due to soft clustering. The strict criterion that our clustering formulation imposes, however, may be a limiting factor of the method. This was evident in both the sample clusters mined and the precision and recall results for information networks containing relatively few natural categories.

Scalability For all information networks, $NClu$ terminated in under 2 minutes of execution time. $MDCFlat$ and especially $MDCAgg$ were not scalable. All experiments involving the full length usenet documents took well over 4 hours with $MDCAgg$, and over 2 hours with $MDCFlat$.

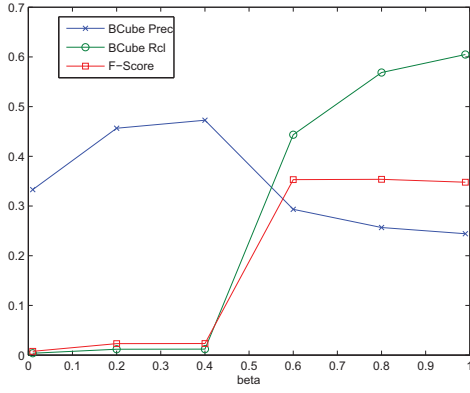
Effect of k and β Parameter selection plays a significant role in most clustering algorithms, and $NClu$ is no different. The effect of K and β on clustering quality was investigated. For each information network two experiments were conducted. First K was held constant while β varied, next β was held constant and k varied. Larger values of β bias $NClu$ towards clusters that contain more instances from sets G_1, \dots, G_{n-1} and fewer instances of G_c . For all information networks experimented with the instances of domains G_1, \dots, G_{n-1} were labeled, so the effect of K and β was measure via BCubed precision and recall. As the results indicate (figure 20), as β grows, the



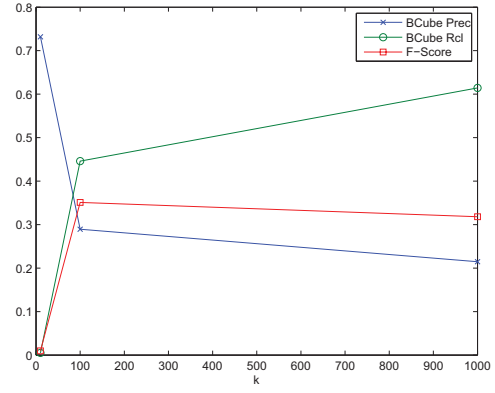
(a) AllPC ($k = 50$)



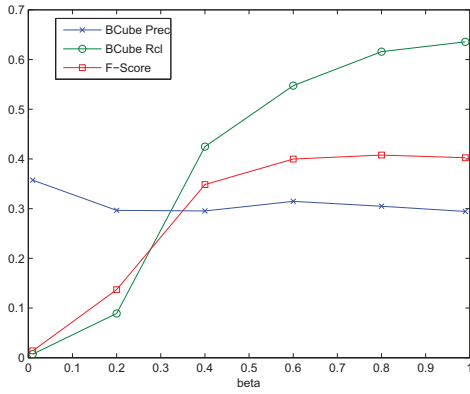
(b) AllPC ($\beta = 0.5$)



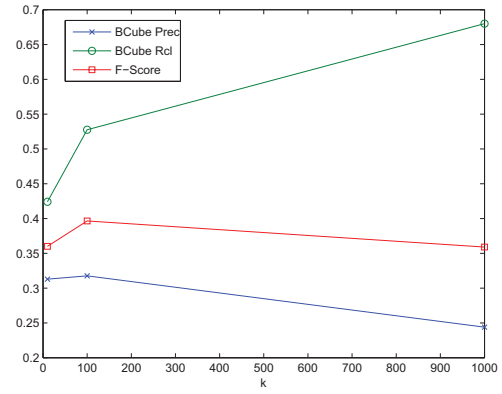
(c) Mideast Christian ($k = 50$)



(d) Mideast Christian ($\beta = 0.5$)



(e) Politics Religion ($k = 50$)



(f) Politics Religion ($\beta = 0.5$)

Figure 20: Effect of K and β on clustering results

Relation Graph Name	Domain	Description
<i>GO_Pheno</i>	Bioinformatics	G_c = genes, G_1 = Gene Ontology Terms, G_2 = Pheno Types
<i>Atheism_Christian</i>	NewsGroups	G_c = words, G_1 = Newsgroup documents from atheism forum, G_2 = documents from Christianity forum

Figure 21: Information networks utilized for clustering experiments

words	Christian Article	Atheist Article
accept	"I don't believe anyone but the Spirit would be able to convince you the Spirit exists. Please don't complain about this being circular"	"Is god not concerned with my disposition? Why is it beneath him to provide me with the evidence I would require to believe?"
bible	"We chose to believe whatever we want, but we are not allowed to define our own Christianity"	"What's a sincerely searching Agnostic or Atheist supposed to do when even the search turns up nothing?"
evidence	"Lets talk about principles. If we accept that God sets the standards for what ought to be included in Scripture - then we can ask"	"Christians have failed to show us how there way of life is in any wy better than ours. I do not see why the attempt to try it is necessary, or even particularly attractive."
god read reason		

(a) Atheism Christian 3-concept

Genes	GO Terms	PhenoTypes
NOTCH1 PPARD PSEN1 PSEN2 SHH	anatomical structure development cell development cell differentiation cellular developmental process cellular metabolic process cellular process developmental process macromolecule metabolic process metabolic process multicellular organismal development multicellular organismal process system development organ development biological regulation growth regulation of biological process regulation of cellular process primary metabolic process cell communication reproduction reproductive process biopolymer metabolic process positive regulation of biological process tissue development positive regulation of cellular process signal transduction ectoderm development epidermis development hair cycle hair cycle process hair follicle development molting cycle molting cycle process developmental growth	abnormal brain morphology abnormal nervous system morphology abnormal postnatal growth/weight/body size abnormal coat/ hair morphology abnormal skin condition/ morphology abnormal epidermal layer morphology abnormal hair follicle morphology abnormal hair follicle structure abnormal hair follicle structure/orientation abnormal skin morphology abnormal embryogenesis/ development embryonic lethality embryonic lethality during organogenesis epidermal hyperplasia thickened epidermis

(b) GO Pheno 3-concept

Figure 22: Sample n -concepts in real world information networks

clusters loose precision but gain recall in all information networks. As β grows, the n -concepts are less specific in terms of G_i , and thus loose precision. Moreover, fewer elements from each G_i are available the lower β is, thus leading to high precision. On the other hand recall will clearly increase as more and more elements from each G_i leading to higher recall values. Interestingly, the $F - Score$ increases significantly at median values of β (0.4-0.6) and remains almost constant thereafter, indicating that the ideal trade-off between precision and recall usually occurs around this range. The effect of K on clustering performance follows the same pattern as β . Lower values of K result in high precision, while higher values allow for higher recall. Once again, the $F - Score$ peaks at a ceratin value of K and steadily declines thereafter. Thus setting larger values of β and K seem to work best when the objective of the multi-way clustering is measured in terms of items from G_1, \dots, G_{n-1} .

The next approach we undertook in validating our results was to simply enumerate n -concepts and view the results and use domain knowledge and basic intuition to judge the utility of the revealed associations among the sets in each n -concepts. In order to accomplish this, we could not simply randomly partition data as was done for previous experiments. Instead, real-world datasets that were created independently of each other were obtained and placed into an information net-

work (figure 21). The resulting n -concepts are depicted in figure 22. The 3-concept mined from *Atheism-Christian* information network contains articles from each context that are related by a few key words. Excerpts of the articles appear in figure 22(a). The author of each usenet document, whether from the Christain newsgroup or the Atheist newsgroup, puts forth their argument pertaining to the existence or non-existence of God. Moreover, several of the articles included in the 3-concept were responses from users of both newsgroups to each other, although the articles never appeared together in the same context.

In the next 3-concept depicted (figure 22(b)), we see several phenotypes related to the morphology of hair appear in the cluster along with gene ontology terms that clearly relate to hair (hair cycle, hair cycle process, hair follicle development).

4.4 Conclusion

In this chapter we extended the Formal Concept Analysis model of bi-clustering to an information network by introducing the notion of an n -concept. It was proven that an n -concept is a generalization of a bi-cluster (2-concept) and that the set of n -concepts form a complete lattice. Moreover, we proposed a cluster quality measure, Ω , that maximizes the number of 1s in local 2-concepts, while accounting for the natural tradeoff between height and width. The search strategies of closed itemset mining algorithms, simultaneous pruning of contexts and the properties of Ω were combined to produce the `NClu` algorithm. It was shown, theoretically, that the computation cost of `NClu` is no greater than that of a closed itemset mining algorithm. Furthermore, we illustrated experimentally that `NClu` outperforms closed itemset mining algorithms in terms of computation cost by orders of magnitudes.

Experiments on labeled data revealed that `NClu` produces better clustering results than the state of the art multi-way clustering algorithm `MDC` when considering data with multiple, overlapping natural clusters. Experimental results, however, also revealed the shortcomings of n -concepts. The strict criterion imposed by n -concepts was a clear limiting factor of the method. While this strict criterion lead to very accurate clusters, the recall was quite poor; especially for information

networks containing relatively few natural categories. In the following chapter we address this shortcoming by reformulating the definition of an information network cluster. The n -concepts will serve as initial candidate clusters, while the local concepts lattices will be utilized to expand these clusters in order to increase the recall.

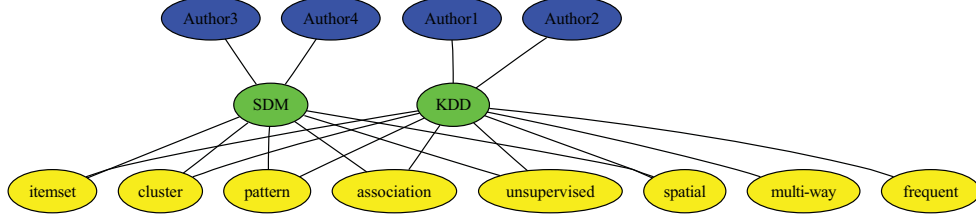


Figure 23: An information network where n -concepts fail

5 Information-Tree Clustering

The n -concepts model of information network clustering suffers from two main limitations. First, n -concepts are defined only over star-shaped networks. Second, as experimental results in the previous chapter illustrated, the strict definition of an n -concept hinders the discovery of clusters whose structure is not immediately available via a single context. For example, consider the simple information network depicted in figure 23. *Author1* and *Author2* only publish in the *KDD* conference, while *Author3* and *Author4* exclusively publish in the *SDM* conference. Two distinct concepts exist in the *Conferences_Authors* context; namely: $(\{Author1, Author2\}, \{SDM\})$ and $(\{Author3, Author4\}, \{KDD\})$; Hence the entire network contains two distinct n -concepts. However, intuition, and domain knowledge suggest that all four authors should be placed in a single cluster since both the *KDD* and *SDM* conferences deal with data mining. Under the strict definition of n -concepts, not only are the authors never placed in a single cluster, the two distinct n -concepts are not neighbors, and only share the null element concept as a common neighbor. This example clearly illustrates phenomenon of n -concepts constituting accurate clusters while suffering in terms of recall. Objects found within an n -concept tend to belong to the same natural category; however, several more objects that belong to the same natural category may never be placed in a cluster due to the strict definition.

In this chapter, both limitations of n -concepts are addressed. First, the specification of an information network cluster is expanded beyond star-shaped networks to information trees (networks that do not contain loops). More importantly, the strict definition of n -concepts is considerably relaxed, while retaining the guiding philosophy of an n -concept that information network clusters

should embody a strong local clustering tendency, sharpened by the additional information available via the network. n -concepts institute this guiding principle by matching concepts from the local concept lattices; in this chapter we agument this by allowing for 1) in-exact matches while 2) matching neighborhoods of concepts across all concept lattices in the information tree. We illustrate, mathematically, that such matchings constitute highly connected subspaces both locally and over the entire information network. In addition, the `TreeClu` algorithm is developed to enumerate such clusters by first making use of a slightly modified version of `NClu` to enumerate initial candidate subspaces. Next, the subspaces are refined iteratively first according to their local concept lattices and secondly in terms of other concepts in the networks. Experimental results with real-world data clearly demonstrate the advantage of `TreeClu` over the state-of-the-art methods `NetClus` and `MDC`.

5.1 Problem Formulation

Information network clusters should exhibit a strong local clustering tendency in each domain, in addition to high connectivity amongst all other domains in the network. The n -concepts model achieves this by enforcing full connectivity across all domains. We reformulate this strict requirement by quantifying the degree of connectivity in a subspace $S = (G_1, \dots, G_n)$ and require this quantity to be greater than the expected connectivity of a random S in \mathcal{T} . In addition, the novel measure, of **self-connectivity** is introduced in order to measure the clustering tendency within a single domain.

5.1.1 Connectivity and Self Connectivity in a Single Context

The notion of connectivity and self connectivity are first formulated with regards to a single context relating two domains. In this manner, the ideas are simpler to demonstrate; this is followed up with the general definitions in terms of an information network. Connectivity of an n -concept was defined as the area of that n -concept, due to the fact that the subspace did not encompass any zeros. Here we generalize the notion of connectivity to any given subspace.

Definition 15. The *connectivity* of the object-set G_i in subspace (G_i, G_j) , denoted as $\phi(G_i, G_j)$, is

$$\phi(G_i, G_j) = \sum_{g \in G_i} |\psi^j(g) \cap G_j| \quad (48)$$

and the *index of connectivity*, denoted as $\rho(C)$, is

$$\rho(G_i, G_j) = \frac{\phi(G_i)}{|G_i| * |G_j|} \quad (49)$$

Viewing \mathbb{K}_{ij} as a bipartite graph, the connectivity of an object-set G_i to object-set G_j is simply a count of the number of edges between elements of G_i and G_j . From the matrix point of view this corresponds to the number of 1s in the sub-matrix induced by the elements of G_i (rows) and G_j (columns). A **fully connected** subspace is one in which $\rho(G_i) = \rho(G_j) = 1$. Intuitively, subspaces in which ρ of both domains is large, imply that the elements of G_i and G_j are highly correlated. Moreover, ϕ and ρ are symmetric, as $\phi(G_i, G_j) = \phi(G_j, G_i)$ and $\rho(G_i, G_j) = \rho(G_j, G_i)$. For simplicity, we will write $\phi(G_i)$ and $\rho(G_i)$ instead of $\phi(G_i, G_j)$ and $\rho(G_i, G_j)$ although these values depend on G_j . The concepts of a context constitute those subspaces that are fully connected and maximal with respect to full connectivity.

Proposition 6. Given \mathbb{K}_{ij} , then for any subspace $C = (G_i, G_j)$, C is fully connected and maximal with respect to fully connectivity if and only if $C \in \mathbb{B}(\mathbb{K}_{ij})$

Proof. • \longrightarrow By definition of ρ , we have that $\psi^j(G_i) \supseteq G_j$, and dually $\psi^i(G_j) \supseteq G_i$. However, by maximality $\psi^j(G_i) = G_j, \psi^i(G_j) = G_i$ implying that $(G_i, G_j) \in \mathbb{B}(\mathbb{K}_{ij})$

• \longleftarrow . In this case $\psi^j(G_i) = G_j, \psi^i(G_j) = G_i$ and C is maximal. Moreover $\sum_{g \in G_i} |\psi^j(g) \cap G_j| = |G_j|$ and dually $\sum_{g \in G_j} |\psi^i(g) \cap G_i| = |G_i|$, thus $\psi(G_i) = |G_i| * |G_j|$, implying that $\rho(G_i) = 1$.

□

Whilst the connectivity measure maybe used to quantify the correlation amongst the object-sets of differing domains, it is not always a good indication of the clustering tendency of the objects within a single domain. Consider an amplified instance of the example in figure 23 in which the connectivity of the Authors and Conferences domain is high; however the subspace is segmented into partitions of authors and conferences that never interact each other. In this case, the local clustering tendency of all the authors in the subspace is not very strong (as two segments exist), although the connectivity is quite high. Hence, in addition to connectivity, a measure of the clustering tendency of the objects within a single domain is also needed to quantify the clustering quality of a subspace (G_i, G_j) . Due to the fact that the information network consists of multiple domains, the clustering tendency of a domain can only be measured in terms of other domains. For a given subspace (G_i, G_j) , the more objects of G_j that every $g \in G_i$ are commonly connected to, the greater the clustering tendency of G_i . In other words, the more objects every pair $g_i^1, g_i^2 \in G_i$ are commonly connected to in G_j the greater the **self-connectivity** of G_i . This quantity is computed as

$$\sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} |(\Psi^j(g_i^k) \cap \Psi^j(g_i^l) \cap G_j)| \quad (50)$$

Normalizing, we attain

$$\begin{aligned} & \sum_{k=1}^{|G_i|-1} \frac{1}{|G_i|-k} \sum_{l=k+1}^{|G_i|} \frac{|(\Psi^j(g_i^k) \cap \Psi^j(g_i^l) \cap G_j)|}{|G_j|} \\ &= \sum_{k=1}^{|G_i|-1} \frac{1}{k} * |G_j| \sum_{l=k+1}^{|G_i|} |(\Psi^j(g_i^k) \cap \Psi^j(g_i^l) \cap G_j)| \\ &= \frac{2}{|G_i| * |G_j| * (|G_i|-1)} \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} |(\Psi^j(g_i^k) \cap \Psi^j(g_i^l) \cap G_j)| \\ &= \frac{1}{\binom{|G_i|}{2} * |G_j|} \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} |(\Psi^j(g_i^k) \cap \Psi^j(g_i^l) \cap G_j)| \end{aligned}$$

The self-connectivity is normalized by counting the total number of pairs of objects in the domain, $\binom{|G_i|}{2}$, and multiplying by the total number of possible connections each pair could share $|G_j|$.

Definition 16. Given a subspace (G_i, G_j) in \mathbb{K}_{ij} the **self-connectivity** of G_i is given by

$$\sigma(G_i, G_j) = \frac{1}{\binom{|G_i|}{2} * |G_j|} \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} |(\Psi^j(g_i^k) \cap \Psi^j(g_i^l) \cap G_j)| \quad (51)$$

if $|G_i| > 1$ and 0 otherwise.

Clearly, the object-sets of a formal concept (G_i, G_j) are fully self-connected with $\sigma(G_i) = \sigma(G_j) = 1$. While this may be desirable in some applications, the introductory example illustrates the need to consider subspaces in which both $\rho(G_i) < 1$ $\sigma(G_i) < 1$ for both domains. To achieve this, the strict restriction of full of connectivity and self connectivity is relaxed by defining clusters to be those subspaces in which $\rho(G_i), \rho(G_j)$ and $\sigma(G_i), \sigma(G_j)$ exceed the expected values of these quantities. The question that remains to be answered is: *what is expected value of $\phi(G_i)$ and $\sigma(G_i)$?*

Assuming the independence of each $g \in G_i$ (and more generally each $g \in G_j$), for a given object-set G_i , and randomly selected object-set G_j , then the expected size of $|\Psi^j(g_i) \cap G_j|$ is the total number of success in a sequence of $|G_j|$ independent yes no experiments. Each experiment, has probability of success of $\frac{|\Psi^j(g_i)|}{|G_j|}$. Hence, the expected size of $|\Psi^j(g_i) \cap G_j|$ is the expected value of a binomially distributed random variable $B(n, k_g)$ with $n = |G_j|$ and $k_g = \frac{|\Psi^j(g_i)|}{|G_j|}$. We are interested in the expected value of the sum of $|G_i|$ such binomially random variables (representing all $g \in G_i$) as this constitutes the expected value of $\phi(G_i)$:

$$\begin{aligned} E \left[\sum_{g \in G_i} B(n, k_g) \right] &= \\ &= E[B(n, k_{g_1})] + \dots + E[B(n, k_{g_{|G_i|}})] \\ &= n * \sum_{i=1}^{|G_j|} p_{g_i} \\ &= \frac{|G_j|}{|G_j|} * \sum_{g \in G_i} |\Psi^j(g)| \end{aligned} \quad (52)$$

Making use of equation 52 we may now define the expected value of $\phi(G_i)$ and $\rho(G_i)$.

Definition 17. Given a subspace $S = (G_i, G_j)$ in context \mathbb{K}_{ij} then

$$E[\phi(G_i)] = \frac{|G_j|}{|\mathbf{G}_j|} \sum_{g \in G_i} |\psi^j(g)| \quad (53)$$

Building on definition 17 it is straightforward to define $E[\rho(S)]$ simply as

Definition 18.

$$E[\rho(G_i)] = \frac{E[\phi(G_i)]}{|G_i| * |G_j|} \quad (54)$$

Notice that while ϕ and ρ are symmetric with respect to G_i and G_j , $E[\phi(G_i)]$ is not (i.e. for a given subspace (G_i, G_j) , $E[\phi(G_i)] \neq E[\phi(G_j)]$), whereas $\phi(G_i) = \phi(G_j)$.

For a given object-set G_i and randomly selected G_j , the expected self-connectivity depends on the expected number of objects in G_j that are commonly connected to by pair of objects $g^1, g^2 \in G_i$. As an analogy, given two coins each of which is flipped $|G_j|$ times we wish to compute the number of heads that occur on the same flip number. Defining an experiment as the simultaneous flip of both coins, the probability of success is $\frac{|\psi^j(g^1)|}{|\mathbf{G}_j|} * \frac{|\psi^j(g^2)|}{|\mathbf{G}_j|} = \frac{|\psi^j(g^1)| * |\psi^j(g^2)|}{|\mathbf{G}_j|^2}$. Hence the number of expected success is the expected value of a binomially distributed random variable $B(n, k_g)$ with $n = |G_j|$ and $k_{g_i}^2 = \frac{|\psi^j(g_i^1)| * |\psi^j(g_i^2)|}{|\mathbf{G}_j|^2}$. The expected self connectivity of an object-set G_i in a subspace (G_i, G_j) is then given as

$$\begin{aligned} & \frac{1}{\binom{|G_i|}{2} * |G_j|} * E \left[\sum_{l=1}^{|G_i|-1} \sum_{m=l+1}^{|G_i|} B(n, k_{g_i}^{g_i^m}) \right] \\ &= \frac{1}{\binom{|G_i|}{2} * |G_j|} * \frac{|G_j|}{|\mathbf{G}_j|^2} \sum_{l=1}^{|G_i|-1} \sum_{m=l+1}^{|G_i|} |\psi^j(g_i^l)| * |\psi^j(g_i^m)| \\ &= \frac{1}{\binom{|G_i|}{2} * |\mathbf{G}_j|^2} * \sum_{l=1}^{|G_i|-1} \sum_{m=l+1}^{|G_i|} |\psi^j(g_i^l)| * |\psi^j(g_i^m)| \end{aligned}$$

Making use of $E[\rho]$ and $E[\sigma]$, an intuitive definition for a 2-cluster in a single context \mathbb{K}_{ij}

Example 5. Consider the following context, \mathbb{K}_{ij} Let $\alpha = 1.5$ and $\delta = 1.5$. Consider the subspace



<i>Domain</i>	ρ	$\alpha * E[\rho]$	σ	$\delta * E[\sigma]$
<i>J</i>	$\frac{1+2+2}{4*2} = 0.625$	$1.5 * \frac{\frac{2}{3} * (1+3+1+2)}{8} = 0.4375$	$\frac{1}{12} * ((1+0+1) + (0+2) + (0)) = 0.33$	$1.5 * \frac{1}{6*16} ((3+1+2) + (3+6) + (2)) = 0.265$
<i>I</i>	$\frac{1+2+2}{4*2} = 0.625$	$1.5 * \frac{\frac{7}{4} * (3+3)}{8} = 0.428$	$\frac{1}{4} * (2) = 0.5$	$1.5 * \frac{1}{49} (9) = 0.275$

$Domain$	ρ	$\alpha * E[\rho]$	σ	$\delta * E[\sigma]$
J	$\frac{1+2+2}{3*2} = 0.833$	$1.5 * \frac{2^4 * (1+3+2)}{6} = 0.5$	$\frac{2}{12} * ((1+1) + (2)) = 0.5$	$1.5 * \frac{2}{16*3*2} ((3+2) + (6)) = 0.229$
I	$\frac{1+2+2}{3*2} = 0.833$	$1.5 * \frac{3^7 * (3+3)}{6} = 0.642$	$\frac{2}{6} * (2) = 0.666$	$1.5 * \frac{2}{2*36} (9) = 0.375$

82

5.1.2 Connectivity in an Information Tree

Connectivity of an object-set in an information-tree is a generalization of connectivity in a single context. Given $S = (G_1, \dots, G_{|V|}) \in \mathcal{T}$

$$\phi(G_i, G_1, \dots, G_{i-1}, G_{i+1}, \dots, G_{|V|}) = \sum_{G_j \in \mathbf{G}_j \in \Gamma(\mathbf{G}_i)} \phi(G_i, G_j) \quad (55)$$

$$\phi(G_i, G_1, \dots, G_{i-1}, G_{i+1}, \dots, G_{|V|}) = \frac{\phi(G_i)}{|G_i| * \sum_{G_j \in \mathbf{G}_j \in \Gamma(\mathbf{G}_i)} |G_j|} \quad (56)$$

Once again, for simplicity we will denote $\phi(G_i, G_1, \dots, G_{i-1}, G_{i+1}, \dots, G_{|V|})$ simply as $\phi(G_i)$. Generalizing expected connectivity of an object-set, G_i entails summing over the expected connectivity of G_i in each neighboring domain.

$$E[\phi(G_i)] = \sum_{G_j \in \mathbf{G}_j \in \Gamma(\mathbf{G}_i)} E[\phi(G_i, G_j)] \quad (57)$$

$$E[\rho(G_i)] = \frac{\sum_{G_j \in \mathbf{G}_j \in \Gamma(\mathbf{G}_i)} E[\phi(G_i, G_j)]}{|G_i| * \sum_{G_j \in \mathbf{G}_j \in \Gamma(\mathbf{G}_i)} |G_j|} \quad (58)$$

Notice, the connectivity of an object-set in a subspace only depends upon the objects-sets from neighboring domains. On the other hand, due to the fact that \mathcal{T} only contains a single component (i.e. all vertices are reachable from any other vertex), the self-connectivity of a an object-set in a subspace (G_1, \dots, G_n) should involve object-sets from every domain in the network. In light of this, we introduce a novel operator ϕ^j . For a given object $g_i \in G_i$ in subspace S , $\phi^j(g_i)$ generates an object-set in domain \mathbf{G}_j that is connected to g_i via any intermediary object-set G_k in the subspace. For example, from figure 23, consider the subspace $(\{Author1, Author3\}, \{KDD\}, \{itemset, cluster\})$, then $\phi^{AUTHOR}(\{itemset\})$ yields the object-set $\{Author1, Author3\}$ since both of these authors are connected to *itemset* via the intermediary object *KDD* in S . To formally define this operator, the notion of hops in the information network must be considered. For any two domains $\mathbf{G}_i, \mathbf{G}_j \in \mathcal{T}$, let $H(\mathbf{G}_i, \mathbf{G}_j)$ denote the minimum number of hops necessary to reach \mathbf{G}_j from \mathbf{G}_i . Formally,

this corresponds to the cardinality of the smallest path connecting \mathbf{G}_i to \mathbf{G}_j . $H(\mathbf{G}_i, \mathbf{G}_j)$ is always non-zero and unique since we are considering only information trees \mathcal{T} that only contain a single component and no loops. Employing, H we may define $\Upsilon(\mathbf{G}_i, \mathbf{G}_j)$ which captures the index of the domain neighboring \mathbf{G}_i and is closest to \mathbf{G}_j in terms of H .

Definition 19. Given an information network, $\mathcal{T} = (V, E)$, for any pair of domains $\mathbf{G}_i, \mathbf{G}_j$, let $H(\mathbf{G}_i, \mathbf{G}_j)$ denote the minimum number of hops necessary to reach \mathbf{G}_j from \mathbf{G}_i . Then

$$\Upsilon(\mathbf{G}_i, \mathbf{G}_j) = k \quad s.t. \mathbf{G}_k = \underset{\mathbf{G}_k \in \Gamma(\mathbf{G}_i)}{\operatorname{argmin}} H(\mathbf{G}_k, \mathbf{G}_j) \quad (59)$$

Utilizing, Υ , the ϕ^j operator is defined recursively.

Definition 20. Given an information tree $\mathcal{T} = (V, E)$, and subspace $S = (G_1, \dots, G_{|V|}) \in \mathcal{T}$, then for any object $g_i \in G_i$

$$\phi^j(g_i, G_1, \dots, G_{i-1}, G_{i+1}, \dots, G_n) = \begin{cases} \psi^j(g_i) \cap G_j & \text{if } \mathbf{G}_j \in \Gamma(\mathbf{G}_i) \\ \bigcup_{g_k \in G_k} \phi^j(g_k) & \text{otherwise} \end{cases} \quad (60)$$

where $G_k = \psi^{\Upsilon(\mathbf{G}_i, \mathbf{G}_j)}(g_i) \cap G_{\Upsilon(\mathbf{G}_i, \mathbf{G}_j)}$.

For brevity, we will write $\phi^j(g_i)$. For a given object $g_i \in G_i$ if the domains \mathbf{G}_i and \mathbf{G}_j share a context (they are neighbors in \mathcal{T} , then $\phi^j(g_i)$ simply enumerates those objects in G_j that are connected to g_i . On the other hand, if the two domains do not share a context then the nearest intermediary domain \mathbf{G}_k between \mathbf{G}_i and \mathbf{G}_j is located. Next, all objects that g_i is connected to in object-set G_k are computed and the operation is repeated recursively on each $g_k \in G_k$ since g_i is connected to each g_k . Applying the ϕ operator, then self-connectivity in an information tree can be generalized as

$$\sigma(G_i) = \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} \sum_{G_j \in C \setminus G_i} |\phi^j(g_i^k) \cap \phi^j(g_i^l) \cap G_j|$$

A damping term, $0 < \lambda \leq 1$ maybe incorporated into the above equation to dampen the effect of large self-connectivity through many intermediate nodes in large information trees. Such a weighting term may be $\lambda^{H(\mathbf{G}_i, \mathbf{G}_j)}$. The more hops required to reach \mathbf{G}_j , the less the value of self connectivity resulting from objects in that domain. Incorporating the damping term and normalizing the self-connectivity we attain:

$$\begin{aligned}
& \sum_{k=1}^{|G_i|-1} \frac{1}{|G_i| - k} \sum_{l=k+1}^{|G_i|} \frac{1}{n-1} \sum_{G_j \in \mathcal{S} \setminus G_i} \frac{\lambda^{H(\mathbf{G}_i, \mathbf{G}_j)-1} * |\varphi^j(g_i^k) \cap \varphi^j(g_i^l) \cap G_j|}{|G_j|} \\
&= \sum_{k=1}^{|G_i|-1} \frac{1}{k(n-1)} \sum_{l=k+1}^{|G_i|} \sum_{G_j \in \mathcal{S} \setminus G_i} \frac{\lambda^{H(\mathbf{G}_i, \mathbf{G}_j)-1} * |\varphi^j(g_i^k) \cap \varphi^j(g_i^l) \cap G_j|}{|G_j|} \\
&= \frac{2}{|G_i| * (|V| - 1) * (|G_i| - 1)} \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} \sum_{G_j \in \mathcal{S} \setminus G_i} \frac{\lambda^{H(\mathbf{G}_i, \mathbf{G}_j)-1} * |\varphi^j(g_i^k) \cap \varphi^j(g_i^l) \cap G_j|}{|G_j|} \\
&= \frac{1}{\binom{|G_i|}{2} * \sum_{G_j \in \mathcal{S} \setminus G_i} |G_j|} \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} \sum_{G_j \in \mathcal{S} \setminus G_i} \lambda^{H(\mathbf{G}_i, \mathbf{G}_j)-1} * |\varphi^j(g_i^k) \cap \varphi^j(g_i^l) \cap G_j|
\end{aligned}$$

For the given object-set, G_i each pair of objects may be connected with up to $\sum_{G_j \in \mathcal{S} \setminus G_i} |G_j|$, with the total number of pairs being $\binom{|G_i|}{2}$.

Definition 21. Given an information tree $\mathcal{T} = (V, E)$ and a subspace $S = (G_1, \dots, G_n)$, the self-connectivity of G_i is

$$\sigma(G_i) = \frac{1}{\binom{|G_i|}{2} * \sum_{G_j \in \mathcal{S} \setminus G_i} |G_j|} \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} \sum_{G_j \in \mathcal{S} \setminus G_i} \lambda^{H(\mathbf{G}_i, \mathbf{G}_j)-1} * |\varphi^j(g_i^k) \cap \varphi^j(g_i^l) \cap G_j| \quad (61)$$

5.1.3 Cluster Definition

To formally define a cluster in an information tree, we must only define the expected self-connectivity of an object-set in such a network. However, we illustrate in appendix, that specifying this value is quite involved and entails utilizing the sieve principle and possibly Boole's inequality; moreover, this quantity is very expensive to compute. As such, we will allow for the user to select a value

$0 < \delta \leq 1$, for which the self-connectivity of each domain must satisfy. Experimental results, have indicated that the cost of computing the expected value is much too costly, while pre-selecting a value for δ only required a few trials to determine (see appendix and experimental section for details).

Definition 22. *Given an information tree \mathcal{T} , and real-numbers $\alpha \geq 1, 0 < \delta \leq 1$, then an n -cluster of \mathcal{T} is a subspace $C = (G_1, \dots, G_n)$, such that the following conditions are met for all $G_i \in C$:*

1. $\rho(G_i) > \alpha * E[\rho(G_i)], \forall G_i \in C$
2. $\sigma(G_i) > \delta, \forall G_i \in C$
3. *No object can be added to any G_i without violating the above conditions*

Conditions 1 and 2 enforce high connectivity and self-connectivity, while condition 3 is set to avoid mining subspaces that trivially satisfy conditions 1 and 2. Let us illustrate with an example.

Example 6. *Consider the information tree and contexts in figure 25. Let $\alpha = 1$ and $\delta = 0.5$. Then the following subspace is a 4-cluster $(\{A^1, A^3, A^4\}, \{M^1, M^2\}, \{V^2, V^3\}, \{W^1, W^3\})$.*

<i>Domain</i>	ρ	$\alpha * E[\rho]$	σ	δ
<i>Actors</i>	1.0	0.733	1.0	0.5
<i>Movies</i>	0.9	0.8525	0.85	0.5
<i>Viewers</i>	0.75	0.481	0.714	0.5
<i>Websites</i>	0.75	0.687	0.85	0.5

The information-network clustering problem in an information trees is: Given a relation tree \mathcal{T} , and real-numbers $1 \leq \alpha$, and $1 \leq \delta$ find all subspaces S , such that S satisfies definition 22. The problem is expected to be hard: In the worst case selecting, $\delta = 1$, and large α , then the problem decomposes into enumerating matching concepts across all contexts in \mathcal{T} . Then in this case, as was illustrated in chapter 4, this problem is at least as hard as enumerating concepts in a single lattice, which is NP-complete in the worst case.

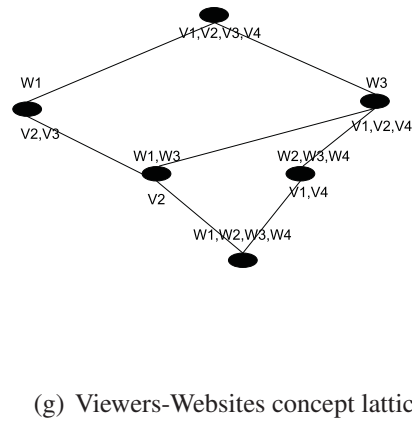
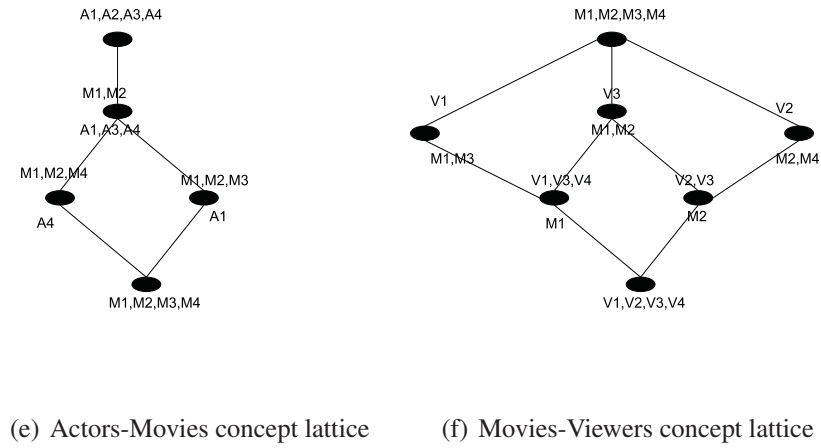
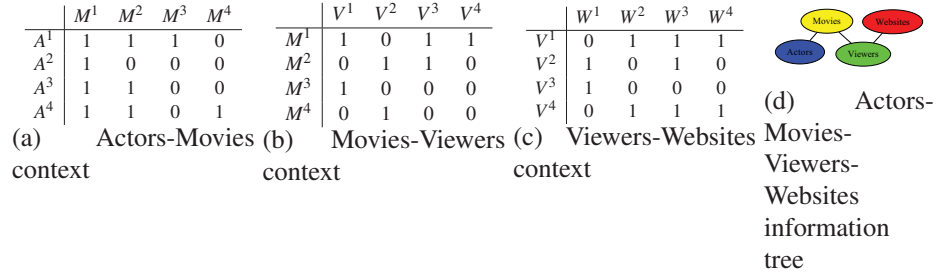


Figure 25: Example contexts, concept lattices and information tree

5.2 TreeClu Algorithm

In this section, the `TreeClu` algorithm is developed to mine n -clusters efficiently. `TreeClu` employs a bottom-up approach in which initial candidate clusters are formed, followed by combining the candidates and iterative refinement. The algorithm consists of three main stages: candidate enumeration, local refinement, and network refinement. In the candidate enumeration phase, `TreeClu` seeks initial subspaces that are guaranteed to satisfy the first two conditions of definition 22; thus fully connected, maximal candidate subspaces are enumerated. As we will show, such subspaces are precisely the semi- n -concepts of the information tree: a relaxation of n -concepts in the information tree. Next, the local refinement phase maps each individual semi-concept of the candidate cluster to a concept in its local context. For each local concept, condition 3 of definition 22 is achieved by adding objects from neighboring concepts. Finally, in the network-refinement phase the local clusters formed in each context are combined together, while objects with low self-connectivity are pruned from the global cluster.

Due to the potentially explosive computational cost of solving the n -cluster problem, `TreeClu` sacrifices completeness: the enumeration of all n -clusters in \mathcal{T} is not guaranteed. On the other hand, `TreeClu` attempts to maximize coverage of all objects, by striving to place every object in a cluster, while still allowing for overlaps. However, because the number of clusters is determined purely from the data and the values of α and δ , not every object will be placed in a cluster.

5.2.1 Initial Candidate Clusters

`TreeClu` initially mines subspaces that are fully connected and fully self-connected (i.e. $\rho(G_i) = 1$ and $\sigma(G_i) = 1, \forall G_i$). Such subspaces clearly satisfy conditions 1 and 2 of the n -cluster definition, although they most likely do not satisfy the third condition. Proposition 6 states that the formal concepts of a context are fully connected and self-connected; however, we present a more general class of subspaces that are also fully connected and self-connected.

Definition 23. *Given an information tree $\mathcal{T} = (V, E)$, and subspace $S = (G_1, \dots, G_{|V|})$, then S is a*

	Input: α, δ , Information tree: $\mathcal{T} = (V, E)$
1	begin
2	foreach $G_i \in V$ do
3	foreach $g_i \in G_i$ <i>not yet in cluster</i> do
4	$C \leftarrow (\emptyset_1, \dots, \emptyset_{ V })$;
5	GenerateCandidate(g_i, C) ;
6	if $C \neq \emptyset$ then
7	LocalRefine(C) ;
8	NetworkRefine(C) ;
9	if C <i>is maximal</i> then
10	Output C as information network cluster ;
11	end

Algorithm 5: Overview of TreeClu Algorithm

semi- n -concept if

$$\forall (G_i, G_j) \in E \quad \psi^j(G_i) = G_j \vee \psi^i(G_j) = G_i$$

Clearly, all n -concepts are also semi- n -concepts; hence a slightly modified version of the NClu algorithm may be utilized to enumerate the candidates. Nevertheless, the fact that the local refinement phase will explore the neighbors of each candidate computational savings can be realized by as follows: Incrementally generate a single candidate from a single object, execute the refinement phases, then repeat the process with all objects not yet enumerated. Following this agend, the high-level pseudocode of TreeClu appears as algorithm 11. Line 2, may suggest that TreeClu does not mine overlapping clusters, however, this check is only performed prior to the candidate generation phase. Objects previously clustered may still appear in other clusters yet to be mined during local refinement and network refinement phases.

Given an object g_i , generating a semi- n -concepts consists of performing a depth first search of \mathcal{T} and applying the ψ^j operator along each edge. If at any point during the search the ψ operator returns an empty set, then the search is terminated since a semi- n -concept cannot formed in the space. In this case, TreeClu proceeds with candidate generation with a new object g_i .

Input: Object-set G_i , Candidate C

Data: \mathcal{T}

```

1 begin
2   foreach  $G_j \in C$  s.t.  $\mathbf{G}_j \in \Gamma(\mathbf{G}_i) \wedge G_i = \emptyset$  do
3     if  $\psi^j(G_i) \neq \emptyset$  then
4        $G_j \leftarrow \psi^j(G_i)$  ;
5       return GenerateCandidate( $G_j, C$ ) ;
6     else
7       return  $\emptyset$  ;
8 end

```

Procedure GenerateCandidate

On the other hand, if a candidate is formed then the LocalRefine procedure is invoked. The GenerateCandidate procedure in figure 8 illustrates this entire process.

5.2.2 Local Refinement

Every candidate cluster $C = (G_1, \dots, G_{|V|})$ enumerated by the GenerateCandidate procedure contains exactly $|V| - 1$ semi-concepts, one in each context. Proposition 1 ensures that each of these semi-concepts can be expanded to a concept in its respective context. Furthermore, proposition 6 ensures that each object-set of these local concepts will still maintain full connectivity and self-connectivity. The local refinement phase constructs clusters by augmenting each of these local concepts with the maximal number of objects subject to the connectivity and self-connectivity constraint. We illustrate that in order to achieve this objective, TreeClu must seek those objects that introduce the least number of 0s when augmented with a local concept. Moreover, we prove that for a given concept, (G_i, G_j) such objects are precisely those contained in the upper and lower neighbors of (G_i, G_j) . Hence, TreeClu navigates the concept lattice of each context augmenting the cluster with objects obtained from neighboring concepts until the thresholds have been met. For ease of notation, given a subspace $(G_1, \dots, G_{|V|})$ we will let $\mathbb{S}((G_1, \dots, G_{|V|}))$ denote the proposition $\forall i p(G_i) \geq E[p(G_i)] \wedge \sigma(G_i) \geq \delta$.

Every semi-concept $(G_i, G_j) \in C$ s.t. $(\mathbf{G}_i, \mathbf{G}_j) \in E$, can be mapped to a concept of \mathbb{K}_{ij} as $C_{ij} =$

$(\Psi^i(G_j), \Psi^j(\Psi^i(G_j))) = (\bar{G}_i, \bar{G}_j)$. Without loss of generality, maximizing C_{ij} consists of seeking objects g_i to be augmented to C_{ij} without violating the required connectivity and self-connectivity requirements:

$$\operatorname{argmax}_{g_i} \rho(G_i \cup g_i, G_j) \quad (62)$$

$$\operatorname{argmax}_{g_i} \sigma(G_i \cup g_i, G_j) \quad (63)$$

Due to the fact, that (G_i, G_j) is a concept, any g_i that is added will introduce at least one zero into the subspace. Therefore an intuitive solution to equations 62 and 63 is g_i that minimizes the number of zeros introduced.

Definition 24. *Given any subspace (G_i, G_j) in \mathbb{K}_{ij} , let $Z(G_i, G_j)$ denote the number of 0s in the subspace.*

$$Z(G_m, G_n) = |G_m| * |G_n| - \phi((G_m, G_n)) \quad (64)$$

Applying definition 24, we formally illustrate that augmenting concepts with objects the minimize Z , in turn maximize ρ and σ .

Proposition 7. *Given a concept (G_i, G_j) then*

$$\begin{aligned} & \operatorname{argmax}_g \rho(G_i \cup g_i, G_j) \\ &= \operatorname{argmin}_g Z(G_i \cup g_i, G_j) \\ &= \operatorname{argmax}_g \sigma(G_i \cup g_i, G_j) \end{aligned}$$

Proof. We will first prove the first equality. Maximizing $\rho(G_i \cup g_i, G_j)$ is equivalent to maximizing $\phi(G_i \cup g_i, G_j)$

$$\phi(G_i \cup g_i, G_j) = |G_i \cup g_i| * |G_j| - Z(G_i \cup g_i, G_j)$$

thus the result clearly follows. Maximizing $\sigma(G_i \cup g_i, G_j)$ is equivalent to maximizing

$$\sum_{k=1}^{|G_i|} \sum_{l=k+1}^{|G_i|+1} |(\Psi^j(g_i^k) \cap \Psi^j(g_i^l) \cap G_j)|$$

Due to the fact that (G_i, G_j) is a concept then, $\Psi^j(g_i^1 \cap G_j) = \Psi^j(g_i^2 \cap G_j), \forall g_i^1, g_i^2 \in G_i$. Hence

$$\begin{aligned} & \sum_{k=1}^{|G_i|} \sum_{l=k+1}^{|G_i|+1} |(\Psi^j(g_i^k) \cap \Psi^j(g_i^l) \cap G_j)| \\ &= \binom{|G_i|-1}{2} * |G_j| + |G_i| * |\Psi^j(g_i) \cap G_j| \end{aligned}$$

Also due to the fact that (G_i, G_j) is a concept $Z(G_i \cup g_i, G_j) = |G_j| - |\Psi^j(g_i) \cap G_j|$. This implies that minimizing $|G_j| - |\Psi^j(g_i) \cap G_j|$ maximizes $|G_i| * |\Psi^j(g_i) \cap G_j|$, hence the result follows. \square

As a result of the equivalence proven in proposition, 7, we state all further results only in terms Z . The basic theorem of FCA and hierarchial order of concepts suggest that for a given concept (G_i, G_j) its neighbors constitute those concepts that introduce the least amount of zeros when taking the union of both concepts.

Proposition 8. *Given concept (G_i, G_j) in \mathbb{K}_{ij} then there exists a concept (\hat{G}_i, \hat{G}_j) s.t. $(\hat{G}_i, \hat{G}_j) \succ (G_i, G_j)$ and $\operatorname{argmin}_g Z(G_i \cup g, G_j) \in (\hat{G}_i, \hat{G}_j)$.*

Proof. For any g , $Z(G_i \cup g, G_j) = |G_j| - |\Psi^j(g) \cap G_j|$. Let $\hat{g} = \operatorname{argmin}_g Z(G_i \cup g, G_j)$. Then, by proposition 1 a concept $(\hat{G}_i, \hat{G}_j) = (\Psi^i(\Psi^j(\hat{g}) \cap G_j), \Psi^j(\hat{g}))$ always exists s.t. $(\hat{G}_i, \hat{G}_j) > (G_i, G_j)$. Assume that $(\hat{G}_i, \hat{G}_j) \neq (G_i, G_j)$. In this case, by definition of the hierarchial order, there exists a concept $(\tilde{G}_i, \tilde{G}_j)$ such that $|\tilde{G}_j \cap G_j| > |\hat{G}_j \cap G_j|$. This implies that $Z(G_i \cup \tilde{g}, G_j) < Z(G_i \cup \hat{g}, G_j)$, where $\tilde{g} \in (\tilde{G}_i, \tilde{G}_j)$ which is a contradiction. \square

Applying proposition 8 to a subspace (G_i, G_j) which maybe decomposed into a set of concepts,

$\Theta((G_i, G_j)), \text{as}$

$$\Theta((G_i, G_j)) = \{(A_i, B_j) | (A_i, B_j) = (\psi^i(\psi^j(g) \cap G_j), \psi^j(g)) \forall g \in G_i\} \quad (65)$$

then $\argmin_g Z(G_i \cup g, G_j)$ is also found by exploring the neighbors of all concepts in $\Theta((G_i, G_j))$.

Theorem 9. *Given subspace (G_i, G_j) in \mathbb{K}_{ij} then w.l.o.g for $\argmin_g Z(G_i \cup g, G_j) \in (\hat{G}_i, \hat{G}_j)$, where $(\hat{G}_i, \hat{G}_j) \succ (A_i, B_j)$ and $(A_i, B_j) \in \Theta((G_i, G_j))$.*

Proof. For any g , $Z(G_i \cup g, G_j) = Z(G_i, G_j) + |G_j| - |\psi^j(g) \cap G_j|$. By proposition 1, there always exists a concept $(\hat{G}_i, \hat{G}_j) > (A_i, B_j)$, where $(A_i, B_j) \in \Theta((G_i, G_j))$. Then also by proposition 8, $(\hat{G}_i, \hat{G}_j) \succ (A_i, B_j)$. \square

As a direct result of theorem 9, TreeClu constructs local clusters by exploring the neighbors of each C_{ij} as follows:

<pre> Input: Candidate C Data: $\mathcal{T}, \alpha, \delta$ 1 begin 2 Decompose C into local concepts $C_{ij} = (G_i, G_j), \forall (G_i, G_j) \in E$; 3 foreach C_{ij} do 4 $PQ \leftarrow \text{Uppers}((G_i, G_j)) \cup \text{Lowers}((G_i, G_j))$; 5 $(G_i^*, G_j^*) \leftarrow \text{concept in } PQ \text{ that minimizes } Z((G_i \cup G_i^*, G_j \cup G_j^*))$; 6 while $\mathbb{S}((G_i \cup G_i^*, G_j \cup G_j^*))$ do 7 $C_{ij} \leftarrow (G_i \cup G_i^*, G_j \cup G_j^*)$; 8 Remove (G_i^*, G_j^*) from PQ; 9 $PQ \leftarrow PQ \cup \text{Uppers}((G_i^*, G_j^*)) \cup \text{Lowers}((G_i^*, G_j^*))$; 10 $(G_i^*, G_j^*) \leftarrow \text{concept in } PQ \text{ that minimizes } Z((G_i \cup G_i^*, G_j \cup G_j^*))$; 11 end </pre>	<p>Procedure LocalRefine</p>
---	-------------------------------------

The procedure initially decomposes the candidate subspace into local concepts, (G_i, G_j) in each context (line 2). For each (G_i, G_j) , a priority queue, PQ , holds both the upper and lower neighbors

of (G_i, G_j) , sorted in increasing order by $Z((G_i \cup \hat{G}_i, G_j \cup \hat{G}_j))$. The upper and lower neighbors of a concept may be computed utilizing one of the methods described in chapter 3 [52, 16]. Line 5 simply entails polling the first concept in PQ . The candidate is refined by continuously adding the objects of every (G_i^*, G_j^*) , until the connectivity and local connectivity have dropped below the expected values (lines 6-10). In order to avoid enumerating duplicate neighboring concepts, a hash-table maybe used to temporarily store all concepts (G_i^*, G_j^*) that have been augmented to the original concept. Furthermore, whenever a new concept is augmented to C_{ij} the priorities of all the elements in PQ must be updated to reflect the new value of $Z((G_i \cup \hat{G}_i, G_j \cup \hat{G}_j))$. At the end of the procedure, the subspace C_{ij} is guaranteed to meet all three conditions of an n -cluster with respect to the two domains \mathbf{G}_i and \mathbf{G}_j .

5.2.3 Network Refinement

Following the local refinement phase, every candidate cluster consists of $|V| - 1$ local clusters. The goal of the network refinement phase is to aggregate all of these local clusterings to form the final information-network cluster. Let $C = ((G_i, G_j)^1, \dots, (G_l, G_m)^{|V|-1})$ be a candidate cluster formed by the `RefineLocal` procedure. For every articulation node \mathbf{G}_a of \mathcal{T} , the candidate cluster contains $|\Gamma(\mathbf{G}_a)|$ object-sets $G_a \subseteq \mathbf{G}_a$; in other words, for each articulation node \mathbf{G}_a , we have $|\Gamma(\mathbf{G}_a)|$ different clusterings in C . Oppositely, for every non-articulation domain (leaf nodes of \mathcal{T}) only a single clustering exists. `TreeClu` utilizes the different local clusterings to incorporate global knowledge into the information network clusterings in two stages. In the first stage, the $|\Gamma(\mathbf{G}_a)|$ clusterings of each articulation node are combined into a single cluster. Next, beginning with the leaf nodes of \mathcal{T} , the second stage of network refinement attempts to add objects to each domain in order to fulfill the maximality criterion.

Stage 1 of network refinement is displayed as `NetworkRefine1`. The information network cluster, $\mathbb{C} = (G_1^*, \dots, G_{|V|}^*)$, is initially formed by assigning the leaf nodes to their respective local clusterings, while each articulation node, G_c^* is assembled as the union of all $|\Gamma(\mathbf{G}_c)|$ local clusterings of G_c (for loops on lines 3 and 5). If \mathbb{C} meets the connectivity requirements, then stage 1

is complete. If this is not the case, then objects in the articulation nodes that are least connected and whose removal improves ρ and σ are removed incrementally until \mathbb{C} meets the requirements or the object-set G_c contains no more objects (lines 7-12). After completing, this refinement, if the \mathbb{C} still does not meet the connectivity requirements, then the local clusterings are too disjoint and the candidate is disregarded.

	<p>Input: Candidate $C = ((G_i, G_j)^1, \dots, (G_k, G_l)^{ V -1})$</p> <p>Data: $\mathcal{T}, \alpha, \delta$</p> <pre> 1 begin 2 $\mathbb{C} \leftarrow (G_1^*, \dots, G_{ V }^*) \leftarrow \emptyset$; 3 foreach <i>Leaf node</i> $\mathbf{G}_l \in V$ do 4 $G_l^* \leftarrow G_l \in C$; 5 foreach <i>Articulation node</i> $\mathbf{G}_c \in V$ do 6 $G_c^* \leftarrow \bigcup_{G_c \in C} G_c$; 7 if not $\mathbb{S}(\mathbb{C})$ then 8 repeat 9 foreach <i>Articulation node</i> $\mathbf{G}_c \in V$ do 10 $g_c \leftarrow$ least connected object in G_c; 11 $\mathbb{C} \leftarrow (G_1^*, \dots, G_c^* \setminus g, \dots, G_{ V }^*)$; 12 until $\mathbb{S}(\mathbb{C})$ or $G_c < 1$; 13 if not $\mathbb{S}(C)$ then 14 $\mathbb{C} \leftarrow \emptyset$; 15 return \mathbb{C}; 16 end </pre>
--	--

Function NetworkRefine1

Following stage 1, if $\mathbb{C} \neq \emptyset$, then the cluster is guaranteed to meet conditions 1 and 2 and be maximal with respect to all articulation nodes. However, maximality is not guaranteed for objects-set $G_l \in \mathbb{C}$, where \mathbf{G}_l is a leaf node domain of \mathcal{T} . This is due to the fact any additional objects added to neighboring object-sets G_c may allow for more objects from \mathbf{G}_l to be added to maximize G_l . Furthermore, if more objects are added to G_l this may allow for more objects from the neighboring articulation node to be added as well. Thus, stage 2 employs an iterative procedure by which

objects are attempted to be added to the leaf node object-sets to achieve maximality. If no maximization is possible, then \mathbb{C} is clearly a cluster; however if maximization is possible, then the process must be repeated for the articulation nodes.

	<p>Input: Candidate $C = ((G_i, G_j)^1, \dots, (G_k, G_l)^{ V -1}), \mathbb{C} = (G_1^*, \dots, G_{ V }^*)$</p> <p>Data: $\mathcal{T}, \alpha, \delta$</p> <pre> 1 begin 2 $X \leftarrow (X_1, \dots, X_{ V }) \leftarrow \emptyset$; 3 repeat 4 foreach <i>Leaf node</i> $\mathbf{G}_l \in V$ do 5 foreach g_c <i>added to local cluster</i> (G_l, G_c) do 6 $X_l \leftarrow X_l \cup \psi^l(g_c) \setminus G_l^*$; 7 foreach $g_l \in X_l$ do 8 if $\rho(G_l^* \cup g_l) > \rho(G^*) \wedge$ then 9 $\mathbb{C} \leftarrow (G_1^*, \dots, G_l^* \cup g, \dots, G_{ V }^*)$; 10 foreach <i>Articulation node</i> $\mathbf{G}_c \in V$ do 11 foreach g_l <i>added to</i> G_l^* do 12 $X_c \leftarrow X_c \cup \psi^c(g_l) \setminus G_c^*$; 13 foreach $g_c \in X_c$ do 14 if $(G_1^*, \dots, G_c^* \cup g_c, \dots, G_{ V }^*)$ <i>improves connectivity</i> then 15 $\mathbb{C} \leftarrow (G_1^*, \dots, G_c^* \cup g, \dots, G_{ V }^*)$; 16 until <i>no additions possible</i>; 17 return \mathbb{C} <i>as n-cluster</i>; 18 end </pre>
--	--

Function NetworkRefine2

Clearly, not all objects of a domain need to be tested when attempting to maximize an object-set. For every leaf node object-set, G_l let G_c be the neighboring articulation node object-set. When maximizing the leaf node object-sets, only those objects $g_c^* \in G_c^*$ that did not appear in the original local cluster, (G_l, G_c) may induce the addition of new objects from G_l . Hence, lines 5-6 of NetworkRefine2() locates all objects in $g_l \in \mathbf{G}_l$ that are connected to all such g_c^* , while the following for loop attempts to add these objects to \mathbb{C} . If any such addition is possible in any leaf node, then the process must repeated for all the articulation nodes (lines 10-16) until no addition

is possible. The entire process is then repeated until no additions are possible, hence ensuring maximality.

Theorem 10. *Every subspace \mathbb{C} returned by `NetworkRefine2()` is an n -cluster.*

5.2.4 Analysis

Theorem 10 ensures that every subspace `TreeClu` mines is indeed an n -cluster. While no guarantee exists that `TreeClu` discovers all n -clusters, `TreeClu` attempts to form an n -cluster with every object in \mathcal{T} . Experimental results suggest the effectiveness of this strategy, while the running time of `TreeClu` may be improved by intelligently selecting only a fraction of the objects. In the sequel we analyze the computational cost of each phase of `TreeClu` in terms of a single attempt to form an n -cluster with a given object $g \in V$. Let \mathbf{G} denote the domain with the largest cardinality.

Candidate Clusters

Along each edge of \mathcal{T} , `GenerateCandidate` performs at most a single application of the ψ operator. This entails at most $O(|V| * |\mathbf{G}|)$ set intersections.

Local refinement

In addition to computing upper and lower neighbors, the major computational tasks of the `LocalRefine` procedure is computing the number of zeros in a given subspace, and computing ρ , σ , and $E[\rho]$. Utilizing an incremental concept generation algorithm such as [16], each neighbor of a given concept (G_i, G_j) maybe generated in $O(|G_i| * |G_j|)$ time. The fact that the initial candidate is a concept, and only concepts are augmented to form the local cluster allows for efficient computation of $Z(G_i \cup G_i^*, G_j \cup G_j^*)$. The maximum number of zeros that may be introduced when augmenting an upper neighbor is

$$|G_i^* \setminus G_i| * |G_j \setminus \psi^j(G_i^*)| \quad (66)$$

Hence computing $Z(G_i \cup G_i^*, G_j \cup G_j^*)$ is done in amortized constant time when enumerating neighbors (since the cost of computing these differences is incorporated into the cost of enumerating neighbors). The computation of ρ , σ , and $E[\rho]$ also benefits from the fact that local clusters are

only constructed with concepts. Analysis of these computations is addressed in a more general form in the next section.

Network refinement

The majority of computation of in `NetworkRefine1()` and `NetworkRefine2()` involves computing ρ , σ , and $E[\rho]$ for each object-set in $(G_1^*, \dots, G_c^* \setminus g, \dots, G_{|V|}^*)$ prior to removing or adding an object g for a particular object-set G_c . Given that ψ , σ , and $E[\psi]$ of $(G_1^*, \dots, G_c^*, \dots, G_{|V|}^*)$ are pre-computed, efficient computation of the new quantities is possible. For simplicity we ignore normalization terms. A node G_c undergoing refinement in `NetworkRefine1` involves removing an object g at each iteration:

$$\begin{aligned}\phi(G_c \setminus g) &= \sum_{G_j \in \Gamma(G_c)} \phi(G_c, G_j) - \phi(g, G_j) \\ &= \phi(G_c) - \phi(g) \\ E[\phi(G_c \setminus g)] &= \sum_{G_j \in \Gamma(G_c)} E[\phi(G_c, G_j)] - E[\phi(g, G_j)] \\ &= E[\phi(G_c)] - E[\phi(g)]\end{aligned}$$

The above simplifications reveal that $\phi(G_c \setminus g)$ and $E[\phi(G_c \setminus g)]$ may be compute utilizing only a single set intersection to compute $\phi(g)$. Furthermore, due to symmetry, the connectivity of all neighboring nodes need not be computed. On the other hand, $E[\rho(G_i)]$ for each neighboring G_i of G_c is computed as

$$\begin{aligned}E[\phi(G_i)] &= \sum_{G_j \in \Gamma(G_i)} E[\phi(G_j)] \\ &= \sum_{G_j \in \Gamma(G_i) \setminus G_c} E[\phi(G_j)] + \frac{|G_c| - 1}{|\mathbf{G_c}|} \sum_{g_i \in G_i} |\psi^c(g_i)| \\ &= \sum_{G_j \in \Gamma(G_i, G_j) \setminus G_c} E[\phi(G_j)] + \frac{|G_c|}{|\mathbf{G_c}|} \sum_{g_i \in G_i} |\psi^c(g_i)| - \frac{\sum_{g_i \in G_i} |\psi^c(g_i)|}{|\mathbf{G_c}|} \\ &= \sum_{G_j \in \Gamma(G_i, G_j)} E[\phi(G_j)] - \frac{E[\phi(G_i, G_c)]}{|G_c|}\end{aligned}$$

Hence, in order to efficiently compute $E[\phi(G_i)]$ with respect to $G_c \setminus g$, we need to store the individual components of $E[\phi(G_i)]$ with respect to G_c . With these stored components, then the computation only requires amortized constant time per removed object. Calculating self-connectivity efficiently is also possible with similar simplifications:

$$\begin{aligned}\sigma(G_c \setminus g) &= \sum_{k=1}^{|G_c|-1} \sum_{l=k+1}^{|G_c|} \sum_{G_j \in C \setminus G_c} |\phi^j(g_c^k) \cap \phi^j(g_c^l) \cap G_j| - \sum_{g_i^k \in G_i \setminus g} \sum_{G_j \in C \setminus G_c} |\phi^j(g_c^k) \cap \phi^j(g) \cap G_j| \\ &= \sigma(G_c) - \sum_{g_i^k \in G_i \setminus g} \sum_{G_j \in C \setminus G_c} |\phi^j(g_c^k) \cap \phi^j(g) \cap G_j|\end{aligned}$$

Computing $\sigma(G_c \setminus g)$ and $\sigma(G_i \setminus g)$ then requires $O(|G_c|^2 * |V|)$ operations. Finally, computing self-connectivity for each all other $G_i \neq G_c$ after removing g is accomplished as

$$\begin{aligned}\sigma(G_i) &= \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} \sum_{G_j \in C \setminus G_i} |\phi^j(g_i^k) \cap \phi^j(g_i^l) \cap G_j| - \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} |(\phi^j(g_i^k) \cap \phi^j(g_i^l) \cap G_c) \setminus g| \\ &= \sum_{k=1}^{|G_i|-1} \sum_{l=k+1}^{|G_i|} \sum_{G_j \in C \setminus G_i} |\phi^j(g_i^k) \cap \phi^j(g_i^l) \cap G_j| - \sum_{g_i^1 \in \phi^i(g) \cap G_i} \sum_{g_i^2 \in (\phi^i(g) \cap G_i) \setminus g_i^1} 1 \\ &= \sigma(G_i) - \binom{|\phi^i(g) \cap G_i|}{2}\end{aligned}$$

Hence the computational cost of computing σ for all other object-sets is accomplished in amortized constant time given that the previous values of self-connectivity are stored.

Total Cost

The cost of each phase is summarized as

Phase	Stage	Complexity
Candidate Generation		$O(\mathbf{G} * V)$
Local Refinement	Decompose to concepts	$O(\mathbf{G} * V)$
	Neighbors	$O(\mathbf{G} * V * r)$
	Compute connectivity for each object-set	$O(V ^2 * \mathbf{G} ^2)$
Network Refinement	Stage 1:	$O(r_2 * \mathbf{G} ^2 * V ^2)$
	Stage 2:	$O(\mathbf{G} ^2 * V ^2 * r_3)$

r_i represents the number of refinement steps at each phase or stage. Experimental results indicate that the value r_i depends on the parameters α and δ , but in general we may regard these as constants. Aggregating all phases, and applying each phase to each object $g \in V$ the worst case computational complexity of `TreeClu` is $O(|\mathbf{G}|^3 * |V|^2)$.

5.3 Experimental Results

In this section we demonstrate the effectiveness of the `TreeClu` algorithm on real-world information networks. Clusterings performance was tested against the state-of-the-art algorithms `MDC` and `NetClus`. Experimental studies on the effects of parameter selection on both clustering effectiveness and run-times for the algorithm are also presented.

5.3.1 Real world datasets

Eight real-world information networks were utilized in the experimental study. Six of the datasets were derived from the 20NewsGroups dataset [7], one from the bioinformatics field and finally a subset of the DBLP was utilized. All information networks are summarized in figure 26. Notice, that the NewsGroups information networks are derived from the data source as the datasets in chapter 4; however, we did not randomly partition the data as was done in that chapter. Instead, we kept the domains as they naturally occur. The FOURAREAS information network, was utilized in [74] and is a subset of the DBLP dataset containing the domains of papers, authors, terms, and conference. Seven of the networks contain class labels: Newsgroup datasets use `usenet`









Name	Network	Domains	Size
MER		mideast and religion docs SubjectLines AllWords	2000 892 21891
AIIPC		pc docs SubjectLines AllWords	5000 2805 27381
AIISCI		science docs SubjectLines AllWords	3999 2074 31340
AIIREC		recreation docs SubjectLines AllWords	3124 1671 21430
PCR		politics docs SubjectLines AllWords	1997 1025 21463
FOURAREAS		Papers Conferences Terms Authors	28569 20 13226 28702
GENES		Genes Pheno types GO terms Micro Rna	1910 3965 3885 318
MERSPLIT		MidEast Docs MidEast KeyWords Religion Docs Religion KeyWords AllWords	1000 472 1000 420 21891

Figure 26: Real-world information networks utilized in empirical evaluation of TreeClu

Parameters	Algorithm	F_1	$F_{0.5}$	F_2
$\alpha = 2.5$	TreeClu	0.242041	0.191622	0.328465
$k = 4$	NetClus	0.225488	0.206883	0.24777
$k = 4$	MDC	0.228272	0.211674	0.247694
$\alpha = 2.75$	TreeClu	0.24411	0.199462	0.314509
$k = 8$	NetClus	0.164438	0.183863	0.148726
$k = 8$	MDC	0.173128	0.195462	0.155374
$\alpha = 3.0$	TreeClu	0.283306	0.219294	0.400094
$k = 16$	NetClus	0.108283	0.148992	0.0850456
$k = 16$	MDC	0.111501	0.157455	0.0863106
$\alpha = 3.25$	TreeClu	0.248256	0.20347	0.318322
$k = 32$	NetClus	0.0731957	0.11887	0.0528781
$k = 32$	MDC	0.0736255	0.125243	0.0521377

(a) ALLPC

Algorithm	F_1	$F_{0.5}$	F_2
TreeClu	0.341985	0.291558	0.413503
NetClus	0.278475	0.287006	0.270437
MDC	0.284599	0.295308	0.274639
TreeClu	0.343541	0.31276	0.381041
NetClus	0.191135	0.241016	0.15836
MDC	0.196604	0.252231	0.161079
TreeClu	0.332871	0.315443	0.352338
NetClus	0.128472	0.193623	0.0961263
MDC	0.123719	0.194651	0.0906756
TreeClu	0.322396	0.314416	0.330792
NetClus	0.0920745	0.157101	0.0651201
MDC	0.0757684	0.141425	0.0517455

(b) ALLREC

Algorithm	F_1	$F_{0.5}$	F_2
TreeClu	0.282161	0.235688	0.351463
NetClus	0.260942	0.25836	0.263575
MDC	0.253242	0.252911	0.253573
TreeClu	0.305063	0.26207	0.364931
NetClus	0.189321	0.224789	0.16352
MDC	0.20547	0.252432	0.17324
TreeClu	0.296717	0.260205	0.345149
NetClus	0.139972	0.193058	0.109784
MDC	0.137501	0.208865	0.102484
TreeClu	0.309464	0.282393	0.342275
NetClus	0.109269	0.168405	0.0808707
MDC	0.0898783	0.164373	0.0618484

(c) ALLSCI

Algorithm	F_1	$F_{0.5}$	F_2
TreeClu	0.447209	0.381246	0.540775
NetClus	0.375971	0.451266	0.322209
MDC	0.366142	0.453309	0.307092
TreeClu	0.45363	0.418364	0.495389
NetClus	0.284878	0.393568	0.22323
MDC	0.245008	0.376406	0.181611
TreeClu	0.460672	0.435264	0.48923
NetClus	0.233927	0.350689	0.175496
MDC	0.153421	0.278065	0.10593
TreeClu	0.448976	0.454443	0.443639
NetClus	0.200725	0.321033	0.146008
MDC	0.114177	0.231198	0.075807

(d) MER

Algorithm	F_1	$F_{0.5}$	F_2
TreeClu	0.466381	0.422473	0.520474
NetClus	0.364267	0.439669	0.310942
MDC	0.344025	0.426891	0.288101
TreeClu	0.438335	0.414682	0.46485
NetClus	0.266409	0.37266	0.207304
MDC	0.218089	0.337279	0.161143
TreeClu	0.414013	0.457857	0.377832
NetClus	0.216861	0.331496	0.161137
MDC	0.14463	0.26357	0.0996577
TreeClu	0.413474	0.460364	0.375253
NetClus	0.177318	0.292182	0.127281
MDC	0.100135	0.203137	0.066444

(e) PCR

Algorithm	F_1	$F_{0.5}$	F_2
TreeClu	0.531512	0.506687	0.55889
NetClus	0.36124	0.366555	0.356076
MDC	0.508535	0.516275	0.501023
TreeClu	0.534896	0.544086	0.526011
NetClus	0.317618	0.356658	0.286281
MDC	0.416542	0.5222	0.346445
TreeClu	0.540569	0.507751	0.577922
NetClus	0.239803	0.306856	0.1968
MDC	0.364935	0.543009	0.274813
TreeClu	0.480845	0.42804	0.548512
NetClus	0.206759	0.377445	0.142375
MDC	0.233105	0.425454	0.160529

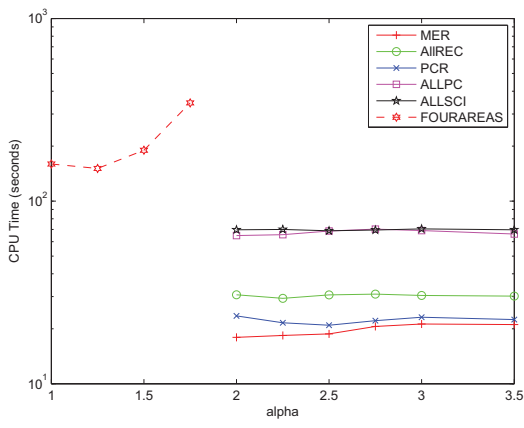
(f) FourAreas

Figure 27: Clustering results with TreeClu algorithm

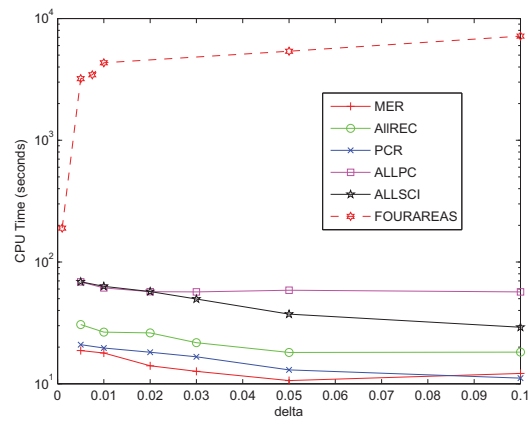
group of a document as a class label, while the FourAreas dataset labels papers according to their research area as data mining, machine learning, database, or information retrieval. Moreover, the newsgroup documents may be assigned non-overlapping labels (each document belongs to a single usenet group) or multiple class labels (such as documents belonging to both electronics and

PC newsgroups). Once again, to test the efficacy of the algorithm we utilized the *BCubePrec* and *BCubeRcl* extrinsic clustering validity measures [26]. The bioinformatics network does not contain class labels, however, we illustrate that the clusters are mined unveil meaningful associations and clusters that were validated by domain experts. For the labeled data, we compared the clusters produced by *TreeClu* with those enumerated by the recently proposed information network clustering algorithm, *NetClus* [74]; this algorithm was shown to outperform the *RankClus* [73] and the *PLSA* algorithm [84]. Additionally, *TreeClu* was compared to the state-of-the-art multi-way clustering algorithm *MDS* [14].

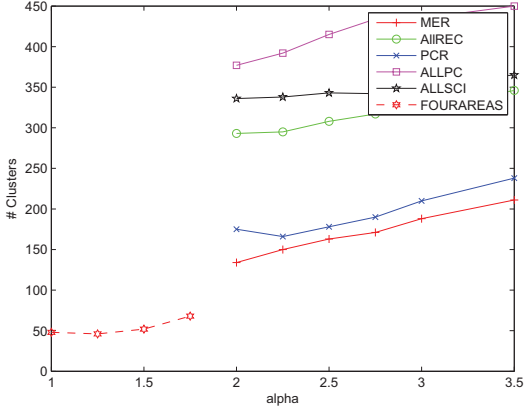
Both *NetClus* and *MDC* require the K parameter to indicate the number of desired clusters, while the number of clusters cannot be explicitly controlled with *TreeClu* as this is determined by the data. Additionally, α was set to 2 for all experiments, while δ was varied in attempts to create an equal number of clusters as *NetClus* and *MDC*. On each run of *TreeClu* we randomized the order in which objects are presented to the algorithm. The results of all experiments are displayed in figure 27; these represent the best results for all algorithms with the selected parameters and 20 runs. For *NetClus* the cluster label is obtained according to the largest posterior probability. As can be seen, *TreeClu* consistency produces higher quality clusters in every single information network. We attribute this result to the fine-grain approach of *TreeClu* and the balancing of locally similar objects (neighboring concepts) versus the global effect of adding or removing objects from the clusters. For example, we observed that the clusters mined by *TreeClu* in the *FourAreas* dataset were extremely accurate, with ≥ 0.9 accuracy (although this is not reflected by the B^3 score because more than 4 clusters were enumerated) mainly due to the paper-conference context. During the local refinement stage each candidate was infused with all other papers that co-occur in a conference; thus creating accurate clusters in terms of the papers domain. On the other hand, this caused low connectivity and associativity in the other domains, thus the network refinement phase counter-acted this, eventually leading to high quality clusters.



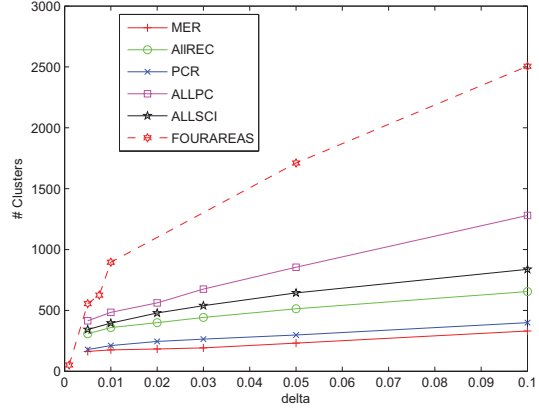
(a) Running time $\delta = 0.01$



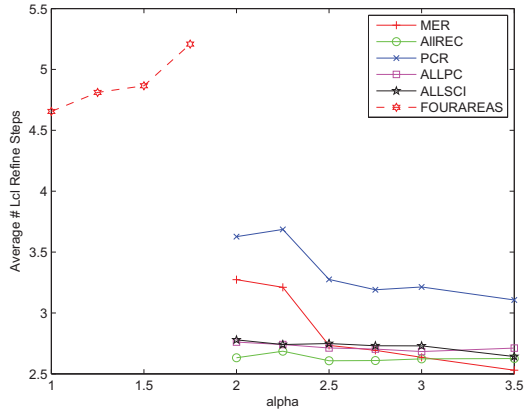
(b) Running time $\alpha = 2.5$



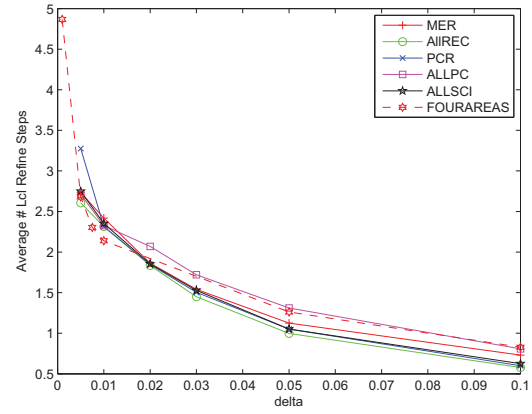
(c) Num. Clusters $\delta = 0.01$



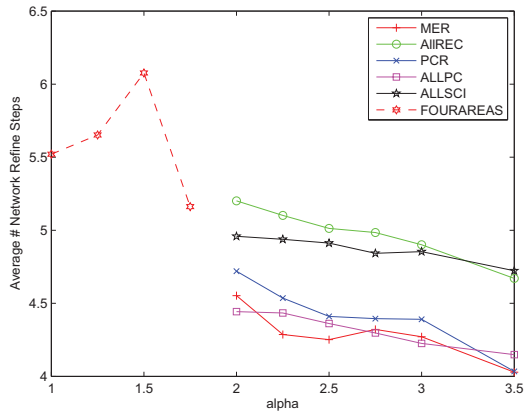
(d) Num. Clusters $\alpha = 2.5$



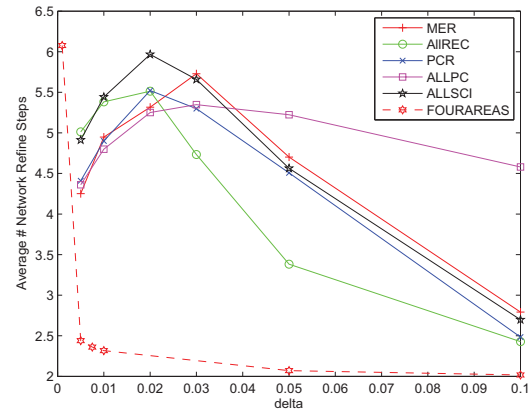
(e) Local refine steps $\delta = 0.01$



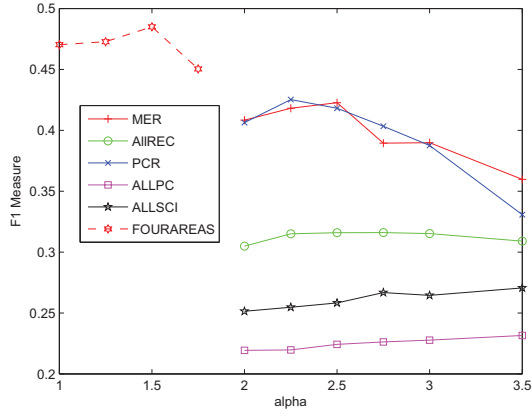
(f) Local refine steps $\alpha = 2.5$



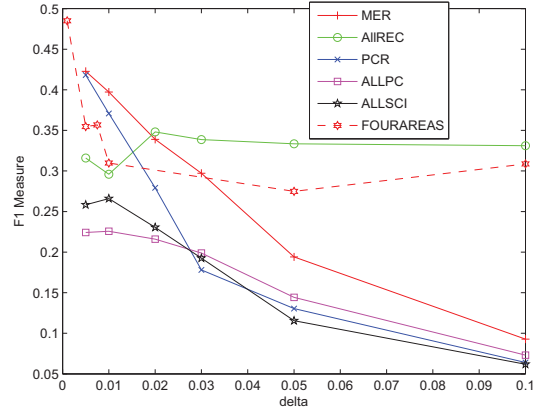
(g) Network refine steps $\delta = 0.01$



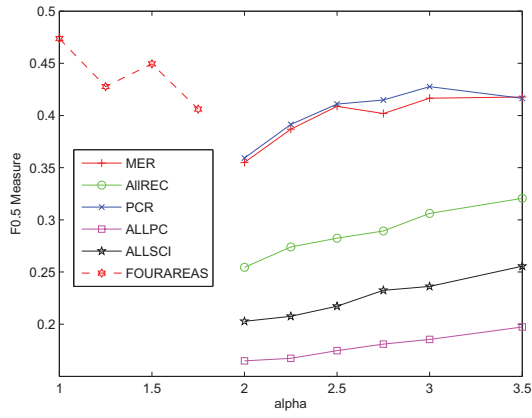
(h) Network refine steps $\alpha = 2.5$



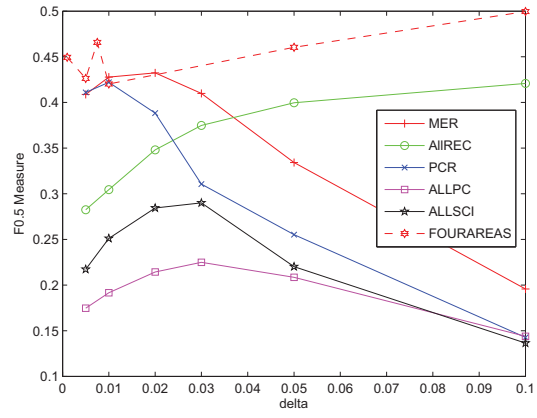
(a) F_1 Scores $\delta = 0.01$



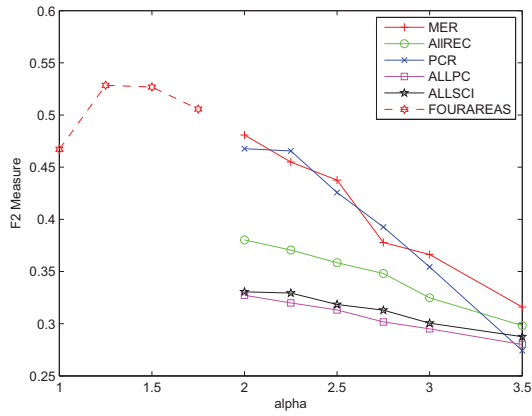
(b) F_1 Scores $\alpha = 2.5$



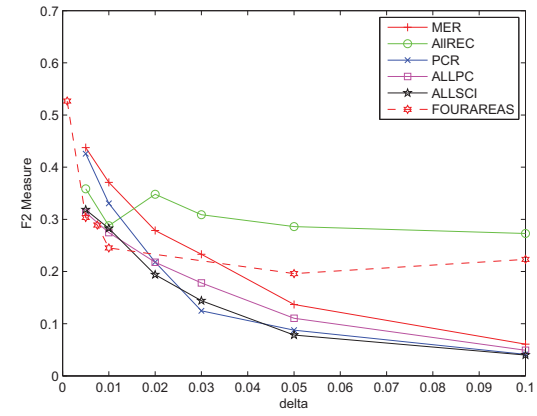
(c) $F_{0.5}$ Scores $\delta = 0.01$



(d) $F_{0.5}$ Scores $\alpha = 2.5$



(e) F_2 Scores $\delta = 0.01$



(f) F_2 Scores $\alpha = 2.5$

Figure 29: Parameter study on TreeClu clustering quality

5.3.2 Parameter Study and Sample clusters

The parameters that govern the nature of the clusters that `TreeClu` enumerates are α and δ . Experiments were performed to study the effects of these parameters. Two sets of tests were performed: first, α was held at a constant value of 2 while δ was varied, next α was varied while δ was held constant at 0.01. Low values of δ are utilized due to the extreme sparsity of the information networks.

The impact of parameter selection on the computational cost of `TreeClu` was studied initially. We expect higher values of α and δ to allow more clusters to form, as less objects will be included in each cluster during formation. This trend is clear in figures 28(c) and 28(d). Noticeably, the number of clusters is much more sensitive to variations in δ . Surprisingly, for the majority of the information networks larger cluster numbers did not translate into longer running times, with the clear exception of `FOURAREAS` (figures 28(a) and 28(b)). While not intuitive, this result is justified by the fact that as the connectivity criterion become stricter, fewer refinement steps are required at each stage (as seen in figures 28(e)-28(h)) hence avoiding the costly computations associated with these stages. On the other hand, the nature and scale of the `FOURAREAS` network enabled the enumeration of orders of magnitude more clusters as the connectivity requirements became stricter. The massive number of clusters outweighed the benefit fewer refinement steps at each phase resulting in explosive growth of the computation cost. This result reflects the worst case cubic growth rate of `TreeClu` and suggests that future work must focus on the scalability of the algorithm. Not surprisingly, a recurring trend throughout all experiments was the much greater sensitivity of `TreeClu` to δ parameter. This trend is a result of the sparsity of the datasets and the much stricter criterion self-connectivity imposes on clusters as opposed to mere connectivity.

The effect of parameter selection on clustering quality was also studied. Intuition suggests that stricter connectivity requirements imply more accurate and precise clusters as this forces `TreeClu` to only mine clusters that are highly connected. On the other hand, stricter criterion may leave several objects un-clustered, as `TreeClu` does contain a mechanism enforcing cluster membership on objects. This expected trend is clearly depicted in figure 29, as large values of α and δ lead to

lower F_1 scores due to significantly lower recall. This notion is reinforced by figures 29(e) and 29(f) where we see the F_2 score decline significantly as stricter connectivity criterion is enforced. The $F_{0.5}$ values, however, exhibit mixed behavior as this scores weights precision twice as much as recall; hence, stricter connectivity requirements may lead to better results due to increase precision. Clearly, parameter selection clearly plays a significant role in the performance of TreeClu and must be considered for each individual dataset. On the positive side, the performance is not overly sensitive to the value of α as the expected connectivity measure is always data driven. On the other hand experimentation and knowledge of the information network is required to set ideal values of δ .

Sample Clusters

Figure 30 displays several clusters mined by TreeClu from the various information networks. In all cases not the entire cluster is displayed but only the objects with greatest connectivity. Common knowledge and domain expertise validate all the clusters displayed: The MER cluster exhibits keywords relating to the primary topic of discussion in Middle East forums: the Israeli-Palestinian conflict. The religious issues cluster clearly contains the most commonly used words in Christianity while politics cluster unveiled posts dealing with global warming issues. Next, figures 30(e) - 30(h) display clusters mined from FOURAREAS; each cluster represents one of the four fields of data mining, information retrieval, machine learning, and databases. The terms clearly correspond to the respective fields as do the conferences. In addition the implicit rankings of the authors truly reflects leaders in each of the respective fields. Of interesting note is the fact that both the data mining and information retrieval clusters both contained conferences that are not explicitly related to the field; this reflects the fact that both these fields draw upon the more general fields of databases of and machine learning. For example, the VLDB conference is by assumption a database conference, however, due to the fact that it deals with large databases, data mining applications and papers are almost always presented. Finally, figure 30(i) displays a cluster from the GENE information network. The validity of this cluster has been verified by domain field experts.

Keywords	All words
proposal	israel
talks	israeli
peace	israelis
justice	arabs
opportunity	palestinian
land	goodwill
negotiations	return
promise	peace

(a) MER middle east word cluster

Keywords	All words
god	jesus
laws	sin
loving	christ
creation	damnation

(b) MER religion word cluster

Keywords	All words
scientists	economic
earth	government
green	climate
house	research
pollutes	temperature

(c) PCR politics cluster

Keywords	All words
mormon	biblical
eternal	angels
marriage	ressurrection
intolerance	jesus

(d) PCR religion cluster

Terms	Authors	Conferences
data	Jiawei Han	VLDB
mining	Philip S. Yu	PAKDD
clustering	Christos Faloutsos	KDD
management	Rakesh Agrawal	
base	Jian Pei	
incremental	Wei Wang	
environment	David J. DeWitt	
warehousing	Charu C. Aggarwal	
	Michael J. Franklin	

(e) FOURAREAS data mining cluster

Terms	Authors	Conferences
retrieval	W. Bruce Croft	SIGIR
information	ChengXiang Zhai	CIKM
model	Clement T. Yu	ECIR
systems	Ophir Frieder	CVPR
automatic	Jian-Yun Nie	
image	Norbert Fuhr	
abstract	Nicholas J. Belkin	
content	Justin Zobel	
processing	Abdur Chowdhury	
databases	James Allan	

(f) FOURAREAS information retrieval cluster

Terms	Authors	Conferences
image	Thomas S. Huang	CVPR
recognition	Takeo Kanade	AAAI
large	Changshui Zhang	ECML
scale	Shree K. Nayar	
view	Dan Roth	
scenes	Rong Jin	
position	Huan Liu	
arbitrary	Xiaoou Tang	
direction	C. Lee Giles	
rendering	Pietro Perona	

(g) FOURAREAS machine learning cluster

Terms	Authors	Conferences
queries	Divesh Srivastava	ICDE
relational	Nick Koudas	VLDB
spatial	Surajit Chaudhuri	
design	Divyakant Agrawal	
dbms	Jeffrey F. Naughton	
top	Amr El Abbadi	
abstract	Nikos Mamoulis	
user	Jiawei Han	
structures	Rakesh Agrawal	

(h) FOURAREAS databases cluster

Genes	MiroRna	PhenoTypes	GO Terms
APOE	hsa-miR-337	abnormal behavior	nervous system development
FYN	hsa-miR-107	abnormal nervous system physiology	multicellular organismal process
PSEN1	hsa-miR-520a	abnormal body size	multicellular organismal development
APP	hsa-miR-103	decreased body size	developmental process
FZD9	hsa-miR-324-5p	abnormal postnatal growth/weight/body size	cellular process
DLG4	hsa-miR-141	abnormal learning/memory/conditioning	system development
PPP1R9B	hsa-miR-525	decreased body weight	cellular developmental process
APBA1	hsa-miR-185	abnormal body weight	cell differentiation
S100B	hsa-miR-373	abnormal brain morphology	cell development
	hsa-miR-96	abnormal telencephalon morphology	cell communication
	hsa-miR-23a	abnormal forebrain morphology	anatomical structure development
	hsa-miR-326	abnormal nervous system morphology	primary metabolic process
	hsa-miR-506	abnormal CNS synaptic transmission	cellular component organization and biogenesis
	hsa-miR-516-3p	abnormal temporal lobe morphology	response to stimulus
	hsa-miR-124a	abnormal neuron morphology	macromolecule metabolic process
	hsa-miR-192	abnormal limbic system morphology	metabolic process
	hsa-miR-200a	abnormal learning/ memory	protein metabolic process
	hsa-miR-369-3p	abnormal hippocampus morphology	neurogenesis
	hsa-miR-24	abnormal synaptic transmission	learning and/or memory
	hsa-miR-153	abnormal spatial learning	generation of neurons

(i) GENE cluster

5.4 Conclusion

In this chapter, we presented a novel information network clustering formulation and the `TreeClu` algorithm to mine information network clusters in information trees. In addition to extending n -concepts beyond star-shaped networks, the clustering formulation addressed shortcomings of n -concepts by relaxing the strict clustering criterion. The connectivity measure and self-connectivity measure were established to capture and quantify the intuition that information network clusters should exhibit a local clustering tendency, while exhibiting global connectivity. Utilizing these measures, n -clusters are defined as maximal subspaces that exceed the expected connectivity and self-connectivity.

The `TreeClu` algorithms mines n -clusters in three phases. First, the fact that n -concepts always exceed the expected connectivity and self-connectivity is exploited to enumerated initial candidate clusters. Next, the candidate clusters are mapped to concepts in each domain, and the number of objects in each domain is maximized by incorporating only local clustering tendency via neighboring concepts. Finally, the network refinement phase aggregates all the local clusterings, and iteratively adds and removes objects until an n -cluster is formed.

Experimental comparisons with the current state-of-the-art algorithms `MDC` and `NetClus` revealed that `TreeClu` produced superior clustering results in real-world data with both overlapping and non-overlapping natural categories. The worst-case running time of `TreeClu` maybe quite expensive at $O(|\mathbf{G}^3|)$, however future work may reduce this cost by incorporating efficient sampling methods. In the next chapter, we illustrate how information network clustering may be reduced to a game theory problem, in which information-network clusters constitute the Nash equilibria points of a game. Utilizing this reduction, a generic algorithm for locating Nash equilibria in a game maybe utilized to mine clusters utilizing different clustering formulations. A variant of the `TreeClu` algorithm is derived utilizing this reduction.

6 Information-Network Clustering and Game Theory

Two definitions of information network clusters have been presented, each with specific advantages and disadvantages. Moreover, the `NClu` and `TreeClu` algorithms relied heavily upon the network topology and cluster definition to mine information network clusters. The aim of this chapter is to develop a general framework for defining and enumerating information network clusters independent of the specific clustering criterion or objective function. In the sequel, we show how differing bi-clustering criteria can be defined in terms of the Nash equilibria of a two-player game. The advantage of this approach is that the specific requirements of applications may be imposed through the reward function (or adding rules to the game), but the definition of a cluster is unchanged and hence the method of enumerating the clusters is unaltered. Moreover, this idea naturally extends to information network clusters by considering Nash equilibria of multi-player games. Selecting differing game rules and reward functions we illustrate how the n -concept and n -cluster definitions are in fact Nash equilibria points of specific multi-player games.

The premise of the framework is that an information network cluster constitutes an equilibrium among local clusterings of each domain. This idea was inspired by the observation in [22] (expanded upon in [4] and chapter 4 to star shaped networks), that a subspace cluster constitutes a trade-off between the number of data points and attributes admitted. The more objects a subspace cluster encompasses, the fewer attributes it will admit, and vice-versa. This observation naturally extends to a relational setting, for example in figure 31, the more *CS_Authors* included in a cluster implies fewer *Math_Authors* that they jointly collaborate with and fewer research areas they are collectively pursuing. Hence, a cluster of the entire information network can be viewed as an equilibrium point between competing clustering influences on the domains. This conception is captured precisely as the Nash equilibrium solution concept of a game involving two or more players. A game is in Nash equilibrium if each player has chosen a strategy and no player can benefit by changing his or her strategy while the other players keep theirs unchanged.

In the sequel, we formally introduce a framework for clustering in information networks by modeling the problem as a game and defining the clusters as the Nash equilibria of that game.

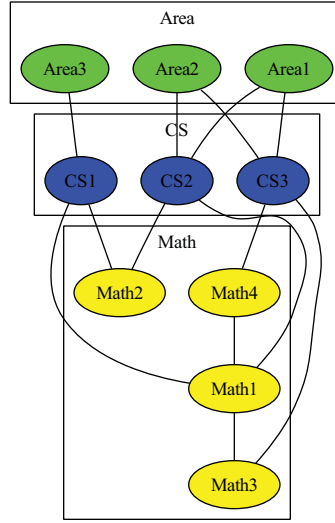


Figure 31: Sample information network

First, a framework for single edge information network clustering (bi-clustering) is established and we show that this approach generalizes the FCA definition of a bi-cluster. Focusing on a specific reward function, the `NashClu` algorithm is developed to enumerate the bi-clusters. Next, the framework is extended to a general information network, and the `NetNashClu` algorithm is presented as a generalized version of `NashClu`. Moreover, it is illustrated, that by slightly altering the reward function, `NetNashClu` is reduced to the `TreeClu` algorithm presented in chapter 5. Finally, experimental results indicate the efficacy of the game theoretic framework utilizing different reward functions.

6.1 Game Theory

In this section, the basic conceptions of game theory are introduced. Applying these conceptions, the clustering problem is formulated as a game. Every game has two or more players. A game consists of a set of positions that players can move to based on the rules. The situation at the start of the game is called the **initial** position. At each position, the rules indicate which player (or players) makes a move from that position and what are the allowable moves from that position to other positions. For every position p there must be a sequence moves from the initial position

to p . Some positions are designated as **terminal** positions, and no moves are allowed from such a position, so that the game ends when such a position is reached. A play of the game consists of a sequence of moves starting at the initial position and ending at a terminal position. Every terminal position determines a reward or pay-off to each of the players. A central notion of game theory is **strategy**. A strategy for a player is a specification that tells the player what to do in every situation that might arise during a game. Notice that once a strategy is chosen, the player's moves are completely determined.

Definition 25. A finite, n -player, normal form game $\mathbb{G} = \langle N, (M_i), (r_i) \rangle$ where

- $N = \{1, \dots, n\}$ is the set of players
- $M_i = \{m_i^1, \dots, m_i^{l_i}\}$ is the set of moves available to player i and l_i is the number of available moves for that player. We will use m_i to denote a particular move m_i^j of player i , and $m = (m_1, \dots, m_n)$ denotes a profile of moves, one for each player.
- $r_i : M_1 \times \dots \times M_n \rightarrow \mathfrak{R}$ is the reward function for each player i . It maps a profile of moves to a value.

Each player i selects a strategy from the set of all available strategies: $\mathcal{P}_i = \{p_i : M_i \rightarrow [0, 1]\}$. The primary solution concept to for a normal form game is that of a **Nash equilibrium**. A strategy profile is a Nash equilibrium if no player has incentive to unilaterally deviate [62, 57]. In other words, given that each player has chosen a strategy and no player can benefit by unilaterally changing his or her strategy while the other players keep theirs unchanged, then the current set of strategy choices and the corresponding payoffs constitute a Nash equilibrium.

Definition 26. A strategy profile $p^* \in \mathcal{P}$ is a **Nash equilibrium** if :

$$\begin{aligned} \forall i \in N \quad , \quad p_i \in \mathcal{P}_i : \\ r_i(p_1^*, \dots, p_{i-1}^*, p_i, \dots, p_n^*) \leq r_i(p_1^*, \dots, p_n^*) \end{aligned} \tag{67}$$

	Player2 chooses 0	Player2 choose 1	Player2 chooses 2
Player1 chooses 0	(0,0)	(1,0)	(2,-2)
Player1 chooses 1	(0,1)	(1,1)	(3,-2)
Player1 chooses 2	(-2,2)	(-2,3)	(2,2)

Figure 32: Sample game displayed in normal form, with Nash equilibrium points highlighted

A fundamental theorem of game theory is that every finite game in which the players have perfect information has a Nash equilibrium [62, 57]. Normal form games may be represented by an n -dimensional array of n -vectors by tabulating the function $r(p_1, \dots, p_n) = (r_1(p_1, \dots, p_n), \dots, r_n(p_1, \dots, p_n))$. In the case of a two-player game $n = 2$, this reduces to a matrix whose elements are pairs of real numbers. Figure 32 illustrates the matrix representation of a two-player normal form game, in which two players simultaneously select an integer between 0 and 2. If the two players select the same integer, the reward is the value of the selected integer. If one player selects a larger number, then they give up that many points to the other player, while the other than the player is additionally rewarded the value of their own number. The Nash equilibrium points of the game are highlighted in yellow.

6.2 Bi-Clustering as a Game

The bi-clustering problem can be viewed as a game. Consider two party planners, P_1 and P_2 whose goal is to plan a party by inviting guests on their client lists. Each party planner is responsible for disjoint sets of clients, G_1 and G_2 (men and women or employers and job hunters for example) and can only send out invitations to members of their set. Moreover, in order to obtain a party lounge each party planner must guarantee a minimum number of attendees from their respective sets, g_1min and g_2min . Party-goers from each set of clients only interact with clients of the other other set, and their satisfaction is based on the amount interaction at the party. Additionally, if party goers encounter guests that they do not care for this may negatively impact their satisfaction. The party planners receive compensation based on the overall satisfaction of their clients. Both P_1 and P_2 are good at their jobs, thus they both know whose company each client enjoys and whose company they don't. Additionally we will assume that P_1 and P_2 do not cooperate, but they are

	M^1	M^2	M^3		M^1	M^1, M^2	M^1, M^2, M^3	M^1, M^3	M^2	M^2, M^3	M^3
G^1	1	1	1	G^1	(1,1)	(1,2)	(1,3)	(1,2)	(1,1)	(1,2)	(1,1)
G^2	1	0	0	G^1, G^2	(2,1)	(-1,-1)	(-2,-3)	(-1,-1)	(-4,-2)	(-4,-4)	(-4,-2)
G^3	1	1	0	G^1, G^2, G^3	(3,1)	(0,0)	(-3,-3)	(-3,-2)	(-3,-1)	(-6,-4)	(-9,-3)
				G^1, G^3	(2,1)	(2,2)	(0,0)	(-1,-1)	(2,1)	(-1,-1)	(-4,-2)
				G^2	(1,1)	(-2,-4)	(-3,-9)	(-2,-4)	(-5,-5)	(-5,-10)	(-5,-5)
				G^2, G^3	(2,1)	(-1,-1)	(-4,-6)	(-4,-4)	(-4,-2)	(-7,-7)	(-10,-5)
				G^3	(1,1)	(1,2)	(-1,-3)	(-2,-4)	(1,1)	(-2,-4)	(-5,-5)

(a) Client Preferences

(b) Normal form of the game

Figure 33: Bi-clustering as the party planning game

each privy to the guest list of the other at any given point. Thus the game is for P_1 and P_2 to create guest lists for each party such that they maximize their compensation. It is randomly selected who begins, then the players alternate selecting individual clients to add or remove from the invitation list. As soon as the minimum number of invitees are needed a party planner may announce this as the final list or continue. The game ends when both party planners have completed creating their lists or after a finite time interval. In the subsequent sections we will refer to the game described above as party planners game or bi-clustering game interchangeably.

Clearly, all subsets $A_1 \subseteq \mathbf{G}_1$ and $B \subseteq \mathbf{G}_2$ s.t. $|A| \geq g_1 \min$ and $|B| \geq g_2 \min$ may constitute the terminal nodes of the bi-clustering game. Let a formal context $\mathbb{K}_{ij} = (G_i, G_j, I_{ij})$, represent the preferences of clients, then any subspace (A, B) represents a party, but can also be thought of as strategy profile (p_i, p_j) . In order to maximize compensation, each party planner certainly must attempt to invite clients with similar preferences in terms of clients of the opposing party planner. Therefore, the parties that maximize compensation for both planners, represent clusters of clients from both \mathbf{G}_i and \mathbf{G}_j , or bi-clusters.

Definition 27. Given $\mathbb{K} = (G, M, I)$, then the bi-clusters of \mathbb{K} are the Nash equilibria pairs (A, B) of the game $\mathbb{G} = \langle \{1, 2\}, (M_i), (r_i) \rangle$.

6.2.1 Satisfaction Reward Function

We now present an intuitive reward function and illustrate that definition 27 generalizes the previous definition of bi-clustering in terms of FCA. As specified by the game, the compensation of P_1 and P_2 is a function of the satisfaction of the clients at each party. Intuitively, the more people

a client encounters whom they like the greater their satisfaction is. On the other hand, if they encounter people they do not care for, this may negatively impact their enjoyment. Hence for a single client $g_i \in A \subseteq \mathbf{G}_i$ (dually $g_j \in \mathbf{G}_j$) attending party (A, B) we define the satisfaction of g_i as

$$sat_i(g_i, B) = \frac{|\Psi^j(g_i) \cap B| - w_i * |B \setminus \Psi^j(g_i)|}{|B|} \quad (68)$$

where w_i represents the weight of a non or negative interaction. Once again we assume P_1 and P_2 are good at their jobs, and hence the value of w_i is known to them. The compensation of each party planner is then the average satisfaction of all clients, magnified by the number of clients, which is simply

$$r_i(A, B) = \sum_{g_i \in A} sat_i(g_i, B) \quad (69)$$

Letting $w_i = 0$, then r_i is exactly $\rho(A)$, hence connectivity is a special case of this reward function. Figure 33 illustrates the normal-form of the bi-clustering game played with $w_1, w_2 = 0.5$ and the context representing the full knowledge of the players. The larger w_i is, the fewer negative interactions both party planners can afford while attempting to maximize r_i . At the same time, the maximum possible number of clients must be invited to the party for larger compensation.

Theorem 11. *For any instance of the bi-clustering game $\mathbb{G} = \langle \{1, 2\}, (M_i), (r_i) \rangle$, there exists w_i^* , $i = 1, 2$ such that $\forall w_i \geq w_i^*$ if $(p_1^*, p_2^*) = (A^*, B^*)$ is a formal concept of \mathbb{K} and $|A^*| \geq g_1 \min, |B^*| \geq g_2 \min$ then (p_1^*, p_2^*) is a Nash equilibrium of \mathbb{G} .*

Proof. Let (A^*, B^*) be a formal concept and $w_1 = w_2 = \max\{\max_{a \in A^*}\{|a' \cap B|\}, \max_{b \in B^*}\{|b' \cap A|\}\}$. Let us assume that (A^*, B^*) is not a Nash equilibrium. Then there must exist (A, B) such that $r_1(A, B^*) > r_1(A^*, B^*)$ and $r_2(A^*, B) > r_2(A^*, B^*)$. W.l.o.g we will consider only $r_1(A, B^*) > r_1(A^*, B^*)$. Two

cases arise, either (A, B^*) is a semi-concept or it is not. If (A, B^*) is a semi-concept we have

$$\begin{aligned} \sum_{g_1 \in A} sat_1(g_1, B^*) &> \sum_{g_1^* \in A^*} sat(g_1^*, B^*) \\ \frac{|A| * |B^*|}{|B^*|} &> \frac{|A^*| * |B^*|}{|B^*|} \\ |A| &> |A^*| \end{aligned}$$

By the maximality of formal concepts, this is a contradiction. On the other hand, if (A, B^*) is not a semi-concept, then $A \supset A^*$ and $|A| = |A^*| + |A \setminus A^*|$. Let $d_1 = |A \setminus A^*|$ and $d_2 = |A^* \setminus A|$, then by definition of a concept $|\Psi^2(A \setminus A^*) \cap B^*| \leq |B^*| - 1$. Then we have

$$\begin{aligned} \frac{d_1 * (|B^*| - 1) + d_2 * |B^*| - w_1 * d_1}{|B^*|} &\geq \sum_{g_1 \in A} sat_1(g_1) \\ w_1 &\geq |B^*| \text{ hence} \\ d_1 * (|B^*| - 1) + d_2 * |B^*| - w_1 * d_1 &> |A^*| * |B^*| \\ d_1 * (|B^*| - 1) + d_2 * |B^*| - |B^*| * d_1 &> |A^*| * |B^*| \\ d_2 * |B^*| - d_1 &> |A^*| * |B^*| \end{aligned}$$

which is a contraction by the properties of set difference. □

Definition 27 and theorem 11 yield a framework for defining bi-clusters in a single relation. Specifically theorem 11 establishes that the framework encompasses previous approaches of bi-clustering in relations including: FCA [65] (and chapter 3), fault-tolerant FCA [65], and quasi-bicliques [71], when the cost of including zeros is high (large w) then the framework. On the other hand, if the cost of including zeros is tolerable (for example in bioinformatics applications) then the framework does not simply call for including all ϵ -maximal bicliques as bi-clusters; rather, the framework determines the bi-clusters based purely on the utility of the subspaces and maintaining an equilibrium between the utility of each clustering of \mathbf{G}_1 and \mathbf{G}_2 .

6.2.2 Connectivity Based Reward Functions

The clustering criterion developed in chapter 5 is also encompassed under the game theoretic framework. Connectivity is a special of the satisfaction of the reward function while the self connectivity measure may be expressed as

$$r_i(G_i, G_j) = \sigma(G_i, G_j) - \sum_{k=1}^{|G_i|-1} \sum_{l=i}^{|G_i|} w_i * |G_j \cap (\psi^j(g_i^k) \triangle \psi^j(g_i^l))| \quad (70)$$

where \triangle is the symmetric set difference. In terms of the game, equation 70 specifies that the more people that clients jointly like, the greater the overall satisfaction and hence the greater the payout to the party planner. Again setting $w_i = 0$ reduces directly to the original self-connectivity measure. It is easy to see that definition 22 can be derived as a special case of Nash equilibrium of the party planner game when the connectivity and self-connectivity are set as the reward functions. The values of w_i dictate the desired level of connectivity and self-connectivity, while the definition of Nash equilibrium ensures maximality. Moreover, theorem 11 is also applicable in this case, as with very large w_i selected, only fully self-connected maximal subspaces would satisfy the Nash equilibrium definition; such subspaces are exactly formal concepts.

6.3 NashClu Algorithm

Given the framework, the problem of enumerating bi-clusters is equivalent to that of finding a Nash equilibrium in a normal form game. This problem is notorious, and has been described as “the most fundamental problem” at the interface of computer science and game theory. However recent efficient algorithms have been proposed [66]. In particular, the algorithm presented in [66] makes use of a simple search strategy and simple heuristics but has been shown to be quite effective. Thus, in general these algorithms may be used in conjunction with the framework. In this dissertation, however, we propose a specialized algorithm specifically for the party planner game with the satisfaction reward function due to the fact that we have a deeper understanding of the structure of the problem.

Consider the matrix representation of a two-player normal form game. A simple technique for locating all Nash equilibria in such a game is to mark all second components that are maximal among all second components in each row. Next, for each column, mark all first components that are maximal among all first components in that column. Any cell in the matrix that has both components marked is then a Nash equilibrium [57]. Clearly, the drawback of this technique is the assumption that the game matrix is pre-computed. However, it is intractable to compute the entire game matrix for the party planner game, in the worst case this would involve enumerating at least $2^{\max(|A|, |G|)}$ cells in the game matrix. In the sequel, we develop the `NashClu` algorithm based on this labeling technique augmented with several heuristics based on the structure of the game and the satisfaction reward function. `NashClu` entails two major steps: 1) identifying candidate strategy pairs 2) refining the pairs to locate Nash equilibria.

6.3.1 Formal Concepts as Candidates

Theorem 11 yields a connection between formal concepts and Nash equilibria. Even when $w_i < w_i^*$, the concepts may or may not be equilibrium points, nonetheless, they serve as good starting points for locating equilibria. This is based on the observation that for a given $A \subseteq \mathbf{G}_1$ (dually $B \subseteq \mathbf{G}_2$), $r_2(A, \psi^2(A)) = |A|$. Building on this observation, for a given object-set, $A \subseteq \mathbf{G}_1$, the object-set B that maximizes $r_2(A, B)$ must be a super-set of $\psi^2(A)$ and vice-versa.

Proposition 9.

$$\hat{B} = \arg \max_B r_2(A, B) \rightarrow \hat{B} \supseteq \psi^1(A) \quad (71)$$

Proof. If $\psi^1(A) = \emptyset$, then the proposition is trivially satisfied. In the other case, $|\psi^1(A)| \geq 1$. Assume that $\hat{B} = \arg \max_B r_2(A, B)$ and $\hat{B} \not\supseteq \psi^1(A)$. Consider the subspace $(A, \hat{B} \cup \psi^1(A))$, then we

```

Input: Context  $\mathbb{K}_{12,w}$ 
Input: Cluster size constraints  $g_{1min}, g_{2min}$ 
Data: max iterations  $mx$ 
1 begin
2    $\hat{\mathbb{K}}_{12} \leftarrow \mathbb{K}_{12}$  ;
3   while  $|\hat{G}_1| > 0 \wedge |\hat{G}_2| > 0$  do
4      $(A_1, A_2) \leftarrow$  candidate concept from  $\hat{\mathbb{K}}_{12}$ ;
5      $cnt \leftarrow 0$  ;
6     repeat
7       for  $i = 1 \rightarrow 2$  do
8          $\hat{A}_i \leftarrow \text{MaxFixedPoint}(A_i, A_{-i})$  ;
9          $A_i \leftarrow \hat{A}_i$  ;
10         $cnt++$  ;
11      until  $\hat{A}_i == A_i$  or  $cnt \geq mx, i = 1, 2$  ;
12      if  $|A_i| \geq g_{imin}$  for  $i = 1, 2$  and  $cnt < mx$  then
13        Output  $(A_1, A_2)$  as cluster ;
14       $\hat{\mathbb{K}}_{12} \leftarrow \hat{\mathbb{K}}_{12} \setminus (A_1, A_2)$  ;
15 end

```

Algorithm 10: NashClu

have

$$\begin{aligned}
r_2(A, \hat{B}) &> r_2(A, \hat{B} \cup \psi^1(A)) \\
r_2(A, \hat{B}) &> r_2(A, \hat{B}) + r_2(A, \psi^1(A) \setminus \hat{B}) \\
&\text{by definition of } \psi \\
r_2(A, \hat{B}) &> r_2(A, \hat{B}) + |\psi^1(A) \setminus \hat{B}|
\end{aligned}$$

which is a contradiction. □

Proposition 9 dually applies to r_1 and a fixed B ; hence aggregating the proposition with theorem 11 clearly indicate formal concepts constitute natural candidates for Nash equilibria. Hence, NashClu initially enumerates formal concepts to serve as candidate equilibria points; the NClu algorithm can be utilized for efficient enumeration of the concepts.

<pre> Input: Fixed point: A_i^* Input: Non fixed: A_j Data: \mathbb{K}, w 1 begin 2 $Z_j \leftarrow \emptyset$; 3 foreach $g_j \in A_j$ do 4 if $\frac{ \Psi^i(A_j) \cap A_i^* }{ A_i^* - \Psi^i(A_j) \cap A_i^* } < w_j$ then 5 $A_j \leftarrow A_j \setminus g_j$; 6 $Z_j \leftarrow Z_j \cup g_j$; 7 foreach $g_j \in A_j \setminus A_j \setminus Z_j$ do 8 if $\frac{ \Psi^i(A_j) \cap A_i^* }{ A_i^* - \Psi^i(A_j) \cap A_i^* } > w_j$ then 9 $A_j \leftarrow A_j \cup g_j$; 10 return A_j; 11 end </pre>

Function MaxFixedPoint

Following the enumeration of a candidate, Nash equilibrium pairs are identified utilizing the simple marking technique introduced earlier. Let (A_1, A_2) be a candidate concept, then we may think of A_1 as a row of the reward matrix, and A_2 as a column. Holding A_2 as a fixed point, while adding and removing objects from A_1 that increase r_2 corresponds to identifying the maximal elements in the column of the reward matrix. This process is next repeated utilizing the refined set A_1 as a fixed point and refining A_2 to identify a maximal element in the new row. The objects to remove or add to from either A_1 or A_2 during refinement are easily determined by considering the properties of the reward function.

Proposition 10. *In the bi-clustering game \mathbb{G} , given party (A_1, A_2) , if $\forall g_1 \in \mathbf{G}_1 \frac{|\Psi^2(g_1) \cap A_2|}{|A_2| - |\Psi^2(g_1) \cap A_2|} > w_1$ then $r_1(A_1 \cup g_1, A_2) > r_1(A_1, A_2)$*

Proof. Assume there exists such a $g_1 \in \mathbf{G}_1$. Then $r_1(A_1 \cup g_1, A_2) = r_1(A_1, A_2) + r_1(g_1, A_2)$. It is

sufficient to show that $r_1(g_1, A_2) > 0$

$$\begin{aligned} \frac{|\Psi^2(g_1) \cap A_2|}{|A_2| - |\Psi^2(g_1) \cap A_2|} &> w_1 \\ |\Psi^2(g_1) \cap A_2| - w_1(|A_2| - |\Psi^2(g_1) \cap A_2|) &> 0 \\ r_1(g_1, A_2) &> 0 \end{aligned}$$

□

It can also be easily shown that for a pair (A_1, A_2) that $r_1(A_1 \setminus g_1, A_2) > r_1(A_1, A_2)$ for any object $g_1 \in A_1$ s.t. $\frac{|\Psi^2(g_1) \cap A_2|}{|A_2| - |\Psi^2(g_1) \cap A_2|} < w_1$.

The `MaxFixedPoint` function (displayed above) directly implements proposition 10. Pseudocode for the `NashClu` algorithm appears as algorithm 10. Following proposition 9, concepts satisfying size constraints are identified. Following the identification of a candidate concept, the `repeat` loop attempts to locate a Nash equilibrium via consecutive calls to the `MaxFixedPoint` function. A Nash equilibrium is discovered if no change occurs between successive fixed points (line 12). If this condition is not met after a number of iterations we terminate the search and enumerate the next candidate. On the other hand, if a Nash equilibrium is found, and the size constraints are met, then by definition a bi-cluster has been located (lines 13-14). Notice that the `MaxFixedPoint` function operates with the original context, \mathbb{K}_{12} , while during candidate enumeration the context is incrementally shrunk. In this manner, the `NashClu` algorithm avoids the costly step of enumerating of all concepts, while maintaining the possibility of overlapping clustering.

Theorem 12. *Given \mathbb{K} , w_1, w_2 , and g_{1min}, g_{2min} all clusters enumerated by `NashClu` are Nash equilibria of the party planners game $\mathbb{G} = \langle \{1, 2\}, (M_i), (r_i) \rangle$.*

6.4 Information-network Clustering as a Game

The information clustering problem can be viewed as a multi-player version of the party planning game. In this case, n party planners attempt to plan a party to maximize their compensation. Party-

goers from each set of clients only interact with clients of other specific sets; the party planners are privy to this information as it is specified via an information network \mathcal{T} . Moreover, to reserve a party hall each client must guarantee a minimum number of attendees, $g_1min, \dots, g_{|V|}min$. The multi-player game proceeds exactly as the two-player game and each planner is privy to the guest lists of all other clients.

The terminal nodes of the multi-player party planner game constitutes all subspaces (A_1, \dots, A_n) of \mathcal{T} where $|A_i| \geq g_i min$ for $i = 1, \dots, |V|$. Adopting the notation (A_{-i}) to denote the collection $(A_1, \dots, A_{i-1}, A_{i+1}, \dots, A_n)$, a generalized version of the previously defined reward function can be given as follows

$$\begin{aligned} sat_i(g_i, A_{-i}) &= \sum_{A_j \in \Gamma(A_i)} sat_i(g_i, A_j) \\ r_i(A_i, A_{-i}) &= \sum_{g \in A_i} sat_i(g_i, A_{-i}) \end{aligned}$$

6.5 NetNashClu Algorithm

Generalizations of propositions 9 and 10 yield a generalized version of the NashClu algorithm. Consider semi- n , $A = (A_1, \dots, A_n)$, of \mathcal{T} ; then these subspaces have the property that $r_i(A_i, A_{-i}) = |\Gamma(\mathbf{G}_i)| * |A_i|$. Utilizing this property it is direct to show that these subspaces constitute good initial candidates for locating Nash equilibria.

Lemma 1.

$$\arg \max_{A_i} r_i(A_i, A_{-i}) \supseteq \bigcap_{\mathbf{G}_j \in \Gamma(\mathbf{A}_i)} \Psi^j(A_i)$$

Proof. Follows directly from proposition 9. □

Lemma1 prescribes utilizing the n -semi-concepts of \mathcal{T} as initial candidates, just as was the case with TreeClu . Refining candidate clusters proceeds exactly as in the two-player game, by holding A_{-i} as a fixed point and refining A_i iteratively until a Nash equilibria is found or terminating condition is met. Once again, considering the properties of the reward function, establishing which


```

Input:  $\mathcal{T}, w$ 
Input: Cluster size constraints  $g_{1min}, \dots, g_{2min}$ 
1 begin
2   foreach articulation edge  $\mathbb{K}_{ij} \in \mathcal{T}$  do
3      $\hat{\mathbb{K}}_{ij} \leftarrow \mathbb{K}_{ij}$ ;
4     while  $|\hat{G}_i| > 0 \wedge |\hat{G}_j| > 0$  do
5        $(A_i, A_j) \leftarrow$  candidate concept from  $\hat{\mathbb{K}}_{ij}$ ;
6        $(A_1, \dots, A_i, \dots, A_j, \dots, A_{|V|}) \leftarrow$  BFS of  $\mathcal{T}$  and applying  $\psi$ ;
7        $cnt \leftarrow 0$ ;
8       if  $(A_1, \dots, A_i, \dots, A_j, \dots, A_{|V|})$  does not contain  $\emptyset$  then
9         repeat
10          for  $i = 1 \rightarrow |V|$  do
11             $\hat{A}_i \leftarrow \text{NetMaxFixedPoint}(A_i, A_{-i})$ ;
12             $A_i \leftarrow \hat{A}_i$ ;
13             $cnt++$ ;
14          until  $\hat{A}_i == A_i$  or  $cnt \geq mx$ ,  $i = 1, \dots, |V|$ ;
15          if  $|A_i| \geq g_{imin}$  for  $i = 1, \dots, |V|$  and  $cnt < mx$  then
16            Output  $(A_1, \dots, A_{|V|})$  as cluster;
17           $\hat{\mathbb{K}}_{ij} \leftarrow \hat{\mathbb{K}}_{ij} \setminus (A_i, A_j)$ ;
18 end

```

Algorithm 12: NetNashClu

objects to remove or add to A_i is easily computed.

Lemma 2. In the multi-player party planner game \mathbb{G} , given party (A_1, \dots, A_n) , $\forall g_i \in \mathbf{G}_i$ s.t.

$$\frac{\sum_{\mathbf{G}_j \in \Gamma(\mathbf{A}_i)} |\psi^j(A_i) \cap A_j|}{|A_j| - |\psi^j(A_i) \cap A_j|} > w_i \text{ then } r_i(A_i \cup g_i, A_{-i}) > r_i(A_i, A_{-i})$$

Proof. Follows directly from definition of $r_i(A_i, A_{-i})$ and proposition 10 □

Pseudocode for NetNashClu and NetMaxFixed appear above, both the algorithm and function directly implement lemmas 1 and 2 and follow the methodology of NashClu and MaxFixedPoint .

6.5.1 TreeClu as a Special Case and Analysis

As was shown earlier, the n -clusters definition presented in chapter 5 may be captured by the game theoretic framework. In addition, the TreeClu algorithm can be viewed as a special case of NetNashClu . Both algorithms generate candidates in the exact same fashion followed by refinement steps. TreeClu adds an extra level of refinement by performing local refinement which only attempts to add objects from local neighboring concepts. In view of NetNashClu the local refinement step will occur as natural result of the MaxFixedPoint procedure; however the special properties of the self-connectivity reward function, allow for TreeClu to perform refinement in this manner. In addition, the network refinement phase of TreeClu is also a more specialized version of the MaxFixedPoint procedure, in which the special property that articulation nodes must exist in \mathcal{T} is taken advantage of.

As was the case with TreeClu , NetNashClu does not guarantee the enumeration of all Nash equilibria, however, theorem 12 ensures that all subspaces identified are indeed equilibria points and hence clusters. The computational cost of NetNashClu is clearly dependent on the number of candidates generated. As was the case with TreeClu it is difficult to predict the number of candidates, however in the worst case $O(|\mathbf{G}|)$ will be enumerated , where \mathbf{G} is the domain with largest cardinality. Each call to NetMaxFixed entails at most $|\mathbf{G}|^2$ operations and this function is called $|\mathbf{V}| * s$, where s is the total number of refine steps needed. Hence the total cost of NetNashClu is $O(|G|^3 * |\mathbf{V}|)$.

6.6 Preliminary Experimental Results

In this section we present preliminary experimental results obtained with the NetNashClu algorithm. The real-world information networks introduced in chapter 5 were reused for the experiments. Our working hypothesis was that TreeClu algorithm would yield superior clustering results due to the fact that it is a specialized approach. On the other hand, NetNashClu is much simpler to implement and is easily modified to accommodate different reward functions. Hence the goal was to examine if NetNashClu would at least yield comparable results. The value w was varied

Parameters	Algorithm	F_1	$F_{0.5}$	F_2
$w = 0.5$	NetNashClu	0.183833	0.127846	0.327065
$\alpha = 2.5$	TreeClu	0.242041	0.191622	0.328465
$k = 4$	NetClus	0.225488	0.206883	0.24777
$k = 4$	MDC	0.228272	0.211674	0.247694
$w = 1$	NetNashClu	0.200589	0.142616	0.337976
$\alpha = 2.75$	TreeClu	0.24411	0.199462	0.314509
$k = 8$	NetClus	0.164438	0.183863	0.148726
$k = 8$	MDC	0.173128	0.195462	0.155374
$w = 1.5$	NetNashClu	0.213688	0.155308	0.34239
$\alpha = 3.0$	TreeClu	0.283306	0.219294	0.400094
$k = 16$	NetClus	0.108283	0.148992	0.0850456
$k = 16$	MDC	0.111501	0.157455	0.0863106
$w = 2.0$	NetNashClu	0.224881	0.167312	0.342849
$\alpha = 3.25$	TreeClu	0.248256	0.20347	0.318322
$k = 32$	NetClus	0.0731957	0.11887	0.0528781
$k = 32$	MDC	0.0736255	0.125243	0.0521377

(a) ALLPC

Algorithm	F_1	$F_{0.5}$	F_2
NetNashClu	0.312233	0.243236	0.435874
TreeClu	0.341985	0.291558	0.413503
NetClus	0.278475	0.287006	0.270437
MDC	0.284599	0.295308	0.274639
NetNashClu	0.33912	0.279691	0.430619
TreeClu	0.343541	0.31276	0.381041
NetClus	0.191135	0.241016	0.15836
MDC	0.196604	0.252231	0.161079
NetNashClu	0.35231	0.307067	0.413189
TreeClu	0.332871	0.315443	0.352338
NetClus	0.128472	0.193623	0.0961263
MDC	0.123719	0.194651	0.0906756
NetNashClu	0.353995	0.31618	0.402084
TreeClu	0.322396	0.314416	0.330792
NetClus	0.0920745	0.157101	0.0651201
MDC	0.0757684	0.141425	0.0517455

(b) ALLREC

Algorithm	F_1	$F_{0.5}$	F_2
NetNashClu	0.433244	0.356725	0.551552
TreeClu	0.447209	0.381246	0.540775
NetClus	0.375971	0.451266	0.322209
MDC	0.366142	0.453309	0.307092
NetNashClu	0.44503	0.391264	0.515927
TreeClu	0.45363	0.418364	0.495389
NetClus	0.284878	0.393568	0.22323
MDC	0.245008	0.376406	0.181611
NetNashClu	0.44701	0.412001	0.48852
TreeClu	0.460672	0.435264	0.48923
NetClus	0.233927	0.350689	0.175496
MDC	0.153421	0.278065	0.10593
NetNashClu	0.443825	0.428689	0.46007
TreeClu	0.448976	0.454443	0.443639
NetClus	0.200725	0.321033	0.146008
MDC	0.114177	0.231198	0.075807

(c) MER

Algorithm	F_1	$F_{0.5}$	F_2
NetNashClu	0.401061	0.383642	0.420138
TreeClu	0.466381	0.422473	0.520474
NetClus	0.364267	0.439669	0.310942
MDC	0.344025	0.426891	0.288101
NetNashClu	0.430635	0.41801	0.444047
TreeClu	0.438335	0.414682	0.46485
NetClus	0.266409	0.37266	0.207304
MDC	0.218089	0.337279	0.161143
NetNashClu	0.423075	0.428359	0.417919
TreeClu	0.414013	0.457857	0.377832
NetClus	0.216861	0.331496	0.161137
MDC	0.14463	0.26357	0.0996577
NetNashClu	0.41851	0.450748	0.390575
TreeClu	0.413474	0.460364	0.375253
NetClus	0.177318	0.292182	0.127281
MDC	0.100135	0.203137	0.066444

(d) PCR

Algorithm	F_1	$F_{0.5}$	F_2
NetNashClu	0.510019	0.496403	0.524403
TreeClu	0.531512	0.506687	0.55889
NetClus	0.36124	0.366555	0.356076
MDC	0.508535	0.516275	0.501023
NetNashClu	0.461402	0.412486	0.523481
TreeClu	0.534896	0.544086	0.526011
NetClus	0.317618	0.356658	0.286281
MDC	0.416542	0.5222	0.346445
NetNashClu	0.4419	0.402695	0.489562
TreeClu	0.540569	0.507751	0.577922
NetClus	0.239803	0.306856	0.1968
MDC	0.364935	0.543009	0.274813
NetNashClu	0.447708	0.386246	0.532432
TreeClu	0.480845	0.42804	0.548512
NetClus	0.206759	0.377445	0.142375
MDC	0.233105	0.425454	0.160529

(e) FourAreas

Figure 34: Clustering results with NetNashClu algorithm

to mimic the setting of the α parameter in `TreeClu` and ensure a fair comparison. Once again the results displayed are the best results after twenty runs. As figure 34 points out, `TreeClu` has a slight advantage over `NetNashClu`. Nonetheless the results are encouraging as the game theoretic approach is quite comparable, and in fact out performs `TreeClu` in some instances. In addition, `NetNashClu` still yields superior clustering when compared to both `MDC` and `NetClus`. These preliminary results coupled with theorem 11 indicate the efficacy of the game theoretic framework.

6.7 Conclusion

In this chapter we have proposed a novel framework for information network clustering based on game theory. Modeling the clustering problem as a game, in which players attempt to maximize their reward, clusters are defined as the Nash equilibria solution concepts of the game. This framework presents a unifying definition of an information network cluster, while allowing for domain knowledge to be incorporated via specifying the rules of the game and the reward function. Utilizing an intuitive reward function, we illustrated that the framework encompasses previous formulations of bi-clustering and information network clustering. Additionally, the `NetNashClu` algorithm was developed in concert with the reward function to identify Nash equilibria. Experimental results on several real world information networks clearly demonstrated the advantage of `NetNashClu` over the recently proposed `NetClus` and state of the art `MDC` algorithms, while illustrating comparable results to the more specialized `TreeClu` algorithm.

7 Distinguishing Sets

In this chapter we turn our attention back to a single context. Mining bi-clusters in a single context has been studied extensively [49]. As was pointed out in the earlier chapter the set of all concepts $\mathfrak{B}(G, M, I)$ tends to be quite large; exponential in the size of the context in the worst case. While concepts have proven their utility in several fields such as bioinformatics, basket data analysis, text mining, web mining and recommender systems; the task of reasoning about the concepts and the relationships that exist among concepts remains a challenge. The lattice structure is fundamental to reasoning about the concepts and has been utilized extensively for rule generation and visualization [82, 13, 35, 58, 61]. However as contexts become larger and larger the lattice structure also grows exponentially making visualization impossible and rule generation very inefficient. Moreover, recall that the overall goal of clustering is rooted in revealing differences between different groups of objects in a context. However, exploring the concepts of $\mathfrak{B}(G, M, I)$ divulges the fact that many concepts differ very slightly and in fact do not reveal much useful information. This is especially true of neighbors in the concept lattice which may differ by only one object or one attribute. These small differences are often times the result of noise or human error and hinder the process of reasoning about the context through clustering.

In order to address these issues we propose the new data mining task of discovering **distinguishing sets**. The goal of this discovery task is to discover sets of attributes and / or objects that most distinguish the concepts of a context from each other. We define the distinguishing sets as the difference between a concept and an immediate parent in the lattice. Therefore each edge in the concept lattice corresponds directly to two distinguishing sets, an attribute distinguishing set and object distinguishing set. As mentioned above, the lattice tends to be cluttered and several paths exist between a concept and its ancestors. This motivates the key question *which distinguishing sets are most significant?* By means of enumerating the significant distinguishing sets we intend to accomplish two objectives. First, identifying the key sets of attributes and/or objects that truly partition the data into concepts are revealed. Second, prioritizing the edges of lattice, which will lead to scalable and informative visualization techniques and rule generation.

In our approach, we model the concept lattice as a directed graph in which the edges are weighted by the degree of distinction between a concept and its upper neighbors. With this formulation we may transform the problem of discovering significant distinguishing sets into the problem of growing a maximum cost spanning tree in the graph. By growing the maximum cost spanning tree both objectives mentioned above are achieved: distinguishing sets are enumerated and the edges of the bi-cluster lattice are prioritized. Furthermore, we intend for this data analysis task to be performed “on the fly”, implying that this problem is at least as difficult as mining formal concepts. Thus the main contributions of this chapter are:

1. Introduce the concept of distinguishing sets
2. Present a quantitative measure for capturing the degree of distinction between neighboring concepts
3. Present the MIDS (mine incremental distinguishing sets) algorithm based on growing a maximum cost spanning tree in the concept lattice

7.1 Problem Model

In dealing with a single context (as opposed to a set of contexts) the column labels of the context are often referred to as attributes (as opposed to the generic title of objects in information networks). Following this convention we will refer to sets of these attributes as attribute-sets. Current algorithms and applications do not attempt to discover the entire set of concepts $\mathfrak{B}(G, M, I)$, rather only a subset of them that such that each concept satisfies two minimum thresholds: *minAttributes* and *minObjects*. These thresholds are imposed in order to make the problem tractable and applicable. We refer to concepts that satisfy these thresholds as **large concepts**. If the support of any attribute-set or object-set is greater or equal to *minObjects* or *minAttributes* then we refer to that set as **frequent**. By theorem 4 in chapter 4, the set of large concepts also form a complete lattice which is a subset of $\mathfrak{B}(G, M, I)$.

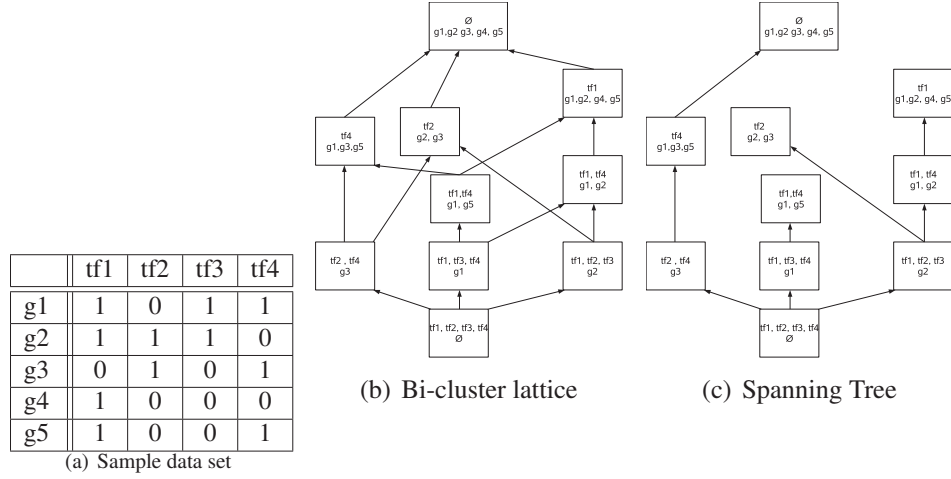


Figure 35: Sample data set and it's bi-cluster lattice

Definition 28. Given a context \mathbb{K} let $\Theta(\mathbb{K}) = (\mathfrak{B}(G, M, I), L)$ denote a directed graph such that $\mathfrak{B}(G, M, I)$ is the vertex set consisting of all concepts of \mathbb{K} , and $L \subseteq \mathfrak{B}(G, M, I)^2$. An ordered pair of concepts $(C^2, C^1) \in L$ if and only if $C^1 \succ C^2$.

Note that we defined Θ in terms of upper neighbors. We could dually define Θ in terms of lower neighbors due to the duality principle for ordered sets [34].

Definition 29. Given two concepts $C^1 = (A^1, B^1)$ and $C^2 = (A^2, B^2)$ such that $C^1 \succ C^2$ then the **distinguishing object-set** between C^1 and C^2 is $A^1 \setminus A^2$. The **distinguishing attribute-set** between C^1 and C^2 is $B^2 \setminus B^1$.

Each edge in Θ corresponds to exactly one distinguishing object-set and one distinguishing attribute-set. For simplicity we will refer to distinguishing sets as *dn*-sets.

Definition 30. A directed path P_n in Θ is a sequence

C^1, C^2, \dots, C^n of distinct concepts with $C^i C^{i+1} \in L$

7.1.1 Motivating Application

A motivating application for detecting significant distinguishing sets comes from the bioinformatics domain. Deciphering mechanisms of gene regulation continues to be a major challenge in

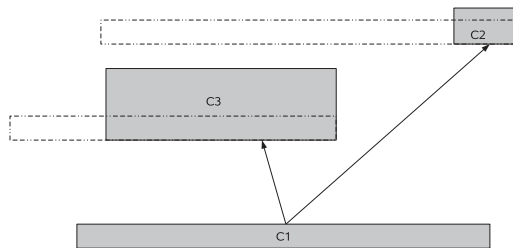


Figure 36: Viewing concepts as maximal rectangles

functional genomics, and many biological investigations are concerned with the interactions between transcription factors (TFs) and target genes. Transcription factors are proteins that bind to genomic regulatory regions of the genes and regulate their expression. Consider the data table in figure 35 which captures interactions between genes and transcription factors. By organizing the concepts of genes and TFs into the lattice structure in figure 35(b) we gain several insights. Specifically, comparing each concept with an immediate parent tells us the difference in activation of genes / TFs that will transform one cellular process into a different one. Discovering the most significant differences that transform cellular processes is of great interest to current biological research.

As can be seen, the lattice structure provides useful insight and visualization which is a starting point for reasoning about the concepts. Growing a maximum cost spanning tree in the lattice has the effect of prioritizing relationships between different gene-TF clusters (as can be seen in figure 35(c)). Furthermore, even greater insight into the data can be gained by post-processing the maximum cost spanning tree; for example enumerating the most frequently occurring distinguishing sets, and the maximal distinguishing set.

7.1.2 Quantifying Distinction in Θ

Consider the example illustrated in figure 36. The distinction between C^1 and C^2 is large in terms of attributes, however in terms of objects the distinction is not very significant. On the other hand the distinction between C^1 and C^3 is large in terms of objects, but not in terms of attributes. This example illustrates the need to utilize both the height and the width of a concept when considering

the degree of distinction. When both height and width change significantly between a concept and one of its upper neighbors, is when the distinction between them is most significant. Another interesting observation can be made by viewing concepts as maximal rectangles. Starting at the infimum and following any path to the supremum, concepts gradually change shape from elongated rectangles height-wise to elongated rectangles width-wise in the concept lattice. This fact follows from the natural properties of the concept lattice. Concepts near the infimum contain the least number of objects and the greatest number of attributes making them the most specific concepts. As we traverse a path in the lattice each upper neighbor becomes more general (more objects and less attributes). The key question is *along which path(s) does the greatest transformation from a specific concept to a general concept occur?* By viewing the concepts as maximal rectangles the degree of “shape change” between a concept C^i and C^{i+1} along a path P_n corresponds exactly to the specific-to-general transformation. Thus we can capture the degree of distinction between concepts by quantifying the “shape change” among concepts. The first step in capturing distinction is then to define a metric that captures the “shape” of a concept. One option to do this is to compute the ratio of width to height (height to width). Given a concept $C = (A, B)$ it’s **shape index** τ is:

$$\tau(C) = s_1(X, Y) = \begin{cases} \frac{|A|}{|B|} & \text{if } |B| \geq |A| \\ \frac{|B|}{|A|} & \text{otherwise} \end{cases}$$

The shape index of a concept captures how square or balanced the concept is. s_1 is maximized when a concept is perfectly square (the number objects and items are exactly equal). Moreover, τ does not distinguish between concepts that are elongated width-wise or elongated height-wise. This property is essential as we wish to capture the change of all shapes and not bias the measure towards any particular shape. Another option to capture the shape of a concept is to simply compute the area of the rectangle. This area will correspond directly to the number of crosses in the concept.

Given a concept $C = (A, B)$, τ then becomes

$$\tau(C) = s_2(A, B) = |A| * |B|$$

Notice that s_2 also does not distinguish between concepts that are elongated width-wise or elongated height-wise. By computing the magnitude of change of τ between a concept $C^i = (A^i, B^i)$ and one of its upper neighbors $C^{i+1} = (A^{i+1}, B^{i+1})$ along a path P_n we capture the intuition discussed above. Computing this change corresponds to the magnitude of the gradient.

$$\|\nabla s_j(A, B)\| = \sqrt{\left(\frac{\partial s_j}{\partial A}\right)^2 + \left(\frac{\partial s_j}{\partial B}\right)^2} \quad (72)$$

where s_j is the chosen shape metric. The partial derivatives in equation 72 capture how s_j changes with respect to the change in A and B along a path $P_n = C^1, C^2, \dots, C^n$. Thus we may compute the partial derivatives utilizing the forward difference operator as follows:

$$\frac{\partial s_j}{\partial A^i} \mapsto s_j(A^{i+1}, B^i) - s_j(A^i, B^i) \quad (73)$$

$$\frac{\partial s_j}{\partial B^i} \mapsto s_j(A^i, B^{i+1}) - s_j(A^i, B^i) \quad (74)$$

Using equations 72, 73, and 74 we define our weighting function on Θ as follows.

Definition 31. Given two concepts $C^1 = (A^1, B^1)$, $C^2 = (A^2, B^2)$ and a shape metric $\tau = s_j$ such that the ordered pair $(C^1, C^2) \in L$ then the weight of the directed edge (C^1, C^2) denoted as $w(C^1, C^2)$ is given as:

$$-\sqrt{(s_j(A^2, B^1) - s_j(A^1, B^1))^2 + (s_j(A^1, B^2) - s_j(A^1, B^1))^2}$$

We assign negative weights to the edges of Θ due to the fact that we wish to grow a maximum cost spanning tree, which is the dual of growing a minimum cost spanning tree.

7.2 MIDS Algorithm

We now present the MIDS algorithm which simultaneously enumerates the concepts of a context \mathbb{K} , builds the concept lattice and grows a maximum (minimum) cost tree in Θ .

7.2.1 Adapting Prim's algorithm

Minimum cost spanning tree algorithms assume that the graph is readily available as input, which is not the case with Θ . However Prim's algorithm has the interesting property that it grows a sequence of n trees T_0, T_1, \dots, T_{n-1} where T_{i+1} is obtained from T_i by adding a single edge e_{i+1} , $i = 0, \dots, n-2$. Furthermore the edge e_{i+1} is selected greedily to be a minimum-weight edge among all edges having exactly one vertex in T_i and one vertex not in T_i ; denoted by $Cut(T_i)$. These properties of Prim's algorithm make it ideal for our problem, because we may dynamically compute $Cut(T_i)$. By definition of Θ , the neighbors of every concept (vertex) are exactly the set of upper neighbors of that concept. Thus intuition tells us that computing the upper neighbors of the concepts in T_i corresponds to computing $Cut(T_i)$. This correspondence in turn allows us to enumerate concepts, build the concept lattice and grow the minimum cost spanning tree all simultaneously. Given a concept C , and current solution of Prim's algorithm T_i let

$$\Lambda(C, T_i) = \{e = (C, C_1) \mid e \in E \wedge C_1 \succ C \wedge C_1 \notin T_i\}$$

In words, $\Lambda(C, T_i)$ is the set of edges between C and all the upper neighbors of C that do not appear in T_i .

Proposition 11. *Given T_i and T_{i-1} let $e_i = (C_1, C_2)$ be the edge added to T_{i-1} to form T_i . Then*

$$Cut(T_i) - Cut(T_{i-1}) = \Lambda(C_2, T_i)$$

Proof. We will prove this by contradiction. Let $e = (X, Y)$ be any edge such that $e \in Cut(T_i) -$

$Cut(T_{i-1}) \wedge e \notin \Lambda(C_2, T_i)$. Then by definition of Cut , $X \in T_i$ and $Y \notin T_i$. Two cases now arise:

1. $\mathbf{X} \in \mathbf{T}_{i-1}$: Clearly $Y \notin T_{i-1}$ implying that $e \in Cut(T_{i-1})$ which is a contradiction.
2. $\mathbf{X} \notin \mathbf{T}_{i-1}$: Thus $X = C_2$ and by definition of Λ , $e \in \Lambda(C_2, T_i)$, which is a contradiction.

□

Proposition 11 gives us a starting point for dynamically computing $Cut(T_i)$, while simultaneously enumerating new concepts in the concept lattice. According to proposition 11 we know what edges to add to $Cut(T_{i-1})$ to form $Cut(T_i)$, now we must specify which edges must drop out. Let $e_i = (C_1, C_2)$ be the edge added to T_{i-1} to form T_i , then clearly by the definition of Cut any edge $e = (D_1, C_2) \in Cut(T_{i-1})$ must be removed from $Cut(T_{i-1})$. We formally define this set of edges as

$$\Psi(Cut(T_i, Cut(T_{i-1}))) = \{e = (d_1, d_2) | e \in Cut(T_{i-1}) \wedge d_2 \in T_i\}$$

Making use of Ψ and proposition 11 we can define an update equation for computing $Cut(T_i)$ directly from $Cut(T_{i-1})$.

$$Cut(T_i) = (Cut(T_{i-1}) - \Psi(Cut(T_i, Cut(T_{i-1})))) \cup \Lambda(C_2, T_i) \quad (75)$$

Utilizing equation 75 we grow a minimum cost spanning tree in Θ by first rooting our tree at a desired concept (usually the infimum). Next we compute $Cut(T_i)$ by generating the upper neighbors of the infimum and compute the distinction w between each upper neighbor and the infimum. The tree is then grown by greedily selecting the minimum weight edge and the associated concept. This process is repeated using equation 75 to update the Cut set at each step until all concepts have been visited. Pseudo code for MIDS algorithm (mine incremental distinguishing sets) appears as Algorithm 20. The upper neighbors of the root concept are first computed and inserted into a priority queue Q (lines 2-7). The current Cut set is maintained implicitly by Q . Moreover once a concept is discovered it is stored in a hashtable along with a hash id. In this manner Q nor T have to explicitly store the concepts; rather they simply store the hash id of each

```

input: root concept  $R$ 
input: shape metric  $\tau$ 
input: minimum thresholds  $minAttributes, minObjects$ 
1 begin
2    $T \leftarrow R$  //root the tree at  $r$  ;
3    $U \leftarrow \emptyset$  //store upper neighbors ;
4    $Q \leftarrow \emptyset$  //priority queue ;
5    $U \leftarrow \text{ComputeUpperNeighbors}(R)$  ;
6   foreach  $u \in U$  do
7      $Q.insert(u, w(R, u))$ ;
8   while not  $Q.empty()$  do
9      $C \leftarrow Q.top()$  ;
10    while  $C \in T$  do
11       $Q.pop()$  ;
12       $C \leftarrow Q.top()$  ;
13     $T \leftarrow T \cup C$  ;
14     $Q.pop()$  ;
15     $R \leftarrow C$  ;
16     $U \leftarrow \text{ComputeUpperNeighbors}(R)$  ;
17    foreach  $u \in U$  do
18       $Q.insert(u, w(R, u))$ ;
19  return  $T$  ;
20 end

```

Algorithm 13: MIDS algorithm

concept, while the weight of each edge is utilized as the priority metric in Q . This allows for quick retrieval of the minimum weight edge in the current *Cut* set and verification that the concept corresponding to that edge is not already in the tree (lines 10-12). Once a concept C has been added to T we compute the upper neighbors of C , as was prescribed in equation 75. The algorithm terminates when no more edges are in Q . Notice that MIDS does not explicitly enumerate the *dn* sets. This can be done in a simple post-processing step by traversing T , or the *dn* sets can be computed by taking the set difference between C and R at line 15, and outputting this result. Storing all the *dn* sets while maintaining T may be infeasible for large databases.

7.2.2 Computing Upper Neighbors

MIDS is centered around computing the upper neighbors of a given concept. Moreover MIDS requires that any algorithm computing the upper neighbors of a concept C depends solely on the concept C . Unfortunately computing the upper neighbors is also the major computational burden of MIDS. Lindig has given the algorithm `Neighbors` to compute the upper neighbors of a concept [52], while Berry et. al have also proposed an algorithm [16]. In this section we propose an improvement to the `Neighbors` algorithm. `Neighbors` computes the upper neighbors of any given concept C based solely on C , without the use of any additional data or complicated data structures. For exact details and proof of correctness refer to [52].

<pre> input: parent concept $P = (A, B)$ Data: $\mathbb{K}_{ij} = (\mathbf{G}_i, \mathbf{G}_j, \mathbf{I}_{ij})$ 1 begin 2 $M \leftarrow \mathbf{G}_i \setminus A$; 3 $N \leftarrow \emptyset$; 4 foreach $g_i \in \mathbf{G}_i \setminus A$ do 5 $Y \leftarrow \psi^j(A \cup \{g_i\})$; 6 $X \leftarrow \psi^i(Y)$; 7 if $M \cap (X \setminus A \setminus \{g_i\}) = \emptyset$ then 8 $N \leftarrow N \cup (X, Y)$; 9 else 10 $M \leftarrow M \setminus \{g_i\}$; 11 return N ; 12 end </pre>

Algorithm 14: Neighbors algorithm

`Neighbors` implements a generate and test strategy based directly on theorem 2. Lines 5-6 of `Neighbors` generate a candidate concept by computing the set S , of all concepts that are greater than C with respect to the hierarchical order, while line 7 performs the upper neighbor check according to theorem 2. The computational complexity of `Neighbors` is $O(|\mathbf{G}_i|^2 \times |\mathbf{G}_j|)$. Computing the closure operator takes $O(|\mathbf{G}_i| \times |\mathbf{G}_j|)$ and this operator is required to execute $O(|\mathbf{G}_i|)$ times. In the following sub-sections we introduce several optimizations to improve the practical running time of `Neighbors` (although the theoretical complexity remains the same) within the context of our

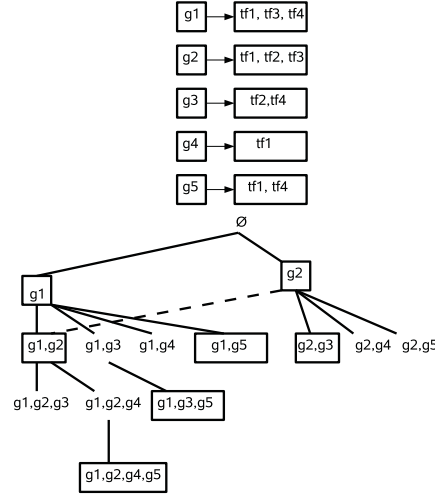


Figure 37: Sample data set and attempt to enumerate upper neighbors with prefix tree

problem.

7.2.3 Optimized neighbors

The major operations involved in `Neighbors` are set intersection, set difference and subset checking. Any reduction in the number of times these operations are performed will improve the practical running time of this algorithm.

Enumerating only large concepts

As mentioned earlier most applications are only interested in large concepts that satisfy the threshold parameters *minAttributes* and *minObjects*. Thus the first standard optimization is to utilize the anti-monotone property of set-intersection to drastically reduce the number of set intersections performed. Furthermore, it is a known fact in data-mining that many of the 2-object-sets (and objectset of size 2) turn out to be infrequent [58]. Thus much computational resources will be wasted performing set intersections when computing $\psi^j(A \cup \{g_i\})$ (line 5) that will eventually lead to infrequent upper neighbors. One option to reduce computation cost is to compute the support of every 2-object-set in a preprocessing step and store the result in 2d upper triangular

boolean array. Efficient methods of computing this are presented in [58]. Moreover the ideas of prefix based classes may be used to reduce the number of set intersections and the length of the sets being intersected. On the down side, the prefix based approach utilized in chapter 4 to enumerate n -concepts will not suffice. This can be demonstrated with a simple example. Consider the example context and the derived prefix tree in figure 37. A square around a prefix node indicates that this prefix node is a concept. The prefix strategy correctly enumerates all upper neighbors of the root concept and of $\{g1\}$ and all ancestors of $\{g1\}$. However, when the search arrives at $\{g2\}$ because of the prefix class formulation we never capture that $\{g1, g2\}$ is an upper neighbor of $\{g2\}$. Following the strict prefix class convention only $\{g2, g3\}$ is discovered as an upper neighbor of $\{g2\}$. Therefore the prefix class formulation is incomplete when attempting to compute upper neighbors. On the other hand, following the strategy of the `Neighbors` algorithm performs several addition set intersections that can and should be avoided. For example consider the $\{g2\}$ once again. Utilizing the `Neighbors` algorithm one would have to re-compute $\Psi^j(\{g1\} \cup \{g2\})$ even though this was already computed when enumerating the upper neighbors of $\{g1\}$.

Due to the incompleteness of prefix based approaches and the expensive method of `Neighbors` we adapt a compromise solution. For each concept $C = (A, B)$ we follow `Neighbors` and perform $\Psi^j(A \cup \{g\})$ with all $g \in \mathbf{G_i} \setminus A$. However, before performing this operation we check if $\Psi^j(A)$ is subsumed (i.e. A is not closed) by any previously enumerated concept. This is identical to the closure check described in chapter 4 when enumerating n -concepts. If $\Psi^j(A)$ is subsumed this implies that $\Psi^i(\Psi^j(A \cup \{g\}))$ has already been enumerated previously in the search and there is no need to recompute it. Now only one more question remains: *is $(\Psi^i(\Psi^j(A \cup g)), \Psi^j(A \cup g))$ an upper neighbor of C ?* Before answering this question though, we consult a hashtable of concepts that have already been enumerated as a result of being in the *Cut* set. If $(\Psi^i(\Psi^j(A \cup g)), \Psi^j(A \cup g))$ was in the *Cut* set at some point then by equation 75 there is no need to visit this concept again, thus no check is needed.

7.2.4 Optimizing upper neighbors check

As mentioned earlier, line 7 checks if the candidate concept $C = (X, Y)$ is an upper neighbor of C . This line involves a set intersection and a set difference operation which can be completely avoided, leading to much accelerated running times. Our new formulation takes advantage of ideas from prefix based classes when checking for upper neighbors, hence $Y = \psi^j(A \cup \{g\})$ is already computed. What remains to be computed $\psi^i(Y)$ and checking if (X, Y) is an upper neighbor. We can accomplish both these tasks simulatensouly avoiding unnecessary addition set intersections and difference operations.

Theorem 13. *Let $C^1 = (A^1, B^1)$ and $C^2 = (A^2, B^2)$ be concepts in context \mathbb{K}_{ij} such that $A^2 = (A^1 \cup X)$ for any object-set X in context \mathbb{K}_{ij} . Let $\langle A^1, X \rangle = \{\psi^j(A^1 \cup \{x\}) \mid \forall x \in X\}$. C^2 is an upper neighbor of C^1 if and only if there does not exist $B^i \in \langle A, X \rangle$ such that $B^2 \subset B^i$.*

Proof. \implies : Assume C^2 is an upper neighbor of C^1 . Assume that there exists $B^i \in \langle A^1, X \rangle$ s.t. $B^2 \subset B^i$. This implies that $\exists x, x \in X$ s.t. $\psi^i(\psi^j(A \cup \{x\})) \subset A^2$, which is a contradiction by theorem 2.

\impliedby : Assume there does not exist $B^i \in \langle A^1, X \rangle$ such that $B^2 \subset B^i$. In that case for all $x \in X$, $\psi^i(\psi^j(A^1 \cup \{x\})) = A_2$. Thus, by theorem 2 C^2 is an upper neighbor of C^1 . \square

Theorem 13 allows for the use of the `Optimize` function (described in chapter 4) to perform both closure and upper neighbor checking. When computing the closure of an object-set A with `Optimize`, if condition 2 of theorem 6 is encountered then by theorem 13 the resulting object-set is not an upper neighbor of A .

Example 7. *Consider the concept $C^1 = (\{\emptyset\}, \{tf1, tf2, tf3, tf4\})$. Let the input to `Optimize` be the object-set $P = \emptyset \cup \{g4\}$. Then we have:*

Input to Optimize	Closure Following Optimize
$P = \{\emptyset \cup g4\}$	$\hat{P} = \{g4, g1, g2, g5\}$
$k = 1$	
$tail(P) = \{g4, g1, g2, g3, g5\}$	
$\langle P \rangle = \{(tf1), (tf1, tf3, tf4), (tf1, tf2, tf3), (tf2, tf4), (tf1, tf4)\}$	

Input: root concept $C = (A, B)$
Input: $\langle A \rangle = \{B^1 = \psi^j(A \cup g^1), \dots, B^n = \psi^j(A \cup g^n)\}$
Input: shape metric τ
Data: the tree T , hash table H , priority queue Q
Data: minimum thresholds $minAttributes, minObjects, N = |G_i|$

```

1 begin
2    $tails[N][N] \leftarrow \emptyset$  // vector of tails ;
3    $contains[N][N]$  //upper triangular 2-d array ;
4   for  $i \leftarrow 1$  to  $n$  do
5      $isUpper \leftarrow true$  ;
6      $X \leftarrow A \cup g^i$  ;
7     for  $j \leftarrow i + 1$  to  $n$  do
8        $Y \leftarrow B^i \cap B^{i+j}$  ;
9        $CheckUpper(C, X, Y, contains, tails, isUpper, i, j)$  ;
10      if  $isUpper$  then
11         $Q.add(C, (X, B^i), tails[i])$  ;
12      else
13         $H.add((X, B^i))$  ;
14      while not  $Q.empty()$  do
15         $E \leftarrow (C^x, C^y, \langle A^y \rangle) \leftarrow Q.top()$  ;
16         $T \leftarrow T \cup E$  ;
17         $MIDS(C^y, \langle A^y \rangle)$  ;
18      return  $T$  ;
19 end

```

Algorithm 15: Improved MIDS algorithm

Thus the resulting concept of this closure is $(\{g1, g2, g4, g5\}, \{tf1\})$. However this concept is not an upper-neighbor of C^1 since $g1, g2$ and $g5$ were all added to the \hat{P} utilizing condition 2 of *Optimize* (line 11). On the other hand consider C^1 , but let the input to *Optimize* be the object-set $P = \emptyset \cup \{g1\}$.

Input to <i>Optimize</i>	Closure Following <i>Optimize</i>
$P = \{\emptyset \cup g1\}$	$\hat{P} = \{g1\}$
$k = 1$	
$tail(P) = \{g1, g2, g3, g4, g5\}$	
$\langle P \rangle = \{(tf1, tf3, tf4), (tf1, tf2, tf3), (tf2, tf4), (tf1), (tf1, tf4)\}$	

In this case no other objects are added to \hat{P} and theorem 13 holds true, as $(\{g1\}, \{tf1, tf3, tf4\})$ is an upper neighbor of C^1 .

Based on the simultaneous closure and upper neighbor check prescribed by theorem 13 we now employ an improved version of `MIDS` (algorithm 15). The *tails* upper triangular 2-d array stores the result of intersecting each B^i with B^j for all $0 \leq i < j \leq N$, where $N = |\mathbf{G}_1|$. The *contains* upper triangular 2-d array tracks if each B^i contains B^j for all $0 \leq i < j \leq N$. The algorithm then proceeds to compute the closure of each $A \cup g^i$ and determine its upper neighbors (lines 6-9). If an upper neighbor is discovered then the edge is added along with the associated sub-search space. In this manner perviously performed set intersections are saved and not repeated (line 11). If concept (X, B^i) is not an upper neighbor of C , then it is added to a hash table H . In this manner we can check if this concept has been previously generated. Lines 14-18 are similar to the previous version of `MIDS` and simply implement equation 75 to grow the maximum cost spanning tree.

The main improvements over the previous version of `MIDS` occurs in the `CheckUpper` procedure. This procedure is almost identical to the `Optimize` function of `NClu` but with a few additional checks. First, before computing closure of X a check is made to see if (Y) is subsumed. If this is the case then the concept (X, Y) has been previously enumerated. If (X, Y) is also in the tree then there is no need to proceed since we have already visited this concept (line 5). If (X, Y) is not in the tree then we proceed even though this concept has already been enumerated, we must check if it is an upper neighbor. Because the `Optimize` procedure utilized prefix classes this guaranteed that no duplicate intersections would take place. However, recall that this approach is incomplete when checking for upper neighbors. For this reason the *for* loop in lines 6-10 checks the *contains* array. If the previous $\psi^j(A \cup g^k)$ contained $(A \cup g^i)$ then they must be added to next tail. By utilizing this check we avoid performing additional set intersections in `MIDS` while still guaranteeing completeness. Lines 11-23 are identical to `Optimize` with the additional setting of flags for upper neighbors based on theorem 13.

We now analyze the computational complexity of `MIDS`. `UpperCheck` greatly improves the practical running time of `MIDS` (as will be shown in the experiments section) however it has no bearing on the computational complexity of the algorithm. Thus we analyze the complexity of `MIDS` utilizing the first version and `Neighbors`. Let N denote the total number of large concepts in

Input: ObjectSet X , attribute-set Y , $contains, tails, isUpper, i, j$

Data: tree T , hash table H , priority queue Q

Data: minimum thresholds $minAttributes, minObjects$

```

1 begin
2   if SubSumed(  $Y$ ) then
3     if  $(X, Y) \in T$  then
4        $isUpper \leftarrow false$  ;
5       return //concept was already visited ;
6   for  $k \leftarrow 0$  to  $i$  do
7     if  $contains[k][i]$  then
8        $X \leftarrow X \cup g^k$  ;
9     else
10       $tails[i][k] \leftarrow Y$  ;
11   if  $|Y| \geq minAttributes$  then
12     if  $B^i = B^j$  then
13        $X \leftarrow X \cup A^j$  ;
14       Remove  $B^j$  from  $\langle A \rangle$  ;
15     else if  $B^i \subset B^j$  then
16        $X \leftarrow X \cup A^j$  ;
17        $contains[i][j] \leftarrow true$  ;
18        $isUpper \leftarrow false$ ;
19     else if  $B^i \supset B^j$  then
20       Remove  $B^j$  from  $\langle A \rangle$  ;
21        $tails[i][j] \leftarrow Y$  ;
22     else
23        $tails[i][j] \leftarrow Y$  ;
24 end

```

Algorithm 16: UpperCheck procedure

\mathbb{K}_{ij} . The body of the **while** loop (lines 8-18) is executed $|L|$ (number of edges in the concept lattice) times. When implemented with a heap-based priority queue each $Q.top()$ operation takes $O(\log N)$ time as does each insertion operation. Checking if a concept is already in the tree can be computed in amortized constant time using the hashtable strategy. The **UpperNeighbors** function is called N times, and each call takes $O(|G_i|^2 \times |G_j|)$ time. Thus the total time for MIDS is $O(|L| \log N + N \cdot |G_i|^2 |G_j|)$.

Proposition 11 along with theorems 2 and 13 guarantee the correctness of MIDS .

7.3 Experimental Results

Experiments were performed on a 2.80 GHz Pentium-D PC with 3.68GB of memory running SUSE Linux. All algorithms were coded in C++ using the STL libraries and data structures. For performance comparison, we used the original source code of CHARM-L [58] downloaded from (<http://www.cs.rpi.edu/~zaki/software/>). It must be noted that CHARM-L and MIDS perform different mining tasks; CHARM-L enumerates and constructs the lattice of closed itemsets for rule generation, while MIDS grows a maximum cost spanning tree in the lattice. However, one approach to growing the *dn*-set tree would be to construct the concept lattice using an algorithm such as CHARM-L weighting the edges, then utilizing any known maximum cost spanning tree algorithm. Growing the maximum cost spanning tree during the mining process is not possible with CHARM-L due to the fact that at any given moment the algorithm does not guarantee that all the upper neighbors of a concept are enumerated. Due to this fact the cut set cannot be maintained and the greedy approach followed by MIDS cannot be implemented. Moreover, only at the termination of CHARM-L is the lattice structure assured to be correct. Many instances exist in the algorithm at which parent pointers of existing lattice nodes have to be adjusted. Thus, clearly a post processing step is needed to ensure correctness when searching for significant *dn*-sets.

We chose three real-world datasets from the bioinformatics domain for the performance tests and the benchmark Mushrooms dataset. The characteristics of the datasets are listed in table 38(a), while the results are displayed in figure 39. We compare the performance of MIDS, to CHARM-L followed by a post processing step to compute the minimum spanning tree. To ensure a fair comparison all experiments were run with *minObjects* = 1, which in effect tells MIDS to search for closed itemsets. We observe that on PhenoType the performance is equivalent at higher levels of support, however for the lower levels MIDS clearly outperforms Charm-L. This scenario is repeated with the Mushrooms dataset: high levels of support yield an advantage to CHARM-L, while MIDS outperforms at lower levels. In order to better understand this phenomenon we computed the ratio of edges to concepts in the lattice for different minimum support levels (figure 38(b)). This value

Database	Num Objects	Num Attributes	Avg. Length
GoTerms	3,385	1,910	42.8094
MicroRna	318	1,910	51.4464
PhenoTypes	3,965	1,910	11.543
Mushrooms	8,124	120	23

(a) Database characteristics

Data Set	Min Attribute %	Density
PhenoTypes	1.26	3.53
	0.75	4.65
	0.50	9.11
	0.37	9.16
Mushrooms	6.15	7.87
	3.07	8.5
	1.23	9.2
	0.01	11.42
GOTerms	1.47	6.43
	1.18	7.42
	1.03	8.0
	0.88	8.5
MicroRna	6.28	4.39
	3.14	5.51
	1.57	6.54
	0.31	8.0

(b) Lattice Density

Figure 38: Data Set and lattice characteristics

indicates the density and complexity of the lattice for each support level and clearly the higher this ratio the more difficult the search for dn -sets becomes. We noticed that as lattice density increases, `MIDS` remains much more scalable compared to `CHARM-L` as evidenced by both the `PhenoType` and `Mushrooms` datasets. On the other hand, `CHARM-L` executes faster in both the `Gene Ontology` and `Micro` datasets which contain lower lattice densities.

The performance experiments illustrate the practical utility of `MIDS`. This is especially true for low levels of support and dense lattices where the difference in running times were significant. This advantage is critical in sparse datasets found not only in the bio-informatics domain, but also in text mining and market-basket analysis.

7.3.1 Application to synthetic data

In this section we present some results after running `MIDS` on different synthetic data sets. We generated three different synthetic datasets and refer to them as **Syn1**, **Syn2**, and **Syn3**. All three datasets contained 41 attributes and 50 objects. Bitmaps of all these datasets appear in figure 40. All three datasets contained different levels of noise and different dense regions in which

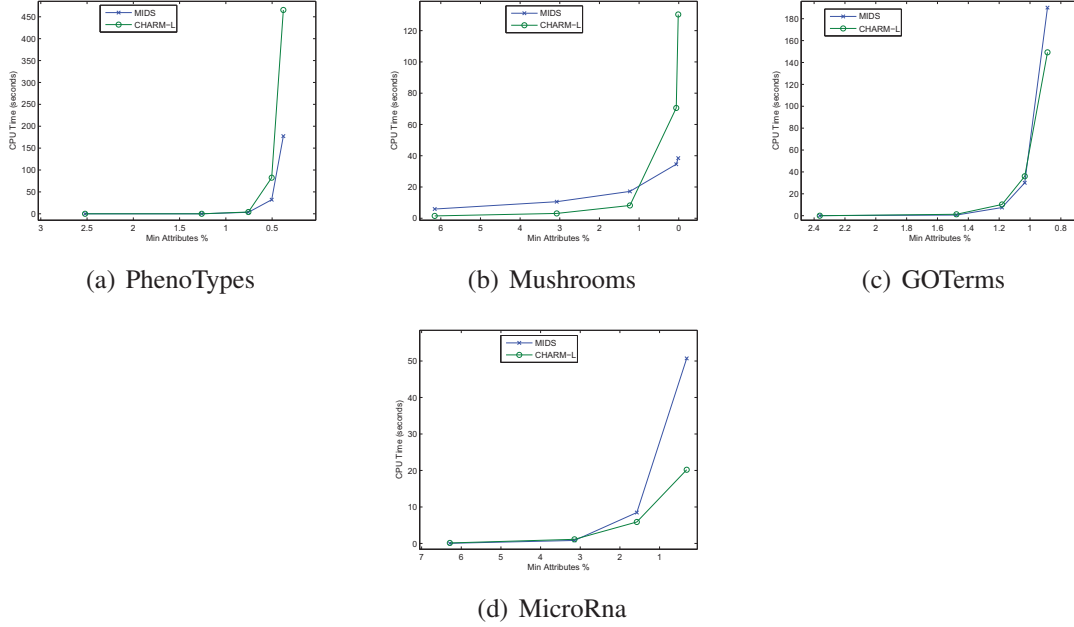


Figure 39: Performance Study Results

concepts are formed. However region 1 (see figure 40(a)) was the main region of interest in each experiment. This is due to the fact that this region contains two fairly large concepts that are very distinct. This can be seen visually in the figures. **Syn1** contained relatively small amounts of noise and only the main dense region 1 of interest. For our first experiment we ran MIDS with $minAttributes = minObjects = 1$ and $\alpha = s_2$ and ordered the resulting dn -sets in decreasing order of w . By visual inspection and intuition we expect MIDS to weight the distinction between the two large concepts in region 1 maximally when growing the dn -set tree. We investigated the results by displaying the upper and lower neighbor pair of the top ranked dn -set (figure 40(b)), as can be seen MIDS correctly picks the the greatest distinguishing set. The results in **Syn1** was used as a reference to study what effect adding noise and other dense regions have on the ability of MIDS to pick out distinctions in this region.

Noise was added to **Syn1** by randomly placing additional 1's to form **Syn2**. Once again we ran MIDS with $minAttributes = minObjects = 1$ and $\alpha = s_2$ and ordered the result the dn -sets. Interestingly, MIDS was able to pick out a single object whose removal resulted in the addition of 10 attributes! This distinction is clearly illustrated in figure 40(d). Upon closer inspection we

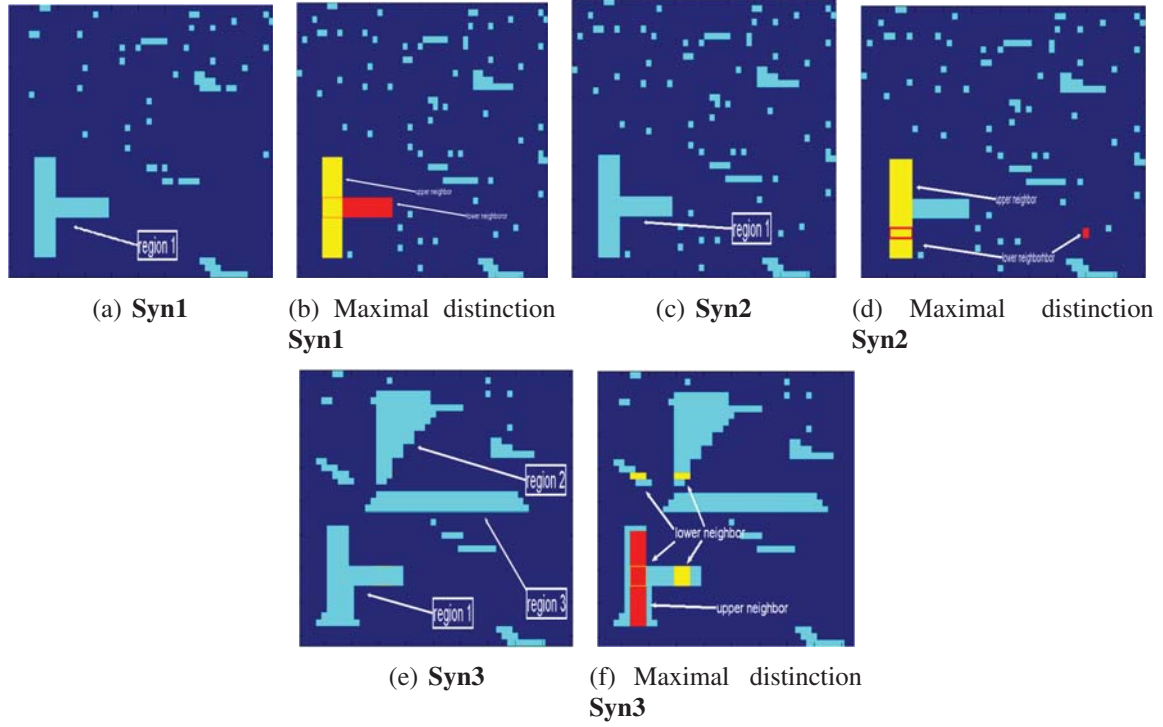


Figure 40: Bitmaps of experimental results on **Syn1**, **Syn2**, **Syn3**

discovered that the two large concepts in region 1 were not neighbors in the concept lattice of **Syn2**. This illustrates the fine grain approach that MIDS takes in mining *dn*-sets as result of relying solely on the lattice structure. On the other hand, by increasing the minimum thresholds *minAttributes* and *minObjects* the granularly of MIDS can be adjusted. We executed the algorithm on **Syn2** with $minAttributes = minObjects = 2$ and in that case the concepts in region 1 were upper neighbors in the concept lattice, (since smaller concepts were filtered out) and we obtained a similar result as in our experiment with **Syn1**. This illustrates the robustness and flexibility of MIDS .

Syn3 contained three main dense regions of concepts, labeled as regions 1,2, and 3. Notice that regions 2 and 3 do not contain clearly distinctive concepts, whereas region 1 does. Region 3 essentially contains three large concepts differentiated by one or two objects and attributes. Whereas region 2 contains several large concepts, but once again the distinction among each concept is minimal. Region 1 is the same region of interest that MIDS pointed to in **Syn1** with additional noise. We conducted experiments to test if MIDS would still mine the interesting distinctions inherent in

Distinguishing Attribute(s)	Lower and Upper neighbors	Class	Distribution
Chocolate spore print color	free gill, close gill spacing, partial veil type, white veil color, one ring	P	58.93 %
		E	41.07 %
	free gill, close gill spacing, partial veil type, white veil color, one ring, chocolate spore print color	P	97.05 %
		E	2.94 %
Path habitat	free gill, close gill spacing, partial veil type, white veil color, one ring	P	58.93 %
		E	41.07 %
	free gill, close gill spacing, partial veil type, white veil color, one ring, path habitat	P	91.30 %
		E	8.69 %
Brown Gill	free gill, partial veil type, white veil color, one ring	P	52.14 %
		E	47.86 %
	free gill, partial veil type, white veil color, one ring, brown gill	P	11.38 %
		E	88.62 %

(a) Mushrooms dataset

Distinguishing Attribute(s)	Lower and Upper neighbors	Class	Distribution
NO physician fee freeze	YES religious-groups-in-schools	R	54.77 %
		D	45.22 %
	YES religious-groups-in-schools , NO physician fee freeze	R	0.95 %
		D	99.05 %
YES adoption of budget	YES religious-groups-in-schools	R	54.77 %
		D	41.07 %
	YES religious-groups-in-schools, YES adoption of budget	R	14.91 %
		D	85.08 %
NO religious groups in school	YES export-administration-act-south-africa	R	35.68 %
		D	64.31 %
	YES export-administration-act-south-africa, NO religious groups in school	R	13.20 %
		D	86.79 %

(b) Congress dataset

Figure 41: Experimental results utilizing real world data sets

region 1 despite the facts that:

1. regions 2 and 3 are larger and contain more mass
2. regions 2 and 3 overlap with region 1
3. the shape of region 1 has been distorted slightly

Executing `MIDS` with $minAttributes = minObjects = 2$ resulted in in figure 40(f). The lower neighbor is primarily located in region 1, but the influence of region 2 is not ignored.

Throughout all experiments the dn -sets located in the primary region(s) of interest were consistently mined and prioritized by `MIDS`.

7.3.2 Experiments in Real Data

In order to measure the true utility and applicability of distinguishing sets we performed several more experiments utilizing the real-world datasets of mushroom attributes and congress voting records. In each experiment `MIDS` was executed and the top 20 distinguishing sets were output.

For each distinguishing set we investigated the content of the upper neighbor and lower neighbor associated with that distinguishing set. Moreover the class labels associated with each object were used to display the distribution of objects in each bi-cluster. Note the class labels were not utilized while executing `MIDS`. The results of the experiments are exhibited in figure 41. The mushrooms datasets consists of 22 nominal attributes that describe 8,124 mushrooms. The class label indicates if a mushroom is poisonous (P) or edible (E). `MIDS` was executed five different times utilizing the s_2 shape metric and varying the minimum attributes parameter between 5-12 %. The choice of minimum attributes was arbitrary to a large degree. The goal was enumerate a suitable number of bi-clusters in order to investigate the utility of the derived distinguishing sets. The resulting distinguishing sets from each of the 5 runs were very similar. Three of the top 20 distinguishing sets mined with min attributes set to 9 % are displayed in figure 41(a). In each case the lower bi-cluster contains a healthy mixture of poisonous and edible mushrooms, while the upper neighbor contains an almost pure bi-cluster with respect to the class label. Even more interesting is the fact that in many cases the distinguishing attribute it self contained a mixture of both edible and posticous mushrooms; however, when the attribute was added to the lower neighbor to form the upper neighbor is when its distinguishing ability was revealed. For example consider the first result in figure 41(a). The distinguishing attribute *Chocolate spore print color* on its own covers 4,640 mushrooms with 45.51 % poisonous and 54.48 % edible. Clearly this attribute on its own does not distinguish the two class labels. Moreover, the attributes of the lower neighbor: *free gill, close gill spacing, white veil color, one ring* do not distinguish between poisonous and edible mushrooms either; however in conjunction with *chocolate spore print color* a clear distinction is made. Thus `MIDS` is able to pick out sets of attributes that induce real distinctions among incremental bi-clusters of the data set, without the use of a class label.

The Congress data set contains 16 votes in which each congress person (435 of them) either voted yes, no, or was absent. The traditional goal of machine learning algorithms is to predict the party affiliation (republican or democrat) of each congress person. `MIDS` was run on this data set once again utilizing the s_2 shape metric and varying the min attribute percentage. Executing

MIDS with low min attributes percentages (below 25 %) resulted in only pure bi-clusters. This is a testament to partisan nature of the votes. However, with higher values we are able to attain results similar to the mushroom data set (figure 41(b)). Clear distinctions arise between incremental bi-clusters as evidenced by the class distributions. The distinguishing sets in this case pointed out what votes induced bipartisanship and which votes induced partisanship.

7.4 Conclusion

In this chapter a shortcoming of FCA based bi-clustering is addressed. The enormous number of concepts contained in large contexts diminishes the effectiveness of the FCA approach, hence the novel idea of mining significant distinguishing sets among incremental concepts was introduced. In this manner, a prioritization is placed on the edges of the concept lattice, and those concepts that only differ slightly are ignored, while uncovering the sets of objects and attributes that truly partition the context into clusters. Quantifying distinction between two neighboring concepts is accomplished by measuring the degree of shape change between the two concepts. Utilizing this as weighting on the edges of the concept lattice, a maximal cost spanning tree is grown via the MIDS algorithm. MIDS takes advantage of some nice properties of Prim's algorithm, allowing us to simultaneously mine the concept lattice and compute the spanning tree. The theoretical complexity and empirical performance studies reveal that MIDS is suited for large datasets and is especially effective as no post processing step is required to grow the tree in order to discover the distinguishing sets. Finally, the efficacy of distinguishing sets was pointed out in experimental results on both synthetic and real data

8 Maximally Banded Sub-Matrices

In addition to the large number of concepts present in large contexts, another possible drawback of the FCA model of bi-clustering is that clusters are restricted to maximal rectangles. However in many applications such as mapping the human genome, palaeontological data, and social networking banded subspace structures are of interest [46, 56, 37]. A binary matrix is said to be fully banded if both the rows and columns can be permuted such that the 1's exhibit a staircase pattern of overlapping rows along the leading diagonal (figure 42).

While the study of banded matrices has its origins in numerical analysis [68], the idea has been studied recently in the data mining community [56, 37]. From the data mining perspective, banded structures have myriad applications and natural interpretations. Consider for example a binary matrix containing documents as the rows and keywords as the columns, where the set of documents revolve around the single theme of “clustering”. Early documents on clustering, from around the 60's, may contain terms like “k-means”, while documents in the 70's may contain both “k-means” and “expectation maximization”, and eventually documents in recent years will contain terms like “subspace clustering”, “bi-clustering”, and “curse of dimensionality”. While recent documents may contain the terms “k-means” and “expectation maximization”, we do not expect documents from the 60's or 70's to contain the terms “bi-cluster” or “subspace cluster”. Thus, an evolution of concepts can be seen in the documents via the terms that occur in the documents, and this pattern will resemble a band in the data matrix. As another example consider a context containing genes and pathways; many situations exist where pathways A and B may start C; B and C may start D and C and D may start D. The whole pathway can be initiated by starting with A and B. If this pattern is encoded in a context it will not look like a maximal rectangle, but more like a parallelogram, or staircase pattern of 1s. Other natural interpretations of a banded structure include overlapping communities in social networks, overlapping roles of genes in various diseases, and patterns of species occurring in spatially correlated sites.

The FCA bi-clustering model does not consider such staircase patterns to be clusters, we show in the sequel that FCA theory can be utilized as a starting point to discover such patterns. This chap-

	A	B	C	D	E
1	1	1	1	0	0
2	0	1	1	0	0
3	0	0	1	0	0
4	0	0	1	1	0
5	0	0	0	1	1

Figure 42: A fully banded matrix

ter presents the novel MMBS algorithm for detecting banded structures and/or approximate banded structures in subspaces of contexts. The algorithm allows for the discovery of multiple, possibly overlapping or segmented, maximally banded sub-matrices from the context. This differs from the previously proposed MBS [37] algorithm which only allows for the discovery of a single band and fixes the column permutations of the data matrix before executing the algorithm. In addition, we formally illustrate the correspondence between the FCA model of bi-clustering and the problem of discovering banded structures. In summary, the main contributions of this chapter are:

1. Establishing correspondence between banded structures and FCA-based bi-clustering.
2. Introducing the novel MMBS algorithm to uncover multiple, possibly overlapping, banded sub-matrices.
3. Empirical results verifying the advantage of MMBS over previous approaches.

8.1 Problem Definition

Consider the binary matrix representation of a context $\text{mat}(\mathbb{K}_{ij})$. We denote the m -th row of $\text{mat}(\mathbb{K}_{ij})$ by $\text{mat}(\mathbb{K}_{ij})^m$ and the n -th column by $\text{mat}(\mathbb{K}_{ij})^n$; however, for simplicity we will simply write \mathbb{K} as opposed to $\text{mat}(\mathbb{K}_{ij})$. Given a permutation π_i of \mathbf{G}_i , and permutation π_j of \mathbf{G}_j , then $\mathbb{K}_{\pi_j}^{\pi_i}$ is the permutation of rows and columns according to π_i and π_j . We will use $g_i^{\pi_i}$ and $g_j^{\pi_j}$ to denote the i -th row and j -th column respectively under permutations π_i and π_j .

Definition 32. A binary matrix \mathbb{K}_{ij} is **fully banded** if there exists a permutation π_i of \mathbf{G}_i and permutation π_j of \mathbf{G}_j such that (1) for every row i in $\mathbb{K}_{\pi_j}^{\pi_i}$ the entries with 1s occur in consecutive

column indices $\{m_i, m_i + 1, \dots, m_i^*\}$ and (2) the values of starting indices for 1s in successive rows (i and $i + 1$) satisfy the conditions $m_i \leq m_{i+1}$ and $m_i^* \leq m_{i+1}^*$. An illustration of this is given in figure 42.

Testing a given matrix to find whether it is banded can be accomplished in polynomial time [37]. In real datasets a fully banded structure may not exist due to noise or irrelevant dimensions, however, we are still interested in discovering 1) approximately banded matrices and 2) maximally banded sub-matrices of the dataset. A maximally banded sub-matrix of a matrix, intuitively, is one in which no more rows from the original matrix can be added while still preserving the bandedness of the selected sub-matrix. The quality of a banded structure can be measured in terms of noise, where noise is the number of 0s or 1s that must be flipped in order to achieve a fully banded matrix. Given an approximately banded matrix $\mathbb{K}_{\pi_j}^{\pi_i}$, let $e(\mathbb{K}_{\pi_j}^{\pi_i})$ denote the noise or error of the banded structure in $\mathbb{K}_{\pi_j}^{\pi_i}$. We say that $\mathbb{K}_{\pi_j}^{\pi_i}$ is ε -banded if $e(\mathbb{K}_{\pi_j}^{\pi_i}) \leq \varepsilon$. Hence the maximal banded matrix problem is: Given binary matrix \mathbb{K} and noise threshold ε , find all sub-matrices $\hat{\mathbb{K}}$ of \mathbb{K} that are ε -banded and maximal.

Unveiling all maximal ε -banded sub-matrices is useful in contexts that contain banded structures in certain subsets of the dimensions along with possibly independent or segmented bands; *segmented bands* are the situations in which cells around multiple, non-principal diagonals are populated by 1s. Moreover, relaxing the requirements from full bandedness to ε -bandedness allows for clear band structures to be identified despite the presence of noise. The algorithmic complexity of this problem is expected to be hard since it is a generalization of both the MBA and MBS problems given in [37], which were both shown to be hard.

8.2 Banded Structures and FCA

Given any concept $C = (A, B)$ of a context \mathbb{K} , the set of rows (columns) specified by A (B) trivially satisfy both conditions of a banded matrix (every element of a bi-cluster is a 1). Thus C is a banded sub-matrix of \mathbb{K} , and can be utilized as a building block to construct larger bands (in larger sub-matrices). Intuitively, any fully banded matrix can be splintered exactly into maximal rectangles

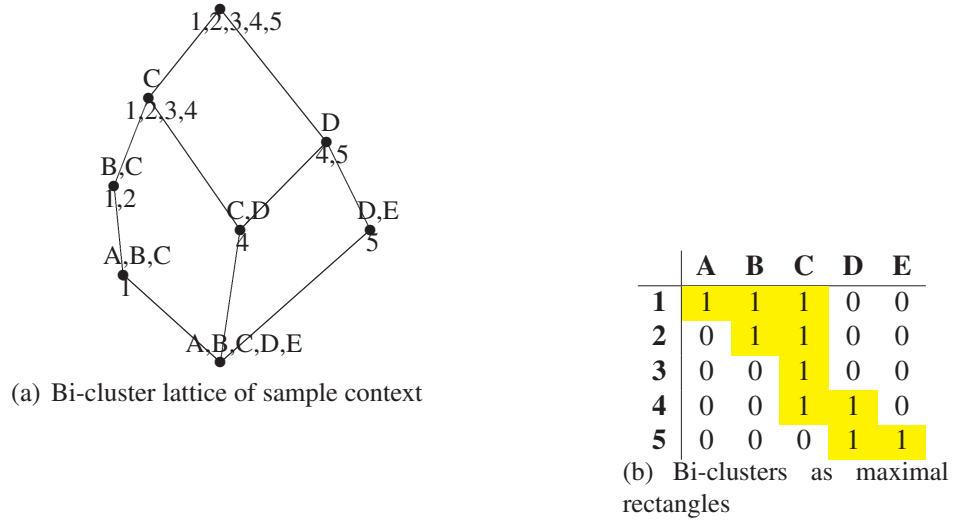


Figure 43: Bi-clusters and maximal rectangles

of 1s, as shown in figure 43(b). Each of these rectangles corresponds exactly to a concept. Thus, a fully banded sub-matrix may be constructed by combining a suitably selected sequence of concepts C^1, \dots, C^n . Formally, given a fully banded matrix, $\mathbb{K}_{\pi_j}^{\pi_i}$, for any row $g_i \in \mathbb{K}_{\pi_j}^{\pi_i}$, let $\Xi(g_i)$ be a mapping from g_i to the set of concepts that contain g_i as a row, other than the null concept element.

$$\Xi(g_i) = \{(A, B) | \{g_i\} \subseteq A \wedge B \neq \emptyset \wedge \psi^j(A) = B \wedge \psi^i(B) = A\}$$

Moreover the objects and attributes of any concept $C \in \Xi(g_i)$ can always be ordered according to π_i and π_j due to the fact that a concept only contains 1s. Let $\mathfrak{F}(\mathbb{K}_{\pi_j}^{\pi_i})$ be the union of all $\Xi(g_i)$ for any $g_i \in G$, that is

$$\mathfrak{F}(\mathbb{K}_{\pi_j}^{\pi_i}) = \bigcup_{g_i \in G_i} \Xi(g_i)$$

Then $\mathfrak{F}(\mathbb{K}_{\pi_j}^{\pi_i})$ can be ordered to make an n -tuple of concepts $\{C^1, \dots, C^n\}$ having a total ordering $\{<_{\pi_i^1, \pi_j^1}, \dots, <_{\pi_i^n, \pi_j^n}\}$ as determined from the lattice structure. Thus we may define a lexicographical order $<_{\pi_i, \pi_j}$ on $C^1 \times C^2 \times \dots \times C^n$. Therefore, considering the concepts in $\mathfrak{F}(\mathbb{K}_{\pi_j}^{\pi_i})$, in order, we may completely specify the permutations π_i and π_j ; hence $\mathfrak{F}(\mathbb{K}_{\pi_j}^{\pi_i})$ constitutes a sequence of concepts

that completely determines the banded structure of \mathbb{K} .

Proposition 12. *Given a context \mathbb{K} , if permutations π_i and π_j exist such that $\mathbb{K}_{\pi_j}^{\pi_i}$ is fully banded then there exists a sequence of concepts $C^1 = (A^1, B^1), \dots, C^n = (A^n, B^n)$ s.t.*

$$\pi_i = \{A^1, A^2 \setminus A^1, \dots, A^n \setminus A^{n-1}\}$$

$$\pi_j = \{B^1 \setminus B^2, \dots, B^{n-1} \setminus B^n, B^n\}$$

The proof of proposition 12 is straightforward: C^1, \dots, C^n can always be constructed by considering the bi-clusters of $\mathfrak{F}(\mathbb{K}_{\pi_j}^{\pi_i})$ in order, while set differences are taken to eliminate duplicate rows and columns. Proposition 12 establishes a clear correspondence between fully banded matrices and FCA; **specifically, if a matrix has a fully banded structure then there always exists a sequence of concepts that stipulates its π_i and π_j .**

Example 8. *Consider the sample context in figure 42. The permutations $\pi = \{1, 2, 3, 4, 5\}$ and $\tau = \{A, B, C, D, E\}$, therefore, lexicographically, $1 < 2 < 3 < 4 < 5$ and $A < B < C < D < E$. The table below illustrates $\mathfrak{F}(\mathbb{K}_{\pi_j}^{\pi_i})$ and the resulting lexicographical ordering.*

$\mathbf{g_i}$	$\Xi(g)$
1	$\{(1, ABC), (12, BC), (1234, C)\}$
2	$\{(12, BC), (1234, C)\}$
3	$\{(1234, C)\}$
4	$\{(4, CD), (45, D)\}$
5	$\{(5, DE), (45, D)\}$
$\mathfrak{F}(\mathbb{K}_{\pi_j}^{\pi_i})$	
$\{(1, ABC) < (12, BC) < (1234, C) < (4, CD) < (45, D) < (5, DE)\}$	

π_i and π_j can be constructed from $\mathfrak{F}(\mathbb{K}_{\pi_i}^{\pi_i})$ as

$$\begin{aligned}\pi_i &= \{1, 12 \setminus 1, \dots, 5 \setminus 45\} \\ &= \{1, 2, 3, 4, 5\} \\ \pi_j &= \{ABC \setminus BC, \dots, D \setminus DE, DE\} \\ &= \{A, B, C, D, E\}\end{aligned}$$

In the next section we show that a banded structure can be grown as a path of concepts in the concept lattice. In addition, we derive an expression for the upper bound of the error when constructing such banded sub-matrices.

8.2.1 Banded sub-matrices and paths in the concept lattice

The concept lattice may be viewed as an undirected graph $\Theta(\mathbb{K}_{ij}) = (\mathfrak{B}(G, M, I), L)$. The set of concepts corresponds to the set of vertices while the set of edges L consists of edges that connect pairs of concepts - upper neighbors to lower neighbors. For a pair of neighbor concepts we can say: $C^1, C^2 \in L \leftrightarrow C^1 \prec C^2 \vee C^2 \prec C^1$. Let $\bar{P} = C^1 = (A^1, B^1), C^2 = (A^2, B^2), \dots, C^n = (A^n, B^n)$ be any path in the lattice graph $\Theta(\mathbb{K})$, then by definition, for every edge $(C^i, C^{i+1}) \in \bar{P}$, $A^{i+1} \subseteq A^i$ and $B^i \subseteq B^{i+1}$ if $C^i \prec C^{i+1}$ (dually $A^i \subseteq A^{i+1}$ and $B^{i+1} \subseteq B^i$ if $C^i \succ C^{i+1}$). Due to the duality of upper and lower neighbors, we restrict the discussion to the case of upper neighbors only without any loss of generality. A banded sub-matrix can be constructed from any edge of the lattice by initiating the matrix to be exactly C^i , by definition every position in this matrix is filled with a 1. Next we place the rows of $A^{i+1} \setminus A^i$ beneath the rows of A^i and shift the columns where A^i and $A^{i+1} \setminus A^i$ differ, to the left. By the definitions of the hierarchal order and concepts, the 0s are located precisely at rows $A^{i+1} \setminus A^i$ and columns $B^i \setminus B^{i+1}$. Consequently, any edge in $\Theta(\mathbb{K}_{ij})$ can be converted into a fully banded sub-matrix.

For example, consider the edge $((1, ABC), (12, BC))$ (figure 44), the two concepts differ by column $\{A\}$ and row $\{2\}$, thus $\{A\}$ is placed at the leftmost position and $\{2\}$ at the bottom-most

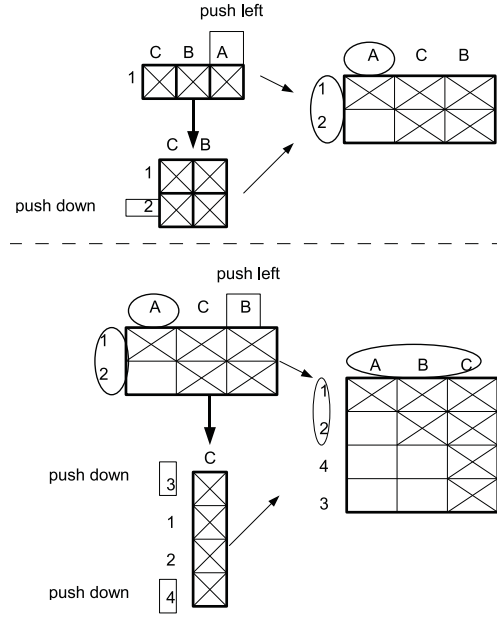


Figure 44: Constructing banded matrices from paths in $\Theta(\mathbb{K})$

position. At this point the positions of rows $\{1\}, \{2\}$ and column $\{A\}$ are fixed and cannot be altered, while the position of columns $\{B\}$ and $\{C\}$ are interchangeable. Thus we obtain the banded sub-matrix with permutations $\pi_i = \{1, 2\}$ and $\pi_j = \{A, C, B\}$ or $\pi_j = \{A, B, C\}$. Next, the path is further expanded by adding the edge $((12, BC), (1234, C))$, resulting in shifting column $\{C\}$ to the right and rows $\{3\}$ and $\{4\}$ to the bottom. This fixes the positions of all the columns but leaves the positions of rows $\{3\}$ and $\{4\}$ interchangeable. In general, the procedure for augmenting a path \bar{P} with an edge (C_n, C_{n+1}) and maintaining the banded structure is described by the `AddToPath` procedure displayed below. Notice that the symmetric difference set operator is used, as opposed to regular set difference, due to the fact that upper or lower neighbors may be added to the path. The procedure determines if the new concept C_{n+1} is an upper or lower neighbor of C_n and if the new rows or columns of C_{n+1} are not already contained in \bar{P} (lines 4 and 10). If this is the case then the new rows \hat{A} are added to the bottom of the band, as signified by the \uplus symbol (line 5), and the differing columns are moved to the leftmost positions within the interchangeable block that they occur in (lines 6-9). This can be easily accomplished if every path maintains integers x and y

indicating the position of fixed blocks of columns or rows. Lines 10-15 implement the exact same procedure, however the rows and columns are reversed because a lower neighbor is added to the path as opposed to an upper neighbor.

```

Input:  $\bar{P} = C^1, C^2, \dots, C^n$ 
Data:  $\mathbb{A} = \bigcup_{j=1}^n A^j, \mathbb{B} = \bigcup_{j=1}^n B^j$ 
Data:  $x, y$  : position of fixed column and row indices
Input: new concept  $C^{n+1}$ 
1 begin
2    $\hat{A} \leftarrow A^{n+1} \ominus A^n$  ;
3    $\hat{B} \leftarrow B^{n+1} \ominus B^n$  ;
4   if  $A^{n+1} \supset \mathbb{A}$  then
5      $\pi_i \leftarrow \pi_i \uplus \hat{A}$  ;
6     for  $b \in \hat{B}$  do
7       if  $pos(b) > x$  then
8         swap element at pos  $x$  in  $\pi_j$  with  $b$  ;
9          $x \leftarrow x + 1$  ;
10  else if  $B^{n+1} \supset \mathbb{B}$  then
11     $\pi_j \leftarrow \pi_j \uplus \hat{B}$  ;
12    for  $a \in \hat{A}$  do
13      if  $pos(a) > y$  then
14        swap element at pos  $y$  in  $\pi_i$  with  $a$  ;
15         $x \leftarrow x + 1$  ;
16 end

```

Procedure AddToPath

The AddToPath procedure illustrates how a path in $\Theta(\mathbb{K}_{ij})$ can be mapped to a sub-matrix of \mathbb{K}_{ij} that is at least partially banded; hence we refer to a sub-matrix $\widehat{\mathbb{K}}_{ij}^{\pi_i}_{\pi_j}$ and a path \bar{P} interchangeably. Due to the fact that each individual edge in \bar{P} is guaranteed to produce a banded structure, the only possible errors when adding a concept $C^{n+1} \succ C^n$ to \bar{P} occurs if a newly added row $a \in \hat{A}$ contains a column index b such that $pos(b) < x$, where x indicates the position of the fixed block of columns.

Proposition 13. Let $\bar{P}_{n-1} = C^1, \dots, C^n$ be a path constructed by repeated calls to procedure AddToPath with associated permutations π_i, π_j and integers x and y . Given concept C^{n+1} s.t.

C^{n+1} is augmented to \bar{P} via `AddToPath` then

$$e(\bar{P}_n) \leq \begin{cases} 0 & \text{if } n \leq 1 \\ e(\bar{P}_{n-1}) + \sum_{a \in \hat{A}} |\Psi^j(a) \cap \mathbb{B}| & \text{if } \mathbf{C}_{n+1} \succ \mathbf{C}_n \\ e(\bar{P}_{n-1}) + \sum_{b \in \hat{B}} |\Psi^i(b) \cap \mathbb{A}| & \text{if } \mathbf{C}_{n+1} \prec \mathbf{C}_n \end{cases}$$

where $\hat{A} = A^{n+1} \ominus A^n$, $\hat{B} = B^{n+1} \ominus B^n$, $\mathbb{A} = \bigcup_{k=1}^y a^{\pi_i^k}$ and $\mathbb{B} = \bigcup_{k=1}^x b^{\pi_j^k}$

Proof. We prove this by induction on n .

Base case, $n \leq 1$: Only one edge C^1, C^2 exists in the path, and without loss of generality we assume $C^1 \prec C^2$. In this case

$$\begin{aligned} \pi_i &= \{A^1, A^2 \ominus A^1\} \\ &= \{a^{\pi_i^1}, \dots, a^{\pi_i^{|A^1|}}, a^{\pi_i^{|A^1|+1}}, \dots, a^{\pi_i^{|A^1|+|A^2 \ominus A^1|}}\} \end{aligned}$$

and

$$\begin{aligned} \pi_j &= \{B^1 \ominus B^2, B^2\} \\ &= \{b_{\pi_j^1}, \dots, b_{\pi_j^{|B^1 \ominus B^2|}}, b_{\pi_j^{|B^1 \ominus B^2|+1}}, \dots, b_{\pi_j^{|B^1 \ominus B^2|+|B^2|}}\} \end{aligned}$$

By definition of a concept, rows $a^{\pi_i^1}, \dots, a^{\pi_i^{|A^1|}}$ contain 1s in all the columns $b_{\pi_j^1}, \dots, b_{\pi_j^{|B^1|}}$. By definition of the hierarchal order and set difference, the remaining rows $a^{\pi_i^{|A^1|+1}}, \dots, a^{\pi_i^{|A^1|+|A^2 \ominus A^1|}}$ contain zeros at precisely $b_{\pi_j^1}, \dots, b_{\pi_j^{|B^1 \ominus B^2|}}$ and ones at $b_{\pi_j^{|B^1 \ominus B^2|+1}}, \dots, b_{\pi_j^{|B^1 \ominus B^2|+|B^2|}}$. Thus by definition of banded matrices \bar{P} is fully banded and $e(\bar{P}_1) = 0$.

Inductive step: Assume true for paths up to length n After adding concept C^{n+1} the procedure only re-arranges column indices that are non-fixed and thus inter-changeable (lines 7-9) therefore not effecting the error. Moreover, every newly added row $a \in A^{n+1} \ominus A^n$ contains 1s in

exactly columns B^{n+1} which is a proper subset of B^n by the definition of the hierarchical order. Thus $\hat{B} = B^n \setminus B^{n+1}$ and all columns $b_{\pi_j^{x+1}}, \dots, b_{\pi_j^{|\pi_j|}}$ contain 1s which cause no errors. Hence by definition of banded matrices, only 1s occurring in a newly added row $a \in A^{n+1} \setminus A^n$ and a column $b_{\pi_j^1}, \dots, b_{\pi_j^x}$ can cause additional error as a result of adding C_{n+1} . These 1s can be precisely counted as $\sum_{a \in \hat{A}} |\Psi^j(a) \cap \mathbb{B}|$, so by the inductive theorem

$$e(\bar{P}_{n+1}) \leq e(\bar{P}_n) + \sum_{a \in \hat{A}} |\Psi^j(a) \cap \mathbb{B}| \quad (76)$$

□

8.3 MMBS Algorithm

Aggregating propositions 12, 13 and the `AddToPath` procedure we have developed the `MMBS` (Mine Maximally Banded Sub-matrix) algorithm. Proposition 12 stipulates that any banded sub-matrix of \mathbb{K}_{ij} can be enumerated as a sequence of concepts of \mathbb{K}_{ij} . Moreover, invoking `AddToPath` repeatedly provides a mechanism to construct a band from a path in the lattice of concepts, and proposition 13 is utilized to compute an upper bound on the error. Consider the graph $\Theta(\mathbb{K})$ of a concept lattice $\underline{\mathfrak{B}}(G, M, I)$, then any edge (C^i, C^j) can be weighted by the amount of error that would be introduced if C^j is added to the path $(\bar{P}) = C^1, \dots, C^i$. **Thus, identifying maximally ε -banded sub-matrices is equivalent to identifying all maximal paths in $\Theta(\mathbb{K})$ with total weight less than ε .** This problem differs from the all-pairs shortest path problem due to the fact that the edge weights are clearly non-constant; to the contrary, the edge weights are a function of all previously added edges along the current path \bar{P} . The `MMBS` algorithm consists of three major steps: 1) computing $\underline{\mathfrak{B}}(G, M, I)$ and $\Theta(\mathbb{K})$, 2) searching the paths of $\Theta(\mathbb{K})$, and 3) determining the top banded sub-matrices to output to the user.

Computing the concept lattice can be accomplished utilizing any of the previously mentioned incremental algorithms [13, 52, 49]; hence, we assume that $\Theta(\mathbb{K})$ is readily available.

8.3.1 Search Space

MMBS conducts a depth first search with backtracking in order to identify maximal ϵ -banded paths. Undoubtedly, in the worst case the search space is exponential in the size of the lattice, due to the number of potential paths. Fortunately, the error associated with a path in the lattice grows monotonically allowing the search to prune entire branches of the space whenever an edge that results in error greater than ϵ is encountered. Despite such pruning, the task of searching all paths is still indeed intractable. We present two heuristic arguments to make the problem approachable. First, let \mathbb{U} be the set of upper neighbors of the bottom element in $\mathfrak{B}(G, M, I)$, then each path is rooted at a bi-cluster $C \in \mathbb{U}$. In essence, this imposes a slight restriction on the possible row orderings since every band will begin with the rows contained in \mathbb{U} . However, it is these rows precisely that contain the largest sets of column indices, therefore allowing the greatest freedom to swap these indices as more concepts are added to the path. Next, we set each vertex to remember the minimum weight edge encountered throughout the search. Due to the monotonicity of the error measure, nodes that have been previously visited are only added to a new path if and only if the newly computed error is less than or equal to the minimum weight edge encountered previously in the search.

8.3.2 Determining top bands

Identifying all maximal ϵ -banded paths in the lattice is not very useful to the user, as there may exist an explosive number of such paths. In order to determine the most interesting banded submatrices we allow several parameters to be set in addition to ϵ . First, *minRows* and *minCols* parameters determine the minimum number of rows and columns any band should contain, and we use the parameter *w* to specify the relative weight that should be assigned to the error in a banded structure. Next *maxOvlp* specifies the maximum overlap allowed between any two bands. Computing the overlap between any two bands consists of computing the Jaccard coefficient of both the rows and columns and weighing each. Given path $\bar{P} = C^1, \dots, C^n$ Let $r(\bar{P})$ denote the

<pre> Input: $\mathfrak{B}(G, M, I), \Theta(\mathbb{K})$ Input: $\varepsilon, w, minRows, minCols, maxOvlp$ 1 begin 2 $\mathcal{W}[] \leftarrow \infty;$ // keep track of min error 3 $\mathcal{R}[] \leftarrow \emptyset;$ // store bands 4 foreach $C \in \mathbb{U}$ do 5 initiate path \bar{P} to C; 6 Search(\bar{P}); 7 end </pre>
--

Algorithm 18: MMBS

rows of \bar{P} and $c(\bar{P})$ the columns, then $ovlp(\bar{P}_1, \bar{P}_2)$ is:

$$c * J(r(\bar{P}_1), r(\bar{P}_2)) + (1 - c) * J(c(\bar{P}_1), c(\bar{P}_2))$$

where $0 \leq c \leq 1$ and $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Finally, a quality measure q is needed to rank candidate ε -banded paths that satisfy all other user parameters. At an intuitive level, the quality of a band should be monotonic in the size of the band and penalized for errors. Following this intuition we utilized the simple quality measure

$$q(\bar{P}) = |r(\bar{P})| * |c(\bar{P})| - w * e(\bar{P}) \tag{77}$$

8.4 Pseudocode and complexity

Pseudo code of MMBS and the search procedure appears as algorithm 18 and procedure Search . We focus on the Search procedure as most of the work is conducted there. The first for loop on line 4 iterates through all possible concepts that the current path \bar{P} can expand to. If the error of augmenting the path with candidate concept \hat{C} is less than ε and at no more than any previous edge weight, then the best edge weight of \hat{C} is updated and the path is expanded to include C (utilizing AddToPath). The search delves deeper in order to attempt to maximize the path (line 10), while

```

Input: Path:  $\bar{P} = C^1, \dots, C^n$ 
1 begin
2    $expand \leftarrow false$  ;
3    $\overline{Pcpy} \leftarrow \bar{P}$  ;
4   foreach  $\hat{C} \succ C^n \vee \hat{C} \prec C^n$  do
5     if  $\hat{C} \notin \bar{P} \wedge (\hat{A} \supset r(\bar{P}) \vee \hat{B} \supset c(\bar{P}))$  then
6        $x \leftarrow e(\bar{P}) - e(\bar{P} \cup \hat{C})$  ;
7       if  $e(\bar{P} \cup \hat{C}) \leq \varepsilon \wedge \mathcal{W}[\hat{C}] \geq x$  then
8          $\mathcal{W}[\hat{C}] \leftarrow x$ ;
9          $\bar{P} \leftarrow \bar{P} \cup \hat{C}$ ;
10        Search(  $\bar{P}$  ) ;
11         $\bar{P} \leftarrow \overline{Pcpy}$  ;
12         $expand \leftarrow true$  ;
13    if ! $expand$  then
14      if  $|r(\bar{P})| \geq minRows \wedge |c(\bar{P})| \geq minCols$  then
15        if  $maxOvl p(\bar{P}, \bar{P}_i \in \mathcal{R}) < maxOvl p$  then
16           $\mathcal{R} \leftarrow \mathcal{R} \cup \bar{P}$  ;
17        else
18          Let  $\bar{P}_x$  be path s.t.  $ovlp(\bar{P}_x, \bar{P}) \geq maxOvl p$ ;
19          if  $q(\bar{P}) > q(\bar{P}_x)$  then
20             $\mathcal{R} \leftarrow \mathcal{R} \cup \bar{P}$  ;
21 end

```

Procedure Search (\bar{P})

the copy on line 11 implements backtracking. If the *expand* flag is never set to true then the current path \bar{P} cannot be further expanded and the algorithm must evaluate if \bar{P} meets all the user defined parameters. If \bar{P} meets the *minRows* and *minCols* parameters, then it is compared to all previously mined bands to determine its degree of overlap (line 15). If the overlap between \bar{P} and all other bands \bar{P}_i does not exceed *maxOvl p* then it is added to \mathcal{R} . On the other hand, if the overlap of \bar{P} and \bar{P}_i does exceed the threshold, then the higher quality band is kept.

The three basic operations of **MMBS** are set difference, set intersection and swap operations. Let $X = \max |A^i \ominus A^j|$ for all $A^i \in C^i, A^j \in C^j$ s.t. $C^j \succ C^i$. Also let $Y = \max |B^i \ominus B^j|$ for all $B^i \in C^i, B^j \in C^j$ s.t. $C^j \prec C^i$. Augmenting a concept to a path utilizing the **AddToPath** procedure involves two set differences and at most $O(X)$ or $O(Y)$ swaps, while computing the error on line 6

involves at most $O(X)$ or $O(Y)$ set intersections. Clearly, `AddToPath` is invoked at least as many times as the error is computed; however the number of times this occurs is difficult to analyze, as it depends on several factors such as ϵ , and the actual structure of the lattice. Nevertheless, the error will never be computed more than $O(|L|)$ times. Lastly, the search procedure invoked $|\mathbb{U}|$ times, thus the total cost of `MMBS` is

$$O(|\mathbb{U}| \times |E| \times \max\{X, Y\}) \quad (78)$$

The memory footprint of `MMBS` is minimal, only a single path needs to be maintained in memory throughout the search, while $\Theta\mathbb{K}$ can be maintained on disk or in main memory. Moreover, our experimental results indicate that it is very plausible to maintain $\Theta\mathbb{K}$ in the main memory comfortably for even moderately large matrices (4000×4000) as 0-1 datasets tend to be sparse.

8.4.1 Speeding up the algorithm

The dominant term in equation 78 is clearly $|L|$, while if $|\mathbb{U}|$ is large then the computational cost is quite expensive. Unfortunately, many of the bi-clusters in $|\mathbb{U}|$ tend to be very similar, and the `Search` procedure often does not yield any new bands, but consumes computation time. Thus, in order to speed up the algorithm, we introduce a pre processing step to eliminate similar concepts and reduce the number of times the `Search` procedure is invoked. Specifically, $ovlp(C^1, C^2)$ is computed for all concepts $C^1, C^2 \in \mathbb{U}$, $C^1 \neq C^2$, if $ovlp(C^1, C^2) > maxOvlp$ then the larger concept is kept. In all performance tests, this pre-processing step accelerated the computation time dramatically (see next section) while producing very comparable results to our original `MMBS` algorithm.

8.5 Experimental Results

In this section we illustrate the efficacy of `MMBS` experimentally. The performance of `MMBS` is compared with the `MBS` algorithm proposed in [37]. We show that `MMBS` consistently outperforms `MBS` in three different ways:

1. `MMBS` uncovers several banded structures as opposed to a single band mined by `MBS`.

2. MMBS consistently uncovers higher quality bands.

3. More scalable computation time.

These results are exhibited with both synthetic and real world datasets. MMBS was implemented in C++ utilizing the STL data structures. All experiments were conducted on a 2.7 GhZ AMD Athlon 64 x2 CPU with 5.8 Gb of RAM, running Ubuntu Linux. Additionally, MMBS_Fast was implemented, in which the pre-processing step described above was applied. Source code for MBS was kindly provided by the authors of [37].

8.5.1 Synthetic datasets

Several synthetic datasets were created to test the capability of MMBS . Three different methods were utilized to create three different classes of datasets. First, single band datasets were created utilizing the method described in [37]. Briefly, for a given a number of rows ($|\mathbf{G}_i|$), columns ($|\mathbf{G}_j|$) and width parameter w a fully banded matrix is generated by means of a random walk starting at the (0,0) coordinate. Initially all cells in the matrix are set to 0, and the walk chooses to either step down or to the right with equal probability. On a step to the right the w cells above and below the current position are all set to 1s. Noise can be added to the band by flipping the original values to 0 or 1 with probabilities p and q . Next, multiple band matrices were created by first producing several single band matrices K_1, \dots, K_n , followed by "concatenating" the single bands into a single "switch matrix" as:

$$\begin{pmatrix} \dots & \dots & 0 & K_1 \\ \dots & \dots & K_2 & 0 \\ 0 & \dots & 0 & \vdots \\ K_n & 0 & \dots & \vdots \end{pmatrix}$$

Finally, random binary matrices with a set sparsity level were also created.

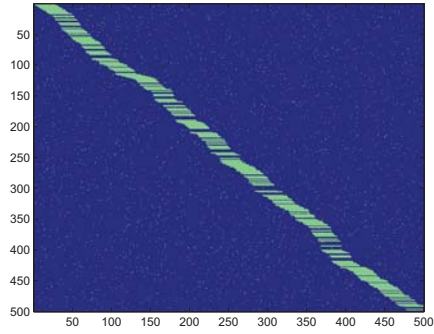
Three sets of experiments were conducted on each class of synthetic data. All experiments

Dataset name	Dataset Size	p	Num. Planted bands	Algorithm	Quality top ranked	Num. bands mined
SynBand100_001	100 × 100	0.01	1	MMBS	3590	6
				MMBS_Fast	3406	4
				MBS_BD	2507	1
				MBS_SD	438	1
SynBand100_005	100 × 100	0.05	1	MMBS	2278	9
				MMBS_Fast	1503	8
				MBS_BD	1050	1
				MBS_SD	1201	1
SynBand500_001	500 × 500	0.01	1	MMBS	8918	7
				MMBS_Fast	8261	6
				MBS_BD	2822	1
				MBS_SD	2145	1
SynMultiBand100_001	100 × 100	0.01	2	MMBS	3367	2
				MMBS_Fast	3367	2
				MBS	4101	1
				MBS_SD	4045	1
SynMultiBand100_001	100 × 100	0.05	2	MMBS	4054	2
				MMBS_Fast	3933	2
				MBS_BD	3910	1
				MBS_SD	3736	1
SynMultiBand500_001	500 × 500	0.01	2	MMBS	28242	8
				MMBS_Fast	21346	5
				MBS_BD	17498	1
				MBS_SD	430	1
SynRandom100_005	100 × 100	0.05	unknown	MMBS	3311	17
				MMBS_Fast	3220	14
				MBS_BD	2801	1
				MBS_SD	1949	1
SynRandom500_001	500 × 500	0.01	unknown	MMBS	18635	73
				MMBS_Fast	16163	64
				MBS_BD	16771	1
				MBS_SD	5229	1

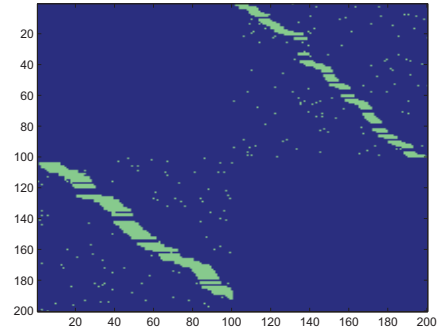
Figure 45: Band quality in synthetic data

were conducted with $w = 1$, $maxOvlp = 0.1$, $minRows = minCols = 5$, and $\epsilon = 99$ (the maximum allowed misplaced 1s or 0s was 99). The `MBS_BD` algorithm allows for bi-directional flipping of both 0s to 1s and 1s to zeros, while `MBS_SD` only allows for flipping 1s to 0s. Several possible initial orderings are possible with the `MBS` algorithms, including Hamiltonian ordering with distance measures and spectral ordering with similarity measures; all orderings were utilized in the experiments and the best results are reported here for comparison.

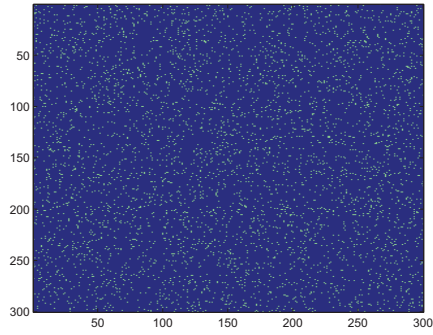
Figure 45 shows the results, and as can be seen `MMBS` and `MMBS_Fast` consistently discover higher quality bands. In the single band case, we see that `MMBS` is more resilient to noise and is able to discover a larger portion of the hidden band; this is a direct consequence of not fixing the column permutations. `MBS_BD` and `MBS_SD` discover bands of higher quality on two of the multi-band class datasets; however upon closer inspection we see that this is due to a failure of these algorithms to recognize that two segmented bands exist in the data, (as can clearly be seen from figure 46(b)). The `MBS` algorithms attempt to lump all the rows into one band, resulting in a larger band than any of the individual segmented bands, but does not complete the job and permute the matrix into a single band. In other words the band produced by `MBS` looses a very natural real world interpretation of two segmented bands. On the other hand, the `MMBS` algorithms was able to discover the two segments almost completely, both of which had very similar quality. Moreover,



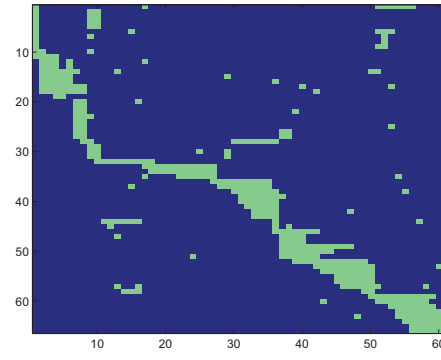
(a) Single band with noise



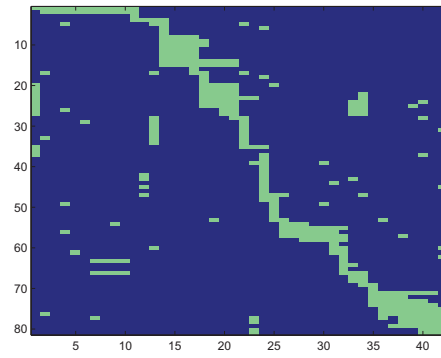
(b) Two segmented bands



(c) Random matrix



(d) One of the segmented bands found by MMBS



(e) Another segmented band found by MMBS

Figure 46: Synthetic data and experimental results for banded matrices

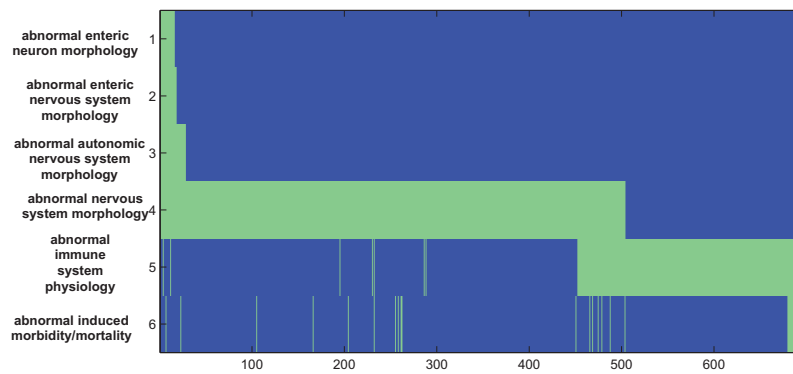
Dataset	Size	Sparsity	Algorithm	Quality top ranked	Num. bands mined
Genes_Phenotypes	1910×3965	0.008	MMBS	6665	56
			MMBS_Fast	6665	43
			MBS_BD	5204	1
			MBS_SD	3578	1
Genes_Drugs	1608×49	0.042	MMBS	6423	18
			MMBS_Fast	6423	13
			MBS_BD	5346	1
			MBS_SD	3047	1
NewsGroups_Mideast_Religion	2000×890	0.003	MMBS	72906	42
			MMBS_Fast	61410	31
			MBS_BD	59781	1
			MBS_SD	58713	1
NewsGroups_AllPC	5000×2805	0.0001	MMBS	93368	5
			MMBS_Fast	93368	5
			MBS_BD	89106	1
			MBS_SD	74125	1

Figure 47: Band quality comparison on real-world data

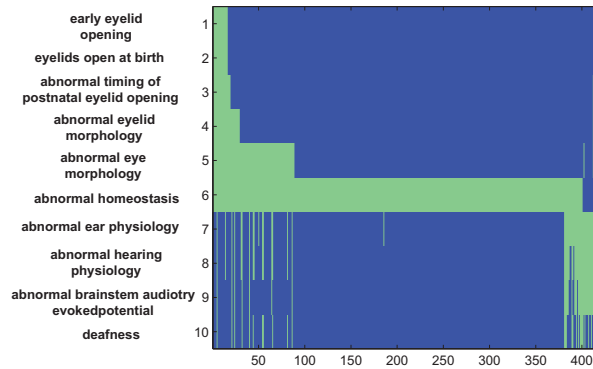
in the largest of the multiple banded datasets the MMBS algorithm discovered both the segmented bands even though the quality of only one of them exceeds the quality of the single band mined by MBS. Finally, the random binary matrices represent the real world scenario of not knowing if a banded structure exists in the data. Once again MMBS algorithms outperform MBS and discovers larger bands, with the same limit of error allowed.

8.5.2 Real world datasets

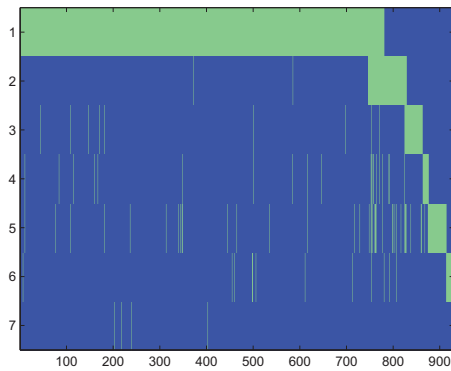
Four real world datasets from differing domains were utilized to examine the true utility of MMBS. The first two datasets, **GenePhenoTypes** and **Genes.Drugs** came from the bioinformatics domain. The rows of each matrix correspond to genes, while the columns to phenotypes and drugs respectively. The final two datasets were obtained from the large **NewsGroups** dataset [7], with rows corresponding to usenet documents and columns to words extracted from the subject lines. **Mideast.Religion** contains documents found in both the Mideast and Religion groups, while **All_PC** contains usenet documents from all of the PC groups. All algorithms were executed with the same parameters discussed above, and the results appear in figure 47. Once again, we notice that MMBS algorithms not only consistently produce better quality bands, but also discover several banded structures in the data. Figure 48 illustrates the data matrices and the resulting bands discovered by MMBS. Investigating figures 48(a) and 48(b) reveals the advantage of mining multiple banded sub-matrices as opposed to a single band. In figure 48(a) the phenotypes all correspond to abnormal nervous system phenomenon; on the other hand, in figure 48(b) the first five phenotypes are abnormalities in the eyes and eyelids, while the last phenotypes indicate abnormalities in



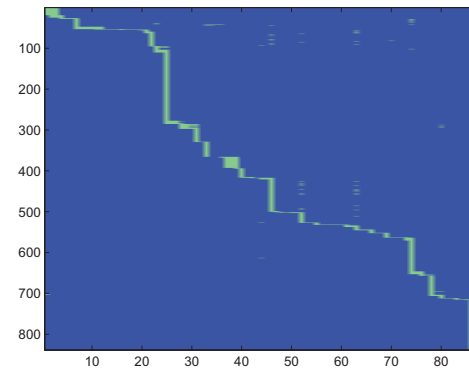
(a) Genes_Phenotypes



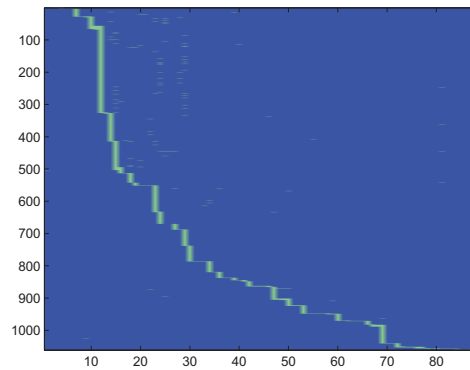
(b) Genes_Phenotypes



(c) Gene_Drugs



(d) NewsGroups_Mideast_Religion



(e) NewsGroups_AllIPC

Figure 48: Real world datasets and mined sub-matrix bands

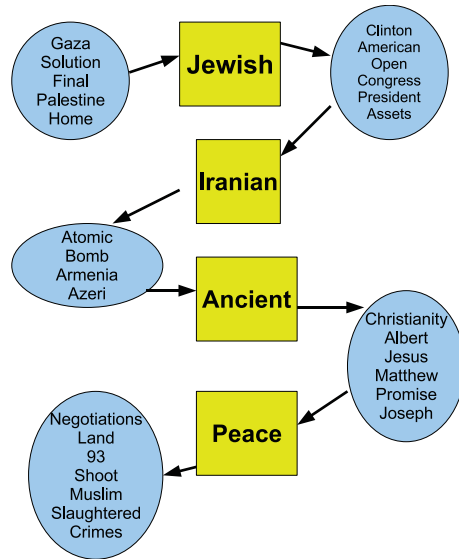
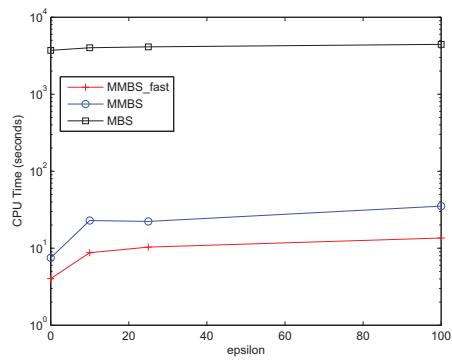


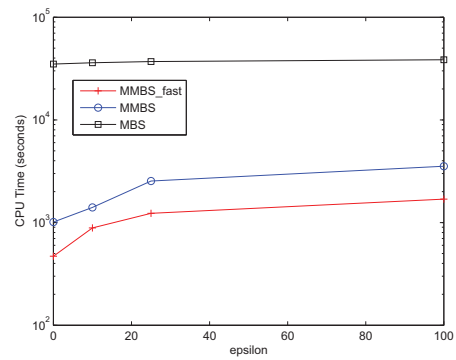
Figure 49: Natural interpretation of band as overlapping communities

the ears. In both cases, the banded structure uncovers the overlapping roles of genes in different phenotypes, but the fact that they are two separate bands emphasizes the correlation between the phenotypes within each submatrix.

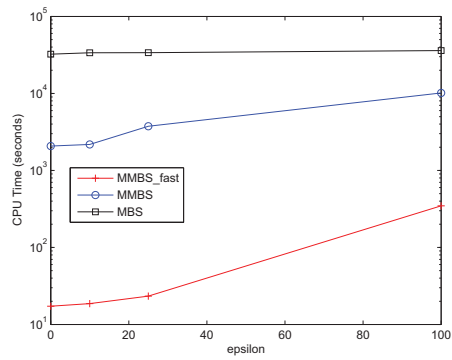
Utilizing the mined band from the **Mideast_Religion** we were able to construct a graph that successfully illustrated different themes discussed among usenet users (figure 49). Specifically, collections of keywords and documents were constructed by following the permutations of the band until a row(s) or column(s) was encountered that was longer than average. All documents up to this point were placed into a collection (blue circles), while the longer than average row(s) or column(s) were placed in the yellow boxes with the bold font. For ease of interpretation we have only included keywords in the graph. At an intuitive level we can see that within each collection (circles) the keywords are highly correlated around a similar theme. At the same time, keywords across collections are quite distinct, however, keywords in the boxes represent overlapping themes that links the two distinct collections of documents and keywords. Thus the banded structure carries with it a natural interpretation of the distinct discussion threads in the newsgroup, along



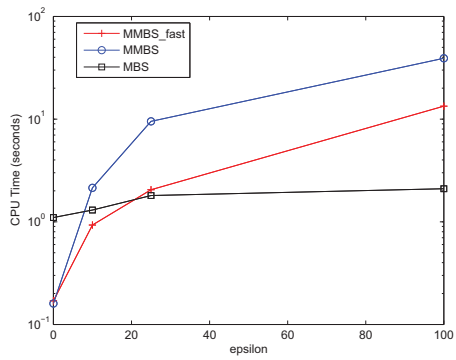
(a) NewsGroups_Mideast_Religion



(b) NewsGroups_AllPC



(c) Genes_Phenotypes



(d) Gene_Drugs

Figure 50: Performance Study on MMBS

with possible links between these distinct threads.

8.5.3 Performance Tests

Performance tests to measure the practical computational cost of \mathbb{MBS} , \mathbb{MBS}_{Fast} , and \mathbb{MBS} were conducted on the real world datasets. Each algorithm was executed ten times, while the ϵ -parameter was varied, and the average CPU times are reported in figure 50. We made use of the $\mathcal{CHARM-L}$ algorithm [58] to compute the concept lattice of each dataset and the cost of this computation is included in the results. Noticeably, the cost of \mathbb{MBS}_{Fast} is significantly lower than the other algorithms in three of the four datasets. In all three of these cases \mathbb{MBS}_{Fast} outperforms \mathbb{MBS} by an order of magnitude, and \mathbb{MBS} by two orders of magnitude. The \mathbb{MBS} algorithm proved not to be very sensitive to the ϵ parameter resulting in more efficient performances in smaller datasets such as the *Gene_Drugs* dataset. On the other hand, both \mathbb{MBS} and \mathbb{MBS}_{Fast} are very sensitive to ϵ . As ϵ increases, less pruning steps occur and the search procedure tends to the worst case cost of exploring all edges of the lattice. Despite this fact, both the \mathbb{MBS} algorithms scaled much better to the three larger datasets than \mathbb{MBS} , as even the original \mathbb{MBS} outperformed \mathbb{MBS} by at least an order of magnitude in each case.

8.6 Conclusion

This chapter explored the connection between bi-clustering and banded structures in binary data. It was shown that banded sub-matrices of a dataset correspond to paths in the concept lattice of that context. Based upon this correspondence mining bands in the data can be accomplished via traversing the paths of the concept lattice. Moreover, when forming bands in this manner an upper bound on the error found with the band can be established. These facts formed the basis of the \mathbb{MBS} algorithm which discovers maximally ϵ -banded sub-matrices by formulating a search and bound on paths in the concept lattice.

Experiments on synthetic and real-world datasets indicated three main advantages of \mathbb{MBS} over previous algorithms. First, multiple banded structures with natural interpretations are uncovered

in the data as opposed to a single structure. Additionally, `MBS` consistently mined higher quality bands, while the performance study illustrated that the computational cost scaled better to larger datasets.

9 Conclusion and Future Work

The body of knowledge concerning clustering has focused on data represented as feature vectors in a single dataset. However, modern, real-world applications encompass multiple interrelated datasets modeled as multi-domain information networks. In the context of knowledge discovery, clustering in such networks accomplishes two main objectives: 1) unveiling associations amongst objects from differing domains and 2) improving clustering of objects within a single domain. However, the problem is challenging for two reasons: First, the data is inherently high dimensional leading to the curse of dimensionality. Second, unlike clustering in a single dataset, the objects of each domain must be clustered simultaneously and only in terms of the available relational information.

In this dissertation, we established Formal Concept Analysis as a solid theoretical model for information network clustering. The FCA model of bi-clustering was expanded to an information network by introducing the notion of an n -concept. It was proven that an n -concept is a generalization of a bi-cluster (2-concept) and that the set of n -concepts form a complete lattice. The search strategies of closed itemset mining algorithms and simultaneous pruning of datasets were combined to produce the `NClu` algorithm. It was shown, theoretically, that the total computational cost of `NClu` is no greater than that of a closed itemset mining algorithm. Furthermore, we illustrated experimentally that in real-world data `NClu` actually outperforms closed itemset mining algorithms (in terms of computation cost) by orders of magnitude. However, further experiments revealed that while n -concepts constitute very highly correlated and accurate clusters, the method also suffers in terms of recall.

Chapter 5 addressed the shortcomings of the n -concept model by relaxing the strict clustering criterion. The measures of connectivity and self-connectivity were defined to capture and quantify the intuition that information network clusters should exhibit a local clustering tendency, while maintaining global connectivity. Utilizing these measures, n -clusters are defined as maximal subspaces that exceed the expected degree of connectivity and self-connectivity. The `TreeClu` algorithm, developed to mine n -clusters, took advantage of several properties of the FCA model to

enumerate ideal local candidate clusters in each dataset, then aggregate the result to finally form high quality information network clusters. Experimental comparisons with the current state-of-the-art algorithms MDC and NetClus revealed that TreeClu produced superior clustering results in real-world data with both overlapping and non-overlapping natural categories. Chapter 6 proposed a framework for defining information network clusters in terms of game theory. Modeling the clustering problem as a game in which players attempt to maximize their reward, clusters are defined as the Nash equilibria solution concepts of the game. This framework presents a unifying definition of an information network cluster, while allowing for domain knowledge to be incorporated via specifying the rules of the game and the reward function. Utilizing an intuitive reward function, we illustrated that the framework encompasses previous formulations of bi-clustering and information network clustering based on FCA. The advantage of such a framework, is that while different clustering criterion and quality measures are incorporated to satisfy specific application needs, the underlying definition of a cluster remains the same. In this manner a generic algorithm for locating Nash equilibria maybe utilized for enumerating clusters despite different clustering criterion in different applications.

Chapters 7 and 8 focused on information networks containing a single edge, or single dataset, but two different domains. We proposed the new data mining task of enumerating significant distinguishing sets. The goal of this task is to discover sets of attributes or objects that most distinguish the concepts of a single dataset from each other. A motivating application for this task came from bioinformatics in which datasets capturing the interaction between genes and transcription factors were available. Comparing each concept in this dataset with an immediate parent tells a biologist the difference in activation of genes and / or transcription factors that will transform one cellular process into a different one. Discovering the most significant differences that transform cellular processes is of great interest to current biological research. Considering the concepts as maximal zero-variance sub-matrices of the dataset, a metric that measures the degree of shape change amongst concepts is utilized to weigh the edges of the concept lattice. Finally, the MIDS algorithm incrementally grows a maximal cost spanning tree in the concept lattice to identify the

most significant distinguishing sets. Experiments in synthetic and real-world datasets indicated the utility of distinguishing sets as `MIDS` extracted attributes that separated classes of labeled data in a completely unsupervised setting.

Chapter 8 explored the connection between bi-clustering and banded structures in binary data. Interest in banded structures of matrices has grown due to the natural interpretations of a banded structure. These include overlapping communities in social networks, overlapping roles of genes in various diseases, and patterns of species occurring in spatially correlated sites. It was shown that banded sub-matrices of a dataset correspond to paths in the concept lattice of the dataset. Utilizing this correspondence, the `MMBS` algorithm was developed to uncover banded structures via intelligent search of the concept lattice. `MMBS` consistently discovered larger bands, utilizing less computational time than the state-of-the-art `MBS` algorithm [37]. Moreover, results obtained from genomic and textual data revealed meaningful bands with natural interpretations in each respective field.

9.1 Future Work

Several avenues for future work are available. In the sequel, we break these opportunities in terms of the major topics covered throughout each chapter.

9.1.1 N-Concept Lattices

The `NClu` algorithm efficiently enumerates the n -concepts of a star-shaped information network, however the actual lattice structure is not unveiled. An immediate area for future work is to develop efficient algorithms for mining the lattice structure of the n -concepts. Even further, an incremental lattice enumeration algorithm would be of great utility as such methodologies could be utilized for either explicitly revealing the lattice structure or implicitly computing only neighboring n -concepts within the context of a bottom-up clustering algorithm such as `TreeClu`.

9.1.2 Information Network Clustering

The most obvious addition to this dissertation is enhancing the current methods to deal with weighted information networks. For example, in text data as opposed to a simple binary relation between words and documents a TF-IDF weighting scheme may be used. In this case, concepts may be defined as statistically significant maximal bi-cliques, while weighted version of connectivity and self-connectivity could also be developed. An interesting study would be to see how a direct generalization of the `TreeClu` algorithm would perform. Moreover, we believe that the game theoretic framework would truly prove its utility in this case, as clusters may still be defined in terms of Nash equilibria, while adjusting the rules of the game to account for the weighted edges.

9.1.3 Distinguishing Sets

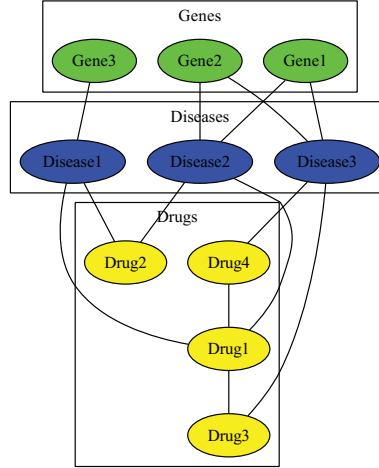
Distinguishing sets were established only for incremental concepts. Defining significant distinguishing sets between neighborhoods of concepts is a logical next step. Moreover, future work may explore how such a scheme could in fact be outfitted as a clustering algorithm, where the degree of shape change can be viewed as a dissimilarity measure among concepts.

9.1.4 Banded Matrices

Future work will focus on more efficient methods of searching the concept lattice for banded structures through stronger bounding criterion on the error and more effective heuristics. In addition, quantitative measures of bandedness should be developed to bias the `MMBS` algorithm towards finding different types of banded structures relevant to different applications.

A Computing Expected Self-Connectivity

The expected value of self-connectivity of two objects within a domain depends on the expected number of objects they are commonly connected to in the entire information tree. Given a single object $g_i \in \mathbf{G}_i$, the prior probability of a link existing between g_i and random $g_k \in \mathbf{G}_k$ depends on the prior probability of a link existing between g_i and random $g_j \in \mathbf{G}_j$, where \mathbf{G}_j is a neighboring domain node of \mathbf{G}_i and along the shortest path from \mathbf{G}_i to \mathbf{G}_k . For example in the figure below, the probability that a link exists between the object *Drug2* and a random gene depends on the probability that a link exists between objects *Disease1* and *Disease2* and some random gene. Hence, the prior probability of a link existing between an object $g_i \in \mathbf{G}_i$ and a random object in



domain \mathbf{G}_k is dependent upon $\varphi^j(g_i)$, where \mathbf{G}_j is a neighboring domain of \mathbf{G}_k and is along the shortest path connecting \mathbf{G}_i and \mathbf{G}_k . Specifically, let $A_1, \dots, A_{|\varphi^j(g_i)|}$ denote the events that objects $g_j^1, \dots, g_j^{|\varphi^j(g_i)|}$ is linked to a single random object in \mathbf{G}_k . Thus, the prior probability of g_i being connected to a single random object in \mathbf{G}_k is $P(A_1 \cup A_2 \cup \dots \cup A_{|\varphi^j(g_i)|})$. Computing this probability requires use of the inclusion-exclusion principle:

$$\begin{aligned}
 P\left(\bigcup_{i=1}^n A_i\right) &= \sum_{i=1}^n P(A_i) - \sum_{i,j:i < j} P(A_i \cap A_j) \\
 &+ \sum_{i,j,k:i < j < k} P(A_i \cap A_j \cap A_k) - \dots + (-1)^{n-1} P\left(\bigcap_{i=1}^n A_i\right)
 \end{aligned} \tag{79}$$

In the context of the `TreeClu` algorithm, computing equation 79 is quite cumbersome and expensive. This is especially true given the fact that expected self-connectivity must be computed at each iteration of each refinement phase. One possibility, is to utilize the upper-bound of the inclusion exclusion principle given by Boole’s inequality:

$$P\left(\bigcup_i A_i\right) \leq \sum_i P(A_i) \quad (80)$$

Given that real-world information networks tend to contain sparse datasets utilizing Boole’s inequality is an attractive prospect. However, the computational cost would still be quite expensive as $P(A_i)$ still depends upon computing $\phi^j(g_i)$ for each object g_i . In the worse case, this entails $O(|V| * |\mathbf{G}|)$ operations per object or $O(|V| * |\mathbf{G}|^2)$ per subspace.

References

- [1] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD ’99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 61–72, New York, NY, USA, 1999. ACM.
- [2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *In SIGMOD*, pages 70–81, 2000.
- [3] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD ’93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA, 1993. ACM.
- [4] F. Alqadah and R. Bhatnagar. An effective algorithm for mining 3-clusters in vertically partitioned data. In *CIKM ’08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1103–1112, New York, NY, USA, 2008. ACM.

- [5] F. Alqadah and R. Bhatnagar. Discovering substantial distinctions among incremental bi-clusters. In *SDM*, 2009.
- [6] S. M. Arindam Banerjee, Sugato Basu. Multi-way clustering on relation graphs. In *Proceedings of the SIAM International Conference on Data Mining*, 2007.
- [7] A. Asuncion and D. Newman. UCI machine learning repository, 2007.
- [8] H. A.V. Methods for demonstrating resemblance in taxonomy and ecology. *Nature*, 214:830–831, 1967.
- [9] C. Aykanat and A. Pinar. Permuting sparse rectangular matrices into block-diagonal form. *SIAM Journal on Scientific Computing*, 25:1860–1879, 2004.
- [10] H. D. Ball G.H. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12:153–155, 1967.
- [11] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 509–514, New York, NY, USA, 2004. ACM.
- [12] S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discoverey*, 5:213–246, 2001.
- [13] P. Becker, J. Hereth, and G. Stumme. Toscanaj - an open source tool for qualitative data analysis. In V. Duquenne, B. Ganter, M. Liquiere, E. M. Nguifo, and G. Stumme, editors, *Advances in Formal Concept Analysis for Knowledge Discovery in Databases. Proc. Workshop FCAKDD of the 15th European Conference on Artificial Intelligence (ECAI 2002). Lyon, France.*, July 23 2002.

- [14] R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 41–48, New York, NY, USA, 2005. ACM.
- [15] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*, 2002.
- [16] A. Berry, J.-P. Bordat, and A. Sigayret. A local approach to concept generation. *Annals of Mathematics and Artificial Intelligence*, 49:117–136, 2007.
- [17] I. Bhattacharya and L. Getoor. Relational clustering for multi-type entity resolution. In *MRDM '05: Proceedings of the 4th international workshop on Multi-relational mining*, 2005.
- [18] H. Bian and R. Bhatnagar. A levelwise algorithm for interesting subspace clusters. *Proceedings of the 2005 IEEE International Conference on Data Mining*, 2005.
- [19] K. Bryan. Biclustering of expression data using simulated annealing. In *CBMS '05: Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, 2005.
- [20] S. Busygin, G. Jacobsen, E. Krmer, and C. Ag. Double conjugated clustering applied to leukemia microarray data. In *In 2nd SIAM ICDM, Workshop on clustering high dimensional data*, 2002.
- [21] S. Busygin, O. Prokopyev, and P. M. Pardalos. Biclustering in data mining. *Comput. Oper. Res.*, 35(9):2964–2987, 2008.
- [22] P. K. A. T. M. Cecilia M. Procopiuc, Micahel Jones. A monte carlo algorithm for fast projective clustering. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 418–427, 2002.
- [23] Y. Cheng and G. Church. Biclustering of expression data. In *8th International Conference on Intelligent Systems for Molecular Biology*, 2000.

- [24] Z. Y. Cheng C. H., Fu A. W. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the Fifth ACM SIGKDD International Conference*, pages 84–93, 1999.
- [25] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172, New York, NY, USA, 1969. ACM.
- [26] P. A. Daniel Crabtree and X. Gao. Qc4- a clustering evaluation method. *Advances in Knowledge Discovery and Data Mining*, 4426:59–70, 2007.
- [27] L. Dehaspe and H. T. T. Y. Discovery of relational association rules. In *Relational data mining*, pages 189–212. Springer-Verlag, 2001.
- [28] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.
- [29] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *KDD*, 1999.
- [30] S. Džeroski. Multi-relational data mining: an introduction. *SIGKDD Explor. Newsl.*, 5(1):1–16, 2003.
- [31] J. A. Enrique Amig, Julio Gonzalo and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, Online, 2008.
- [32] L. M. Everitt B., Landau S. *Cluster Analysis*. Arnold, London, 2001.
- [33] K. S. G. Liu and J. Li. Efficient mining of large maximal bicliques. *Dawak*, pages 437–448, 2006.
- [34] B. Gamter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, 1999.

- [35] B. Ganter, G. Stumme, and R. Wille, editors. *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science*. Springer, 2005.
- [36] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus-clustering categorical data using summaries. In *Proc. of ACM SIGKDD*, pages 73–83, 1999.
- [37] G. C. Garriga, E. Junttila, and H. Mannila. Banded structure in binary matrices. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 292–300, New York, NY, USA, 2008. ACM.
- [38] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 73–84, June 1998.
- [39] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.
- [40] R. B. H. Bian. An algorithm for lattice-structured subspace clustering. *Proceedings of the SIAM International Conference on Data Mining*, 2005.
- [41] A. C. H. Nagesh, S. Goil. Adaptive grids for clustering massive data sets. *SIAM Data Mining Conf*, 2002.
- [42] R. B. Haiyun Bian and B. Young. An efficient constraint-based closed set mining algorithm. *Proceedings of the sixth international conference on Machine Learning*, pages 172–177, 2007.
- [43] J. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67, No. 337:123–129, 1972.
- [44] S. M. I. S. Dhillon and D. S. Modha. Information-theoretic co-clustering. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 89–98, 2003.

- [45] I. jen Lin, M. K. Sen, and D. B. West. Classes of interval digraphs and 0,1-matrices. In *28th S.E. Conf.Comb.Graph.Th. and Congr.Numer*, 1997.
- [46] M. F. Kai Puolamaki and H. Mannila. Seriation in paleontological data using markov chain monte carlo methods. *PLoS Comput Biol.*, 2, 2006.
- [47] M. J. Z. Karlton Sequeira. Schism: A new approach to interesting subspace mining. *International Journal of Business Intelligence and Data Mining*, 1 (2):137–160, 2005.
- [48] A. J. Knobbe and E. K. Y. Ho. Maximally informative k-itemsets and their efficient discovery. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 237–244, New York, NY, USA, 2006. ACM.
- [49] S. O. Kuznetsov and S. A. Obiedkov. Algorithms for the construction of concept lattices and their diagram graphs. In *PKDD '01: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 289–300, London, UK, 2001. Springer-Verlag.
- [50] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:12, 2000.
- [51] J. Li, G. Liu, H. Li, and L. Wong. Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: A one-to-one correspondence and mining algorithms. *IEEE Trans. Knowl. Data Eng.*, 19(12):1625–1637, 2007.
- [52] C. Lindig. Fast concept analysis. *8th International Conference on Conceptual Structures*, 2000.
- [53] B. Long, X. Wu, Z. M. Zhang, and P. S. Yu. Unsupervised learning on k-partite graphs. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 317–326, New York, NY, USA, 2006. ACM.

- [54] B. Long, Z. M. Zhang, X. Wú, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 585–592, New York, NY, USA, 2006. ACM.
- [55] O. A. Madeira S.C. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1 (1):24–45, 2004.
- [56] H. Mannila and E. Terzi. Nestedness and segmented nestedness. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 480–489, New York, NY, USA, 2007. ACM.
- [57] E. Mendelson. *Introducing Game Theory and Its Applications*. Chapman & Hall / CRC, 2004.
- [58] C.-J. H. Mohammed J. Zaki. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17 (4), 2005.
- [59] I. A. T. S. Mohammed J. Zaki, Markus Peters. Clicks: An effective algorithm for mining subspace clusters in categorical datasets. *Data and Knowledge Engineering special issue on Intelligent Data Mining*, 60 (2):51–70, 2007.
- [60] K. G. Mohammed J. Zaki. Fast vertical mining using diffsets. *9th International Conference on Knowledge Discovery and Data Mining*, 2003.
- [61] M. O. Mohammed J. Zaki. Theoretical foundations of association rules. *3rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, 1998.
- [62] G. Owen. *Game Theory*. Academic Press, 1995.
- [63] L. H. Parsons L., Haque E. Subspace clustering for high dimensional data: A review. *ACM SIGKDD Explorations Newsletter*, 6 (1):90–105, 2004.
- [64] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Appl. Math.*, 131:651–654, 2003.

- [65] R. Pensa and J. Boulicaut. Towards fault-tolerant formal concept analysis. *Congress of the Italian Association for Artificial Intelligence AI* IA*, 3673:21–23, 2005.
- [66] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a nash equilibrium. In *Games and Economic Behavior*, pages 664–669, 2004.
- [67] D. G. P. R. R. Agrawal, J. Gehrke. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Rec.*, 27, Number 2:94–105, 1997.
- [68] R. Rosen. Matrix bandwidth minimization. In *Proceedings of the 1968 23rd ACM national conference*, pages 585–595, New York, NY, USA, 1968. ACM.
- [69] A. Satsangi and O. R. Zaiane. Contrasting the contrast sets: An alternative approach. In *IDEAS '07: Proceedings of the 11th International Database Engineering and Applications Symposium*, 2007.
- [70] M. Sen and B. K. Sanyal. Indifference digraphs: A generalization of indifference graphs and semiorders. *SIAM J. Discret. Math.*, 7(2):157–165, 1994.
- [71] K. Sim, J. Li, V. Gopalkrishnan, and G. Liu. Mining maximal quasi-bicliques to co-cluster stocks and financial ratios for value investment. *Data Mining, IEEE International Conference on*, 0:1059–1063, 2006.
- [72] L. S.P. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:84–95, 1980.
- [73] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: Integrating clustering with ranking for heterogeneous information network analysis. In *Proc. 2009 Int. Conf. on Extending Data Base Technology (EDBT'09)*, 2009.
- [74] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proc. 2009 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'09)*, 2009.

- [75] H. A. T. Uno, M. Kiyomi. Lcm ver 3.: Collaboration of array, bitmap and prefix tree for frequent itemset mining. *Proceedings of the 2005 ACM OSDM Open Source Data Mining Workshop*, 2005.
- [76] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. In *In Proceedings of ISMB 2002*, 2002.
- [77] K. K. Theodoridis S. *Pattern Recognition*. Academic Press, San Diego, CA, 2006.
- [78] G. I. Webb, S. Butler, and D. Newlands. On detecting differences between groups. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.
- [79] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee. Findit: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, March 2004.
- [80] P. S. Y. X. Yin, Jiawei Han. Linkclus: Efficient clustering via heterogeneous semantic links. In *VLDB'2006: Proceedings of the 32nd international conference on Very large data bases*, 2006.
- [81] J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. In *BIBE '03: Proceedings of the 3rd IEEE Symposium on BioInformatics and BioEngineering*, 2003.
- [82] S. Yevtushenko. Conceptexplorer. <http://conexp.sourceforge.net/> .
- [83] X. Yin, J. Han, and P. S. Yu. Cross-relational clustering with user's guidance. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005.
- [84] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

- [85] L. Zhao and M. J. Zaki. Tricluster: an effective algorithm for mining coherent clusters in 3d microarray data. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005.