CLUSTERING WITH CLUSTER-LEVEL CONSTRAINTS

by

Rong Ge
B.Sc., Beijing Institute of Technology, 1997
M.Sc., Michigan Technological University, 2002

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
2008

APPROVAL

Name:	Rong Ge
Degree:	Doctor of Philosophy
Title of thesis:	Clustering With Cluster-level Constraints

Examining Committee: Dr. Jiangchuan Liu
Chair

Dr. Martin Ester
Associate Professor of Computing Science
Senior Supervisor

Dr. Binay Bhattacharya
Professor of Computing Science
Supervisor

Dr. Ramesh Krishnamurti
Professor of Computing Science
SFU Examiner

Dr. Christian Böhm, University of Munich
Associate Professor of Informatics
External Examiner

Date Approved: Jan 7, 2008

Abstract

The task of clustering is to group data objects into clusters which exhibit internal cohesion and external isolation. The generated clusters provide useful knowledge to support decision making in many applications. However, clustering methods may fail to discover satisfactory results due to the lack of user involvement, especially in the form of supplying background knowledge about target domains and application needs. Normally, background knowledge can be captured by three types of constraints, i.e., instance-level constraints, cluster-level constraints, and model-level constraints. In this thesis, we study how cluster-level constraints are used to capture the background knowledge and users' special requirements in several real life clustering tasks, e.g., catalog segmentation, community identification, and privacy preservation etc. We design appropriate clustering models that integrate those constraints. We analyze the complexity of the proposed models, develop efficient clustering algorithms, and evaluate the clustering results on synthetic and real data sets.

To Zengjian, Ben, Dad, and Mom

"Science is a wonderful thing if one does not have to earn one's living at it."

— *Albert Einstein*

Acknowledgments

Finally I would like to thank so many people for a variety of reasons. Without them I could not complete this thesis.

First of all, I wish to express my deep gratitude to my senior supervisor, Dr. Martin Ester, for his valuable guidance, constant encouragement, and generous financial support. He had been actively involved in the discussions of my research topics, and so patient to revise my writeups round after round. From him, I had learned how to conduct research and present results to the academic community. Besides of being a wonderful supervisor, Martin has been a good friend to me. I had also learned how to balance work and family effectively.

I also want to thank my supervisor, Dr. Binay Bhattacharya for his valuable advise on not only research but also future career path. Sincere appreciation goes to my committee members, Dr. Ramesh Krishnamurti, Dr. Christian Böhm from University of Munich, and Dr. Jiangchuan Liu for reading my thesis and giving me valuable suggestions to improve it. Special thanks also go to Dr. Funda Ergun for serving on my depth exam committee.

I would like to thank my lab mates, Wen Jin, Byron Gao, Qidan Cheng, Flavia Moser, Richard Frank, Recep Colak, and Yabo Xu for creating a fun and collaborative environment. In particular, I thank Recep Colak for preparing one of the datasets used in this thesis. My appreciation also goes to my friends from Computer Science Department: Zhengbing Bian, Yuanzhu Peter Chen, Lei Duan, Zhe Fang, Baohua Gu, Mayu Ishida, Hao Jiang, Chiyoko Kawano, Mike Letourneau, Yudong Liu, Cheng Lu, Zhongmin Shi, Qiaosheng Shi, Yi Sun, Dan Wang, Feng Wang, Yang Wang, Yong Wang, Weihua Xiong, Yinan Zhang, and Senqiang Zhou. Without them, life would not be as joyful as it was here.

I feel a deep sense of gratitude for my parents, and my brother for their endless support and encouragement throughout these years of study. I am also grateful for my parents in-laws for taking care of my son and many tedious house work to allow me to focus on my study. Last but not least, I would like to thank my husband, Zengjian. He is always there to discuss with me on my research, to help me improve my writing, and to support me emotionally. His love has accompanied me to get through the hard times. This thesis would not have been here without him. This thesis is dedicated to him and our son, Ben, who has given us great joy.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The task of clustering is to group data objects into clusters based on certain criteria so that the generated clusters provide useful knowledge to support decision making in many applications. Many clustering methods have been developed to solve problems in real life applications. For example, clustering is often used as a tool in biological sciences to find groups of co-expressed genes. In image recognition, pixels are grouped into clusters corresponding to objects in an image.

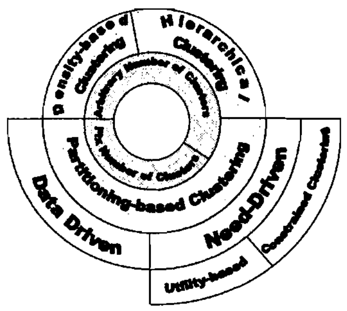


Figure 1.1: Classification of Clustering Methods

From different perspectives, clustering methods can be classified into different classes (see Figure 1.1 for a classification of clustering methods). In a broad sense, clustering methods can have the number of clusters specified a priori or decided on the fly. Based on the properties of the generated clusters, clustering methods can be classified into three major classes, namely, partitioning-based clustering [86, 73, 72, 91], hierarchical clustering [75, 69, 74, 120] and density-based clustering [46, 8]. Generally speaking, partitioning-based methods tend to find sphere shaped clusters; hierarchical methods can discover a hierarchy of clusters and then extract clusters at different resolutions; density-based methods are good for finding arbitrarily shaped clusters.

In this thesis, we study partitioning-based clustering which is often modeled as an optimization problem. In partitioning-based clustering, every data object is assigned to one cluster. The quality of a clustering is measured by an objective function, which typically measures the compactness of the generated clusters. Depending on the goal, partitioning-based methods can be either *data-driven* or *need-driven* [12]. Data-driven methods aim at discovering the true structure of the underlying data distribution. Most traditional partitioning-based methods, such as k-Center and k-Means, can be viewed as data-driven methods.

Need-driven methods achieve the same goal as that of data-driven methods while complying with application needs or background knowledge. Thus, the clusters generated by need-driven methods are more useful and actionable to meet certain application requirements. There are two typical classes of need-driven methods, the utility-based method [79] and constrained clustering. To capture simple application needs, people define utility functions as objectives that measure, instead of the compactness of the generated clusters, the utility of a clustering in decision making. As the utility function is typically defined based on business needs, the utility-based method can yield more useful and actionable clusters than the ones generated by data-driven methods. Yet, the utility-based method may fail to discover desired clusters for applications with complex application needs or background knowledge, which can be modeled alternatively as constrained clustering problems. For example, in market segmentation, relatively balanced customer groups are preferable so that the knowledge extracted from each group has similar significance and are thus easier to evaluate [56]. This special requirement can be effectively captured by imposing several balancing constraints [12, 20, 108, 121]. Recently, clustering with constraints, which integrates users' background knowledge and application needs to achieve desired clustering results, has become a significant research topic.

The work on constrained clustering can be classified into three categories [19]: (1) clustering with instance-level constraints (2) clustering with cluster-level constraints and (3) clustering with model-level constraints. Clustering with instance-level constraints is often referred to as semi-supervised clustering [111, 112, 17, 32]. Instance-level constraints, such as must-link and cannot-link constraints, impose specific requirements on pairs of instances. Knowledge on groups of data objects can be modeled as cluster-level constraints, such as the minimum number of objects per group [20, 108]. As an example of a model-level constraint, users may specify that a clustering should be different enough from a given clustering [57]. In this thesis, we focus on clustering with cluster-level constraints.

Cluster-level constraints, in general, capture users' knowledge on a group of data objects by specifying some properties of the group, e.g., the minimum number of data objects in the group or the maximum value on a SQL aggregate of the group [108]. By enforcing such constraints, we can identify clusters which are cohesive on attributes and at the same time possess certain desirable properties. Cluster-level constraints have played an important role in capturing application needs and domain knowledge in many real life applications. Among them, to name a few, are catalog segmentation, community identification in social networks, and privacy preservation. In the rest of this chapter, we overview these applications and the corresponding clustering models respectively.

1.1 Catalog Segmentation

Catalogs, containing a line of products, are one of the primary media in direct marking [93]. The best strategy in catalog campaigns is to design one catalog for each individual customer based on his/her personal interests, which can be collected from the customer's purchasing history. However, this approach is infeasible due to budget restrictions. Instead, a common approach is to ask an enterprise to design k product catalogs of size r that maximize the total number of catalog products matching customers' interests [77]. This application, named *Catalog Segmentation*, provides an effective way to boost sells under a limited budget in retail stores. From the point of view of clustering, the task of catalog segmentation is to find k clusters of customers where each cluster is described by a set of r products and each customer is assigned to the cluster with the most similar cluster description.

In this particular setting, we often observe that a customer, once attracted to an enterprise, will purchase more products beyond the ones contained in the catalog. In the case of traditional brick-and-mortar retailers, for example, a customer typically would purchase additional products if the catalog has attracted him to visit the store. In the case of electronic commerce companies, there is still a substantial overhead involved in visiting a company's website, and customers that have done so are likely to purchase other products from the website that match their interests.

Motivated by the observation above, we propose a variant of the catalog segmentation problem with a new constraint called the *minimum interest constraint*. The original definition of the catalog segmentation problem is thus reformulated so that the overall utility of a set of catalogs is measured by the number of customers that have at least a specified minimum interest t in the catalog sent to them. We name this new problem *Customer-Oriented Catalog Segmentation*.

1.2 Community Identification in Social Networks

Our second motivating scenario is *Community Identification* in social networks. A community is a group of people who share common interests and are bonded with some social relations. Community identification is an important social network analysis task which involves grouping people into clusters (or communities) [115]. In this application, communities are formed based on two basic criteria: tight social relations as well as common interests. The main interest of community identification has been focused on social relationship among entities. Graph-based clustering methods, performed solely on relationship (network) data, have been the standard tool for the task [115]. However, the results of the graph-based clustering methods are often not satisfactory since the common interests of community members are overlooked.

In a social network, every entity is often associated with two types of information, individual interests and social relations among people, which are normally represented in the form of attribute data and relationship (network) data respectively. For the task of identifying communities, it is intuitive that attribute data can impact community formation significantly [95, 64]. For example, given a scientific collaboration network, scientists can be separated into different research communities such that community members are not only connected (e.g., by co-author relations) but also share similar research interests. Such information on research interests can be automatically extracted from authors' homepages and used as attribute data.

Therefore, a clustering model considering both attribute and relationship data is in high demand. As a natural assumption, a community should be at least internally connected with possibly more constraints on the degree of connectivity. To capture the simple requirement on the connectivity of a community, we introduce an *internal connectedness constraint*. Meanwhile, the requirement on common interests can be realized by minimizing the objective function defined on attribute data. We propose a joint clustering model called *Connected k-Center*, which integrates the constraint and the objective function together. The model aims at discovering clusters which are cohesive with respect to both attribute data and relationship data.

1.3 Privacy Preservation

Clustering with cluster-level constraints can also be applied to the problem of privacy preservation. Generally speaking, data mining is a process of extracting potentially useful information from data. In some applications, data contain sensitive features and cannot be revealed to the public directly. Imagine that the government wants to release some census data. It is desirable to reveal data as accurately as possible to support better decision making. Yet on the other hand, the individual records cannot be released due to privacy concerns. A typical approach is to release the data altered by random perturbation. However, this approach could destroy the correlation among records, and thus reduce the accuracy of extracted knowledge dramatically.

Alternatively, we can group records into small clusters and release the summary of each cluster to the public. The summary reflects the overall properties of a small group of records. Subsequent analysis can be applied to summaries of the generated clusters to extract meaningful knowledge. In this context, the utility of a clustering is evaluated by how much privacy is preserved in the clustering and how truthfully the summaries represent the individual records. To protect individual records, the generated clusters have to contain at least a certain number of individual records. Furthermore, the clusters must have a minimum variance to prevent these individuals from having similar or even identical attribute values, so that an adversary cannot accurately estimate the sensitive attribute values with high confidence from the summaries. In the context of constrained clustering, those requirements are naturally translated into two types of cluster-level constraints, the *minimum significance constraint* and the *minimum variance constraint*. The goal of generating compact clusters ensures that summaries accurately reflect the properties of individual records. Furthermore, in this application, it is necessary to have the constraints decide the appropriate number of clusters instead of specifying this number in advance. Thus, we propose the *Constraint-Driven Clustering* model to capture these needs and incorporate the proposed constraints. We note that the constraint-driven clustering model is general and can be applied to many applications in addition to privacy preservation.

Model	Objective	Number of Clusters	Purpose of Constraints
Customer-Oriented Catalog Segmentation	Utility Function	Specified	Specify the minimum similarity of cluster members and representatives
Joint Cluster Analysis	Compactness	Specified	Force the generated clusters to be internally connected on relationship data
Constraint-Driven Clustering	Compactness	Not Specified	Specify the cardinality and compactness of the generated clusters

Table 1.1: Clustering models introduced in this thesis

1.4 Contribution and Organization of the Thesis

In this thesis, we introduce three clustering models (see Table 1.1 for their characteristics) in which cluster-level constraints are incorporated for discovering meaningful clusters. The rest of this thesis is organized as follows:

1. **Related Work.** We review the related work in Chapter 2. In particular, we focus on the most related topic, i.e., clustering with instance-level and cluster-level constraints.

2. **Customer-Oriented Catalog Segmentation.** In Chapter 3, we discuss the minimum interest constraint and formally introduce several variations of the Customer-Oriented Catalog Segmentation problem incorporating this constraint. We propose several heuristic algorithms to design good catalogs efficiently.

3. **Joint Cluster Analysis.** Chapter 4 introduces the novel Connected k-Center (CkC) problem, a clustering model integrating the internal connectedness constraint. The model provides a general way for joint cluster analysis of attribute data and relationship data. We analyze the complexity of the CkC problem, prove its NP-hardness and provide a constant factor approximation algorithm. For the special case of the CkC

problem where the underlying graph is a tree, we propose a dynamic programming method producing an optimal solution in polynomial time. Based on the principles of the approximation algorithm, we design NetScan, a heuristic algorithm that efficiently computes a "good" clustering solution for the *CkC* problem on large datasets.

4. **Constraint-Driven Clustering.** Chapter 5 studies the Constraint-Driven Clustering, which finds an a priori unspecified number of compact clusters that satisfy the minimum significance constraint and the minimum variance constraint. We prove the NP-hardness of the proposed clustering problem with different constraints and develop an efficient algorithm based on a novel data structure named *CD*-Tree.

5. **Conclusion.** We conclude this thesis in Chapter 6 and discuss some future directions.

Chapter 2

Related Work

In this chapter, we begin with a classification of existing clustering methods. We next review two types of partitioning-based clustering methods, data-driven methods and need-driven methods. For need-driven methods, we discuss the utility-based method and constrained clustering.

2.1 Classification of Clustering Methods

Clustering, one of the most important unsupervised learning approaches, aims at discovering underlying patterns in data. There are a wide variety of clustering methods which can be classified in multiple ways. For example, clustering methods can be differentiated by whether to specify the number of clusters. For example, the k-Means [86], k-Center [73] and k-Median [72] models all have the number of clusters, k, specified a priori. On the contrary, DBSCAN [46] is able to identify an arbitrary number of clusters. Depending on different types of patterns being identified, clustering methods can be classified into three major categories, namely, partitioning-based methods, hierarchical methods and density-based methods. Partitioning-based methods, e.g. k-Means and CLARANS [91], search for a partitioning of data objects where the quality of a clustering is measured by an objective function typically measuring the compactness of the generated clusters. Consequently, those methods tend to find sphere shaped clusters.

Different from the partitioning-based methods, hierarchical methods, such as AGNES, DIANA [75] and BIRCH [120], can discover a hierarchy of clusters. The result of hierarchical clustering is often represented by a dendrogram [87]. A cut through the dendrogram corresponds to a valid partitioning of data objects. Given a dendrogram, clusters can be extracted at different resolutions by cutting the dendrogram at different levels. Besides, the hierarchy can be obtained by either a top-down approach or a bottom-up approach. Top-down approaches, e.g., X-Means [92], initially assign all data objects into one cluster. Then in each round, a chosen cluster is broken down into two small clusters until each object belongs to its own cluster or certain conditions are satisfied. Alternatively, bottom-up approaches, such as single-link, complete-link and average link [69], start from the state where each data object belongs to its own cluster. In each round that follows, two similar clusters are fused into one until only one cluster is left.

Density-based methods, such as DBSCAN [46], are good at finding clusters that satisfy a predefined density threshold. These methods have two major advantages: First, the generated clusters can have arbitrary shapes; Second, outliers, the data objects that do not belong to any clusters, can be identified.

Partitioning-based methods can be further classified into two categories: data-driven methods and need-driven methods [12]. Data-driven methods aims at discovering the true structure of the underlying data distribution while the need-driven methods have the same objective while complying with application needs or background knowledge. Most traditional partitioning-based clustering methods, such as k-Center and k-Means, can be viewed as data-driven methods. These methods have been applied to many applications to identify useful clusters. However, they may fail to discover clusters that satisfy individual application needs due to the lack of background knowledge about target domains. Need-driven methods come into play to overcome such shortcomings.

There are two typical need-driven methods, the utility-based method and constrained clustering. As an example of the utility-based method, Kleinberg *et al.* [79] proposed a general microeconomic framework in which clusterings are evaluated by their utilities in decision making. Since the utility function is defined based on business needs, the resulting clusters are more useful and actionable than the ones generated by generic data-driven approaches.

Nevertheless, many complex application needs or background knowledge cannot be captured even using sophisticated utility functions. As an example, consider the problem to group students sharing similar interests into classes. The number of students in each cluster cannot exceed a given threshold since each classroom has a limited capacity. This type of background knowledge is naturally captured by constraints and needs to be incorporated into a clustering model. Constrained clustering, which aims at forming coherent data groups that satisfy given constraints, is particularly suitable for these clustering applications with background knowledge.

2.2 Data-Driven Methods

As mentioned in the previous section, data-driven methods can discover cohesive clusters by optimizing certain objective functions. Various data-driven clustering models have been investigated in the literature, to name a few, k-Center [41, 67, 59, 47, 3], k-Median [72, 83, 28, 70], k-Means [85], min-diameter(pairwise clustering) [25], min-sum [16, 62] and min-sum of diameters (or radii) [38, 29], which optimize the cluster radius, the cluster diameter, the sum of intra-cluster pairwise distances, the sum of diameters (or radii), and the compactness (sum of squared distances from data objects to corresponding cluster centers), respectively. For many of these problems, rigorous complexity studies and polynomial approximation algorithms are provided.

We overview a well studied data-driven model, the k-Center problem, as follows.

Definition Given a set of data objects P in d-dimensional space, an integer k and a distance measure *dist*, find a set of k data objects C as centers such that the maximum distance from a data object to its nearest center, i.e., $\max_{p \in P} \min_{c \in C} dist(p, c)$, is minimized.

If *dist* is the Euclidean distance, the above problem is called Euclidean k-Center. It is well known that both k-Center and Euclidean k-Center are NP-hard for d (dimensionality) ≥ 2 and arbitrary k [88]. For any metric space, Hochbaum and Shmoys [67] gave a greedy algorithm with approximation ratio two. Feder and Greene [47] also gave a 2-approximation algorithm with a better running time of $O(n \log k)$. When $d = 1$, the Euclidean k-Center problem is polynomially solvable using dynamic programming techniques [89, 50]. For $d \geq 2$ and fixed k, the k-Center problem can be solved easily by enumerating all the k centers and assign each data object to its nearest center.

Many of the above-mentioned clustering models are closely related to the general facility location problem [107], which has been extensively studied in the operation research literature. Given a set of facilities and a set of customers, the facility location problem is to decide which facilities to open and which customers should be served by which facilities so as to minimize the total cost of serving all the customers. Theoretic results and practical algorithms for various facility location problems are beyond the scope of this thesis. A detailed survey of related work on facility location problems can be found in [40].

2.3 Utility-based Methods

Kleinberg *et al.* [79] proposed a microeconomic data mining framework which evaluates data mining results, in particular clustering results, by their utility in decision-making. With this new objective function, the goal of clustering is shifted from identifying the true structure of the underlying distribution to discovering useful clusters.

Traditionally, finding patterns is normally considered the major task of data mining. However, finding patterns itself should not be the ultimate goal of data mining. Instead, the goal should be "turning the data into information, the information into action, and the action into value"(stated in [22]). In this regard, the microeconomic data mining framework provides an approach to evaluate patterns (clusterings) in an enterprise's decision-making process. The utility of an enterprise's decision is measured by $\max_{x \in D} f(x)$ where D is the decision space and $f(x)$ is the utility or value of decision $x \in D$.

Kleinberg *et al.* consider a linear utility function $f(x) = c \cdot x$ where c is the objective vector of customers. Normally, each customer has his/her own objective vector c_i. When the set of customers C is large, it is often infeasible to make a customized decision for each individual customer. Instead, we partition C into k segments $C_1, \ldots, C_k$ such that all the customers in the same segment share the same decision, and the total utility is maximized.

We can have different utility functions for different segmentation problems. Catalog segmentation, a special segmentation problem under the microeconomic framework, measures the utility by the total number of catalog products matching customers' interests. Formally, the catalog segmentation problem is defined as follows:

Definition Given n vectors $C = \{c_1, \ldots, c_n\}, \forall c_i \in \{0,1\}^d$, and integer k and r, the goal is to partition C into k disjoint groups $C_1, \ldots, C_k$ such that the following utility function is maximized:

$$\sum_{i=1}^{k} \sum_{c_j \in C_i} c_j \cdot x_i,$$

where vector x_i containing exactly r ones is the representative of group C_i.

Kleinberg *et al.* [79] proved that the catalog segmentation problem and its several variants are NP-hard even when $k = 2$. They also showed how sensitivity analysis of the microeconomic optimization problem distinguishes the interesting changes from the uninteresting changes of an enterprise's decisions. Furthermore, the authors outlined a sampling-based approximation algorithm and proved probabilistic bounds for the quality of the result and runtime. The algorithm first draws a sample from the given instance and then enumerates all possible partitions of the customers in the sample.

Subsequently, approximation algorithms for the catalog segmentation problem have received considerable attention in the algorithms community. Asodi and Safra [9] proved that a polynomial time $(\frac{1}{2} + \epsilon)$-approximation algorithm, for any constant $\epsilon > 0$, would imply $NP = P$. Xu *et al.* [119] developed an approximation algorithm based on semi-definite programming that has a performance guarantee of $1/2$ for general r and of strictly greater than $1/2$ for special instances where $r \geq \frac{n}{3}$. In particular, when $k = 2$, the catalog segmentation problem can be approximated by a factor of 0.67 when $r = n/2$.

2.4 Constrained Clustering

In this section, we review related work on constrained clustering, especially on clustering with instance-level and cluster-level constraints, which are closely related to the clustering models presented in this thesis. In the following, we adopt the term *constrained clustering* for the class of clustering problems where constraints are integrated. In the literature, constrained clustering problems are also referred to as *constraint-based clustering.*

2.4.1 Different Types of Constraints

There are three kinds of background knowledge based on their scopes: knowledge on individual data objects, knowledge on a group of data objects and knowledge on overall clustering of underlying data. Accordingly, the corresponding constraints are classified into three categories, i.e., instance-level constraints, cluster-level constraints and model-level constraints [35]. We discuss them in detail as follows.

Instance-level Constraints Instance-level constraints, as suggested by the name, specify relationships between data objects. Originally, such constraints are used to capture information in labeled data. For example, two data objects with the same label would suggest a must-link constraint through them. Similarly, two data objects with a different label can be attached with a cannot-link constraint [111]. Labeled data are considered as a form of partial knowledge in the target domain. Although instance-level constraints mostly come from labeled data, they can also be used to capture more general knowledge on individual data objects. For example, in some applications, domain experts know that two data objects should not be in the same cluster although they are not labeled. As there is only limited supervision, clustering with instance-level constraints is also referred to as *semi-supervised clustering.*

Cluster-level Constraints Cluster-level constraints capture users' knowledge on a group of data objects, e.g., the minimum number of data objects in the group or constraints on SQL aggregates of the group [108]. By enforcing the constraints on clusters, we are able to identify groups of objects which are coherent on attributes and possess certain properties at the same time. In the next section, we discuss cluster-level constraints in more depth.

Model-level Constraints There are often multiple ways to cluster a dataset as it may contain multiple aspects. It is natural to assume that for a certain clustering task only one aspect is relevant. Finding the "right" clustering is to discover coherent groups which agree with the relevant information and disagree with the irrelevant information. To achieve this goal, irrelevant information is often quantized and treated as constraints on clustering problems. Similarly, a previously discovered clustering can be treated as negative information. Then we can design clustering models to discover discriminative clusterings which have not been discovered by previous clustering processes [57, 58]. In those approaches, the irrelevant or negative information is modeled as model-level constraints. Different from both instance-level and cluster-level constraints, model-level constraints are enforced on a clustering. In this chapter, we do not discuss model-level constraints in detail since they are not the focus of this thesis.

2.4.2 Semi-Supervised Clustering

Instance-level constraints are considered in the following four types [111, 32]:

(a) **Must-link constraints:** For two data objects o_i and o_j $(i \neq j)$, a must-link constraint specifies that o_i and o_j have to belong to the same cluster in any feasible clustering.

(b) **Cannot-link constraints:** For two data objects o_i and o_j $(i \neq j)$, a cannot-link constraint specifies that o_i and o_j cannot belong to the same cluster in any feasible

Feasibility Problem under	Complexity
must-link constraints	$O(n+m)$ [31].
cannot-link constraints	NP-Complete. By a reduction from GRAPH K-COLORABILITY [32].
δ-Constraints	$O(n^2)$ [32].
ϵ-Constraints	$O(n^2)$ [32].

Table 2.1: Complexity Results of the Feasibility Problems under different Constraints

clustering.

(c) **δ-Constraint (or Minimum Separation Constraint) :** [1] For any pair of data objects o_i and o_j from clusters Cl_i, respectively Cl_j, $dist(o_i, o_j) \geq \delta$ where *dist* is any distortion measure.

(d) **ϵ-Constraint:** For any cluster Cl_i containing two or more data objects, for any data object $o_i \in Cl_i$, there must exist a data object $o_j \in Cl_i$ s.t. $dist(o_i, o_j) \leq \epsilon$.

With those constraints, two natural research questions are: Do instance-level constraints change the complexity of a clustering problem? How feasible is it to combine two types of instance-level constraints into a clustering model? Regarding the first question, the following feasibility problem under instance-level constraints was studied in the literature.

Definition Given a set of data objects $D = \{o_1, \ldots, o_n\}$, a collection of constraints $C = \{C_1, \ldots, C_m\}$ where $C_i = (o_{i_1}, o_{i_2})$. Furthermore, two constants K_l and K_u where $1 \leq K_l \leq K_u \leq n$ are given. Note that if C is a δ-constraint or ϵ-constraint, δ and ϵ are given as well. Is there a partition of D into K clusters such that $K_l \leq K \leq K_u$ and all constraints in C are satisfied?

The complexity results of these feasibility problems are summarized in Table 2.1. Among the four types of constraints, only the feasibility problem with cannot-link constraints is NP-hard. Furthermore, it is natural to study the complexities of the feasibility problems under combinations of constraints. We list the results from [32] for all other combinations in Table 2.2.

[1]The δ-Constraint and the ϵ-Constraint are also considered as cluster-level constraint in several papers. But in this thesis, we classify them as instance-level constraints since they are defined on pairs of data objects and can be reduced to instance-level constraints.

	Must-link constraints	δ-Constraints	ϵ-Constraints
Must-link constraints	–	Solvable in $O(n^2)$	NP-Complete
δ-Constraints	Solvable in $O(n^2)$	–	Solvable in $O(n^2)$
ϵ-Constraints	NP-Complete	Solvable in $O(n^2)$	–

Table 2.2: Complexity Results of the Feasibility Problems under Combinations of Constraints

[33] demonstrates that given a set of n data objects, adding a small number of cannot-link constraints helps to provide a better clustering. However, when the number of cannot-link constraints increases, current clustering algorithms experience difficulties to provide a feasible clustering that satisfy all the constraints. [33] also identifies several sufficient conditions from graph theory which can be used to identify the easy problem instances in advance.

Basu *et al.* [18] proposed a probabilistic framework based on Hidden Markov Random Fields (HMRF) which incorporates supervision into k-clustering algorithms. This method solves the clustering problem by minimizing the posterior energy of the HMRF defined below:

$$\begin{aligned}\mathcal{J}_{obj} = & \underbrace{\sum_{o_i \in D} dist(o_i, \mu_{l_i})}_{(1)} + \underbrace{\sum_{(o_i,o_j)\in\mathcal{M}} w_{ij}\varphi_{dist}(o_i,o_j)\mathbb{I}[l_i \neq l_j]}_{(2)} \\ & + \underbrace{\sum_{(o_i,o_j)\in\mathcal{C}} \overline{w}_{ij}(\varphi_{dist\_max} - \varphi_{dist}(o_i,o_j))\mathbb{I}[l_i = l_j]}_{(3)} + \log Z \end{aligned} \quad (2.1)$$

where $\mathbb{I}$ is an indicator function satisfying $\mathbb{I}[true] = 1$ and $\mathbb{I}[false] = 0$.

In the objective function, part (1) is a distortion measure between two observed data objects. Part (2) is for penalizing the violation of a must-link constraint where φ_{dist} is a *penalty scaling function*. The penalty scaling function is chosen as a monotonically increasing function of the distance between o_i and o_j to reflect the fact that the penalty for violating a must-link constraint between two distant data objects is higher than that between nearby data objects. Similarly, part (3) defines the penalty for violating a cannot-link. The experiment in [18] demonstrates that considering instance-level constraints is beneficial and also shows that the probabilistic framework can achieve good clustering results.

A recent work [82] shows that the objective function of the HMRF-based semi-supervised clustering model, as well as that of some graph clustering models, can be expressed as special cases of weighted kernel k-Means objective. Based on this observation, a unified algorithm is proposed to perform semi-supervised clustering on data given either as vectors or a graph.

In semi-supervised clustering, instance-level constraints are used to capture users' background knowledge on individual data objects. Many studies [111, 112, 18, 17] have demonstrated that by incorporating instance-level constraints one could improve the clustering performance. For example, the experiments on document clustering in [18] show that a small number of instance-level constraints can significantly improve the accuracy of clustering results.

2.4.3 Clustering with Cluster-level Constraints

In many applications, users' background knowledge on a target domain is limited to the cluster level. For example, in the application of analyzing coexpressed genes[66], a quality threshold on each generated cluster helps to achieve better clusterings. Note that in such applications, constraints are enforced on the generated clusters instead of on the pairs of data objects. In order to capture these requirements, people introduce cluster-level constraints, which result in more meaningful clusterings.

The term *cluster-level constraint* [110] is relatively new despite that clustering models with some special requirements have been studied for decades. Tung *et al.* [108] studied this problem with the name "constraint-based clustering". The proposed constrained clustering model is essentially a combination of cluster-level constraints and k-clustering.

Definition (Constrained Clustering) Given a data set D with n objects, a distance function $dist : D \times D \rightarrow \mathcal{R}$, a positive integer k, and a set of constraints $\mathcal{C}$, find a k-clustering $(Cl_1, \ldots, Cl_k)$ such that $DISP = (\sum_{i=1}^{k} \sum_{o \in Cl_i} dist(o, rep_i))$ is minimized, and each cluster Cl_i satisfies the constraints $\mathcal{C}$.

Tung *et al.* proposed a typical SQL aggregate constraint and a special case named the existential constraint (defined below), which provides a general form of cluster-level constraints.

Definition (SQL Aggregate Constraint [108]) Given a database D, each data object $o_i \in D$ is associated with a set of d attributes $\{A_1, \ldots, A_d\}$. Consider the aggregation functions $agg \in \{max(), min(), avg(), sum()\}$. Let θ be a comparator function, i.e., $\theta \in \{\leq, \leq, \neq, =, \geq, \geq\}$, and c represent a numeric constant. Given a cluster Cl, an SQL aggregate constraint on Cl is a constraint in one of the following forms: (1) $agg(o_i[A_j]|\forall o_i \in Cl)\theta c$ where $o_i[A_j]$ denotes the value of an attribute A_j of object o_i; (2) $count(Cl)\theta c$.

Definition (Existential Constraints [108])
Given a database D, each data object $o_i \in D$ is associated with a set of d attributes $\{A_1, \ldots, A_d\}$. Let $W \subseteq D$ be any subset of data objects. Let m be a positive integer. An existential constraint on a cluster Cl is a constraint of the form: $count(\{o_i|\forall o_i \in Cl, o_i \in W\}) \geq m$.

The existential constraint specifies the minimum number of data objects in a cluster that belong to a certain set W. Note that in this model W may be the same as D. In such a case, the existential constraint becomes the minimum significance constraint which requires every cluster to contain at least a certain number of data objects.

With constraints specified, the clustering problem exhibits some new properties. In the original k-clustering problem, grouping a data object with the nearest cluster representative always ends up with an optimal assignment. This property is referred to as *Nearest Representative Property* (NRP). However, as pointed out in [108], in constrained clustering NRP may not always be satisfied due to possible conflicts between NRP and constraints. In other words, clustering algorithms sometimes have to sacrifice clustering cost by not assigning some data objects to its nearest representative in order to satisfy all constraints.

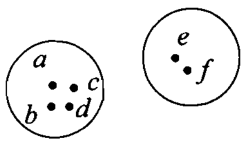


Figure 2.1: Demonstration of the nearest representative property

Constraints introduce new challenges to clustering algorithms, especially to the cluster assignment procedure. With the NRP property, given k cluster representatives, we can simply assign each data object to its nearest representative to achieve an optimal clustering with respect to the given representatives. But when NRP is no longer valid due to constraints, it is hard to find a valid clustering with the minimal cost. For example, in Figure 2.1 we are given six data objects in two-dimensional space where a, b, c, d are close to each other but far away from e and f. We want to partition the dataset into two clusters with at least three data objects each. Assume that we have picked a and e as two representatives of the two clusters. In order to satisfy the existential constraint, exact one of b, c, d has to be grouped with e, due to which NRP is violated. However, none of these data objects should be assigned to e if NRP holds. This shows that the process of finding such an optimal solution for given representatives can be expensive.

Algorithms to Solve the Clustering with Existential Constraints Based on the research of [108] where the SQL aggregate constraint provides a general form of cluster-level constraints, the following efforts mainly concentrate on how to solve the constrained clustering problem efficiently.

An algorithm for finding a valid clustering satisfying existential constraints is proposed in [108]. The algorithm starts with an initial solution that satisfies user-provided constraints and then refines the solution by performing confined object movements under constraints. Tung *et al.* [108] showed that the problem of moving objects to achieve optimal solution is NP-hard and suggested several heuristics. Bennett *et al.* [20] proposed a modified k-Means algorithm to assign data objects to clusters by solving a minimum cost network flow problem.

To deal with a special existential constraint, minimum number of data objects constraint, Banerjee and Ghosh [11] proposed an algorithm which modifies the k-Means algorithm to satisfy the constraint on each generated cluster. They suggested a three-step scheme to cluster n data objects into k clusters where each cluster contains at least m data objects for some given $m \leq \frac{n}{k}$ in $O(kn \log n)$ time. The three steps are *sampling*, *soft-balanced clustering* and *populating the clusters while keeping the balance.* [11] also demonstrates that the three-step process gives a very general methodology for scaling up balanced clustering algorithms. For the same problem, [102] proposes to convert the clustering problem into a graph partition problem to achieve a similar goal. However, the complexity of this approach is higher than the one in [11].

In general, cluster-level constraints are used to capture users' background knowledge on groups of data objects. They can be used in many applications to discover meaningful clusters [20, 56]. For example, the existential constraints can help improve the current k-Means algorithm since the k-Means algorithm often outputs a set of tiny clusters or even empty clusters when k is large. Bennett *et al.* [20] suggested to add k existential constraints to the underlying clustering optimization problem while each existential constraint specifies the minimum number of data objects in a cluster. The existential constraint is also utilized in market segmentation [56] to discover balanced customer groups which are preferable so that the knowledge extracted from each group has similar significance and are thus easier to evaluate.

Chapter 3

Customer-Oriented Catalog Segmentation

Kleinberg *et al.* [79] proposed a microeconomic framework to evaluate data mining results by their utilities in decision making. The microeconomic framework for data mining has in particular been investigated for segmentation (clustering) problems where the enterprise does not make an optimal decision per individual customer but chooses one optimal decision per customer segment. In particular, the *catalog segmentation* problem is to design k product catalogs of size r that maximize the overall number of catalog products matching customers' interests. In this framework, the goal of generating useful clusters is achieved by a sophisticated objective function which is defined based on business needs. In many applications, the utility function cannot fully capture special requirements which alternatively can only be captured by constraints. In this chapter, we study a variant of the catalog segmentation problem where application needs are captured by both the utility function and constraints. A preliminary version of Chapter 3 was published in [45].

3.1 Overview

The goal of knowledge discovery in databases is to extract knowledge from databases that is implicit, valid and potentially useful [49]. While the validity of a pattern is relatively easy to formalize and to check, this turns out to be much harder for the usefulness of a pattern. The common data mining approach is to automatically find interesting patterns, i.e. patterns that are interesting from a statistical point of view since they are very confident or unexpected, and to let the user judge the usefulness of the discovered patterns w.r.t. some given application. This approach, however suffers from two fundamental problems. First, without a notion of usefulness the search in the space of all patterns cannot be effectively focused which leads to inefficient data mining algorithms. Second, and even more importantly, the user will be overwhelmed by a huge number of patterns so that it becomes very difficult for him to identify the really useful pieces of knowledge. Several researchers have discussed the necessity of the notion of usefulness and pushing it into the pattern discovery. So far, however, the only formal framework for mining useful knowledge from data has been proposed by Kleinberg *et al.* [79]. In their microeconomic framework for data mining, they consider an enterprise with a set of possible decisions and a set of customers that, depending on the decision chosen, contribute different amounts to the overall utility of a decision from the point of view of the enterprise. It is assumed that the contribution of a customer is a, possibly complicated, function of the data available on that customer. The enterprise chooses the decision that maximizes the overall utility over all customers.

The microeconomic framework for data mining has in particular been investigated for segmentation (clustering) problems where the enterprise does not make an optimal decision per individual customer but chooses one optimal decision per customer segment. Catalog Segmentation, a specialized segmentation problem, has received considerable attention [78, 79, 101]: the enterprise wants to design k product catalogs of size r that maximize the overall customer purchases after having sent the best matching catalog to each customer.

The Catalog Segmentation problem measures the utility of a customer in terms of catalog products purchased. But there are many applications where a customer, once attracted to an enterprise, would purchase more products beyond the ones contained in the catalog. In the case of traditional brick-and-mortar retailers, for example, a customer typically would purchase additional products if the catalog has attracted him to visit the store. In the case of electronic commerce companies, as another example, there is still a substantial overhead involved in visiting a company's website, and customers that have done so are likely to purchase other products from that website that match their interests. Therefore, we introduce a *minimum interest constraint* and investigate an alternative formulation incorporating the constraint. We measure the overall utility by the number of customers that have at least a specified minimum interest t in the catalog sent to them. A similar problem has been mentioned as an open problem in [78]. We call the new problem *Customer-Oriented Catalog*

Segmentation problem.

The rest of this chapter is organized as follows. Section 3.2 reviews research closely related to the catalog segmentation problem. In Section 3.3, we formally introduce the Customer-Oriented Catalog Segmentation problem and analyze its complexity. Section 3.4 presents efficient algorithms for the Customer-Oriented Catalog Segmentation problem. Section 3.5 reports the results of our experimental evaluation and comparison. Finally, we discuss directions for future research on this topic in Section 3.6.

3.2 Background

The microeconomic approach to data mining was introduced by Kleinberg *et al.* [78]. This approach formalizes the optimization problem of enterprises based on data and allows the enterprise to predict the utility of a customer with respect to a chosen decision. As a model in the microeconomic framework, our customer-oriented catalog segmentation model is an extension of the catalog segmentation problem utilizing both the utility function and constraints to capture complex application needs.

In the data mining community, the Catalog Segmentation problem has been treated as a clustering problem. Steinbach *et al.* [101] showed that the sampling-based enumeration algorithm[79] is impractical for realistic problem sizes. Instead, they proposed two alternative heuristic algorithms and a hybrid algorithm (HCC) combining both of them. The first algorithm, Indirect Catalog Creation (ICC), groups together similar customers using the k-Means algorithm and then determines the optimal catalog for each cluster. The second one, Direct Catalog Creation (DCC), iteratively optimizes k catalogs in a manner similar to the EM paradigm. Their experimental evaluation demonstrates that DCC and HCC obtain higher overall profit than ICC.

Another research direction that is inspired by the microeconomic view of data mining is the extension of association rule mining to take into account the indirect profit of products that are frequently purchased together with some other products. Brijs [24] proposed PROFSET to model the cross-selling effects by identifying "purchase intentions" in the transactions. Lin *et al.* [84] introduced a value added model of association rule mining where the value could represent the profit, the privacy or other measures of the utility of a frequent itemset. Wang *et al.* [114] presented a method for proposing a target item whenever a customer purchases a non-target item. This method maximizes the total profit of target items for future customers. Wang *et al.* [113] applied the principle of mutual reinforcement of hub/authority web pages in order to rank items taking into account their indirect profits. Addressing a similar problem, Wong *et al.* [117] studied the problem of selecting a maximum profit subset of items based on modeling the cross-selling effects with association rules. While all these approaches incorporate the notion of utility into the process of association rule mining, they analyze the relationships between sets of products without considering which customers have purchased these products. These methods aim at suggesting products to individual customers or selecting subsets of (globally) profitable items, but not at Catalog Segmentation or clustering customer databases.

3.3 Problem Formulation

In the microeconomic framework for data mining, there is an enterprise with a set of possible decisions and a set of customers that, depending on the decision chosen, contribute different amounts to the overall utility of a decision from the point of view of the enterprise. It is assumed that the contribution of a customer can be determined based on the data available on that customer. In our case, these data represent the set of products that a customer is interested in. The customer interest can be obtained either from aggregating the history of transactions of that customer or from obtaining explicit customer votes on the set of products. We assume that the (possibly very large) collection of customer data is stored in a Customers Database (Customers DB).

In order to formally introduce our problem and for the presentation of our algorithms, we choose to represent the Customers DB as a bipartite graph. We distinguish two sets of vertices, one for the customers and another one for the products, and an edge denotes the fact that the corresponding customer is interested in the corresponding product. In the following, we introduce the graph-based representation and the related notation.

Notation:

- $G(P, C, E)$ denotes a bipartite graph with two vertices sets P, C and the edges set E:
  $P = \{$ all products $\}$,
  $C = \{$ all customers $\}$,
  $E = \{ (p, c) \mid c$ is interested in $p, p \in P, c \in C \}$.
- For $\forall P' \subseteq P$, $\omega(P') = \{$all the vertices in C which have at least an edge connecting to the vertices in P' $\}$.

- For $\forall P' \subseteq P$, $t \geq 1$, $\psi(P', t) = \{$ all the vertices in C which have at least t edges connecting to the vertices in P' $\}$.
- For $\forall P' \subseteq P$, $C' \subseteq C$, $\theta(P', C') = \{$ all the edges in E with one end in P' and the other in C' $\}$.
- $||$ denotes the cardinality.

The original Catalog Segmentation problem can be defined in a more illustrative graph-based manner (partition version) as follows [119]: given a bipartite graph $G = (P, C, E)$ with $|P| = m$ and $|C| = n$, find a partition of $C = C_1 \cup C_2 \cup \ldots \cup C_k$, and k subsets $P_1, P_2, \ldots, P_k$ of P, such that $|P_i| = r$ and $\sum_{i=1}^{k} |\theta(P_i, C_i)|$ is maximized. Note that $\forall i,j$, $i \neq j$, $|P_i \cap P_j|$ does not have to be empty.

In the following, we formalize our Customer-Oriented Catalog Segmentation model. We first introduce *MEC* (*Maximum Element Cover*), a well known combinatorial problem, whose goal is to find *one* catalog such that the maximum number of customers is interested in at least *one* of its products.

Definition (Maximum Element Cover)
Given a bipartite graph $G = \{P, C, E\}$ and a positive integer r, find a subset $P' \subseteq P$ with size r such that $|\omega(P')|$ is maximized.

We generalize the problem for the case of k catalogs and call it k-MEC (k-Maximum Element Cover) problem.

Definition (k-Maximum Element Cover)
Given a bipartite graph $G = \{P, C, E\}$ and positive integers r, k, find k subsets $P'_1, \ldots, P'_k \subseteq P$, each with size r, such that $|\omega(P'_1) \cup \ldots \cup \omega(P'_k)|$ is maximized.

Note that in the k-MEC problem, the subset $P'_1, \ldots, P'_k$ do not have to be disjoint. Finally, we introduce the minimum interest constraint t representing the minimum interest in a catalog necessary to attract a customer, and extend k-MEC to k-MECWT (k-Maximum Element Cover With t).

Definition (k-Maximum Element Cover With t)
Given a bipartite graph $G = \{P, C, E\}$ and positive integers r, k, t, find k subsets $P'_1, \ldots, P'_k \subseteq P$, each with size r, such that $|\psi(P'_1, t) \cup \ldots \cup \psi(P'_k, t)|$ is maximized.

The task of the k-MECWT problem is to find k catalogs maximizing the number of distinct customers who have at least t interesting products in the catalog that is sent to them.

MEC is a well known NP-Complete problem and can be easily reduced from *Set Cover* [51]. In [48], Feige proved that the simple greedy algorithm, iteratively selecting the next product that covers the largest number of uncovered customers, approximates MEC by a ratio of at least $1 - 1/e \approx 0.632$. He showed that this ratio cannot be further improved by any constant number unless $P = NP$. More generally, by a simple Turing reduction from the Set Cover problem, we can show that the k-MECWT problem is NP-hard for any $k, t \geq 1$[1]. Thus, k-MECWT is even harder than the classical Catalog Segmentation problem which is NP-hard only for $k \geq 2$. As an example, for $k = 1$, there is an $O(|P| + r \log r)$ algorithm solving the Catalog Segmentation problem that simply picks the r products with the largest number of interested customers[2]. But for k-MECWT and $k = 1$, the simple algorithm enumerating and testing all combinations of r products has a runtime complexity of $O(|P|^r \cdot |C|)$.

From the point of view of clustering, k-MECWT can be understood as follows. The task of k-MECWT is to find k clusters of customers where each cluster is described by a set of products and each customer is assigned to the cluster with the most similar cluster description. There are two constraints for acceptable clusterings: (1) the cardinality of each cluster description is r and (2) customers can only be assigned to a cluster if they have a minimum similarity of t to the cluster description. The clustering objective is to maximize the number of customers assigned to some cluster.

[1]The bipartite graph formulation of Set Cover is the following: Given a bipartite graph $G = \{P, C, E\}$, an integer $r > 0$, we want to decide whether there exists a subset $R \subseteq P$, with $|R| = r$, which covers all elements of C, i.e., $\omega(R) = C$. The proof idea is that we take an instance $I = (G, r)$ of Set Cover and reduce it to an instance I' of k-MECWT. To construct the instance I', we first attach $t - 1$ dummy vertices to P and link each dummy vertex to every vertex in C, and then copy the new graph by k times to get a graph G' of I'. Based on the construction, it is easy to verify that I' is valid if and only if I is a valid instance of Set Cover. Therefore, k-MECWT is NP-hard.

[2]Given a graph representation, we need to select r products with the largest degree. This can be done by a typical Top-r algorithm with running time $O(|P| + r \log r)$.

3.4 Algorithms

Since the Customer-Oriented Catalog Segmentation problem is NP-hard, in this section, we present several efficient heuristic algorithms. All algorithms are based on the graph representation of the Customers DB. We employ adjacency lists as our major data structure: for each product, the corresponding adjacency list contains all customers interested in that product. The list head records the total number of customers in the list. The Customers DB is read once and transformed into the (main memory) adjacency lists that efficiently support the manipulation of the graph structure from the point of view of products. For each customer, we need a counter denoting the number of additional interesting products that this customer requires to be attracted by the current catalog. This data structure is much smaller than the adjacency lists, and the overall space complexity of these data structures is $O(|E| + |C|) = O(|E|)$, i.e. proportional to the number of edges in the graph G. In Section 3.4.1, we explore different greedy, deterministic algorithms. In particular, the Best-Product-Fit algorithm is a greedy algorithm that constructs one catalog at a time by choosing the next product for that catalog based on some heuristic quality criteria. The Best-Product-Fit algorithm is very efficient but, due to its greedy nature, may return a solution which is only locally optimal. Therefore, we also investigate a randomized algorithm (subsection 3.4.2), i.e., Random-Product-Fit, that iteratively optimizes the result of a greedy algorithm.

3.4.1 Greedy Algorithm for the k-MECWT Problem

The basic idea of greedy algorithms for the Customer-Oriented Catalog Segmentation problem is as follows: one catalog is constructed at a time by choosing the "best" next product for the current catalog. The "goodness" of a product is measured by criteria such as the number of customers interested and the products already chosen for the catalog.

Since our objective is to maximize the overall number of customers that have enough interests in at least one of the catalogs, a naive greedy algorithm would always pick the remaining product with the largest number of interested customers. Customers that are already interested in at least t products from the current catalog cannot increase the overall number of customers attracted by that catalog and are not considered by the calculation of this product goodness.

While the naive greedy algorithm is very efficient, it does not take the threshold t into account. This decreases the quality of its resulting solutions whenever the product with the maximum number of interested customers does not cover (a good number of) customers that have already been covered by catalog products.

Since the naive greedy algorithm does not take the threshold t into consideration when choosing the next product, it may choose products interesting for customers whose overall interests in the catalog may never reach the specified threshold. To avoid this waste of resources, we need to increase the priorities of products connected to customers which are already interested in other catalog products. The Best-Product-Fit algorithm (illustrated in Algorithm 1) assigns a score to each product based on the (remaining) customers interested in that product and the number of additional interesting products that these customers need to be attracted to the current catalog. Before stating our algorithm, we define some notions based on a Customers DB in graph representation $G = (P, C, E)$.

$CustomersCovered$={customers who have already t interests in one of the catalogs};

$\overline{C} := C - CustomersCovered.$

$Counter(c)$ =the counter associated to customer c. Initially, $Counter(c) = t$.

We define the score of product p w.r.t the current catalog *cat* by the following equation: $Score(p) = \sum_{c \in \overline{C}, (p,c) \in E} \frac{1}{Counter(c)} + \left|\{c \in \overline{C} \,|\, \exists p' \in cat, (p', c) \in E, (p, c) \in E\}\right|.$

The score depends on two terms. The first term represents the weighted sum of all customers interested in product p, where the weight of the customer is the inverse of its counter (the weight is the higher, the more interests the customer already has in the current catalog). The second term focuses only on customers that are already interested in at least one of the current catalog products and measures how many of these customers are also interested in p. As an optimization, for the second term, we do not count the customers who need more than $r - |cat|$ further products to be fully covered. The pseudocode of the Best-Product-Fit algorithm is illustrated in Algorithm 1.

To show the difference to the naive greedy algorithm, we apply this algorithm to the example in Figure 3.1. The computation is shown in Table 3.1.

With the Best-Product-Fit algorithm, the first catalog includes customer 1 and 2 which is actually the best solution we can get for this particular example.

The runtime complexity of the Best-Product-Fit algorithm is $O(kr|E|)$ where $|E|$ is the total number of edges in the graph, i.e. the total number of interests over all customers.

Algorithm 1 Best-Product-Fit:

```
1: Input: (1)Customers $DB$ $G = (P, C, E)$, (2)number of catalogs $k$, (3)number of products
   in each catalog $r$, (4)the threshold $t$
2: Output: $k$ catalogs of customers
3: Initialize $Counter(c) = t$ for all customers.
4: for $i = 1$ to $k$ do
5:   for $j = 1$ to $r$ do
6:     for each $p \in P$ do
7:       Calculate $Score(p)$
8:     Add $p$ with the largest $Score(p)$ to Catalog $i$
9:     for each $c$ with $(p, c) \in E$ do
10:      $Counter(c) = Counter(c) - 1$
11:    Remove customers whose counter is 0 and recalculate $Score(p)$ for all products
       interesting to those customers.
12:  for each $c \in C$ do
13:    if $Counter(c) \geq 0$ then
14:      $Counter(c) = t$
15: Return $k$ Catalogs
```

The algorithm requires only one scan of the database if the memory can hold the necessary data structures. Otherwise, we can adopt the divide-and-conquer approach to scale up the Best-Product-Fit algorithm. First, we partition the Customers DB into several subsets $DB_1, DB_2, \ldots, DB_p$ that each can fit into the memory. Then we apply the Best-Product-Fit algorithm to each subset DB_i to determine k catalogs and combine those $k \cdot p$ catalogs into k final catalogs. This algorithm still requires only one database scan.

c	$Counter(c)$	
	step 0	step 1
1	2	1
2	2	1
3	2	1
4	2	2
5	2	2

p	$Score(p)$	
	step 0	step 1
Beer	1	2
Diaper	1	1.5
Pencil	1	1.5
Milk	1	1
TV	1.5	–
Candidate	*TV*	*Beer*

Table 3.1: Illustration of Best-Product-Fit

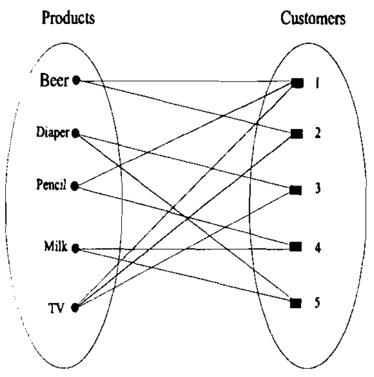


Figure 3.1: Example Customers DB in Graph Representation.

3.4.2 Randomized Algorithm

The greedy algorithm finds a local optimum only. It may include products into its catalogs that are interesting for many customers that ultimately may not have enough interest (i.e. t interesting products) in the catalog. This weakness is due to the heuristic nature of the quality criterion for individual products and to the deterministic nature of the algorithm. The proposed greedy algorithm has no means of backtracking from some suboptimal choice of a catalog product. This problem is illustrated by the example in Figure 3.2 with $k = 2$, $t = 2$ and $r = 2$. The Best-Product-Fit algorithm would pick *Diaper* first since it has largest number of interested customers and then select *Beer* as the second product of $Catalog_1$ because it covers two customers who have already been interested in *Diaper*. *Diaper* is still the product with the largest number of interested customers who have not yet been covered by the $Catalog_1$ and is therefore chosen as the first product of $Catalog_2$. As the second product, VCR is chosen because it is interesting for customer 7 (that is already interested in the first catalog product) and for customer 8 and 9. In this solution, only customer 7 meets the interest threshold, while $Catalog_2 = \{VCR, Coke\}$ covers three customers (7,8,9). Due to the lack of a look-ahead mechanism, the choice of *Diaper* as the first product of $Catalog_2$ leads into a local optimum that cannot be escaped by the greedy method.

In order to overcome these limitations, randomized algorithms seem to be promising. Since the performance of randomized algorithms crucially depends on appropriate initial solutions, we propose to combine the greedy deterministic algorithm with a randomized

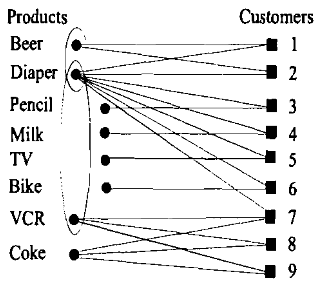


Figure 3.2: Customers DB in Graph Representation

algorithm in a two-step approach (Random-Product-Fit):

1. A greedy deterministic algorithm (e.g. Best-Product-Fit) is used to efficiently determine a good solution of the Customer-Oriented Catalog Segmentation problem.

2. The resulting catalogs and corresponding clusters are iteratively optimized by randomly replacing one catalog product by a non-catalog product (Random-Product-Switch).

There are different alternatives for the second randomized step with more or less deterministic aspects. A fully randomized algorithm would randomly select one of the catalogs, one of its products and one non-catalog product for replacement. More deterministic versions would select the catalog and the product to be replaced in a deterministic way, e.g. in a round robin fashion. There are two major types of termination conditions for randomized algorithms. They can terminate either after a user-specified number of iterations or as soon as the number of customers covered no longer increases. For simplicity, we propose a fully randomized algorithm with a user-specified number of iterations.

To efficiently support our randomized algorithm, we introduce an additional data structure for each customer c consisting of a customer id (Id) and one list of products for each of the catalogs recording the interesting products (CatalogInterests[k]). This data structure enables us to efficiently calculate the gain (δ) in the number of customers covered caused by the replacement of catalog product p by p'. The space requirement of this additional data structure is $4k \cdot |C|$ bytes. The pseudocode of the Random-Product-Switch algorithm is provided in Algorithm 2.

Algorithm 2 Random-Product-Switch:

```
1: Input: (1)$k$ catalogs & $k$ corresponding clusters of customers (2)number $r$ of products
   in each catalog, (3)the threshold $t$ and (4)number of iterations $s$
2: Output: $k$ catalogs & $k$ corresponding clusters of customers
3: Calculate the number $N_{customer}$ of all customers
4: for $n = 1$ to $s$ do
5:   Randomly select a catalog $cat$
6:   Randomly select a product $p$ from $cat$
7:   Randomly select a non catalog product $p'$ from $\{P - cat\}$
8:   Calculate the gain $\delta$ in $N_{customer}$ by replacing $p$ with $p'$
9:   if $\delta \geq 0$ then
10:    Replace $p$ by $p'$ in $cat$
11:    $N_{customer} = N_{customer} + \delta$
12: Return $k$ clusters of customers with $k$ catalogs
```

The runtime complexity of the Random-Product-Switch method is $O(sk|C|)$ since in the second step, in each iteration, for each customer we need to access all k elements of CatalogInterests[k] in order to update the number $N_{customer}$ of all covered customers. As the two-step Random-Product-Fit approach consists of (1) Best-Product-Fit (2) Random-Product-Switch methods, the overall runtime complexity of Random-Product-Fit is $O(kr|E|+sk|C|)$.

3.5 Experimental Evaluation

In this section, we report the results of our experimental evaluation using synthetic as well as real datasets. The synthetic datasets were generated using the well-known IBM data generator [6]. The real dataset records the purchasing transactions of the customers of a large Canadian retailer over a period of several weeks. Since the Customer-Oriented Catalog Segmentation problem has not yet been addressed in the literature, we compare our proposed algorithms with one of the state-of-the-art algorithms [101] for the related Catalog Segmentation problem. We choose DCC as our comparison partner because of the following two reasons. First, the experimental evaluation in [101] shows that DCC, together with HCC, achieves the highest quality results. Second, DCC scales better to large customers databases than HCC because DCC can, different from HCC, use storage efficient adjacency lists instead of an adjacency matrix. We report the results of the algorithms w.r.t. utility(quality).

We evaluate the quality of the catalogs obtained by our algorithms Best-Product-Fit and Random-Product-Fit as well as DCC. Since DCC has been developed for a related, but different problem formulation, we measure the resulting quality w.r.t. both the objective functions of classical Catalog Segmentation (catalog products purchased) and Customer-Oriented Catalog Segmentation (customers covered). To demonstrate the extra profit achievable by Customer-Oriented Catalog Segmentation, we also measure the number of non-catalog products that are additionally purchased by customers interested in their corresponding catalogs.We report results for a dataset with $|C| = 50,000$, $|P| = 7,374$ and $|E| = 376,713$ that seems to be representative for a medium-sized customers database. For the real dataset, $|C| = 45,394$, $|P| = 23,182$ and $|E| = 355,908$.

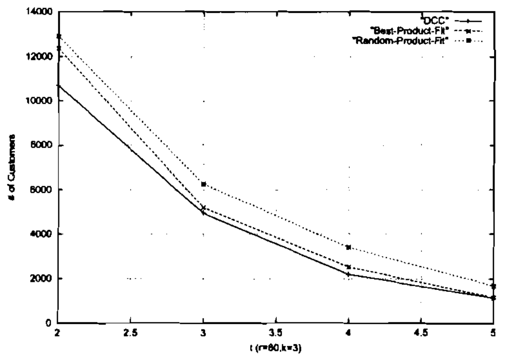
We compare the numbers of the customers covered (i.e., interested in at least t catalog products) w.r.t. t, k and r on the synthetic dataset. The impact of different values of t w.r.t. the numbers of customers covered is depicted in Figure 3.3(a) in the case of $r = 80$ and $k = 3$, both Best-Product-Fit and Random-Product-Fit yield higher coverages of customers than DCC, while Random-Product-Fit always covers more customers than Best-Product-Fit.

While the effects of different k values on the total number of covered customers in the case of $r = 60$ and $t = 2$ shown in Figure 3.3(b), reflects the fact that more customers will be covered if more catalogs are created. Random-Product-Fit method always covers the largest number of customers since it has more chances to check and switch more catalog products to cover more customers. Best-Product-Fit still covers more customers than DCC. These results are confirmed by our experiments on the real dataset. The corresponding numbers of covered customers in Figure 3.6(a) shows the corresponding numbers of covered customers for $r = 30$ and $t = 2$. It also illustrates that the advantage of Random-Product-Fit compared to Best-Product-Fit grows with increasing k values.

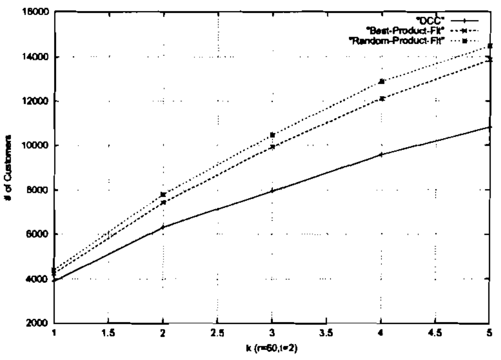
The relationship between the size r of the catalog and the number of covered customers with $k = 3$ and $t = 2$ provided in Figure 3.4(a), has similar results as Figure 3.3(b). For example, for $r = 100$, the catalog generated by Random-Product-Fit attracts 2,700 (22%) more customers than the DCC catalogs.

The profit in terms of the total numbers of catalog products and the numbers of extra products (beyond the catalogs) w.r.t. t, k and r are also investigated on the synthetic dataset. When measuring the number of catalog products covered, these experiments favor DCC due to the different objectives of the comparison partners.

We observe the numbers of products covered w.r.t different values of t in the case of

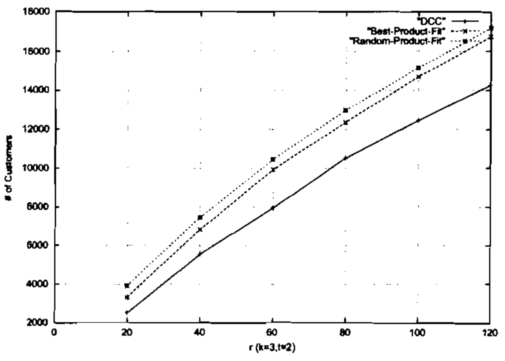


(a) Customers covered vs. t

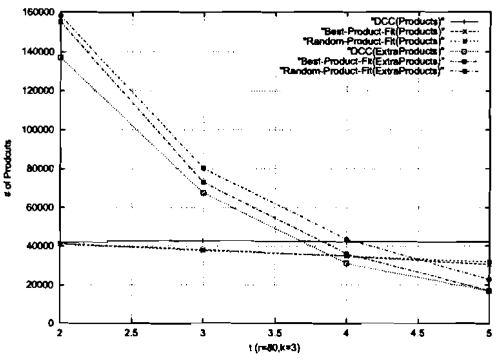


(b) Customers covered vs. k

Figure 3.3: Synthetic Dataset Test 1



(a) Customers covered vs. r

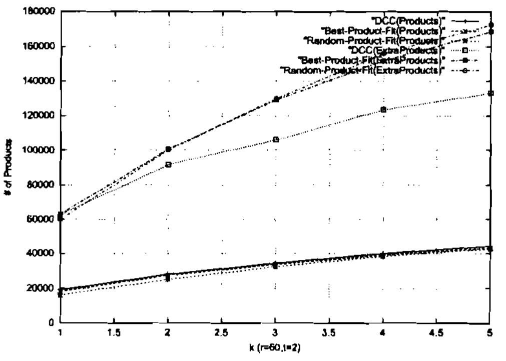


(b) Products covered vs. t

Figure 3.4: Synthetic Dataset Test 2

$r = 80$ and $k = 3$ in Figure 3.4(b). It is expected that DCC covers more products in the catalogs than our methods, but both Best-Product-Fit and Random-Product-Fit have higher extra profits on non-catalog products than DCC. Finally, Random-Product-Fit always covers more extra products than Best-Product-Fit.

It is clear to see the effects of different k values on the same quality measures for $r = 60$ and $t = 2$ in Figure 3.5(a). All three methods have similar performance w.r.t. the profit on catalog products. However, both of our methods clearly outperform DCC w.r.t. the extra profit from non-catalog products. For example, for $k = 5$, Random-Product-Fit achieves an extra profit of 40,000 products (30%) compared to DCC. We obtain similar results on the real dataset, e.g. with $r = 30$ and $t = 2$ (Figure 3.6(b)). Figure 3.5(b) shows how the


(a) Products covered vs. k

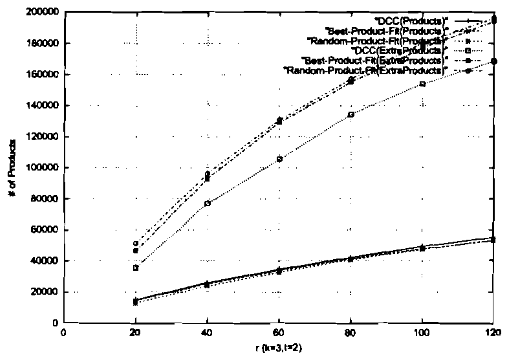
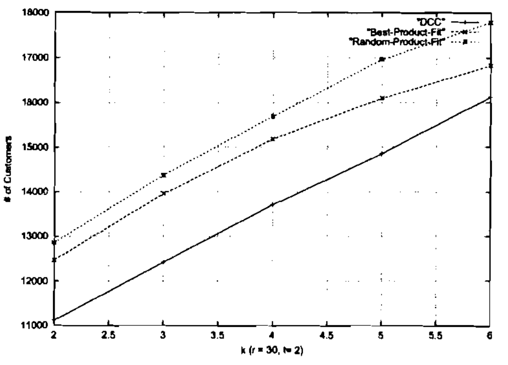

(b) Products covered vs. r

Figure 3.5: Synthetic Dataset Test 3


(a) Customers covered vs. k

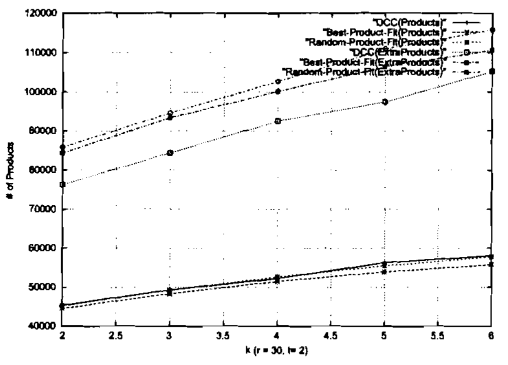

(b) Products covered vs. k

Figure 3.6: Real Data Test 1

size r of the catalog affects the number of covered catalog products and extra products with $k = 3$ and $t = 2$. The results are comparable to the results in Figure 3.5(a).

3.6 Summary

The microeconomic view of data mining is one of the most promising theoretical frameworks evaluating data mining results by their utilities in decision making. Since the utility function is often designed based on business needs, the generated clusters are more useful and actionable than the ones discovered by generic data-driven methods. This data mining framework has in particular been investigated for segmentation problems such as the Catalog Segmentation problem. In this chapter, we have investigated an alternative problem formulation integrating a minimum interest constraint t which allows us to measure the overall utility by the number of customers who are interested in at least t products in the catalog. We have formally introduced the Customer-Oriented Catalog Segmentation problem and discussed its complexity. We have presented efficient, heuristic algorithms adopting the paradigms of greedy and randomized algorithms. Our experimental evaluation on synthetic and real data showed that the new algorithms yield catalogs of significantly higher utility compared to classical Catalog Segmentation algorithms. Our best algorithm, Random-Product-Fit, achieves an excellent tradeoff between quality and runtime by optimizing a greedily determined initial solution in a randomized manner.

This research does not only have many promising applications, but also indicates several interesting directions that deserve further investigation. In order to better judge the relative utility values obtained by different algorithms, it is necessary to develop methods to estimate the utility of the optimal solution. The optimum can only be approximated since k-MECWT is NP-hard even for $k = 1, t = 1$. To make the k-MECWT model even more realistic, it could be generalized by replacing the crisp threshold by a probabilistic threshold, i.e., a customer would be attracted to a catalog with some probability. The Customer-Oriented Catalog Segmentation model could also be studied in the case that the number of catalogs is not set in advance, but there is a fixed cost for each catalog. Finally, Customer-Oriented Catalog Segmentation could be combined with association rule mining techniques to find novel types of customer purchase patterns.

Chapter 4

Joint Cluster Analysis

Attribute data and relationship data are two principal types of data, representing the intrinsic and extrinsic properties of entities. While attribute data have been the main source of data for cluster analysis, relationship data such as social networks or metabolic networks are becoming increasingly available. It is also common to observe both data types carry complementary information such as in market segmentation and community identification, which calls for a joint cluster analysis of both data types so as to achieve better results. In this chapter, we introduce the novel Connected k-Center problem, a joint clustering model taking into account both attribute data and relationship data. In the model, an internal connectedness constraint is defined on relationship data and the objective function measuring the compactness of the generated clusters is defined on attribute data. Integrating the constraints and the objective function allows us to search for clusters that are both cohesive (within clusters) and distinctive (between clusters) in both attribute data and relationship data. A preliminary version this Chapter was published in [44, 52].

4.1 Overview

Entities can be described by two principal types of data: attribute data and relationship data. Attribute data describe intrinsic characteristics of entities whereas relationship data represent extrinsic influences among entities. While attribute data have been the standard and dominant data source in data analysis applications, more and more relationship data are becoming available. Among them, to name a few, are acquaintance and collaboration networks as social networks, and ecological, neural and metabolic networks as biological networks. Consequently, network analysis [115, 95, 116] has been gaining popularity in the study of marketing, community identification, epidemiology, molecular biology and so on.

Depending on the application and the chosen data representation, the two types of data, attribute data and relationship data, can be more or less related. If the dependency between attribute data and relationship data is high enough such that one can be deduced from or closely approximated by the other, a separate analysis on either is sufficient. However, often relationship data contain information that goes beyond the information represented in the attributes of entities. For example, two persons may have many characteristics in common but they never got to know each other; on the other hand, even with very different demographics, they may happen to become good acquaintances. Due to rapid technological advances, the mobility and communication of humans have tremendously improved. As a consequence, the formation of social networks is slipping the leash of confining attributes [115, 95].

The unprecedented availability of relationship data carrying important additional information beyond attribute data calls for a joint analysis of both. Cluster analysis, one of the major tools in exploratory data analysis, has been investigated for decades in multiple disciplines such as statistics, machine learning, algorithms, and data mining. Varied clustering problems have been studied driven by numerous applications including pattern recognition, information retrieval, market segmentation and gene expression profile analysis, and emerging applications continue to inspire novel clustering models with new algorithmic challenges. The task of clustering is to group entities into clusters that exhibit internal cohesion and external isolation. Given both attribute data and relationship data, it is intuitive and necessary to require clusters to be cohesive (within clusters) and distinctive (between clusters) in both ways, which can only possibly result from a joint cluster analysis.

With profound differences in nature between the two data types, it is difficult to obtain a single combined objective measure for joint cluster analysis. Instead, we propose to optimize some objective derived from the continuous attribute data and to constrain the discrete relationship data. In this chapter, we introduce and study a novel clustering model taking into account both data types, the Connected k-Center (CkC) problem, which is essentially the k-Center problem with the constraint of internal connectedness on relationship data. The internal connectedness constraint requires that any two entities in a cluster are connected by an internal path, i.e., a path via entities only from the same cluster. The k-Center problem, as a classical clustering problem, has been intensively studied in the algorithms community from a theoretical perspective. The problem is to determine k cluster heads (centers) such that the maximum distance of any entity to its closest cluster head, the radius of the cluster, is minimized.

The *CkC* problem can be motivated by market segmentation, community identification, and many other applications such as document clustering, epidemic control, and gene expression profile analysis. In the following, we further discuss the first two applications.

Market segmentation is the process of dividing a market into distinct customer groups with homogeneous needs, such that firms can target groups effectively and allocate resources efficiently, as customers in the same segment are likely to respond similarly to a given marketing strategy. Traditional segmentation methods are based only on attribute data such as demographics (age, sex, ethnicity, income, education, religion, etc.) and psychographic profiles (lifestyle, personality, motives, etc.). Recently, social networks have become more and more important in marketing [68]. Ideas and behaviors are contagious. The relations in networks are channels and conduits through which resources flow [68]. Customers can hardly hear companies but they listen to their friends; customers are skeptical but they trust their friends [116]. By word-of-mouth propagation, a group of customers with similar attributes have much more chances to become like-minded. Depending on the nature of the market, social relations can become vital in forming segments, and purchasing intentions or decisions may rely on customer-to-customer contacts to diffuse throughout a segment, for example, for cautious clients of risky cosmetic surgery or parsimonious purchasers of complicated scientific instruments. The *CkC* problem naturally models such scenarios: a customer is assigned to a market segment only if he has similar purchasing preferences (attributes) to the segment representative (cluster center) and can be reached by propagation from customers of similar interest in the segment (the internal connectedness constraint).

Community identification is one of the major social network analysis tasks, and graph-based clustering methods have been the standard tool for the task [115]. In this application, clustering has generally been performed on relationship (network) data solely. Yet it is intuitive that attribute data can impact community formation in a significant manner [95, 64]. For example, given a scientific collaboration network, scientists can be separated into different research communities such that community members are not only connected (e.g., by co-author relations) but also share similar research interests. Such information on research interests can be automatically extracted from homepages and used as attribute data for the *CkC* problem. As a natural assumption, a community should be at least internally connected with possibly more constraints on the degree of connectivity. Note that most graph-based clustering methods used for community identification in network analysis also return some connected components.

The rest of the chapter is organized as follows. Related work is reviewed in Section 4.2. Section 4.3 introduces the *CkC* clustering problem and analyzes its complexity. In Section 4.4, we provide a constant factor approximation algorithm for the *CkC* problem. Section 4.5 studies the *CkC* problem for the special case of tree-structured relationship data. To achieve better scalability, in Section 4.6 we present the efficient heuristic algorithm NetScan. We report experimental results in Section 4.7. Finally, we discuss future directions in Section 4.8.

4.2 Background

We survey the related work to our model as follows.

Theory and algorithms: The *CkC* problem we study is essentially the k-Center problem with the constraint of internal connectedness on relationship data. It is well known that both the k-Center and Euclidean k-Center problems are NP-hard for d (dimensionality) ≥ 2 and arbitrary k [88]. However, as we will see in Section 4.3, the *CkC* problem remains NP-Complete even for $k = 2$ and $d = 1$. In this sense, the *CkC* problem is harder than the Euclidean k-Center problem.

Some facility location problems, such as the recently studied Connected k-Median problem [103], are closely related to our *CkC* problem. As a variant of the facility location problem, the Connected k-Median problem additionally considers the communication cost among facilities, whereas our *CkC* problem requires within-cluster connectedness. While all of these optimization problems are related to our study in the sense that they also study clustering from a theoretical perspective, they do not perform joint cluster analysis, and they do not require clusters to be cohesive with respect to both attribute data and relationship data.

Data mining: In the data mining community, clustering research emphasizes more on real life applications and development of efficient and scalable algorithms. While most clustering algorithms, e.g. k-Means, assume data to be represented in a single table, recently multi-relational clustering algorithms have been explored which can deal with a database consisting of multiple tables related via foreign key references. In particular, Taskar *et al.* [106] presented a multi-relational clustering method based on Probabilistic Relational Models (PRMs). PRMs are a first-order generalization of the well-known Bayesian Networks. The problem addressed in this chapter, i.e. clustering a single table with attributes and relationships, can be understood as a special case of multi-relational clustering. However, the approach by Taskar *et al.* is not applicable in this scenario since PRMs do not allow cycles which often occur in relationships within a single table.

Social network analysis and graph clustering: Recently, the increasing availability of relationship data has stimulated research on network analysis [115, 95, 64]. Clustering methods for network analysis are mostly graph-based, separating sparsely connected dense subgraphs from each other as in [23]. A good graph clustering should exhibit few between-cluster edges and many within-cluster edges. More precisely, graph clustering refers to a set of problems whose goal is to partition nodes of a network into groups so that some objective function is minimized. Several popular objective functions, e.g. *normalized cut* [97] and *ratio cut* [27], have been well studied. Those graph clustering problems can be effectively solved by spectral methods that make use of eigenvectors. Recently, Dhillon *et al.* [37] discovered the equivalence between a general kernel k-Means objective and a weighted graph clustering objective. They further utilize the equivalence to develop an effective multilevel algorithm, called GraClus, that optimizes several graph clustering objectives including the normalized cut. The experiments in [37] show that GraClus can beat the best spectral method on several clustering tasks.

Graph clustering methods can be applied to data that are originally network data. The original network can be weighted where weights normally represent the probability that two linked nodes belong to the same cluster [97]. In some cases, the probability is estimated by the distance between linked nodes on attribute data. Moreover, graph clustering methods can also be applied to similarity graphs representing similarity matrices, which are derived from attribute data. A similarity graph can be a complete graph as in the agglomerative hierarchical clustering algorithms, e.g., single-link, complete link, and average link [69], or incomplete retaining those edges whose corresponding similarity is above a threshold [60, 65]. CHAMELEON [74] generates edges between a vertex and its k nearest neighbors, which can be considered as relative thresholding.

There are two major differences between graph clustering, in particular the normalized cut, and CkC. On the one hand, the graph clustering model does not require the generated clusters to be internally connected which makes it somewhat more flexible than *CkC*. Yet, for some applications such as market segmentation and community identification, *CkC* fits better than the graph clustering model as these applications often require the generated clusters to be internally connected. On the other hand, in graph clustering, attribute data is used only indirectly by using distances between nodes as edges weights, which may lose important information since it reduces the d-dimensional attribute data of two connected nodes to a single, relative distance value. In *CkC*, attribute data are used directly, avoiding information loss.

Constrained clustering and semi-supervised clustering: Joint cluster analysis is also related to the emerging areas of semi-supervised clustering. The existing semi-supervised methods are similar to our research in the sense that they also adopt a k-clustering approach under the framework of constrained clustering. Nevertheless, in semi-supervised clustering, links represent specific instance-level constraints on attribute data. They are provided by the user to capture some background knowledge. In our study, links represent relationship data. They are not constraints themselves, but data on which different constraints can be enforced, such as being "internally connected". Enforcing the connectedness constraint should lead to cohesion of clusters with respect to relationship data, so that clusters can be cohesive in both ways.

Bioinformatics: There have been several research efforts that consider both attribute and relationship data in the bioinformatics literature. With the goal of identifying functional modules, Hanisch *et al.* [63] proposed a co-clustering method for biological networks and gene expression data by constructing a distance function that combines the expression distance and the network distance. However, their method cannot guarantee that the resulting clusters are connected. Segal *et al.* [96] introduced a probabilistic graphical model, combining a Naive Bayes model for the expression data and a Markov random field for the network data. While the probabilistic framework has the advantage of representing the uncertainty of cluster assignments, it cannot ensure the connectedness of the resulting clusters. Ulitsky and Shamir [109] presented an algorithmic framework for clustering gene data. Given a gene network and expression similarity values, they seek heavy subgraphs in an edge-weighted similarity graph. Similar to our model, this model requires the generated clusters to be connected. Different from the *CkC* model, their model does not search for a partition of the whole dataset, i.e. , not every gene needs to be assigned to a cluster.

4.3 Problem Definition and Complexity Analysis

In this section, we formally define the Connected k-Center (CkC) problem. We prove the NP-completeness of the decision version of the CkC problem through a reduction from the 3SAT problem. The key observation in this proof is the existence of so-called "bridge" nodes, which can be assigned to multiple centers and are crucial to link some other nodes to their corresponding centers within a certain radius. We construct a polynomial reduction showing that finding the assignment of these bridge nodes is at least as hard as finding the satisfying assignment for any instance of 3SAT.

4.3.1 Preliminaries and problem definition.

Attribute data can be represented as an $n \times m$ entity-attribute matrix. Based on a chosen similarity measure, pairwise similarities can be calculated to obtain an $n \times n$ entity-entity similarity matrix. Relationship data are usually modeled by networks comprised of nodes and links, which we call entity networks. In this chapter, we concentrate on symmetric binary relations, thereby entity networks can be naturally represented as simple graphs with edges (links) as dichotomous variables indicating the presence or absence of a relation of interest such as acquaintance, collaboration, or transmission of information or diseases.

Nodes in an entity network do not have meaningful locations. With attribute data available, attributes for each entity can be represented as a coordinate vector and assigned to the corresponding node, resulting in what we call an "informative graph". Informative graphs, with both attribute data and relationship data embedded, are used as input for our Connected k-Center problem.

In this chapter, the terms "vertex" and "node" are used interchangeably, so are "edge" and "link". In the following sections, "graph" will refer to "informative graph" since we always consider the two data types simultaneously.

The Connected k-Center (CkC) problem performs a joint cluster analysis on attribute data and relationship data, so that clusters are cohesive in both ways. The problem is to find a disjoint k-clustering (k-partitioning) of a set of nodes, such that each cluster satisfies the internal connectedness constraint (defined on the relationship data), and the maximum radius (defined on the attribute data) is minimized. The radius of a cluster is the maximum distance of any node in the cluster to the corresponding center node. A formal definition of the CkC problem is given in the following.

Definition (*CkC* problem) Given an integer k, a graph $G = (V, E)$, a function $w : V \rightarrow \mathcal{R}^d$ mapping each node in V to a d-dimensional coordinate vector, and a distance function $|| \cdot ||$, find a k-partitioning $\{V_1, \dots, V_k\}$ of V, i.e. , $V_1 \cup \dots \cup V_k = V$ and $\forall 1 \leq i < j \leq k, V_i \cap V_j = \phi$, such that the partitions satisfy the *internal connectedness constraint*, i.e., the induced subgraphs $G[V_1], \dots, G[V_k]$ are connected, and the maximum radius defined on $|| \cdot ||$ is minimized.

In this study, we assume the given graph G is connected, which is reasonable for many application scenarios, e.g., social networks are normally considered to be connected. Even if the entire graph is not connected, the problem can still be applied to individual connected components.

4.3.2 Complexity analysis.

Due to the similarity of the *CkC* problem to the traditional k-Center problem, it is natural to ask the following question: *How much has the traditional k-Center problem been changed in terms of hardness by adding the constraint of internal connectedness?* To answer this question, we analyze the complexity of the *CkC* problem. In the following, we define the decision version of the *CkC* problem and prove its NP-completeness. Note that in this subsection of complexity analysis, the names of the problems refer to their decision versions.

Definition (*CkC* problem, decision version) Given an integer k, a graph $G = (V, E)$, a function $w : V \rightarrow \mathcal{R}^d$ mapping each node in V to a d-dimensional coordinate vector, a distance function $|| \cdot ||$, and a radius threshold $r \in \mathcal{R}^+$, decide whether there exists a k-partitioning $\{V_1, \dots, V_k\}$ of V, i.e. , $V_1 \cup \dots \cup V_k = V$ and $\forall 1 \leq i < j \leq k, V_i \cap V_j = \phi$, such that in addition to the internal connectedness constraint, the partitions also satisfy the *radius constraint*, i.e., $\forall 1 \leq i \leq k$, there exists a center node $c_i \in V_i$, such that $\forall v \in V_i$, $||w(v) - w(c_i)|| \leq r$.

Intuitively, the problem is to check whether the input graph can be divided into k disjoint connected components, such that each component is a cluster with radius less than or equal to r, i.e. , in each cluster, there exists a center node c and all the remaining nodes are within distance r to c.

We will prove an NP-completeness result for fixed k. As the formal analysis is rather technical, we precede it with an intuitive explanation. We say a solution (or partitioning) is *legal* if all the k partitions (or clusters) are disjoint and the corresponding induced subgraphs are connected. Since k is fixed as a constant, a naive algorithm would enumerate all the combinations of k centers, and for each combination assign the remaining nodes to the centers such that both the internal connectedness and radius constraints are satisfied. However, we note that there may exist some "bridge" node v which can connect to multiple centers within distance r and is critical to connect some other nodes to their corresponding centers. In a legal partitioning, every bridge node must be assigned to a unique center. If there are many such bridge nodes, it is difficult to assign each of them to the "right" center in order to maintain the connection for others. Therefore, the naive algorithm may fail to determine a legal partitioning. Intuitively, the CkC problem is hard even for a fixed k. In the following, we prove a hardness result for the CkC problem by a reduction from 3SAT. For convenience, we state the 3SAT problem as follows:

Definition (*3SAT problem*) Given a set $U = \{u_1, \ldots, u_n\}$ of variables, a boolean formula $I = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ where each clause $C_i = l_i^1 \vee l_i^2 \vee l_i^3$ contains three literals and each literal l_i^x, $x = 1, 2, 3$, is a variable or negated variable, decide whether there exists a truth assignment of U that satisfies every clause of C.

Theorem 4.3.1 *For any $k \geq 2$ and $d \geq 1$, the CkC problem is NP-Complete.*

Proof. We only construct a proof for the case of $k = 2$ and $d = 1$, the proof can be easily extended to any larger k and d.

First, we show $C2C$ is in NP. We can nondeterministically guess a partitioning of graph G and pick a node as center from each partition. For each partition, we can traverse the corresponding subgraph in polynomial time to verify whether it is a legal partitioning satisfying the radius constraint.

Next, we perform a reduction from 3SAT to show the NP-hardness of $C2C$. Let $L = \{u_1, \overline{u}_1, \ldots, u_n, \overline{u}_n\}$ be a set of literals. For any 3SAT instance $I = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, we construct a $C2C$ instance $f(I) = (G, w, r)$, where $G = (V, E)$ is the underlying graph, $w : V \rightarrow R$ is the function mapping nodes to coordinate vectors, and $r \in \mathcal{R}^+$ is the radius constraint, by the following procedure:

1. Create a set of nodes $V = P \cup L \cup C \cup A \cup B$. $P = \{p_0, p_1\}$ where p_0 and p_1 are two center nodes. L and C are the sets of literals and clauses respectively. $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ are two sets of nodes introduced only for the purpose of the reduction.

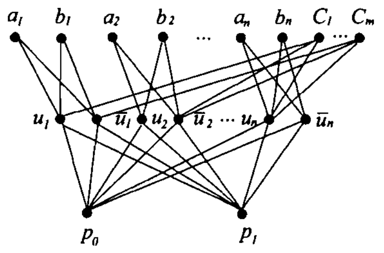


Figure 4.1: Constructed graph G.

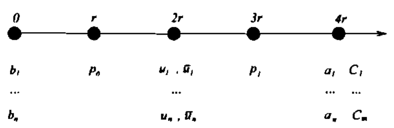


Figure 4.2: Deployment of nodes on the line.

2. Connect the nodes created in step (1). We link each literal $l \in L$ to both p_0 and p_1. For each literal $l \in L$ and clause $C_i \in C$, we link l to C_i if $l \in C_i$. For each $i \in \{1, 2, \ldots, n\}$, we link a_i and b_i to both u_i and $\overline{u}_i$.

3. Set an arbitrary positive value to r and assign each node $v \in V$ a coordinate as follows:

$$w(v) = \begin{cases} 0, & \text{if } v \in B; \\ r, & \text{if } v = p_0; \\ 2r, & \text{if } v \in L; \\ 3r, & \text{if } v = p_1; \\ 4r, & \text{if } v \in A \cup C. \end{cases}$$

Steps (1) and (2) construct the underlying graph G. A visual explanation of the construction method is provided in Figure 4.1. Note that every node in A, B, C can only connect to the center nodes p_0 and p_1 via some nodes in L.

Step (3) assigns each node in V a carefully chosen coordinate, such that each node in A, B, C is within distance r to one unique center node p_0 or p_1. Figure 4.2 illustrates the deployment of nodes on the line.

In order to have a legal partitioning (partitions are disjoint and satisfy the internal connectedness constraint), every node in L must be assigned to an appropriate center (cluster). For the reduction, we associate a truth value (true or false) to each cluster; accordingly, the allocations of these nodes can then be transferred back to a truth assignment for the input 3SAT instance I. Besides, we need to guarantee that the truth assignment for I is proper, i.e., $\forall i \in \{1, 2, \ldots, n\}$, node u_i and $\overline{u}_i$ belong to different clusters. Node sets A and B are two gadgets introduced for this purpose.

Clearly the above reduction is polynomial. Next, we show I is satisfiable if and only if $f(I) = (G, w, r)$ has a legal partitioning satisfying the radius constraint. We use V_0 and V_1 to refer to the clusters centered at p_0 and p_1 respectively.

If $f(I) = (G, w, r)$ has a legal partitioning satisfying the radius constraint, we have the following simple observations:

1. Both p_0 and p_1 must be selected as centers, otherwise some node cannot be reached within distance r.
2. For the same reason, each node in A and C must be assigned to cluster V_1 and each node in B must be assigned to V_0.
3. For any $i \in \{1, \ldots, n\}$, u_i and $\overline{u}_i$ cannot be in the same cluster. If u_i and $\overline{u}_i$ are both assigned to cluster V_0 (or V_1), some node in A (or B) would not be able to connect to p_1 (or p_0).
4. For each clause $C_i \in C$, there must be at least one literal assigned to cluster V_1, otherwise C_i will be disconnected from p_1.

We construct a satisfying assignment for I as follows: For each variable $u_i \in U$, if u_i is assigned to V_1, set u_i to be true, otherwise false. Note by observation (3), u_i and $\overline{u}_i$ are always assigned different values, hence the assignment is proper. Moreover, the assignment satisfies I since by observation (4), all the clauses are satisfied.

If I is satisfiable, we construct a partitioning $\{V_0, V_1\}$ as follows:

$$V_0 = B \cup \{p_0\} \cup \{l_i \in L \mid l_i = false\}$$
$$V_1 = V \setminus V_0$$

It is easy to verify that the above partitioning is legal. In addition, the radius constraint is satisfied since every node in V is within distance r from its corresponding center node, p_0 or p_1.

Finally, we show that the above proof can be easily extended to any larger k and d. When $k > 2$, one can always add $k-2$ isolated nodes (hence each of them must be a center) to graph G and apply the same reduction; when $d > 1$, one can simply add $d-1$ coordinates with identical values to the existing coordinate for each node. $\square$

The internal connectedness constraint poses new challenges to the traditional k-Center problem. Table 4.1 compares the hardness of these two problems in different settings. Note that the referred problems are decision versions.

	Traditional k Center	*CkC*
k is fixed	Polynomially Solvable	NP-complete
k is arbitrary, $d = 1$	Polynomially Solvable	NP-complete
k is arbitrary, $d > 1$	NP-complete	NP-complete

Table 4.1: Complexity results.

Remarks:

1. Theorem 4.3.1 implies the *CkC* problem defined in Definition 4.3.1 is NP-hard.
2. Similar to the *CkC* problem, one can define the connected k-Median and connected k-Means problems. In fact, the proof of Theorem 4.3.1 can be extended to these problems to show their NP-Completeness.

4.4 Approximation Algorithms

In this section we study the *CkC* problem defined in Definition 4.3.1. We prove that the problem is not approximable within $2 - \epsilon$ for any $\epsilon > 0$ unless $P = NP$. When the distance function is metric, we provide approximation algorithms with ratios of 3 and 6, respectively, for the cases of fixed and arbitrary k. The idea is to tackle an auxiliary CkC' problem. Based on the solution of CkC', we show the gap between these two problems is at most 3, i.e., a feasible solution of CkC' with radius r can always be transferred to a feasible solution of *CkC* with radius at most $3r$. We also prove a lower bound result indicating that the gap cannot be closed within 2.64.

4.4.1 Inapproximability result for *CkC*

In the following, we prove an inapproximability result for the *CkC* problem, which can be viewed as a corollary of Theorem 4.3.1.

Theorem 4.4.1 *For any $k \geq 2$, $\epsilon > 0$, the CkC problem is not approximable within $2 - \epsilon$ unless $P = NP$.*

Proof. We only prove the case of $k = 2$, the proof can be easily extended to any larger k based on the same argument as in the proof of Theorem 4.3.1.

Let *opt* denote the optimal radius of the CkC problem. We show if there is a polynomial algorithm $\mathcal{A}$ guaranteed to find a feasible solution within $(2 - \epsilon)opt$, it can actually be used to solve the 3SAT problem. The reduction is similar to the proof of Theorem 4.3.1. First, for a given 3SAT instance I, we construct a $C2C$ instance $f(I) = (G, w, r)$ by the same procedure as in the proof of Theorem 4.3.1. Then, we invoke Algorithm $\mathcal{A}$ on the input (G, w, r).

Since the coordinates of all the nodes are multiples of r, the optimal radius must also be a multiple of r. If Algorithm $\mathcal{A}$ returns a solution smaller than $2r$, the optimal radius must be r. By the same argument as in the proof of Theorem 4.3.1, I is satisfiable. Otherwise if Algorithm $\mathcal{A}$ returns a solution bigger than or equal to $2r$, since Algorithm $\mathcal{A}$ is guaranteed to find a solution within $(2 - \epsilon)r$, the optimal radius is at least $2r$ and consequently I is not satisfiable. Hence, unless $P = NP$, the CkC problem cannot be approximated within $2 - \epsilon$.

□

4.4.2 Approximation results for metric CkC

In the following, we study approximation algorithms for the CkC problem in the metric space. Our approximation results rely on the triangle inequality. However, our hardness results presented in Section 4.3 remains valid even for non-metric spaces.

We provide approximations with ratios 3 and 6 for the cases of fixed and arbitrary k. For this purpose, we introduce the CkC' problem, which is a relaxed version of the CkC problem without stipulating the disjointness requirement on the clusters. Then we show that CkC' can be solved in polynomial time for fixed k and approximated within a factor of 2 for arbitrary k. We then show the gap between these two problems is at most 3.

Definition (CkC' problem) Given an integer k, a graph $G = (V, E)$, a function $w : V \rightarrow \mathcal{R}^d$ mapping each node in V to a d-dimensional coordinate vector, and a distance function $||\cdot||$, find k node sets $V_1, \ldots, V_k \subseteq V$ with $V_1 \cup \ldots \cup V_k = V$, such that the node sets satisfy the internal connectedness constraint and the maximum radius defined on $||\cdot||$ is minimized.

Algorithm 3 Polynomial exact algorithm for CkC'.

```
1: Calculate all the pairwise distances for the nodes in $V$ and store them in set $R$;
2: Sort $R$ in increasing order;
3: $low = 0; high = |R|$;
4: while $low \leq high$ do
5:   $middle = (low + high)/2$;
6:   $r = R[middle]$;
7:   for each set of $k$ centers $\{c_1, \ldots, c_k\} \subseteq V$ do
8:     Perform BFS from each center $c_i$ and mark all the nodes that are reachable from $c_i$ w.r.t. $r$;
9:     if all nodes are marked then
10:      if $low = high$ then
11:        Return $r$ and the $k$ clusters;
12:      else
13:        $high = middle - 1$;
14:    else
15:      $low = middle + 1$;
```

Solving CkC' for fixed k

We propose an exact algorithm to solve the CkC' problem for fixed k in polynomial time. We define the reachability between any two nodes as follows:

Definition Given a graph $G = (V, E)$, for $u, v \in V$, v is *reachable* from u w.r.t. r, $r \in \mathcal{R}^+$, if there exists a path $\{u = s_0 \to s_1 \to \ldots \to s_l \to s_{l+1} = v\}$, $s_1, \ldots, s_l \in V$, such that $\forall 1 \leq i \leq l+1$, $(s_{i-1}, s_i) \in E$ and $||w(u) - w(s_i)|| \leq r$.

Intuitively, v is reachable from u w.r.t. r if and only if v can be included in the cluster with center u and radius r. Clearly it can be decided in polynomial time by performing a breadth first search (BFS) for node v from node u. This forms the main idea of Algorithm 3, which returns the optimal solution for CkC' in polynomial time.

Runtime complexity: Algorithm 3 performs $O(n^k \log n)$ times of BFS since it iterates over all possible sets of k centers, and a binary search is performed for all possible $r \in R$ where $|R| = \binom{n}{2}$. Since every BFS takes $O(n^2)$ steps, the total running time of Algorithm 3 is $O(n^{k+2} \log n)$.

Approximating CkC' for arbitrary k

For the case k is fixed, we show an approach providing a 2 approximation for the CkC' problem. We define the *reaching distance* between any two nodes as follows:

Definition Let G, u, v, p be defined as in the above definition. The distance between u and v w.r.t p is defined as $D(u,v)_p = \max_{s_i, s_j \in p} ||w(s_i) - w(s_j)||$. The reaching distance between u and v is defined as $D(u,v) = \min_{p \in P} D(u,v)_p$, where P is the set of all paths between u and v.

Note that the reaching distance is symmetric, i.e., $\forall u, v \in V, dist(u,v) = dist(v,u)$. It also satisfies the triangle inequality, i.e., $\forall u, v, s \in V, dist(u,s) \leq dist(u,v) + dist(v,s)$. We can obtain a $|V| \times |V|$ matrix, storing reaching distances for all the nodes in V. Then, we can apply the 2-approximation algorithm proposed in [67] on V with the reaching distance matrix replacing the pairwise distance matrix. The maximum radius of the k clusters resulting from this algorithm is at most twice as big as the optimal solution.

Back to CkC

In Algorithm 4, we present a method transferring a feasible solution of CkC' with radius r to a feasible solution of CkC with radius at most $3r$. Combining the CkC' results and Algorithm 4 gives approximations for the CkC problem.

Let $\{V'_1, \ldots, V'_k\}$ be a clustering returned by Algorithm 3 or the approximation algorithm specified in Section 4.4.2 where $V'_i \subseteq V$ and the node sets (clusters) $V'_1, \ldots, V'_k$ may not be disjoint. Algorithm 4 determines a clustering $\{V_1, \ldots, V_k\}$ with disjoint node sets $V_1, \ldots, V_k$. Let $c_1, \ldots, c_k$ be the centers of $V_1, \ldots, V_k$. Since the algorithm retains the cluster centers, they are also the centers of $V'_1, \ldots, V'_k$. Algorithm 4 assigns every node in V to a unique cluster V_i for $1 \leq i \leq k$. For each iteration $1 \leq i \leq k$, line 3 assigns the nodes in V'_i that have not been assigned to any previous clusters $V_1, \ldots, V_{i-1}$ and are connected to c_i to V_i. Afterwards, there may still be some unassigned nodes in V'_i, and line 5 assigns them to one of the clusters $V_1, \ldots, V_{i-1}$ to which they are connected.

Figure 4.3 provides an illustration for Algorithm 4. The circles with dashed lines represent the three initial (overlapping) clusters V'_1, V'_2 and V'_3 generated by Algorithm 3. Applying Algorithm 4, we obtain three new disjoint clusters V_1, V_2 and V_3. The center nodes were not changed.

Algorithm 4 Algorithm converting a solution of CkC' to a solution of CkC.

```
1: for $i$ from 1 to $k$ do
2:   $V_i = \phi$, $c_i \leftarrow c'_i$;
3:   Add all the nodes reachable w.r.t. $r$ from $c_i$ in $G[V'_i \setminus \cup_{j=1}^{i-1} V_j]$ to $V_i$ (by performing a
     BFS from $c_i$ in $G[V'_i \setminus \cup_{j=1}^{i-1} V_j]$);
4:   for every node $v \in \left(\cup_{j=1}^{i-1} V_j\right) \cap V'_i$ do
5:     Add all the nodes connected to $v$ in $G[V'_i]$ to the cluster of $v$ (by performing a BFS
       from $v$ in $G[V'_i]$);
6: Output clusters $V_1, \ldots, V_k$;
```

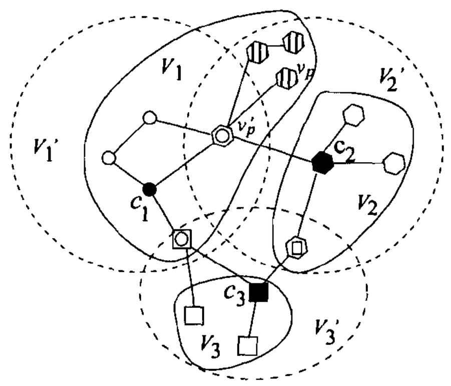


Figure 4.3: Illustration of Algorithm 4.

Lemma 4.4.2 *Let r be the maximum radius associated with a feasible solution for the CkC' problem. Algorithm 4 is guaranteed to find a feasible solution for the CkC problem with maximum radius at most $3r$.*

Proof. First we show that Algorithm 4 assigns each node $u \in V$ to a unique cluster. There are two cases. In case 1, u can be reached via a path from center node c_i without having any node previously assigned to $V_1, \ldots, V_{i-1}$ on the path; then, u is assigned to V_i in line 3 of Algorithm 4. In case 2, u is connected to c_i via some node $v \in \cup_{j=1}^{i-1} V_j$; then, in line 5 of Algorithm 4, u is assigned to the cluster that v belongs to.

Next, we bound the maximum radius of a node u to the corresponding center node. In case 1, since u is assigned to V_i, the distance between u and c_i is at most r. In case 2, observe that the distance between u and v is at most $2r$ due to the triangle inequality and the fact

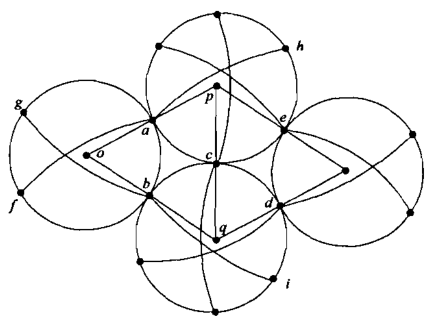


Figure 4.4: Lower bound for the gap between *CkC* and *CkC'*.

that u and v were in the same set V'_i. Besides, we observe that the distance between v and its corresponding center node c_j is at most r. Therefore, again by the triangle inequality, the distance between u and its corresponding center node is at most $3r$.

□

Let *opt* and *opt'* be the optimal solutions for the *CkC* and *CkC'* problems respectively. Clearly *opt'* $\leq$ *opt* since *opt* is also a feasible solution for *CkC'*. Based on this observation, we obtain the following approximation results for *CkC*:

Theorem 4.4.3 *Combining Algorithm 3 and Algorithm 4 gives a polynomial 3-approximation for the CkC problem for fixed k.*

Theorem 4.4.4 *Combining the approach proposed in 4.4.2 and Algorithm 4 gives a polynomial 6-approximation for the CkC problem for arbitrary k.*

Lower bound for the gap between *CkC* and *CkC'*

In Lemma 4.4.2, we have proved that the gap between *CkC* and *CkC'* is at most 3. In the following, we show the gap is at least 2.64.

In the graph shown in Figure 4.4, each node is associated with a 2-dimensional coordinate vector, and the nodes are placed according to their coordinates. The edges between nodes are also shown. The graph is symmetric and the circles are contingent to one another. The nodes f, o, a, p, h are on the same line, so are g, o, b, q, i. In the *CkC'* model, nodes can have multiple memberships. Let $k = 4$, it is easy to construct a feasible clustering for CkC', represented by the circles, where each cluster has radius r.

Now, we examine the optimal clustering of CkC. By the pigeonhole principle, at least two of the five nodes, a, b, c, d, e, have to be assigned to the same cluster. Without loss of generality, we assume a and b belong to the same cluster. Since h and f only connect to a and g and i only connect to b, h, f, g, i, a, b have to be assigned to the same cluster. To minimize the cluster size, either a or b must be the cluster center. The cluster radius is then $dist(a, i)$ or $dist(b, h)$, which is $\sqrt{7}r \approx 2.64r$[1]. We omit the mathematical details.

To summarize, we list the approximability results of CkC in Table 4.2.

Approximation Ratio	Upper bound	Lower bound
k is fixed	3	$2 - \epsilon$
k is arbitrary	6	$2 - \epsilon$

Table 4.2: Approximability results.

4.5 Exact algorithm for *CkC* on Trees

As shown in Theorem 4.3.1, the CkC problem is NP-hard for general graphs. A natural question to ask is whether the problem is tractable for certain subclasses of graphs, e.g., tree graphs. Tree structure is exhibited in common *organization charts*, which graphically illustrate how authority and responsibility are distributed within organizations. Although organization charts generally capture formal relationships only, they can be used to approximate the pattern of social relationships, as informal (social) relationships may develop in accordance with formal relationships in many circumstances.

In this section, we present a dynamic programming approach giving an optimal solution for the CkC problem on trees in $O(n^2 \log n)$ time.

4.5.1 Polynomial exact algorithm for *CkC* on trees

Algorithm 5 returns an optimal solution for the CkC problem on trees in $O(n^2 \log n)$ time. The algorithm starts with calculating all $|V|(|V| - 1)/2$ pairwise distances of the nodes

[1]Since o, b, q, i are on the same line and the angle aoi is 60 degree, we can calculate $dist(a, i)$ based on cosine law.

Algorithm 5 Polynomial exact algorithm for *CkC* on trees.

```
1: Calculate all the pairwise distances for the nodes in $V$ and store them in set $R$;
2: Sort $R$ in increasing order;
3: $low = 0; high = |R|$;
4: while $low \leq high$ do
5:   $middle = (low + high)/2$;
6:   $r = R[middle]$;
7:   Call the dynamic programming algorithm specified in section 4.5.2 on the decision
     version of $CkC$ with $r$ as the radius threshold;
8:   if the dynamic programming algorithm returns "yes" then
9:     if $low = high$ then
10:      Construct the $k$-partitioning from the dynamic programming table and return $r$;
11:    else
12:      $high = middle - 1$;
13:  else
14:    $low = middle + 1$;
```

in V and sorts them in increasing order. Then we perform a binary search to find the smallest distance which is feasible. The feasibility can be decided by invoking a dynamic programming procedure presented as follows.

4.5.2 Dynamic programming algorithm

The dynamic programming algorithm solves the decision version of the CkC problem defined in Definition 4.3.2 where the underlying graph is a tree. For simplicity, we consider binary trees. An arbitrary tree can be transformed to a binary one by the approach presented in [105]. The algorithm returns "yes" if and only if it finds a feasible k-partitioning of nodes satisfying the internal connectedness and radius constraints. Each partition represents a cluster.

Let $T = (V, E)$ be a binary tree with $|V| = n$. For an arbitrary node v, let $T(v)$ be the subtree rooted at v and $C(v)$ be the set of direct descendants of v. Let $P(v_i, v_j)$ denote the set of nodes on the path from v_i to v_j. Note that the path between any pair of nodes is unique for trees. In a feasible (partial) solution, we say node v_j *serves* node v_i if v_i is assigned to some cluster centered at v_j. This requires that $\forall v_k \in P(v_i, v_j)$, $||w(v_k) - w(v_j)|| \leq r$. Let $f(v_i, v_j)$ be the minimum number of clusters in a feasible clustering of $T(v_i)$ w.r.t. r when v_i is served by v_j. Note that v_i can be served by itself and v_j does not have to be an element of $T(v_i)$.

Initial Step: Initially, we arbitrarily pick a non-leaf node of T as the root. Our dynamic programming algorithm starts by computing $f(v_l, v_i)$ for every leaf node v_l and every node $v_i \in V$. Note that $f(v_l, v_i) = \infty$ if $\exists v_j \in P(v_l, v_i), ||w(v_j) - w(v_i)|| > r$; otherwise, $f(v_l, v_i) = 1$.

Recursive Step: We recursively compute f values for all non-leaf nodes. For any non-leaf node v, $f(v, v_i) = \infty$ if $\exists v_j \in P(v, v_i), ||w(v_j) - w(v_i)|| > r$. Note that $f(v, v_i)$ can be computed whenever the f values of the nodes in $C(v)$ are available.

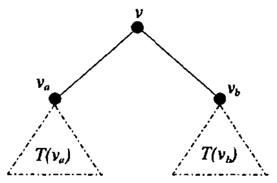


Figure 4.5: The tree structure.

Fix some arbitrary node v where the f values of all of its descendants have been computed, Figure 4.5 shows a partial binary tree. There are four cases for computing $f(v, v_i)$ depending on how v is served by v_i.

$$f(v, v_i) = \begin{cases} f(v_a, v_i) + f(v_b, v_i) - 1 & \text{if } v_i \text{ serves } v, v_a \text{ and } v_b, \\ f(v_a, v_i) + \min\limits_{v_j \in T(v_b)} f(v_b, v_j) & \text{if } v_i \text{ serves } v \text{ and } v_a \text{ but not } v_b, \\ f(v_b, v_i) + \min\limits_{v_j \in T(v_a)} f(v_a, v_j) & \text{if } v_i \text{ serves } v \text{ and } v_b \text{ but not } v_a, \\ 1 + \min\limits_{v_j \in T(v_a)} f(v_a, v_j) & \\ + \min\limits_{v_k \in T(v_b)} f(v_b, v_k) & \text{if } v_i \text{ serves } v \text{ but not } v_a \text{ nor } v_b. \end{cases}$$

In the first case, we need to subtract 1 since the cluster was counted twice when $f(v_a, v_i)$ and $f(v_b, v_i)$ were computed. In the second case, note that $v_i \notin T(v_b)$ since otherwise v_i must serve v_b in order to serve v and v_a. Similarly, in the third case $v_i \notin T(v_a)$.

Observe that in an optimal solution, there are only four possible cases to assign node v, v_a and v_b. For simplicity, we only elaborate on one of these cases, i.e., v_i serves v and v_a but not v_b. We show that the *CkC* problem on trees has an optimal-substructure property for this case. Similar arguments hold for other cases.

In an optimal clustering where v_i serves v and v_a but not v_b, $f(v_a, v_i)$ represents the minimum number of clusters in $T(v_a)$ when v_i serves v_a. $\min_{v_j \in T(v_b)} f(v_b, v_j)$ is the minimum number of clusters in $T(v_b)$ since v_b cannot be served by any node outside of $T(v_b)$ due to the internal connectedness constraint. Summing up the two items gives the minimum number of clusters in this optimal clustering.

Thus, we compute an $n \times n$ table with $f(v_i, v_j), \forall 1 \leq i, j \leq n$. There is a feasible k-partitioning, and a "yes" is thus returned, if and only if $\min_{v_i \in V} f(v_{root}, v_i) \leq k$.

Runtime complexity: To trade space for efficiency, we can use an additional data structure of size n to store $\min_{v_j \in T(v_a)} f(v_a, v_j)$ for every node v_a. Then every table entry can be calculated in $O(1)$ steps. Since there are n^2 table entries to be filled in, the runtime of the dynamic programming algorithm is $O(n^2)$. Since the dynamic programming is executed at most $O(\log n)$ times due to binary search, the overall runtime for Algorithm 5 is $O(n^2 \log n)$.

4.6 Heuristic Algorithm

The complexity analysis has demonstrated the hardness of the general case of the *CkC* problem. Meanwhile, algorithm 4 provides a way to guarantee the performance. Moreover, Theorem 4.3.1 implies that even the assignment step alone, i.e., given k centers finding an optimal assignment of the remaining nodes to minimize the radius, is NP-hard. While providing an algorithm with guaranteed approximation performance is important from the theoretical point of view, the expensive enumeration operation makes the approach infeasible for real large datasets. In this section, we present *NetScan*, a heuristic algorithm that efficiently produces a "good" solution for the *CkC* problem.

4.6.1 Overview of NetScan

NetScan follows a three-step approach. It starts by picking k centers randomly, then assigns nodes to the best centers and refines the clusters iteratively.

- Step I: Randomly pick k nodes as initial cluster centers.
- Step II: Assign all the nodes to clusters by traversing the input graph.
- Step III: Recalculate cluster centers.

The algorithm repeats steps II and III until no change of the cluster centers occurs or a certain number of iterations have been performed. In step III, finding the optimal center

	Cluster 1	Cluster 2
R_0	$\{g\}$	$\{e, f\}$
R_1	$\{g, h, i\}$	$\{e, f, a, b\}$

Table 4.3: Node assignment w.r.t. R_0 and R_1.

from a group of n nodes requires $O(n^2)$ time. For efficiency, we select the node closest to the mean of the cluster as the new center. Typically, the mean provides a reasonably good approximation for the center.

The three-step framework resembles the k-Means algorithm. However, unlike the straightforward assignment step in k-Means, given k centers, finding an optimal assignment satisfying the connectedness constraint requires a search through an exponential space, as shown in Section 4.3. Thus, the major challenge of NetScan is finding a good membership assignment, i.e., step II.

From the design principles of the approximation algorithm, we observe that the BFS-based approach provides an efficient way of generating clusters without violating the internal connectedness constraint. Inspired by this observation, we start the membership assignment from the centers, and neighboring nodes (directly connected by some edge of the graph) of already assigned nodes are gradually absorbed to the clusters. The whole step II may take multiple rounds to finish until all the nodes are assigned, and each round i is associated with a radius threshold R_i. For the first round, the assignment starts from cluster centers with the initial radius threshold R_0. Each node is tested and assigned to the first cluster for which its distance to the cluster center is no larger than R_0. If all the centers have been processed but not all the nodes have been assigned, the next assignment round tries to assign them with an incremented radius threshold R_1. The process continues until all the nodes are assigned. A running example is illustrated in Figure 4.6 with the assignment rounds given in Table 4.3. In the figure, g and e are chosen as the initial cluster centers. In the first round with R_0 as the radius threshold, cluster 1 has no new members while cluster 2 has f added. In the second round with R_1 as the radius threshold, h and i are assigned to cluster 1 while a and b are assigned to cluster 2. The pseudocode of step II is given in Algorithm 6, and more details of NetScan will be discussed shortly.

Algorithm 6 Step II of NetScan.

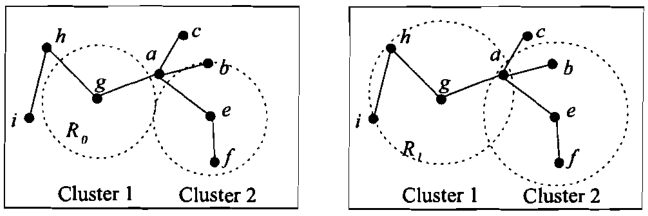
```
1: $R_i = R_0$;
2: Empty working queue $Q$;
3: for every center $c_j$ of cluster $C_j$ do
4:   Append all unassigned neighbors of $c_j$ to $Q$;
5:   while $Q$ is not empty do
6:     Pop the first element $q$ from $Q$;
7:     if $||q - c_j|| \leq R_i$ then
8:       if $q$ is a potential bridge node then
9:         Invoke the look-ahead routine to decide the membership for $q$. If $q$ should be
           assigned to $C_j$, append $q$'s unassigned neighbors to $Q$; otherwise, only assign
           $q$ to the right cluster without appending $q$'s neighbors to $Q$;
10:      else
11:        Assign $q$ to $C_j$ and append $q$'s unassigned neighbors to $Q$;
12: if all the nodes are assigned to some $C_j$ then
13:   Stop;
14: else
15:   Increment $R_i$ and goto 2;
```

4.6.2 More details on NetScan

How to choose initial cluster centers

The initialization has a direct impact on the NetScan results as in many similar algorithms. Instead of using a naive random approach, we weight each node with its degree so that nodes with higher degrees have higher probabilities to be chosen. Since NetScan relies on edges to grow clusters in step II, the weighted random approach allows clusters to grow fast. More importantly, due to the improved edge availability, true cluster contents can be absorbed during early rounds of membership assignment, reducing the possibility that they would be assigned to some other clusters inappropriately.

For some datasets in which most nodes have small degrees, the weighted random approach becomes less effective. We propose a heuristic to achieve better initialization in such cases. The heuristic requires the user to input the minimum size of the clusters, i.e., *minSize*. The introduction of this parameter is reasonable, since in many applications domain experts have the knowledge of the minimum size of the clusters, and tiny clusters are not interesting to users. Instead of choosing one object to start cluster assignment (step II), we create initial clusters consisting of at most *minSize* objects in a round robin fashion in the initialization step. At the beginning, each cluster contains only one object, i.e., the



(a) w.r.t. R_0 (b) w.r.t. R_1

Figure 4.6: Node assignment in NetScan.

initial seed. Then, each cluster is asked to absorb an unassigned object whose distance to its initial seed is the smallest. This step is skipped if no unassigned object is available. We continue this process for *minSize* rounds. After the formation of those initial clusters, the regular cluster assignment step starts. This heuristic allows clusters to have more candidate objects to choose from in the beginning of the cluster assignment step. Thus, the unassigned objects would be more likely to be absorbed by the right cluster.

How to choose R_i

In step II of NetScan, we assign cluster membership to all the nodes in multiple rounds. The radius threshold R_i is gradually incremented from round to round. R_i plays an important role in minimizing the maximum radius of the resulting clusters. Figure 4.7 gives an example where a larger threshold R_j allows node a to be assigned to cluster 1, resulting in a larger radius of cluster 1. Instead, by using a smaller threshold R_i, this case is avoided because a can only be assigned to cluster 2. From the point of view of minimizing the maximum radius, we want the increment of R_i to be as small as possible. However, a too small increment of R_i may lead to the case that no additional node can be assigned for many rounds, which may greatly and unnecessarily increase the runtime.

As a trade-off, we propose the increment to be the average pairwise distance of nodes. That is, the radius threshold R_{i+1} is chosen as $R_i + \overline{D}$ where $\overline{D}$ is the average pairwise distance of nodes. This choice of increment makes it likely that at least some further nodes can be assigned in the next round. $\overline{D}$ can be obtained efficiently by drawing a small set of samples and calculating the average pairwise distance of the samples.

Algorithm 4 suggests that the nodes located in the overlapping area of two clusters w.r.t.

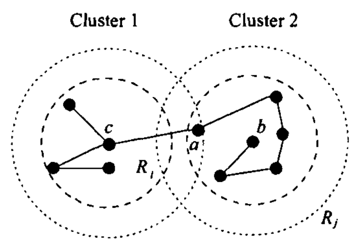


Figure 4.7: Radius increment.

a given radius threshold are more difficult to assign than the others. Thus, to start with, we choose R_0 to be half of the smallest distance among all pairs of cluster centers. This choice of R_0 does not create overlap that introduces any ambiguity in the node assignment, thus reducing the problem size.

How to assign nodes

In step II of NetScan, nodes are assigned to clusters generally based on their distances to the cluster centers. Special attention, however, needs to be paid to those nodes in the overlap area of two or more clusters w.r.t. R_i. Inspired by the concept of bridge nodes introduced in Section 4.3, we call these nodes *potential bridge nodes.* We assign potential bridge nodes not only based on their distances to the different cluster centers, but also on their neighborhood situations. For example, in Figure 4.6 (b), a is a potential bridge node and its assignment has an impact on the assignment of its neighbors b and c. If node a is assigned to cluster 1, both b and c have to be assigned to cluster 1, resulting in a larger radius compared to assigning all three nodes to cluster 2.

Whether a node is a potential bridge node depends on three factors: (1) the node has neighbors who have been assigned membership and those neighbors are from more than one cluster, e.g., C_i and C_j. (2) the node is within R_i distance from both centers of C_i and C_j. (3) the node has unassigned neighbors.

We propose the following look-ahead approach for the cluster assignment of potential bridge nodes. For the sake of efficiency, for each potential bridge node, we only check its unassigned neighbors (if any) which have a degree of 1, the so-called *unary neighbors.* These unary neighbors are especially critical since they can be connected to any cluster only via the node under consideration. The membership assignment decision is made only based on the unary neighbors. A potential bridge node is assigned to its closest center unless the node has a direct unary neighbor which is closer to some other center. In the case that more than one unary neighbors exist, the cluster center leading to the smallest radius increase is chosen. Our algorithm could benefit from looking into indirect neighbors of potential bridge nodes as well, however, this would significantly increase the runtime without guarantee of quality improvement.

Postprocessing to eliminate outliers

As in the traditional k-Center problem, the *CkC* problem faces the same challenge of "outliers", which may cause significant increase in radius of the resulting clusters. In many applications such as market segmentation, it is acceptable and desirable to give up a few customers to meet most customers' preference. We propose an optional step, which utilizes a graphical approach to eliminate outliers from the NetScan results. Each node remembers the radius threshold at which it was assigned, and all the nodes are sorted by these thresholds. We filter out the node (and its following nodes) which causes a sudden increase of the radius. The "cut-off" point can be determined by automatic detection as well as manual inspection from a chart displaying the sorted nodes, as illustrated in Figure 4.8 (b). Part (a) shows the corresponding input graph. Only f would be removed as an outlier in the example.

Runtime complexity: In each iteration of step II and III, the NetScan algorithm generates k clusters one by one. During membership assignment of each cluster, the nodes sharing edges with the assigned nodes of that cluster are considered. The distances between these nodes and the cluster center are calculated. Thus, the overall runtime complexity is bounded by the total number of nodes being visited. For the purpose of minimizing the maximum radius, NetScan gradually increases the radius threshold R_i. Let $\overline{D}$ represent the amount of radius increment, the total number of radius increases in one iteration is a constant, $\frac{diam}{\overline{D}}$, where *diam* is the longest distance among all pairs of nodes. In the worst case, every edge is visited k times for each R_i, hence the total number of node visits in an iteration is $O(k|E|\frac{diam}{\overline{D}})$, where $|E|$ is the total number of edges. We assume the NetScan algorithm converges after t iterations. Hence, the worst case runtime complexity of NetScan is $O(tk|E|\frac{diam}{\overline{D}})$.

However, in each iteration, we only need to consider those edges connecting to the nodes in the *frontier*, i.e., a set of unassigned nodes that are direct neighbours of the assigned

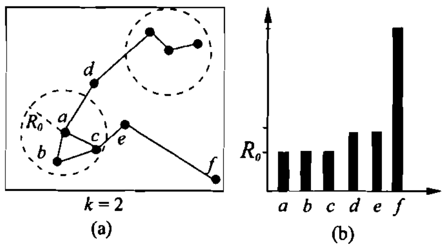


Figure 4.8: Outlier elimination.

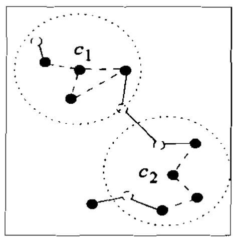


Figure 4.9: Runtime.

nodes. The worst case rarely happens, in which all the edges are connected to the frontier nodes. For example, the dashed edges in Figure 4.9 do not have to be considered in the next radius increment round. In the figure, the nodes in the frontier are dashed. In practice, the number of edges visited in one iteration can be reasonably assumed to be $O(|E|)$ on average, and the expected runtime of NetScan would be $O(t|E|)$ under this assumption.

4.6.3 Adaptation of NetScan to the Connected k-Means problem

As we have discussed in related work (Section 4.2), various clustering problems can be formulated depending on different objectives. The well-known k-Means problem [85] minimizes the compactness, i.e., the sum of squared distances from data objects to their corresponding cluster centers. The corresponding k-Means algorithm [86] is widely used as a practical and robust clustering method. Also motivated by joint cluster analysis as in *CkC*, we can define the Connected k-Means problem, which finds a k-partitioning of nodes minimizing the compactness under the internal connectedness constraint.

As a straightforward extension, NetScan can be adapted to the Connected k-Means problem. We can simply use the means of clusters to replace the actual center nodes. Then in step II of NetScan for node membership assignment, the radius threshold is incremented from round to round with respect to the means instead of the center nodes. Similarly in step III, the new cluster means are relocated instead of the center nodes. The algorithm terminates when there is no change in node membership or a certain number of iterations have been performed. The adapted NetScan algorithm was used in part of our experiments to compare to the traditional k-Means algorithm and validate the concept of joint cluster analysis.

4.7 Experimental Results

In this section, we demonstrate the meaningfulness of our joint cluster analysis models on three real datasets and show the efficiency of the NetScan algorithm using synthetic datasets.

4.7.1 Experimental design

We evaluated the *CkC* model on two applications, community identification and gene clustering, in which attribute data and relationship data carry complementary information and clusters are often required to be internally connected. In community identification, communities are naturally defined as connected sub-networks. In gene clustering, an independent study [109] shows that connected sub-networks with highly similar expression profiles often correspond to important gene groups.

Datasets. We derived two datasets from the DBLP data [36] for community identification. The DBLP I dataset includes 50 researchers from three computer science communities. The researchers are represented by keywords of their research interests and collaboration relationships to other researchers. The true label (community) of each researcher was manually determined. Due to the difficulty of determining true labels for a large number of researchers, we constructed the DBLP II dataset for document clustering which was used as an analogue to community identification. In this dataset, 1436 papers were collected from 9 major conferences of three communities. The attributes of each paper were collected from its abstract representing keyword frequencies and the relationship data were extracted from the coauthorship network. The true cluster label of each paper was determined by the community corresponding to the conference in which it appears. The availability of true cluster labels allowed us to evaluate the *CkC* model on this much larger dataset. For gene clustering, we constructed a dataset where each gene is represented by its expression profile and relationships to other genes. The gene attributes were collected from the Spellman dataset [98], while the relationship data was extracted from the protein interaction network of Saccharomyces Cerevisiae, which was downloaded from the BioGRID database [99]. We preprocessed the data to get 2149 genes by eliminating the ones with missing expression data and selecting the largest connected component of the interaction network. An overview of the three real datasets can be found in Table 4.4.

Comparison partners. Related work, as discussed in Section 4.2, can be categorized as partitioning vs. overlapping, distance-based vs. probabilistic, and unsupervised vs.

Datasets	Objects	Attributes	Relationships
DBLP I	Researchers	Research Interests	Co-authorship
DBLP II	Papers	Keywords	Co-authorship
Adapted Spellman	Genes	Expression profiles	Protein interaction

Table 4.4: Overview of the real datasets

semi-supervised. Since the *CkC* model is partitioning, distance-based and unsupervised, we compare against such methods to ensure a fair comparison. We chose k-Means and GraClus[2], a state-of-the-art graph clustering algorithm, as representatives of attribute-based and relationship-based methods, respectively. Note that graph clustering algorithms can exploit some of the attribute information in the form of edge weights. GraClus outperforms the best existing spectral algorithms [37] on an important graph clustering problem, the normalized cut [97], which has been shown to be effective for discovering meaningful clusters in several real life applications [37].

DBLP dataset I is small and is only used to provide anecdotical evidence that the clusters of the *CkC* model are meaningful. We report the results of the k-Center algorithm and the k-Center version of the NetScan algorithm. DBLP dataset II was constructed for document clustering. The traditional k-Means algorithm is known to work well for document clustering [100] utilizing only the attribute data. We applied our adapted NetScan (for the Connected k-Means problem, see Section 4.6.3) to the dataset and compared the results with k-Means and GraClus.

For gene clustering, various clustering methods, such as k-Means, have been applied on gene expression profiles to gain insight of how genes are grouped and to predict the function of a gene. In addition to the gene expression profiles, large-scale interaction data, such as protein-protein interactions, often carry important biological knowledge [109]. In order to fully make use of all the knowledge, joint clustering analysis of both types of data is necessary. For the Spellman dataset, we compared the adapted NetScan (for Connected k-Means) with the traditional k-Means algorithm and GraClus.

[2]Downloaded from http://www.cs.utexas.edu/users/dml/Software/graclus.html

4.7.2 DBLP dataset I: clustering researchers

The first real dataset includes 50 researchers from three major computer science communities: theory, databases and machine learning. The attributes of each researcher were collected from his/her homepage representing the keyword frequencies of his/her research interests. The relationship data used a connected subgraph extracted from the DBLP [36] coauthorship network, an edge was created for pairs of researchers who have coauthored at least one paper. We applied the NetScan algorithm to identify communities from this dataset in an unsupervised manner. The relatively small size of the dataset allowed us to manually determine a researcher's true community (cluster label) from his/her lab affiliation and professional activities. These true labels were then compared to the labels determined by our algorithm.

We used the Cosine Distance as the distance measure for the attributes, a standard measure for text data. We ran NetScan for the Connected k-Center problem and a known heuristic algorithm (Greedy k-Center) [67] for the traditional k-Center problem, both with $k = 3$. Due to the small size of this dataset, we set *minSize* to 0 for NetScan. Table 4.5 reports the best clustering results over 20 runs for both algorithms, recording the number of correctly identified researchers for each community together with the overall accuracy. To calculate the accuracy, we associated each of the three communities with one of the clusters such that the best overall accuracy was achieved. Compared to Greedy k-Center, NetScan improved the accuracy from 54% to 72%. Note that we perform unsupervised learning, which accounts for the relatively low accuracy of both algorithms compared to supervised classification algorithms.

Communities	Size	Greedy k-Center	NetScan
Theory	20	11	14
Databases	20	12	15
Machine Learning	10	4	7
Sum	50	27	36
Accuracy		54%	72%

Table 4.5: Comparison of NetScan and Greedy k-Center on dataset DBLP I .

The main reason why NetScan outperforms Greedy k-Center is that both relationship and attribute data make contributions in the clustering process, and considering only one data type may mislead the clustering algorithm. Figure 4.10 illustrates the differences

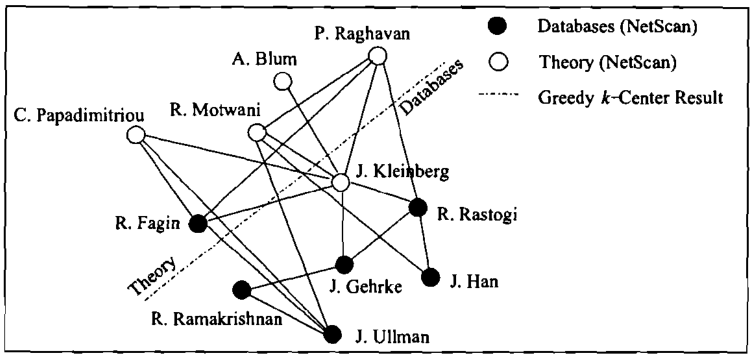


Figure 4.10: Partial clustering results on dataset DBLP I.

between NetScan and Greedy k-Center on real dataset I. For example, J. Kleinberg lists interests in Clustering, Indexing and Data Mining, also Discrete Optimization and Network Algorithms. From this attribute information, it seems reasonable to identify him as a researcher in databases. Nevertheless, after taking his coauthorship information into consideration, NetScan clustered him into the theory community, which is a better match for his overall research profile. On the other hand, J. Ullman has broad coauthorship connections, which alone cannot be used to confidently identify his community membership. However, he claims research interests mainly in databases, and NetScan clustered him into the database community as expected.

4.7.3 DBLP dataset II: clustering papers

The second real dataset was constructed for document clustering. As summarized in Table 4.6, the dataset includes 1436 selected papers from DBLP [36] published from 2000 to 2004 in nine major conferences of three communities: theory, database data mining, and machine learning. The attributes of each paper are vectors representing the keyword frequencies in the abstract obtained from CiteSeer [30]. After deleting stop words and applying stemming and word occurrence thresholding, we obtain a dataset whose attribute vector has 603 dimensions. The relationship data are based on the DBLP coauthor-ship network [36]. If two papers share a common author, an edge is added between them. Note that the papers were chosen so that the relationship graph is connected. We also removed papers that make the true clusters unconnected, since otherwise, due to the connectedness constraint, NetScan will have no chance to achieve 100% accuracy. The true cluster label of a paper is defined

Communities	Conferences	# of Papers
Theory	FOCS, STOC, SODA	519
Databases and Data Mining	SIGMOD, VLDB, KDD	434
Machine Learning	ICML, NIPS, COLT	483

Table 4.6: Summaries of dataset DBLP II.

by the community corresponding to the conference in which it appears.

We ran NetScan, k-Means and GraClus on the DBLP II dataset with $k = 3$. To run NetScan, we set *minSize* to be 20 assuming that a community with less than 20 people would not be interesting. To run GraClus, we set the edge weights to be the inverse of the (attribute) distance between the two corresponding nodes, as suggested by Shi *et al.* in [97]. In order to measure the accuracy, a cluster is labeled by a majority vote of its members. Table 4.7 lists the clustering results of the three comparison partners, recording the number of correctly identified papers for each community as well as the overall accuracy. For both k-Means and NetScan, we chose the best results over 20 runs and the accuracy is defined as the number of correctly clustered papers divided by the total number of papers. Our results show that NetScan clearly outperforms k-Means and GraClus, improving the accuracy from 60%, and 73% to 85% respectively.

Communities	Size	k-Means	GraClus	NetScan
Theory	519	359	387	504
Databases	434	351	405	423
Machine Learning	483	156	260	300
Sum	1436	866	1052	1227
Accuracy		60%	73%	85%

Table 4.7: Comparison of NetScan, GraClus and k-Means on dataset DBLP II.

To illustrate the benefits of joint cluster analysis, we present in Figure 4.11 a small portion of the DBLP II dataset and the clusterings produced by k-Means and by NetScan. Table 4.8 lists the details of the chosen papers, which allows readers to verify and compare the clustering results shown in Figure 4.11. As a particular example, the STOC'02 paper (id:602) is about "Query strategies for priced information". Many of the keywords appearing in its abstract are commonly used in database papers, e.g., "Query", "Search", "Framework", and "Algorithm". From this attribute information alone, the paper would

Paper ID	Title	Authors	Conference
50	SECRET: a scalable linear regression tree algorithm	A. Dobra J. Gehrke	KDD'02
51	Detecting change in data streams	D. Kifer S. Ben-David J. Gehrke	VLDB'04
13	Query optimization in compressed database systems	Z. Chen J. Gehrke F. Korn	SIGMOD'01
306	RE-Tree: an efficient index structure for regular expressions through a social network	C. Chan M. Garofalakis R. Rastogi	KDD'03
54	Processing complex aggregate queries over data streams	A. Dobra M. Garofalakis J. Gehrke R. Rastogi	SIGMOD'02
804	Gossip-based computation of aggregate information	D. Kempe A. Dobra J. Gehrke	FOCS'03
755	Protocols and impossibility results for gossip-based communication mechanisms	D. Kempe J. Kleinberg	FOCS'02
763	Building low-diameter P2P networks	G. Pandurangan P. Raghavan E. Upfal	FOCS'01
885	Can entropy characterize performance of online algorithms?	G. Pandurangan E. Upfal	SODA'01
754	Algorithms for facility location problems with outliers	M. Charikar S. Khuller D. Mount G. Narasimhan	SODA'01
761	Stability of load balancing algorithms in dynamic adversarial systems	E. Anshelevich D. Kempe	STOC'02
602	Query strategies for priced information	M. Charikar R. Fagin V. Guruswami J. Kleinberg P. Raghavan A. Sahai	STOC'02

Table 4.8: Papers used in Figure 4.11.

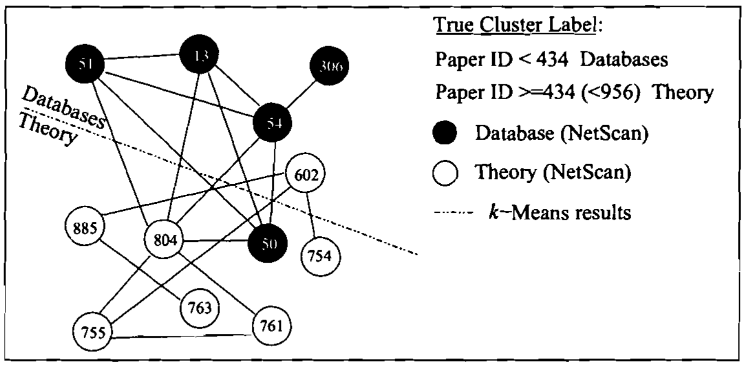


Figure 4.11: Partial clustering results on dataset DBLP II.

be clustered as a database paper. However, considering the neighboring papers that share at least one author with it, NetScan correctly identified the paper as a theory paper. As another example, the FOCS'03 paper (id:804) has extensive connections with database papers. However, taking its attributes into consideration, NetScan correctly identified it as a theory paper.

The GraClus algorithm was shown to be so far the best spectral method for solving the normalized cut problem [97]. In our above experiment, NetScan clearly outperformed GraClus on the DBLP II dataset in terms of clustering accuracy. Moreover, since k-Means and GraClus do not enforce the internal connectedness constraint, each generated cluster may contain more than one connected component. In fact, the clusters generated by k-Means and GraClus form 241 and 13 connected components respectively instead of the three connected components corresponding to the three communities. These results confirmed that the *CkC* model is effective in discovering clusters which are cohesive on both types of data in the DBLP dataset. Given that coauthorship networks are known to be typical social networks [14], the *CkC* model is able to handle many real life applications such as community identification and market segmentation, in which internal connectivity is requested.

4.7.4 Spellman dataset: clustering genes

The third experiment was conducted on the Spellman gene dataset. For this dataset, the attribute data is the expression profile of 6026 yeast genes [98] which were measured at 73 time points corresponding to different stages of cell cycle. The relationship data was extracted from the protein interaction network of Saccharomyces Cerevisiae, which was downloaded from the BioGRID database [99]. Edges in the informative graph were created for every pair of genes for which BioGRID records at least one experiment reporting some type of interaction between the corresponding proteins. Since our model requires complete attribute data and a connected relationship network, we preprocessed the data to obtain an adapted dataset with 2149 genes by eliminating the ones with missing expression data and selecting the largest component. For this dataset, there is no ground truth available to measure clustering accuracy. Alternatively, we computed the biological significance of the generated clusters using FuncAssociate [21]. FuncAssociate measures the over-representation of GO[3] (Gene Ontology) terms within a cluster using negative log P-values, a standard cluster validation method in the bioinformatics literature [109]. The P-value measures the probability that a certain over-representation of GO terms appears by chance, i.e., the smaller the P-value (the larger the negative log P-value), the more significant the cluster.

Algorithm	Top 5 Enriched GO Terms	$-\log_{10}$(P-value)
k-Means	Nucleolus	33
	Ribosome biogenesis and assembly	33
	Ribosome biogenesis	31
	Transcription from Pol I promoter	26
	rRNA processing	26
GraClus	Endoplasmic reticulum	40
	Mitochondrial ribosome	35
	Organellar ribosome	35
	mRNA splicing	33
	Transcription regulator	32
NetScan	Ribosome biogenesis and assembly	47
	Ribosome biogenesis	38
	Transcription from Pol I promoter	35
	Nucleolus	35
	rRNA processing	33

Table 4.9: Comparison of NetScan, GraClus and k-Means on the Adapted Spellman dataset.

We ran k-Means, GraClus and NetScan with $k = 15$ on the adapted Spellman dataset. We adopted Euclidean distance as the similarity metric since mean vector calculation is not meaningful for other popular similarity metrics such as Pearson Coefficient. To run GraClus, we set the edge weights to be the inverse of the Euclidean distance between the two corresponding genes. As the GraClus implementation used requires integer edge

[3]http://www.geneontology.org/

weights, we further multiplied the edge weights by an appropriate constant (i.e., 1000). Due to the unavailability of the knowledge on the minimum size of clusters, we set *minSize* to 0 for NetScan. For both k-Means and NetScan, we chose the best results over 20 runs. The enriched GO terms together with the corresponding negative log P-value are listed in Table 4.9.

From this set of experiments, we observed that the clusters generated by NetScan have significance at least comparable to the ones generated by GraClus. Meanwhile, both NetScan and GraClus clearly outperformed the k-Means algorithm. These results confirmed that relationship data provides additional knowledge and joint analysis of both types of data is helpful to identify significant gene clusters. Moreover, NetScan identifies connected components in the interaction network, while the clusters found by GraClus may be unconnected. As demonstrated by Ulitsky *et al.*, identifying connected subnetworks in the interaction data helps to ensure that clusters are biologically relevant [109].

4.7.5 Synthetic datasets.

We used synthetic data for the efficiency evaluation. According to the complexity analysis in Section 4.6, the runtime is directly related to the number of edges instead of the number of nodes. We thus fix the number of nodes and vary the degree of each node to evaluate the efficiency of NetScan. We took the UCI PIMA dataset [1], which was also used in [32] to evaluate the quality of the k-Means algorithm. Since the PIMA dataset contains only numerical attributes, we automatically generated the relationship data based on two random graph models, the Erdös-Rényi model [43] and the Power-Law model [15]. In the Erdös-Rényi model, every pair of nodes is connected with the same probability. In the Power-Law model, which is often used to model internet structure, there are many nodes with few edges and only a few nodes with a large number of neighbors. All the experiments were conducted on an Intel Celeron 1.6G processor with 512M RAM running the Window XP operating system.

We studied the effect of the average number of edges on the runtime. Figure 4.12 (a) shows the results over 50 restarts for both models. The average degree was set starting from 4 since with a smaller degree the data generator often failed to generate a connected network. We also evaluated the effect of k on the runtime, as reported in Figure 4.12(b). Our results show that NetScan scales well with both the average degree and k, confirming our average runtime complexity analysis.

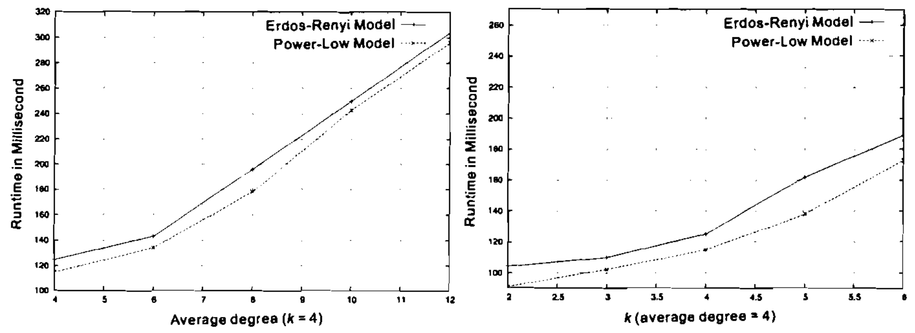


Figure 4.12: NetScan on synthetic data. (a) runtime vs. average degree. (b) runtime vs. k.

4.8 Summary

Existing cluster analysis methods are based on either attribute data or relationship data. However, in scenarios where these two data types contain complementary information, a joint cluster analysis of both promises to achieve better results. In this chapter, we have introduced the novel Connected k-Center (CkC) problem, a clustering model that integrates an internal connectedness constraint defined on relationship data and an objective function specified on attribute data. In this model, the constraint provides a way to capture application needs in several applications, e.g. market segmentation, and community identification. We have also relaxed the CkC problem to allow multiple membership which results in the CkC' problem. The CkC' problem captures the requirements in some applications where entities can belong to multiple clusters.

We have proved the NP-hardness of both the CkC problem and the CkC' problem and studied their connections which lead to a constant factor approximation algorithm for CkC. For the special case of the CkC problem where the underlying graph is a tree, we have proposed a dynamic programming method giving an optimal solution in polynomial time. To improve the scalability for large real datasets, we have developed the efficient heuristic algorithm NetScan. Our experimental evaluation using real datasets for community identification and gene clustering demonstrated the meaningfulness of the NetScan results. In addition, experiments on synthetic datasets demonstrated the efficiency and scalability of the NetScan algorithm.

The framework of joint cluster analysis of attribute and relationship data suggests several interesting directions for future research. Firstly, to better model practical applications, the connectivity constraints can be generalized to suit varied requirements. For example, the internal connectedness constraint can be replaced by specifications on any combination of properties of graphs such as length of paths, degree of vertices, connectivity, etc. Secondly, the relationships can be non-binary, and the edges can be weighted to indicate the degree of relationship. For example, friendship can go from intimate and close to just nodding acquaintanceship. The relationships can also be non-symmetric such as in citation or superior-subordinate relations. Clustering models for these types of relationship data and corresponding algorithms are worth to be explored. Finally, we believe that with the increasing availability of attribute data and relationship data, data mining in general, not only cluster analysis, will benefit from the joint consideration of both data types.

Chapter 5

Constraint-Driven Clustering

There have been many clustering models integrating constraints to enable us to find useful clusters. However, most of them require users to provide the number of clusters, K, which is often unknown in advance. Furthermore, even if the most appropriate number of clusters determined from the data distribution is known, it may not suit the application needs [12]. Furthermore, an inappropriate number of clusters may result in generating distorted and less actionable clusters. We argue that a more natural way to generate actionable clusters is to let the constraints decide the number of clusters. In this chapter, we propose the so called *Constraint-Driven Clustering*, which aims at utilizing user-provided cluster-level constraints to discover an arbitrary number of compact and balanced clusters. The compactness of a clustering is measured by the sum of the squared distances of all data objects to their corresponding cluster representatives. Two general types of cluster-level constraints are considered, i.e., minimum significance constraints and minimum variance constraints, as well as combinations of these two types. A preliminary version of Chapter 5 was published in [53].

5.1 Overview

Various balancing constraints have been designed to restrict the generated clusters in order to make them actionable. In particular, we are interested in two types of cluster-level constraints, i.e., minimum significance constraints and minimum variance constraints. The minimum significance constraint specifies the minimum number of objects in a cluster. The minimum variance constraint poses a lower bound on the variance of a cluster. By imposing a minimum significance constraint or/and a minimum variance constraint, our model searches for clusters which are balanced in terms of cardinality or/and variance.

To motivate Constraint-Driven Clustering with minimum significance and variance constraints, we use applications in energy aware sensor networks and privacy preservation as our running examples.

Energy Aware Sensor Networks [55, 61, 12]: Grouping sensors into clusters is an important problem in sensor networks since it can drastically affect the network's communication energy consumption [55]. Normally, a master node is chosen from sensors in each cluster or deployed to the central area of each cluster. Other sensors will communicate with the outside world through the closest master node. In this context, it is desirable to require each cluster to contain at least a certain number of sensors in order to balance the work load of master nodes. To prolong the lifetime of a sensor network, evenly distributing energy consumption among clusters is desired. Since the energy consumption of message transmissions increases quadratically with the distance between communicating sensors [26], the variance of a group of sensors corresponds to the amount of energy consumed by those sensors on average. The minimum variance constraint allows to group sensors into clusters which are balanced in terms of energy consumption. Moreover, in this application, it is natural to have the constraints decide the appropriate number of clusters instead of specifying a number in advance.

Privacy Preservation [94, 104]: In a privacy preservation application, we may want to release personal records to the public without a privacy breach. To achieve this, we can group records into small clusters and release the summary of each cluster to the public. In this context, the usability of a clustering is evaluated by how much privacy is preserved in the clustering. To preserve individual privacy, the k-anonymity model [104] requires that each cluster has to contain at least a certain number of individuals. However, these individuals could have very similar, even identical attribute values, allowing an adversary to accurately estimate their sensitive attribute values with high confidence from the summary. We argue, therefore, that the clusters to be published should also have a minimum variance which translates into the width of the confidence interval of the adversary estimate. In the context of privacy preservation, again, it is typically unreasonable to specify the number of groups in advance.

The PPMicroCluster model [71] requires both minimum significance and minimum radius constraints to preserve privacy. Compared to this model, our Constraint-Driven Clustering model adopts a more practical constraint, i.e., minimum variance constraint, which describes the statistical properties of a cluster better and can be used for a wide range of applications. Besides, we systematically study the complexity of the Constraint-Driven Clustering problem and propose a dynamic algorithm which is shown to be more efficient than the static algorithm for solving the PPMicroCluster problem.

In this chapter, we introduce the Constraint-Driven clustering problem. We prove the NP-hardness of the proposed clustering problem with different constraints. Inspired by the NP-hardness proof, we propose a novel data structure, named *CD-Tree*, which organizes data objects in leaf nodes such that each leaf node approximately satisfies the significance and variance constraint and minimizes the sum of squared distances. Based on *CD*-Trees, we develop an efficient algorithm to generate constrained clusters with good quality. Furthermore, benefiting from the hierarchical tree structure, the *CD*-Tree algorithm can easily adapt to dynamic updates of the data.

The rest of the chapter is organized as follows. Section 5.2 surveys related work. Section 5.3 introduces the new cluster model and analyzes its complexity. Section 5.4 presents the *CD*-Tree algorithm. We report experimental results in Section 5.5. Section 5.6 summarizes the constraint-driven clustering problem and discusses future directions.

5.2 Background

The Constraint-Driven Clustering problem is related to many topics. In this section, we review related research on actionable clustering, cluster-level constraints, instance-level constraints, cluster methods in sensor networks, and k-anonymity for privacy preservation.

Cluster-level Constraints. Our proposed clustering model belongs to the category of clustering with cluster-level constraints. Different from all existing models, our model does not require the number of clusters as an input.

Clustering methods in Sensor networks. Ghiasi *et al.* proposed to cluster sensor nodes such that the number of sensors in each cluster (which has a master node) is in the range of $[\frac{n}{k} - \delta, \frac{n}{k} + \delta]$ and the total distance between sensor nodes and K master nodes is minimized [55]. The clustering problem presented in [55] is different from the Constraint-Driven Clustering in that it specifies the number of clusters. More practical protocols are studied in the sensor network literature to minimize the energy consumption or message transmissions by grouping sensors to clusters. Bandyopadhyay *et al.* [10] proposed a randomized algorithm to find the optimal number of cluster heads by minimizing the total energy spent on communicating between sensors and the information-processing center through the cluster heads. Authors in [80] present a clustering method for self-organizing sensor networks, for the purpose of grouping sensors into the optimal number of clusters that minimize the number of message transmissions. Similar approaches on clustering in sensor networks also include [7, 13, 76] etc. However, these clustering methods focus on dealing with engineering constraints instead of systematically studying the properties of the proposed clustering models.

k-Anonymity. The k-Anonymity model [94, 104] was proposed for the purpose of protecting data privacy. The k-anonymity framework archives the goal by generalizing entries in a table with minimum cost such that every record becomes textually indistinguishable from $k-1$ other records in the table. [90] and [5] prove that k-Anonymity with Suppression is NP-hard and study approximation algorithms. The k-Anonymity model is defined on categorical data, and thus has different properties from our model which assumes a geometric space with the Euclidean distance. [4] introduces a k-nearest neighbor based algorithm to solve the k-anonymity problem for numerical data. Domingo-Ferrer *et al.* [39] studied the optimal k-partition problem which can be considered as the k-Anonymity model in the Euclidean space. And it is a special case of our model where only significance constraints are allowed. But in [39] the complexity of the proposed problem is not analyzed. [54] considers privacy preservation as a problem of finding the minimum number of hyperspheres with a fixed radius to cover a dataset satisfying that each hypersphere covers at least a certain number of data objects. A similar model, named PPMicroCluster, is studied in [71] which requires both significance and radius constraints. Compared to the PPMicroCluster model, our model adopts a more practical constraint, i.e., minimum variance constraint which describes the statistical properties of a cluster better and can be used for a wide range of applications. Besides, [71] does not analyze the complexity and proposes a static algorithm.

5.3 Problem Definition and Complexity Analysis

In this section, we introduce the Constraint-Driven Clustering problem and analyze its complexity under different types of constraints.

5.3.1 The *CDC* Problem

First we define the general Constraint-Driven Clustering problem, also referred to as *CDC* problem, as follows:

Definition Constraint-Driven Clustering(*CDC*)
Given a set of data objects P in d-dimensional space, a set of constraints $\mathcal{C}$, the task is to partition P into disjoint clusters $\{P_1, \cdots, P_m\}$, $\forall i, j, i \neq j, P_i \cap P_j = \emptyset$, $P = P_1 \cup \cdots \cup P_m$, such that: (1) each cluster $P_i, 1 \leq i \leq m$ satisfies all constraints in $\mathcal{C}$ and (2) the sum of squared distances of data objects to their corresponding cluster representatives is minimized.

Note that a cluster representative can be either a real data object or the mean vector of a cluster. In this thesis, we study the *CDC* problem under the following two types of constraints.

Definition For each cluster $P_i, 1 \leq i \leq m$,

- **Minimum Significance Constraint** $Sig \geq 1$: $|P_i| \geq Sig$.
- **Minimum Variance Constraint** $Var \geq 0$: $\frac{\sum_{p \in P_i} dist(p,\mu)^2}{|P_i|} \geq Var$, where μ is the representative of P_i.

Remark. The *CDC* model is general in that the constraint set $\mathcal{C}$ can include one or more constraint types. Note that when $Sig = 1$, the significance constraint is trivially satisfied, similarly the variance constraint when $Var = 0$. For the *CDC* problem to be meaningful, at least one of the constraints has to be non-trivial.

In the current definition of *CDC*, we require the generated clusters to be non-overlapping. Yet, the *CDC* problem can also be extended to allow overlapping. For privacy preservation, for example, possible overlaps among the generated clusters cannot only make the model more flexible, but also enhance privacy protection. When overlapping is allowed, a data object assigned to multiple clusters would contribute to the objective function (sum of squared distances) multiple times. The complexity analysis in the following section remains valid when overlapping is permitted since the optimal solution for the instance in the proof can never contain overlapping clusters.

5.3.2 Complexity Analysis

In this section we show that the *CDC* problem is NP-hard under the minimum significance or the minimum variance constraints. We shall focus on the significance constraint, and show how the same proof can be easily extended for the minimum variance constraint.

In order to study the complexity of the *CDC* problem, we consider the decision version of the *CDC* problem with a significance constraint only (*sig-CDC*) as follows.

Definition (*sig-CDC*). Given a set of data objects P in d dimensional space, a constant $Sig > 1$ and a cost threshold W. Decide whether P can be partitioned into disjoint clusters $\{P_1, \cdots, P_m\}$, which satisfies the following conditions:

1. $\forall P_i$, $|P_i| \geq Sig$ (the minimum significance constraint).
2. $\sum_{j=1}^{m} \sum_{p \in P_j} (dist(p, p_j))^2 \leq W$, where p_j is the medoid of cluster P_j and $dist(p, p_j)$ is the Euclidean distance between p and p_j.

In the following we prove that the *sig-CDC* problem is NP-complete by a reduction from a known NP-complete problem, PLANAR X3C.

Definition (PLANAR X3C [42])
Given a set Q with $|Q| = 3q$ and a set T of triples from $Q \times Q \times Q$ such that (1) every element of Q occurs in at most three triples and (2) the induced graph is planar. (The induced graph G contains a vertex for every element of Q and for every triple in T. There is an edge connecting a triple to an element if and only if the element is a member of the triple. Clearly, G is bipartite with vertex bipartition Q and T.) Decide whether there exists a subset of q triples in T which contains all the elements of Q.

Theorem 5.3.1 *sig-CDC is NP-complete for $Sig \geq 3$.*

Proof. First, the problem is in NP since it takes polynomial time to verify whether a given clustering solution is feasible. To prove the NP-hardness, we perform a reduction from PLANAR X3C. Given an instance $I = (Q, T)$ of PLANAR X3C, we create an instance $I' = (P, Sig, W)$ of the *sig-CDC* problem by the following procedure.

1. Construct a planar bipartite graph $G(V, E)$ of the instance I where $V = Q \cup T$ and $E = \{(q, t) | q \in Q, t \in T, q \text{ is a member of t}\}$.

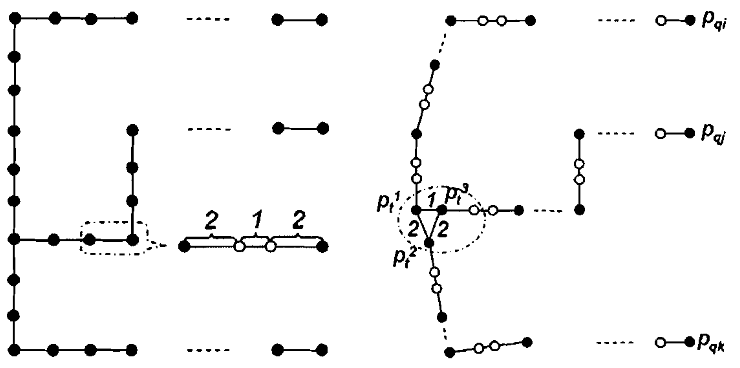


Figure 5.1: (a) Rectilinear layout L, (b) Final layout L'

2. Compute a rectilinear layout of G where each vertex v of G is mapped to a data object p_v on the integer lattice. We further enlarge the layout by a factor of 1000 to ensure that every two distinct horizontal (vertical) line segments are far away enough from each other. Each edge $e = (q, t)$ of G is broken into a sequence of line segments of length 5 by placing data objects in the rectilinear layout (Figure 5.1(a)). The resulting layout is denoted as L. [118] proposed a linear time algorithm to compute such a layout. A similar rectilinear layout is used in [34] to prove the NP-completeness of the feasibility problem for the Must-Link and ϵ constraints.

3. Replace the corresponding point p_t of a triple $t \in T$ by a point set $U_t = \{p_t^1, p_t^2, p_t^3\}$. p_t^1, p_t^2, p_t^3 form an isosceles triangle with two sides of length 2 and one side of length 1. U_t is called *triple set* in the following. Then, we connect p_t^1, p_t^2, and p_t^3 each with a different path of p_t leading to the corresponding element points in the original layout. To adapt to this change, the layout L needs to be adjusted by allowing edge segments to be inclined. See Figure 5.1(b) for such a transformation. In the following, we refer to points whose corresponding vertices are in Q as *element points*, and refer to points in the triple sets as *triple points.*

4. Break each line segment of length 5 into three segments with length 2, 1, 2 respectively, by adding two auxiliary points to the line segment (see Figure 5.1(a)).

5. Let L' denote the final layout. Let P be the set consisting of all the points in L' and set $Sig = 3$ and $W = 5|P|/3$.

Let m be the number of edge segments in L. Note that $|P|$ is a multiple of 3 since $|P| = 3m + |Q|$ and $|Q|$ is a multiple of 3. For any $t \in T$, let $Q_t \subset Q$ be the set of the three elements covered by t. We define a *path* as a collection of line segments from a triple point $p_t^i \in U_t$ to its corresponding element point p_q where $q \in Q_t$ without going through any other triple points. Furthermore, note that the distance between two neighboring points on a path is 1 or 2, and the distance between two non-neighboring points is greater than 2.

Assume that there is a set of triples $Y \subset T$ which covers all elements of Q exactly once. We construct a feasible clustering of instance I' with a cost of $5|P|/3$ as follows. For every triple set U_t, we first group all the three triple points of U_t into one cluster if $t \in Y$. We then start from the first unassigned point on each path originating from the points in U_t and group every three consecutive points together. Figure 5.2 illustrates how points are grouped along a path. Note that for any element $q \in Q_t$ that is covered by $t \in Y$, the corresponding point $p_q \in P$ is grouped with its two closest auxiliary points on the path from p_t^i to p_q. Since q is only covered by one triple in Y, p_q is uniquely assigned to one cluster. Furthermore, every cluster has a cost of 5 (including those clusters consisting of all the points in a triple set). Thus, we obtain $|P|/3$ clusters each with a cost of 5 so that the total cost is $5|P|/3$.

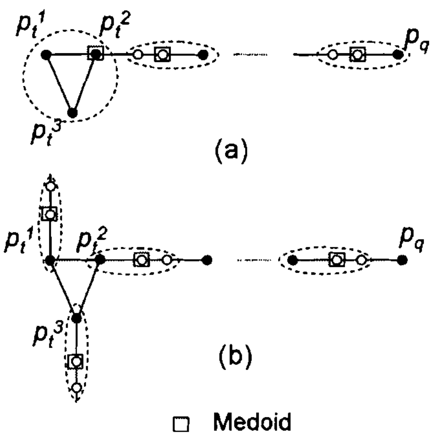


Figure 5.2: (*a*). Points in a triple set are grouped into one cluster. (*b*). Points in a triple set are grouped separately with auxiliary points.

Now assume that there is a feasible clustering with the total cost smaller than or equal to $5|P|/3$. We demonstrate how to obtain a subset of triples $Y \subset T$ that covers every element in Q exactly once. First we prove that any feasible clustering consists of $|P|/3$ disjoint clusters each consist of 3 points. For any $i \geq 3$, let H_i denote the number of clusters consisting of i points. We have $|P| = \sum_i iH_i$ and the total number of clusters is $\sum_i H_i$. Note that the minimum cost of a cluster with three points is 5 by grouping three points connected through two consecutive edges and choosing the middle point as the medoid. Furthermore, every additional node in a cluster with more than three points contributes at least 4 to the total cost (check the cluster containing four points in the dashed circle in Figure 5.1(b)). Thus,

$$\begin{aligned} 5|P|/3 &\geq \text{total clustering cost} \\ &\geq \textstyle\sum_i 4(i-3)H_i + 5\sum_i H_i = 4\sum_i iH_i - 7\sum_i H_i. \end{aligned}$$

Hence $\sum_i H_i \geq |P|/3$ since $|P| = \sum_i iH_i$. Besides, since the cost of a cluster is at least 5 and the total cost is at most $5|p|/3$, $\sum_i H_i \leq |P|/3$. Thus $\sum_i H_i = |P|/3$. Thus we can conclude that there must be $|P|/3$ disjoint clusters, each consists of 3 points and has a cost of 5.

In a feasible clustering, observe that all the points in a triple set either (a) form a single cluster, or (b) belong to three different clusters, otherwise at least one cluster would cost more than 5. We call a triple set U_t is of type (a) if (a) happens. Define $G = \{t \in T \,|\, U_t \text{ is of type } (a)\}$. It remains to show that G covers every element in Q exactly once. If U_t is of type (a), each of the three points in Q_t must be grouped into different clusters with the two nearest auxiliary points along the path from U_t (See Figure 5.2(a)). If U_t is of type (b), none of the points in Q_t is grouped with points along the paths from U_t (See Figure 5.2(b)). Since every element in Q is uniquely assigned to a cluster, T must consist of $|Q|/3$ type (a) triples (i.e., $|G| = |Q|/3$). These triples must cover every element in Q exactly once. □

Next we demonstrate that the *sig-CDC* problem remains NP-Complete if using the mean vector (instead of the medoid) of a cluster as the representative. We refer to this model as *μ-sig-CDC*. Following a similar proof technique, we prove the NP-hardness of the following problem.

Theorem 5.3.2 *The μ-sig-CDC problem is NP-complete.*

Proof. (Sketch.) The proof is by a reduction from PLANAR X3C that is similar to the one for Theorem 5.3.1. We first construct a rectilinear layout for a PLANAR X3C instance and replace each triple set by an isosceles triangle with two sides of length $\sqrt{37}$ and one side of length 2. See Figure 5.3 for the final layout. Next, we set the significance constraint

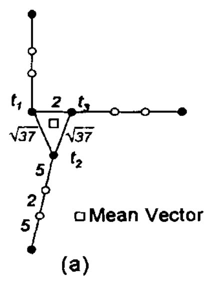


Figure 5.3: Transformed Layout for Theorem 3.2 and Theorem 3.3.

$Sig = 3$ and set the total cost threshold $W = 26|P|/3$. Similar to the proof of Theorem 5.3.1, we can show that in a feasible clustering, there must be exactly $|P|/3$ disjoint clusters, each consists of three points and has a cost of 26. Given this, we can show that there is a feasible clustering if and only if the corresponding instance of PLANAR X3C has a feasible solution. Hence, the *μ-sig-CDC* problem is NP-complete. □

Finally, we apply the previous technique to show that the *CDC* problem is NP-Complete if only a minimum variance constraint is specified and mean vectors are used as the cluster representatives. We refer to this model as *μ-var-CDC*.

Theorem 5.3.3 *The μ-var-CDC problem is NP-complete.*

Proof. (Sketch) To prove that the *μ-var-CDC* problem is NP-Complete, we use the same construction for proving Theorem 5.3.2. We set the minimum variance constraint $Var = 26/3$ and set the total cost $W = 26|P|/3$. Let P_i be an arbitrary cluster and denote by $Cost(P_i)$ the total cost of P_i. Note that due to the minimum variance constraint, $Cost(P_i) \geq 26|P_i|/3$, and by a case analysis we can show that the equality holds only when $|P_i| = 3$. Hence, in order to have a total cost of $26|P|/3$, we must have $|P|/3$ disjoint clusters each of which consists of 3 points. The rest of the proof is similar to that of Theorem 5.3.2. □

5.4 Algorithm

In the last section we have shown that the *CDC* problem with either a minimum significance or a minimum variance constraint is NP-hard. In order to efficiently solve the *CDC* problem, we design a heuristic algorithm which builds a compact tree structure to generate clusters satisfying user specified constraints. The algorithm is general in that it can handle both constraints separately or together.

We observe that a solution to the CDC problem has the following characteristics. (1) To minimize the objective function (the sum of squared distances), the generated clusters in an optimal solution should be balanced in terms of given constraints. For example, when only a minimum significance constraint is provided, the generated clusters in an optimal solution should contain similar number of data objects. (2) The membership assignment of any data object can be decided by considering its close neighbors. Thus, easily retrieving the local neighborhood of data objects is critical to the design of a universal algorithm that can handle different constraints. Guided by these observations, we propose an algorithm based on a novel data structure, called the *CD-Tree*, which is similar to the other index structures, e.g., B-Tree, R-Tree and CF-Tree [120]. Yet, the CD-Tree is specially designed for the CDC problem in that building the tree takes into account the minimization of the objective function of CDC.

5.4.1 The CD-Tree

The CD-Tree has two input parameters, i.e., a significance parameter S, a variance parameter V. Normally we set $S = Sig$ and $V = Var$, i.e., the parameters of the CDC problem. If one of the given constraints is trivial, methods for automatically determining appropriate parameter values are needed which are discussed in Section 5.4.4.

In a CD-Tree, the maximum capacity of leaf nodes is set to $2S - 1$ and the variance of data objects in leaf nodes is upper bounded by $2V$. Note that the CDC problem specifies minimum constraints on the significance and variance of a cluster, while in the CD-Tree we specify upper bounds for the significance and variance of leaf nodes in order to keep them compact. Keeping leaf nodes compact matches our goal of minimizing the sum of squared distances of generated clusters since data objects in leaf nodes will be used to generate constrained clusters to solve the CDC problem.

It is appropriate to upper-bound the variance, because a too small variance may yield too many leaf nodes, while a too large variance will make the statistical information of this leaf node less meaningful when it is used to direct a new data object to its closest leaf node. We set $2V$ as the upper bound of the variance since it makes leaf nodes to be reasonably compact and to likely satisfy the minimum variance constraint Var. The maximum capacity of leaf nodes is set to $2S - 1$ since, as we show in the following lemma, there is always an optimal solution where the number of data objects in every cluster is smaller than $2Sig$.

Lemma 5.4.1 *In the μ-sig-CDC problem, there exists an optimal clustering s.t. the number of data objects in any cluster is less than $2Sig$ and greater than or equal to Sig.*[1]

Proof. By contradiction. Assume that in every optimal clustering, there is a cluster P_i containing l data objects where $l \geq 2Sig$.

We arbitrarily split P_i into two clusters P_{i_1} and P_{i_2} where $|P_{i_1}| = Sig$ and $|P_{i_2}| = l - Sig$. Let $\overrightarrow{M}$ be the mean vector of the data objects in P_i. Similarly let $\overrightarrow{M_1}$ and $\overrightarrow{M_2}$ be the mean vectors of the data objects in P_{i_1} and P_{i_2} respectively. Note that

$$\begin{aligned}\overrightarrow{M} &= \frac{1}{l}\left(\sum_{p \in P_{i_1}} \overrightarrow{p} + \sum_{q \in P_{i_2}} \overrightarrow{q}\right) \\ &= \frac{1}{l}[Sig\overrightarrow{M_1} + (l - Sig)\overrightarrow{M_2}] \qquad (5.1)\end{aligned}$$

Let $f(P_i)$ be the objective value of Cluster P_i. We have

$$f(P_i) = \sum_{p \in P_i} (\overrightarrow{p} - \overrightarrow{M})^2 = \sum_{p \in P_i} \overrightarrow{p}^2 - l\overrightarrow{M}^2,$$

since $\sum_{p \in P_i} \overrightarrow{p} = l\overrightarrow{M}$. Similarly, $f(P_{i_1}) = \sum_{p \in P_{i_1}} \overrightarrow{p}^2 - Sig\overrightarrow{M_1}$ and $f(P_{i_2}) = \sum_{q \in P_{i_2}} \overrightarrow{q}^2 - (l - Sig)\overrightarrow{M_2}^2$.

Applying Equation 5.1 and straightforward algebra, we get

$$\begin{aligned}& f(P_i) - (f(P_{i_1}) + f(P_{i_2})) \\ = \ & Sig\overrightarrow{M_1}^2 + (l - Sig)\overrightarrow{M_2}^2 - l\overrightarrow{M}^2 \\ = \ & \frac{Sig(l - Sig)}{l}(\overrightarrow{M_1}^2 + \overrightarrow{M_2}^2 - 2\overrightarrow{M_1}\overrightarrow{M_2}) \geq 0 \qquad (5.2)\end{aligned}$$

Thus, we could obtain a clustering whose objective value is smaller than or equal to the previous one by splitting P_i into P_{i_1} and P_{i_2}, yielding a contradiction. □

In the *CD*-Tree, each entry of a leaf node represents an individual data object. The maximum capacity of a non-leaf node is set to Z which is a constant and can be set arbitrarily. Every entry of a non-leaf node corresponds to the subtree rooted at one of its child

[1] *Note that a similar result is presented in [39].*

nodes. The entry stores a pointer to the child node, as well as the statistical information (the mean vector, linear sum and squared sum) of all the data objects in the corresponding subtree. Similar to the CF-Features [120], the statistical information is used to direct a new data object along a path to the closest leaf node. Besides, all the leaf nodes are linked together for easy access of their neighborhood.

The construction of a CD-Tree relies on two basic operations: *insertion* and *split*. The CD-Tree algorithm takes one data object at a time and inserts it into an appropriate leaf node following the path from the root. A tree node is split into two nodes whenever its capacity is exceeded. The CD-Tree is constructed by repeatedly invoking these two operations until all data are processed. After the CD-Tree is ready, some post-processing is needed to generate clusters that satisfy the constraints of the CDC problem.

This approach has three advantages: (1) Building a CD-Tree requires only one scan of a dataset so that the disk access is minimized. (2) Benefiting from the tree structure, our approach can easily deal with incremental updates of the dataset. (3) The CD-Tree algorithm ensures that the data objects in the same leaf node are similar to each other. Thus it is sufficient to examine only the neighboring leaf nodes whenever there is some change to the required constraints.

Insertion. Given a new data object p, we first locate the leaf node that p shall be inserted into. Starting from the root of the CD-Tree, every time we pick the subtree whose mean vector (which is part of the statistical information) is the closest to p. Repeat this process until we reach some leaf node. If the variance of a leaf node exceeds the threshold after inserting p, we need to create a new leaf node to accommodate it.

Split. In the construction of the CD-Tree, a split is invoked whenever the capacity of a node is exceeded. The proof of Lemma 5.4.1 provides an efficient way to evaluate the drop of the objective value during a split. Based on Equation 5.2, we design an efficient algorithm to split a group of $2S$ data objects P_i into two clusters P_{i_1} and P_{i_2} (Algorithm 7). The algorithm proceeds in a greedy fashion. First we pick the data object that is farthest from P_i's mean vector and add it to P_{i_2}. Then we iteratively add data objects that are closest to P_{i_2}'s mean vector into P_{i_2} until the objective value stops decreasing. And P_{i_1} keeps all remaining data objects in P_i. Splitting non-leaf nodes can be done similarly by considering the mean vector saved in the entries of the non-leaf nodes.

Finally, after each split, we need to decide how to link the newly created leaf nodes with the existing nodes. Suppose we just split a leaf node P_i into P_{i_1} and P_{i_2}. Let P_{prev} and

Algorithm 7 Splitting a group of $2S$ data objects.

```
1: Input: $P_i$ contains $2S$ data objects
2: Output: $P_{i_1}$ and $P_{i_2}$
3: $P_{i_1} = P_i$; size $= |P_{i_1}|$;
4: Set $\overrightarrow{M_1}$ to the mean vector of data objects in $P_{i_1}$;
5: Pick the data object $p$ in $P_{i_1}$ that is farthest from $\overrightarrow{M_1}$;
6: Remove $p$ from $P_{i_1}$ and update $\overrightarrow{M_1}$;
7: $P_{i_2} = \{p\}$; $\overrightarrow{M_2} = \overrightarrow{p}$;
8: MaxObjDrop $= \frac{(size-1)}{size}(\overrightarrow{M_1}^2 + \overrightarrow{M_2}^2 - 2\overrightarrow{M_1}\overrightarrow{M_2})$;
9: for $i = 2$ to $S$ do
10:   Pick a data object $p'$ from $P_{i_1}$ that is nearest to $\overrightarrow{M_2}$;
11:   Remove $p'$ from $P_{i_1}$ and update $\overrightarrow{M_1}$;
12:   Add $p'$ to $P_{i_2}$ and update $\overrightarrow{M_2}$;
13:   ObjDrop $= \frac{(size-i)}{size}(\overrightarrow{M_1}^2 + \overrightarrow{M_2}^2 - 2\overrightarrow{M_1}\overrightarrow{M_2})$;
14:   if ObjDrop $\geq$ MaxObjDrop then
15:     Continue;
16:   else
17:     Remove $p'$ from $P_{i_2}$ and add $p'$ back to $P_{i_1}$;
```

P_{next} be the both neighbors of P_i before split. There are two ways to link P_{i_1} and P_{i_2}. Let $\mu(P_i)$ be the mean vector of P_i. If $|\mu(P_{prev}) - \mu(P_{i_1})| \leq |\mu(P_{prev}) - \mu(P_{i_2})|$, we create links $P_{prev} \rightarrow P_{i_1} \rightarrow P_{i_2} \rightarrow P_{next}$, otherwise $P_{prev} \rightarrow P_{i_2} \rightarrow P_{i_1} \rightarrow P_{next}$.

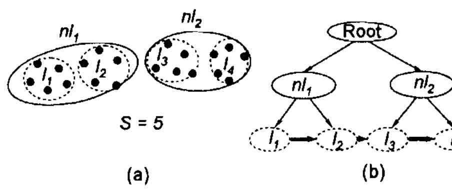


Figure 5.4: (*a*) Data Objects. (*b*) Visualization of the *CD*-Tree.

5.4.2 Solving the *CDC* problem

After we construct a *CD*-Tree, the *CDC* problem can be solved by post-processing the leaf nodes of the *CD*-Tree. Similar to many other hierarchical tree structures, a *CD*-Tree has the property that data objects located in the same subtree tend to be more similar than the data objects located in different subtrees. By linking tree nodes appropriately, we are able to retrieve the neighbors of any data object easily. For example, Figure 5.4 shows a small dataset and its corresponding *CD*-Tree. In this figure, dashed circles represent leaf nodes and solid circles correspond to non-leaf nodes of the *CD*-Tree. Since every new data object is inserted into the tree based on its distance to the mean vectors of subtrees, data objects in l_1 are more similar to data objects in l_2 than to data objects in l_3 or l_4. This allows us to postprocess the neighboring leaf nodes to obtain constrained clusters.

We propose a sliding window approach for solving the *CDC* problem. A sliding window consists of exactly Z leaf nodes with the same parent node. Starting with the first leaf node in the window, we examine one leaf node at a time. Depending on the given significance constraint Sig and variance constraint Var, we distinguish between two types of a leaf node L. L is called *qualified* if L has at least Sig data objects and variance at least Var, otherwise *not-qualified*. For every qualified leaf node L, we output its "kernel", which is a subset of data objects of L that just satisfies the given constraints. Kernels can be easily calculated using a greedy approach (Case *a* of Algorithm 8). The remaining data objects in L are treated together with those under-qualified leaf nodes. We repeatedly absorb data objects from other leaf nodes in the same window to form clusters that satisfy the given constraints (Case *b* of Algorithm 8). Note that all the data objects which have been assigned to some clusters are not considered in case *b*. After looping through all the leaf nodes, a set of constrained clusters is generated.

Algorithm 8 Cluster leaf nodes

```
1: Input: significance constraint $Sig$, variance constraint $Var$, a leaf node $L$.
2: Output: a set of constrained clusters
3: $Q = \emptyset$
   Case a:
4: if $L.size \geq Sig$ and $L.var \geq Var$ then
5:   insert the data object closest to $L.mean$ to $Q$
6:   while $Q.size < Sig$ or $Q.var < Var$ do
7:     Add the data object closest to the $Q.mean$ from $L$ to $Q$
8:   Output $Q$
9:   $L = L \setminus Q$, apply Case b to $L$
   Case b:
10: while $L.size < Sig$ or $L.var < Var$ /*not-qualified*/ do
11:   Absorb similar data objects from other leaf nodes in the same window following the link, shift the
      window if all data objects in the current window are absorbed
12: Output $L$
```

5.4.3 Runtime Analysis

We assume that both *CDC* constraints are non-trivial. In order to analyze the runtime of the *CD*-Tree algorithm, we first bound the height of a *CD*-Tree. In the worst case, a *CD*-Tree can have $O(n)$ levels if every inserted data object triggers a split and every split results in two leaf nodes containing $2Sig - 1$ data objects and one data object respectively.

An insertion of a single data object into the CD-Tree involves two operations: locating the right leaf node to insert the data object and splitting the leaf node if its capacity is exceeded. The time to locate the right leaf node is $O(n)$ since the height of the CD tree is $O(n)$. In a split (Algorithm 7), we create a new node starting with the farthest data object from the mean of the old node (which can be found in $O(Sig)$ steps), and gradually absorb the closest data object to the mean of the new node (each of the $O(Sig)$ absorptions takes $O(Sig)$ steps). Hence, the runtime of a split is $O(Sig^2)$. For building a CD-Tree with n data object, the total runtime is $O(n^2 + Sig^2 n)$ in the worst case.

In the phase of postprocessing the CD-Tree to solve the CDC problem, if the variance constraint Var is not specified, generating a valid cluster consisting of $O(Sig)$ data objects requires $O(Z \cdot Sig^2)$ steps since there are at most $O(Z \cdot Sig)$ data objects in a sliding window of length Z which is often considered as a constant. Thus, generating a set of $O(n/Sig)$ valid clusters takes $O(n \cdot Sig)$ steps. If a variance constraint Var is also specified, the number of data objects in a valid cluster could be $O(n)$ in the worst case (imagine that there is only one valid cluster containing all data objects). In such a case the runtime of the second phase is $O(n^2)$. Therefore, the overall runtime is $O(n^2 + Sig^2 n)$.

Yet, in practice, Sig is typically small compared to n. If the data distribution is not highly skewed, the height of the CD-Tree is usually small. In such cases, our algorithm is very efficient as demonstrated by the experimental evaluation in Section 5.5.

5.4.4 Discussion

How to handle a trivial variance constraint? If a variance constraint is trivial, i.e., $Var = 0$, we need to set the variance parameter V used for the CD-Tree construction automatically. A suitable threshold should allow Sig closest data objects to be grouped together. Ideally, if we know the average Sig nearest neighbor distance of a dataset, this distance can be used to approximate the variance parameter. However, the exact computation of the average Sig nearest neighbor distance is expensive. Therefore, we propose to estimate the Sig nearest neighbor distance based on the following lemma.

Lemma 5.4.2 *Given a dataset of n data objects that are uniformly distributed in a d dimensional space with volume Vol. Fix any data object p, let R be the random variable indicating the radius of the smallest enclosing ball of the Sig nearest neighbors of p. Let*

$R^* = \pi^{-1/2} \sqrt[d]{Sig \cdot Vol\Gamma(d/2+1)n^{-1}}$ *where* Γ *is the Gamma Function [2]. Then*

$$\Pr[2^{-3/d}R^* \leq R \leq 2^{2/d}R^*] \geq 1 - 2e^{-Sig}.$$

Proof. We first show that $\Pr[R > 2^{2/d}R^*] \leq n^{-2}$. Let H be the hypersphere with radius $2^{2/d}R^*$ centered at p. Let *vol(H)* be the volume of H. Then, from [2], we have

$$vol(H) = \pi^{\frac{d}{2}}(2^{2/d}R^*)^d(\Gamma(d/2+1))^{-1} = 4Sig \cdot Vol/n.$$

Let Q be a random variable indicating the number of data objects in H. We have $E[Q] = n \cdot vol(H)/Vol = 4Sig$. Using Chernoff's bound we obtain $\Pr[Q < Sig] \leq \Pr[Q < (1 - 3/4)E[Q]] < e^{E[Q]\cdot(3/4)^2/2} \leq e^{-Sig}$. Consequently $\Pr[R > 2^{2/d}R^*] \leq \Pr[Q < Sig] < e^{-Sig}$. Similarly we can show that $\Pr[R < 2^{-3/d}R^*] < e^{-Sig}$ so the result follows. □

Lemma 5.4.2 shows that we can estimate the *Sig* nearest neighbor distance with high probability if we know the volume *Vol* of a dataset for uniformly distributed data. In practice, if the volume is not known, we can draw a small set of samples to obtain a good estimation.

How to handle a trivial significance constraint? A *CD*-Tree requires a meaningful significance constraint to set its leaf node capacity. If the supplied significance constraint is 0 and only a variance constraint *Var* is given, we can estimate the number of data objects located in a hypersphere of radius $\sqrt{Var}$ for uniformly distributed data and set the number to be *Sig*. Let $vol(H)$ be the hypersphere with radius $\sqrt{Var}$ and *Vol* be the volume of a dataset. The expected number of data objects located in this hypersphere is $n \cdot vol(H)/Vol$, n is the total number of data objects in the dataset.

5.5 Experimental Evaluation

In this section, we experimentally demonstrate the efficiency and effectiveness of the *CD*-Tree algorithm using real and synthetic datasets.

5.5.1 Methodology

In order to evaluate the *CD*-Tree algorithm for sensor network applications, we generated a synthetic dataset (DS1) consisting of 5000 two dimensional data objects which simulates a sensor network with 5000 sensors uniformly deployed in a two dimensional space. A similar simulation was used in [61] to evaluate clustering results for sensor networks. In addition, we evaluated the *CD*-Tree algorithm on two real datasets. The first one is the "Abalone" dataset and the second one is the "Letter" dataset. Both datasets are from UCI machine learning repository [1]. The "Abalone" dataset was also used by [4] for evaluating the quality of the condensation group approach for privacy preservation applications. The original abalone dataset contains 4177 data objects and 9 attributes. We preprocessed the dataset and kept 7 out of total 8 continuous attributes since one of the attributes is the class label. The Letter dataset includes 20,000 instances and has 16 continuous attributes. Finally, we generated three large synthetic datasets, containing 0.5 million, 1 million, and 2 million three dimensional data objects, to evaluate the scalability of the *CD*-Tree algorithm.

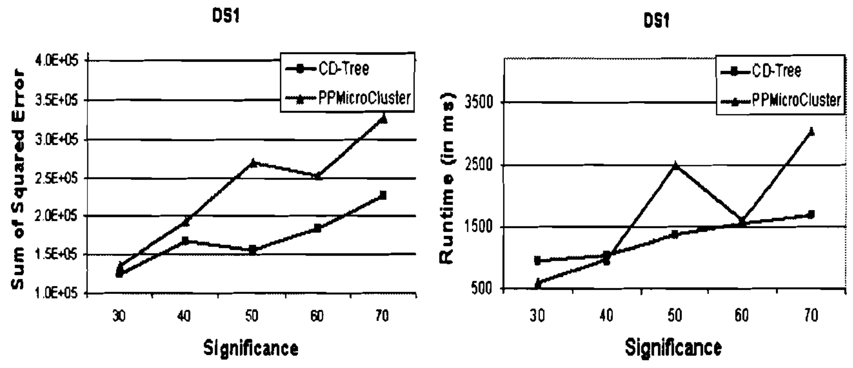


Figure 5.5: Results for Dataset DS1 (Only significance constraints are specified).

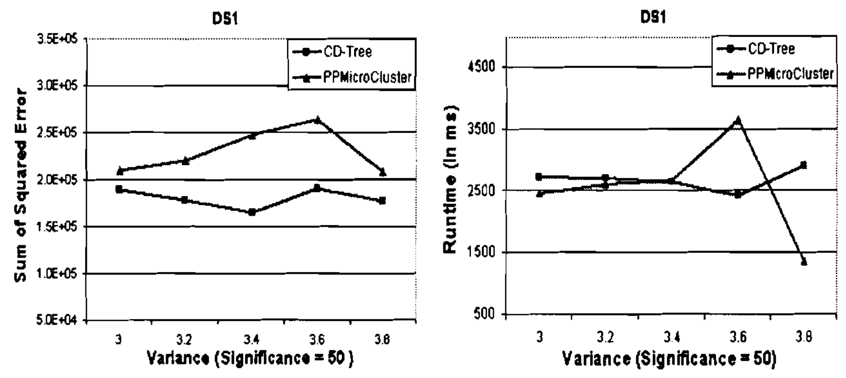


Figure 5.6: Results for Dataset DS1 (Both significance and variance constraints are specified).

Two related approaches, the PPMicroCluster algorithm [71] and the condensation group

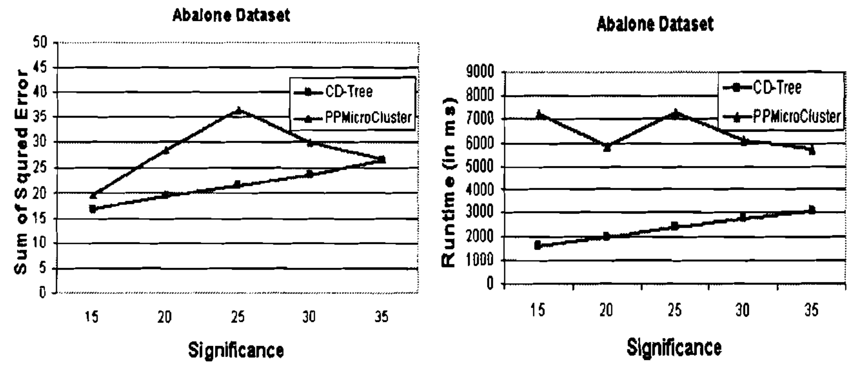


Figure 5.7: Results for Abalone Dataset (Only significance constraints are specified).

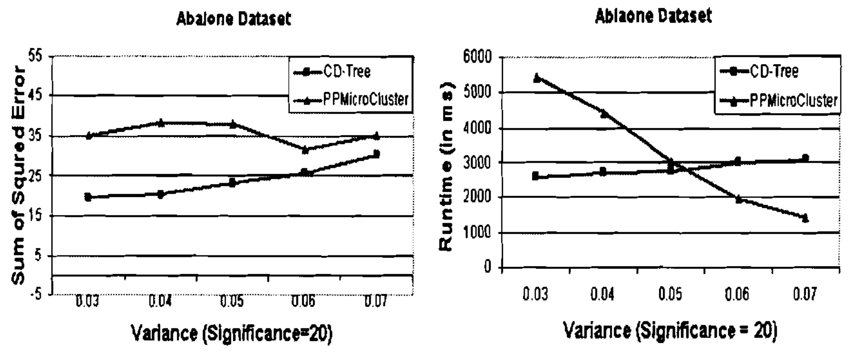


Figure 5.8: Results for Abalone Dataset (Both significance and variance constraints are specified).

approach [4], solve a constrained clustering problem similar to the *CDC* problem. We chose the PPMicroCluster algorithm as our comparison partner due to the following reasons. First, its problem definition is equivalent to the *CDC* problem except that is specifies a radius constraint instead of a variance constraint. Different from the variance, the radius of a cluster is the maximum distance between all data objects in a cluster to the cluster representative. We have adapted the PPMicroCluster algorithm to handle the variance constraint in order to have a meaningful comparison. Second, the PPMicroClustering problem generalizes the condensation group approach (which has only the significance constraint), and the significance constraint is handled by the PPMicroCluster algorithm in the same style as by the condensation group approach. Note that the following experiments were conducted on the static phase of the PPMicroCluster algorithm since we did not evaluate the incremental updates of databases and the static phase of the PPMicroCluster algorithm is more effective due to the availability of the global knowledge for all data objects.

5.5.2 Results

We compared the two algorithms from two aspects, clustering quality and runtime. We set the same parameters for both algorithms and ran them on the aforementioned datasets. The clustering quality is measured by the sum of squared distances of data objects to their corresponding cluster representatives. Note that the *CD*-Tree and the adapted PPMicroCluster algorithm both satisfy the same constraints, but with possibly different compactness of the discovered clusters. For the *CD*-Tree algorithm, we set the capacity of non-leaf nodes to 20 and the runtime is the total time spent on building a tree and generating constrained clusters from the tree. For the PPMicroCluster algorithm, we assume an index structure (*R*-Tree) existing for supporting fast k-nearest-neighbor query. The runtime only records the time spent on generating valid clusters based on the index structure, excluding the index construction time. All the experiments were conducted on a server with an Intel Pentium IV 3.0GHz CPU and 2GB memory running the Window Server 2003.

For dataset DS1, the results are presented in Figure 5.5 and 5.6 for significance constraints only and both constraints respectively. For both comparisons, the *CD*-Tree algorithm outperforms the PPMicroCluster algorithm in terms of clustering quality. Similar behavior is observed on the abalone dataset, for which the results are depicted in Figure 5.7 and 5.8. Due to the small size of the abalone dataset, both algorithms require comparable time.

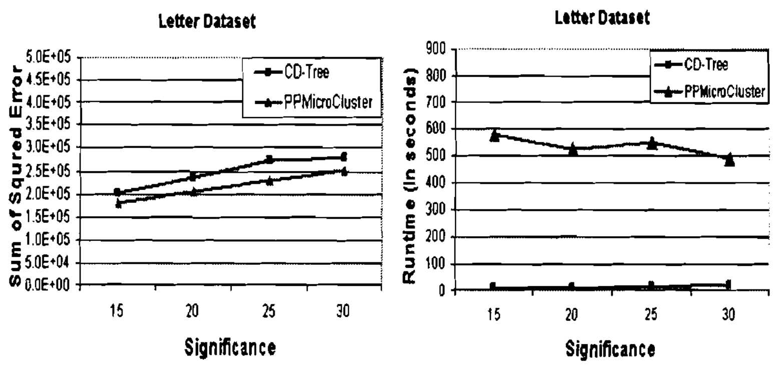


Figure 5.9: Results for the Letter Dataset (Only significance constraints are specified).

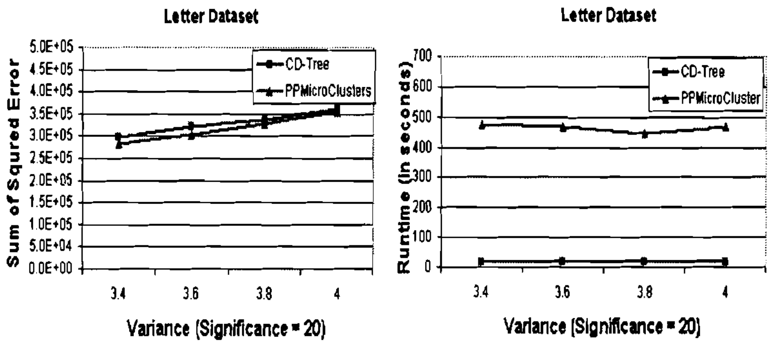


Figure 5.10: Results for the Letter Data set (Both significance and variance constraints are specified).

For the larger Letter dataset (see Figure 5.9 and 5.10), we observe that the PPMicroCluster algorithm slightly outperforms the *CD*-Tree algorithm in terms of clustering quality. This behavior is expected since the PPMicroCluster algorithm relies on an index structure to maintain an accurate neighborhood relations among data objects, while the tree structure built by the *CD*-Tree algorithm only keeps approximate neighborhood relations among data objects. However, the *CD*-Tree algorithm runs more than 100 times faster than the PPMicroCluster algorithm. A small sacrifice of the clustering quality is reasonable for a dynamic algorithm like the *CD*-Tree algorithm.

# of data objects	0.5 million	1 million	2 million
Runtime (in seconds)	160	295	677

Table 5.1: Scalability vs. Number of Data Objects.

In order to evaluate the scalability of the *CD*-Tree algorithm to large datasets, we generated three synthetic datasets with 0.5 million, 1 million, and 2 million three dimensional data objects. We evaluated only the *CD*-Tree algorithm on these synthetic datasets since the PPMicroCluster algorithm cannot handle such large datasets. The runtime results of the *CD*-Tree algorithm with the significance constraint $Sig = 30$ are presented in Table 5.1. In order to evaluate the impact of different values of the constraints, we apply the *CD*-Tree algorithm on the dataset with 1 million data objects. Table 5.2 contains the runtime results.

Constraint Combinations	Runtime (in seconds)
$Sig = 20, Var = 0$	231
$Sig = 30, Var = 0$	295
$Sig = 40, Var = 0$	402
$Sig = 30, Var = 0.7$	408
$Sig = 30, Var = 1.1$	438
$Sig = 30, Var = 1.5$	471

Table 5.2: Scalability vs. Different Constraints.

5.6 Summary

Clustering methods can be either data-driven or need-driven. Among need-driven methods, constrained clustering captures background knowledge or application requirements by specifying constraints. In this chapter, we have introduced a novel clustering model, Constraint-Driven Clustering (*CDC*), which aims at utilizing constraints to drive the cluster formation. We have focused on two constraint types, i.e., minimum significance constraints and minimum variance constraints, for discovering actionable clusters for applications such as energy aware sensor networks and privacy preservation applications. We have proved the NP-hardness of the proposed *CDC* problem with difference constraints. We have also proposed a novel dynamic data structure, the *CD*-Tree, which keeps dataset summaries that approximately satisfy the given constraints by minimizing the sum of squared distances during its construction. Based on *CD*-Trees, an efficient algorithm is developed for the new clustering problem. Our experimental evaluation on synthetic and real datasets showed that our algorithm yields good clusters efficiently.

This study suggests several interesting directions for future research. First, some issues related to our heuristic algorithm, such as how the maximum capacity of a non-leaf node influences the final partition and how effective is our heuristic algorithm on high dimensional data, can be further studied. Second, we want to evaluate the *CDC* model in real life applications. Third, the application needs may not always suggest exact values for the significance and variance constraints. Therefore, it is worthwhile to explore variants of the *CDC* model that allow the user to specify ranges instead of a fixed minimum value. Finally, we believe that the *CDC* framework and the CD-Tree algorithm can be generalized to include other constraint types, such as minimum separation constraints [32], to produce actionable clusters in an even broader category of applications.

Chapter 6

Conclusion

In this thesis, we have studied clustering problems arising from several real life applications, namely, catalog segmentation, community identification in social networks, privacy preservation and energy aware sensor networks. We have modeled all these problems under the unified framework of clustering with cluster-level constraints. We have shown that cluster-level constraints can capture complex application needs and domain knowledge in these applications, and the corresponding clustering models integrating those constraints yield meaningful results.

Customer-Oriented Catalog Segmentation The microeconomic framework for data mining assumes that an enterprise chooses a decision maximizing the total utility over all customers where the contribution of a customer is a function of the data available on that customer. In Catalog Segmentation, the enterprise wants to design k product catalogs of size r that maximize the utility defined by the total number of catalog products purchased. From the point of view of clustering, the task of catalog segmentation is to find k clusters of customers where each cluster is described by a set of products and each customer is assigned to the cluster with the most similar cluster description. The utility function specifically designed for maximizing the total number of catalog products purchased leads to a better partitioning of customers in terms of the profit of an enterprise.

There are many applications where a customer, once attracted to an enterprise, would purchase more products beyond the ones contained in the catalog. In this thesis, we have investigated an alternative problem formulation, named Customer-Oriented Catalog Segmentation, to capture the requirement on maximizing the number of customer visiting the enterprise. In the new model, the overall utility is measured by the number of customers that satisfy the minimum interest constraint t, i.e., having at least a minimum interest t in the catalogs. The model utilizes both a utility function and a minimum interest constraint to obtain desired clustering results. More specifically, we have formally introduced the Customer-Oriented Catalog Segmentation problem and discuss its complexity. We have presented efficient, approximate algorithms adopting the paradigms of greedy and randomized algorithms. Our experimental evaluation on synthetic and real data showed that the new algorithms yield catalogs of significantly higher utility compared to classical Catalog Segmentation algorithms. Our best algorithm, Random-Product-Fit, achieves an excellent tradeoff between quality and runtime by optimizing a greedily determined initial solution randomly.

Joint Cluster Analysis Attribute data and relationship data are two principal types of data, representing the intrinsic and extrinsic properties of entities. It is also common to observe both data types carry complementary information such as in market segmentation and community identification. In this thesis, we have introduced the novel Connected k-Center (*CkC*) problem, a clustering model that takes into account both attribute data and relationship data. The *CkC* problem integrates an internal connectedness constraint defined on relationship data and an objective function specified on attribute data in order to generate clusters which are cohesive (within clusters) and distinctive (between clusters) in both ways. In this model, the internal connectedness constraint is a novel cluster-level constraint which provides a way to capture application needs in several applications, e.g., market segmentation, and community identification.

We have proved the NP-hardness of the problem and provided a constant factor approximation algorithm. For the special case of the *CkC* problem where the underlying graph is a tree, we have proposed a dynamic programming method giving an optimal solution in polynomial time. To improve the scalability for large real datasets, we have developed the efficient heuristic algorithm NetScan. Our experimental evaluation using real datasets for community identification and gene clustering demonstrated the meaningfulness and accurary of the NetScan results. In addition, experiments on synthetic datasets demonstrated the efficiency and scalability of the NetScan algorithm.

Constraint-Driven Clustering Many existing clustering models with constraints rely on the user-provided number of clusters. In many applications, the number of clusters is unknown a priori. Furthermore, an inappropriate number of clusters may result in generating distorted and less actionable clusters. In this thesis, we have introduced a novel clustering model, Constraint-Driven Clustering (*CDC*), which aims at utilizing cluster-level constraints to drive the cluster formation.

The SQL aggregate constraint [108] provides a general form of cluster-level constraints, but it cannot capture the complex requirements in many other applications. For example, in privacy preservation, the requirement to specify the minimum variance of each generated cluster cannot be captured by the SQL aggregate constraint. In this thesis, we have proposed a new type of cluster-level constraints, i.e., minimum variance constraints, to capture such requirement. The minimum variance constraints together with the well studied minimum significance constraints are integrated to the *CDC* model to discover meaningful clusters for applications such as energy aware sensor networks and privacy preservation applications.

We have proved the NP-hardness of the proposed *CDC* problem with difference constraints. We have also proposed a novel dynamic data structure, the *CD*-Tree, which keeps dataset summaries that approximately satisfy the given constraints by minimizing the sum of squared distances during its construction. Based on *CD*-Trees, an efficient algorithm is developed for the new clustering problem. Our experimental evaluation on synthetic and real datasets showed that our algorithm yields good clusters efficiently.

Future Work For future work, we plan to explore the following directions.

1. *Define a general form of cluster-level constraints.* Similar to the work in [108], we can study a general framework of cluster-level constraints. The general framework should be powerful to capture all existing cluster-level constraints and at the same time be flexible for future extensions. The SQL aggregate constraint or the existential constraint aim at describing the properties of a cluster in terms of aggregate functions. As shown in this thesis, cluster-level constraints can specify more properties than just aggregates. For example, the internal connectedness constraint in the joint cluster analysis for identifying communities specifies the property of the objects in a cluster on the relationship data. The goal of capturing the rich characteristics of clusters makes the task both interesting and challenging.

2. *Combine instance-level constraints and cluster-level constraints.* Other than the constraints on the generated clusters, instance-level constraints, e.g., a small set of labeled

data or domain knowledge on pairs of data objects, are often available in practice and they are usually helpful for obtaining good clustering results. For example, in the problem of identifying academic communities, the difficulty of cluster assignment can be alleviated if the knowledge on pairs of objects allows the "bridge" objects to be uniquely assigned to one cluster. Yet, how to effectively incorporate both types of knowledge is a difficult task. Two important issues need to be addressed: How should we design efficient and effective algorithms which take all these information into account? How should we cope with the case where the two types of knowledge contradict?

3. *Incremental clustering with cluster-level constraints.*

   Another interesting direction is incremental clustering with clustering-level constraints. Clustering, as a learning method, is an iterative process that could take many rounds to fully comprehend a certain domain. In such a process, little is known about the target domain initially. Afterwards, more and more knowledge may be collected in subsequent iterations. Consequently, it is desirable to design clustering methods that can handle the scenario where the cluster-level constraints are provided incrementally. A naive method would be to calculate a clustering from scratch whenever the constraints change. Unfortunately, as the input data grows quickly, this method becomes prohibitively expensive. Efficient incremental methods are highly demanded for this task.

Bibliography

[1] D.J. Newman A. Asuncion. UCI machine learning repository, 2007.

[2] Milton Abramowitz and Irene A. Stegun(Eds.). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* New York: Dover, 1972.

[3] Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

[4] Charu C. Aggarwal and Philip S. Yu. A condensation approach to privacy preserving data mining. In *Proceedings of the 9th International Conference on Extending Database Technology*, pages 183 - 199, 2004.

[5] Gagan Aggarwal, Tomas Federand Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, 2005.

[6] Rakesh Agrawal. Ibm synthetic data generator, 1994.

[7] Alan D. Amis, Ravi Prakash, Dung Huynh, and Thai Vuong. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings IEEE INFOCOM 2000*, pages 32–41, 2000.

[8] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 49–60, 1999.

[9] Vera Asodi and Shmuel Safra. On the complexity of the catalog segmentation problem. Unpublished manuscript.

[10] Seema Bandyopadhyay and Edward J. Coyle. An energy-efficient hierarchical clustering algorithm for wireless sensor networks. In *Proceedings IEEE INFOCOM 2003*, 2003.

[11] Arindam Banerjee and Joydeep Ghosh. On scaling up balanced clustering algorithms. In *Proceedings of the 2nd SIAM International Conference on Data Mining*, 2002.

[12] Arindam Banerjee and Joydeep Ghosh. Scalable clustering algorithms with balancing constraints. *Data Mining Knowledge Discovery*, 13(3), 2006.

[13] Suman Banerjee and Samir Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *Proceedings IEEE INFOCOM 2001*, 2001.

[14] A.L. Barabási, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicseks. Evolution of the social network of scientific collaborations. *Physica A*, 311(3-4):590–614, 2002.

[15] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[16] Yair Bartal, Moses Charikar, and Danny Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings on the 33rd Annual ACM Symposium on Theory of Computing*, pages 11–20, 2001.

[17] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 4th SIAM International Conference on Data Mining*, 2004.

[18] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68, 2004.

[19] Sugato Basu and Ian Davidson. Clustering with constraints: Theory and practice. *KDD'06 Tutorial*, 2006.

[20] Kristin P. Bennett, P.S. Bradley, and Ayhan Demiriz. Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[21] Gabriel F. Berriz, Oliver D. King, Barbara Bryant, Chris Sander, and Frederick P. Roth. Characterizing gene sets with funcassociate. *Bioinformatics*, 19(18):2502–2504, 2003.

[22] M. J. Berry and G. Linoff. *Data Mining Techniques.* John-Wiley, 1997.

[23] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Experiments on graph clustering algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms*, pages 568–579, 2003.

[24] Tom Brijs, Bart Goethals, Gilbert Swinnen, Koen Vanhoof, and Geert Wets. A data mining framework for optimal product selection in retail supermarket data: The generalized profset model. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 300–304, 2000.

[25] Peter Brucker. On the complexity of clustering problems. In R. Hehn, B. Korte, and W. Oettli, editors, *Optimization and Operations Research*, pages 45–54. Springer-Verlag, 1977.

[26] Julien Cartigny, David Simplot, and Ivan Stojmenovic. Localized minimum-energy broadcasting in ad-hoc networks. In *Proceedings of IEEE INFOCOM 2003*, 2003.

[27] Pak K. Chan, Martine D. F. Schlag, and Jason Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, 1994.

[28] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant factor approximation algorithm for the k-median problem. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 1–10, 1999.

[29] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *Journal of Computer and System Sciences*, 68(2):417–441, 2004.

[30] CiteSeer. Scientific literature digital library. http://citeseer.ist.psu.edu/, 2006.

[31] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[32] Ian Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 5th SIAM International Conference on Data Mining*, pages 138–149, 2005.

[33] Ian Davidson and S. S. Ravi. Identifying and generating easy sets of constraints for clustering. In *Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*, 2006.

[34] Ian Davidson and S. S. Ravi. The complexity of non-hierarchical clustering with constraints. *Journal of Knowledge Discovery and Data Mining*, To Appear.

[35] Ian Davidson and S.S. Ravi. The complexity of non-hierarchical clustering with instance and cluster level constraints. *To Appear in the Journal of Knowledge Discovery and Data Mining*, 2006.

[36] DBLP. Computer science bibliography. http://www.informatik.uni-trier.de/~ley/db/index.html.

[37] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page To appear, 2007.

[38] Srinivas Doddi, Madhav V. Marathe, S. S. Ravi, David S. Taylor, and Peter Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. In *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory*, pages 237–250, 2000.

[39] Josep Domingo-Ferrer and Josep M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.

[40] E. Drezner. *Facility Location: A Survey of Applications and Methods.* Springer-Verlag, Heidelberg, 1995.

[41] Martin Dyer and Alan M. Frieze. A simple heuristic for the p-center problem. *Operations Research Letters*, 3:285–288, 1985.

[42] Martin E. Dyer and Alan M. Frieze. Planar 3dm is np-complete. *J. Algorithms*, 7(2), 1986.

[43] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.

[44] Martin Ester, Rong Ge, Byron J. Gao, Zengjian Hu, and Boaz Ben-mosha. Joint cluster analysis of attribute data and relationship data: the connected k-center problem. In *Proceedings of the 6th SIAM Conference on Data Mining*, pages 246–257, 2006.

[45] Martin Ester, Rong Ge, Wen Jin, and Zengjian Hu. A microeconomic data mining problem: customer-oriented catalog segmentation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[46] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[47] Tomás Feder and Daniel H. Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th annual ACM symposium on Theory of computing*, pages 434–444, 1988.

[48] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal ACM*, 45(4):634 – 652, 1998.

[49] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. In *Knowledge Discovery in Databases*, pages 1–30. AAAI/MIT Press, 1991.

[50] G. N. Frederickson and D. B. Johnson. Optimal algorithms for generating quantile information in $x + y$ and matrices with sorted columns. In *Proceedings of the 13th Annual Conference on Information Science and Systems*, pages 47–52, 1979.

[51] Michael R. Garey and David S. Johnson. *Computers and Intractability, a guide to the Theory of NP-completeness.* W.H. Freeman and company, 1979.

[52] Rong Ge, Martin Ester, Byron J. Gao, Zengjian Hu, Binay Bhattacharya, and Boaz Ben-mosha. Joint cluster analysis of attribute data and relationship data: the connected k-center problem, algorithms and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, To appear.

[53] Rong Ge, Martin Ester, Wen Jin, and Ian Davidson. Constraint-driven clustering. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[54] Rong Ge, Martin Ester, Wen Jin, and Zengjian Hu. A disc-based approach to data summarization and privacy preservation. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*, pages 321 – 332, 2006.

[55] Soheil Ghiasi, Ankur Srivastava, Xiaojian Yang, and Majid Sarrafzadeh. Optimal energy aware clustering in sensor network. *Sensor*, 2(7):258–269, 2002.

[56] Joydeep Ghosh and Alexander Strehl. Clustering and visualization of retail market baskets. In N. R. Pal and L. Jain, editors, *Knowledge Discovery in Advanced Information Systems*. Springer, 2002.

[57] David Gondek and Thomas Hofmann. Non-redundant data clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining*, pages 75 – 82, 2004.

[58] David Gondek, Shivakumar Vaithyanathan, and Ashutosh Garg. Clustering with model-level constraints. In *Proceedings of the 5th SIAM International Conference on Data Mining*, pages 126 – 137, 2005.

[59] Teofilo F. Gonzalez. Clustering to minimize the maximum inter-cluster distance. *Theoretical Computer Science*, 38(2-3), 1985.

[60] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: a robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, pages 512–521, 1999.

[61] Gaurav Gupta and Mohamed Younis. Load-balanced clustering of wireless sensor networks. *IEEE International Conference on Communications*, pages 1848–1852, 2003.

[62] Nili Guttman-Beck and Refael Hassin. Approximation algorithms for min-sum p-clustering. *Discrete Applied Mathematics*, 89(1-3):125–142, 1998.

[63] Daniel Hanisch, Alexander Zien, Ralf Zimmer, and Thomas Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18:S145–S154, 2002.

[64] Robert A. Hanneman and Mark Riddle. *Introduction to social network methods.* http://faculty.ucr.edu/~hanneman/, 2005.

[65] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4-6):175–181, 2000.

[66] Laurie J. Heyer, Semyon Kruglyak, and Shibu Yooseph. Exploring expression data: Identification and analysis of coexpressed genes. *Genome Research*, 9(11):1106 – 1115, 1999.

[67] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.

[68] Dawn Iacobucci. *Networks in marketing.* Sage Publications, 1996.

[69] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data.* Prentice Hall, 1988.

[70] Kamal Jain and Vijay Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual scheme and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

[71] Wen Jin, Rong Ge, and Weining Qian. On robust and effective k-anonymity in large databases. In *Proceedings of the 10th Pacific-Asia Conference*, pages 621–636, 2006.

[72] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. Part II: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979.

[73] O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems. Part I: The p-centers. *SIAM J. Applied Mathematics*, 37(3):513–528, 1979.

[74] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[75] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, New York, 1990.

[76] Vikas Kawadia and P. R. Kumar. Power control and clustering in ad-hoc networks. In *Proceedings of IEEE INFOCOM*, 2003.

[77] Jon Kleinberg. The small-world phenomenon: an algorithmic perspective. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–170, 2000.

[78] Jon M. Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. A microeconomic view of data mining. *Journal of Data Mining and Knowledge Discovery*, 2(4):311–324, 1998.

[79] Jon M. Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. Segmentation problems. In *Proceedings of the 13th Annual ACM Symposium on the Theory of Computing*, pages 473–482, 1998.

[80] Rajesh Krishnan and David Starobinski. Efficient clustering algorithms for self-organizing wireless sensor networks. *Journal of Ad-Hoc Networks*, 4(1):36 – 59, 2005.

[81] Joseph Kruskal and Myron Wish. *Multidimensional Scaling.* Sage Publications, 1978.

[82] Brian Kulis, Sugato Basu, Inderjit S. Dhillon, and Raymond J. Mooney. Semi-supervised graph clustering: a kernel approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 457–464, 2005.

[83] Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992.

[84] Tsau Lin, Y.Y. Yao, and Eric Louie. Value added association rules. In *Proceedings of the 6th Pacific-Asia Conference*, pages 328 – 333, 2002.

[85] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982.

[86] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[87] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

[88] Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984.

[89] Nimrod Megiddo, Arie Tamir, Eitan Zemel, and R. Chandrasekaran. An $o(n \log^2 n)$ algorithm for the k-th longest path in a tree with applications to location problems. *SIAM Journal on Computing*, 10(2):328–337, 1981.

[90] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223 - 228, 2004.

[91] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 144–155, 1994.

[92] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.

[93] Mary Lou Roberts and Paul D. Berger. *Direct Marketing Management.* Prentice Hall, 1999.

[94] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, page 188, 1998.

[95] John Scott. *Social Network Analysis: A handbook.* Sage Publications, 2000.

[96] E. Segal, H. Wang, and D. Koller. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics (Suppl. 1)*, pages 264–272, 2003.

[97] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[98] Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, Kirk Anders, Michael B. Eisen, Patrick O. Brown, David Botstein, and Bruce Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273– 3297, 1998.

[99] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*, 34:D535–D539, 2006.

[100] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[101] Michael Steinbach, George Karypis, and Vipin Kumar. Efficient algorithms for creating product catalogs. In *Proceedings of the WebMining Workshop at the 1st SIAM international conference on Data Mining*, 2001.

[102] Alexander Strehl and Joydeep Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proceedings of the 7th Conference on High Performance Computing*, pages 525 – 536, 2000.

[103] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[104] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557 – 570, 2002.

[105] Arie Tamir. An $o(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.

[106] Ben Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In *Proceedings of 17th International Joint Conference on Artificial Intelligence*, pages 870–878, 2001.

[107] Constantine Toregas, Ralph Swan, Charles Revelle, and Lawrence Bergman. The location of emergency service facilities. *Operations Research*, 19:1363–1373, 1971.

[108] Anthony K. H. Tung, Raymond T. Ng, Laks V. S. Lakshmanan, and Jiawei Han. Constraint-based clustering in large databases. In *Proceedings of the 8th International Conference on Database Theory*, pages 405–419, 2001.

[109] Igor Ulitsky and Ron Shamir. Identification of functional modules using network topology and high-throughput data. *BMC System Biology*, 1(8), 2007.

[110] Kiri Wagstaff. *Intelligent Clustering with Instance-Level Constraints.* PhD thesis, Cornell University, 2002.

[111] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1103–1110, 2000.

[112] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning*, pages 577–584, 2001.

[113] Ke Wang and Ming-Yen Thomas Su. Item selection by "hub-authority" profit ranking. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 652–657, 2002.

[114] Ke Wang, Senqiang Zhou, and Jiawei Han. Profit mining: From patterns to actions. In *Proceedings of the 8th International Conference on Extending Database Technology*, pages 70 – 87, 2002.

[115] Stanley Wasserman and Katherine Faust. *Social Network Analysis: methods and applications.* Cambridge University Press, 1994.

[116] Cynthia M. Webster and Pamela D. Morrison. Network analysis in marketing. *Australasian Marketing Journal*, 12(2):8–18, 2004.

[117] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, and Ke Wang. Mpis: Maximal-profit item selection with cross-selling considerations. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 371–378, 2003.

[118] Donald R. Woods. Drawing planar graphs. Technical report, Report No. STAN-CS-82-943, Computer Science Department, Stanford University, 1981.

[119] Dachuan Xu, Yinyu Ye, and Jiawei Zhang. Approximate the 2-catalog segmentation problem using semidefinite programming relaxation. *Optimization Methods and Software*, 18(6):705–719, 2003.

[120] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103 – 114, 1996.

[121] Shi Zhong and Joydeep Ghosh. Scalable, balanced model-based clustering. In *Proceedings of the 3rd SIAM International Conference on Data Mining*, pages 71 – 82, 2003.