Joint Cluster Analysis of Attribute Data and Relationship Data: the Connected k-Center Problem

Martin Ester, Rong Ge, Byron J. Gao, Zengjian Hu, Boaz Ben-Moshe
School of Computing Science, Simon Fraser University, Canada, V5A 1S6
{ester, rge, bgao, zhu, benmoshe}@cs.sfu.ca

Abstract

Attribute data and relationship data are two principle types of data, representing the intrinsic and extrinsic properties of entities. While attribute data has 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 orthogonal information such as in market segmentation and community identification, which calls for a joint cluster analysis of both data types so as to achieve more accurate results. For this purpose, we introduce the novel Connected k-Center problem, taking into account attribute data as well as relationship data. We analyze the complexity of this problem and prove its NP-completeness. We also present a constant factor approximation algorithm, based on which we further design NetScan, a heuristic algorithm that is efficient for large, real databases. Our experimental evaluation demonstrates the meaningfulness and accuracy of the NetScan results.

1 Introduction

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. By entities we mean corporeal or intangible objects of study interest that have distinctions, such as humans, organizations, products, or events. An entity holds endogenous properties and at the same time bears relations to other entities.

While attribute data continue to be the standard and dominant data source in data analysis applications, more and more relationship data are becoming available due to computerized automation of data probing, collection, and compiling procedures. 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 [26, 23, 27] has been gaining popularity in the study of marketing, community identification, epidemi-

ology, molecular biology and so on.

The two types of data, attribute data and relationship data, can be more or less related. A certain relation between entity A and B may imply some common attributes they share; on the other hand, similar attributes of A and B may suggest a relation of some kind between them with high probability. If the dependency between attribute data and relationship data is high enough such that one can be soundly deducted from or closely approximated by the other, a separate analysis on either is sufficient. For example, the classical facility location problem [24] only manipulates locations (attributes) since the Euclidean distance between A and B can be used to well approximate their reachability via route connections (relationships).

On the other hand, often relationship data contains information that is independent from the attributes of entities. For example, two persons may share many characteristics in common but they never get to know each other; or in contrast, even with totally 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. The small world phenomenon, as a consensus observation, has received voluminous cross-discipline attention [19].

The unprecedented availability of relationship data carrying important additional information beyond attribute data calls for 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, algorithm, 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 cluster models with new algorithmic challenges. The task of clustering is to group entities into clusters that exhibit internal co-

hesion and external isolation. Given both attribute data and relationship data, it is intuitive to require clusters to be cohesive (within clusters) and distinctive (between clusters) in both ways.

As a new clustering model taking into account attribute and relationship data, we introduce and study the Connected k-Center (CkC) problem; i.e., the k-Center problem with the constraint of internal connectedness. 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.

Motivating applications: 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 divides 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 were based on attribute data only such as demographics (age, sex, ethnicity, income, education, religion and etc.) and psychographic profiles (lifestyle, personality, motives and etc.). Recently, social networks have become more and more important in marketing [16]. Ideas and behaviors are contagious. The relations in networks are channels and conduits through which resources flow [16]. Customers can hardly hear companies but they listen to their friends; customers are skeptical but they trust their friends [27]. 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 even 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.

Community identification is one of the major social network analysis tasks, and graph-based clustering methods have been the standard tool for this purpose [26]. In this application, clustering has generally been performed on relationship (network) data solely. Yet it is intuitive [23, 13] that attribute data can impact community formation in a significant manner. 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 relationships) 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 intensity of connectivity. Note that most graph-based clustering methods used for community identification in network analysis also return some connected components.

Contributions and outline: This paper makes the following contributions:
1. We advocate joint cluster analysis of attribute data and relationship data, introduce the novel CkC clustering problem.
2. We analyze the complexity of the CkC clustering problem, prove its NP-completeness and present a corresponding 3-approximation algorithm.
3. Based on principles derived from the approximation algorithm, we propose a heuristic algorithm NetScan that efficiently computes a "good" clustering solution.
4. We report results of our experimental evaluation, demonstrating the meaningfulness of the clustering results and the scalability of the NetScan algorithm.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. Section 3 introduces the CkC clustering problem and analyzes its complexity. In Section 4, we present an approximation algorithm for the proposed clustering problem. To provide more scalability, we also present an efficient heuristic algorithm in Section 5. We report experimental results in Section 6 and conclude the paper in Section 7.

2   Related Work

Theoretical approaches to cluster analysis usually formulate clustering as optimization problems, for which rigorous complexity studies are performed and polynomial approximation algorithms are provided. Depending on the optimization objective, many clustering problems and their variants have been investigated, such as the k-center and k-median problems for facility lo-

cation [24], the min-diameter problem [7], and so on. In the areas of statistics, machine learning, and data mining, clustering research emphasizes more on real life applications and development of efficient and scalable algorithms. Clustering algorithms can be roughly categorized [12] into partitioning methods such as k-means [20], k-medoids [19], and CLARANS [22], hierarchical methods such as AGNES and DIANA [19], and density-based methods such as DBSCAN [10] and OPTICS [1]. The above methods generally take into account only attribute data.

We also summarize some theoretical results on the k-center problem that are related to our theoretical analysis. It is well known that both the k-center and Euclidean k-center problem is NP-Complete for $d \geq 2$ when k is part of the input [21]. Besides, in the case of $d = 1$, the Euclidean k-center problem is polynomially solvable using dynamic programming techniques. Meanwhile, when $d \geq 2$, if k is treated as a constant, the k-center problem can also be easily solved by enumerating all the k centers. However, as we will see in section 3, the CkC problem remains NP-Complete even for $k = 2$ and $d = 1$. Hence in this sense, the CkC problem is harder than Euclidean k-center.

Recently, the increasing availability of relationship data stimulated research on network analysis [26, 23, 13]. Clustering methods for network analysis are mostly graph-based, separating sparsely connected dense subgraphs from each other as seen in [6]. A good graph clustering should exhibit few between-cluster edges and many within-cluster edges. Graph clustering methods can be applied to data that is originally network data as well as to similarity graphs representing a similarity matrix, i.e., derived from the original attribute data. A similarity graph can be a complete graph as in agglomerative hierarchical clustering algorithms of single-link, complete link, or average link [17]; or incomplete retaining those edges whose corresponding similarity is above a threshold [11, 14]. CHAMELEON [18] generates edges between a vertex and its k nearest neighbors, which can be considered as relative thresholding. Note that none of the above methods simultaneously considers attribute and relationship data that represent independent information.

Finally, our research is also related to the emerging areas of constraint-based clustering and semi-supervised clustering. Early research in this direction allowed the user to guide the clustering algorithm by constraining cluster properties such as size or aggregate attribute values [25]. More recently, several frameworks have been introduced that represent available domain knowledge in the form of pairwise "must-links" and "cannot-links". Objects connected by a must-link are supposed to be-long to the same cluster; those with a cannot-link should be assigned to different clusters. [3] proposes a probabilistic framework based on Hidden Markov Random Fields, incorporating supervision into k-clustering algorithms. [8] also considers additional minimum separation and minimum connectivity constraints, but they are ultimately translated into must-link and cannot-link constraints. A k-means like algorithm is presented using a novel distance function which penalizes violations of both kinds of constraints. The above two papers are similar to our research in the sense that they also adopt a k-clustering (partitioning) approach under the framework of constraint-based clustering. Nevertheless, in semi-supervised clustering, links represent specific 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; e.g., "internally connected" as in this paper.

3 Problem Definition and Complexity Analysis

In this section, we introduce the Connected k-Center problem and analyze its complexity.

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. Sometimes, similarity matrixes are transformed into graphic representations, called similarity graphs, with entities as vertices and pairwise similarities, possibly thresholded, as edge weights. Relationship data are usually modeled by networks comprised of nodes and links, which we call entity networks. In this paper, 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. Entity networks can as well be represented by adjacency matrices or incident matrices as graphs in general.

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 to perform joint cluster analysis.

For low dimensional (attribute) data, informative graphs can be directly visualized. For higher dimen-

sional data, multidimensional scaling techniques can be invoked to provide such a graphical visualization. These visualizations can intuitively demonstrate the dependency and orthogonality of attribute data and relationship data. In the former case, the visualizations exhibit relatively short edges only, as shown in Figure 1 (a); in the latter case, the visualizations exhibit randomness in terms of edge length, as shown in Figure 1 (b).
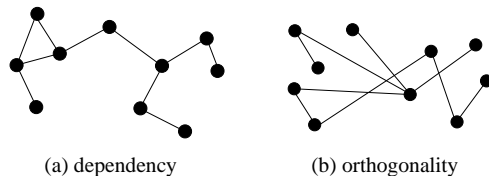

(a) dependency   (b) orthogonality

Figure 1: Attribute data and relationship data.

In this paper, 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 two data types simultaneously.

Now we introduce the decision version of the CkC(Connected k-Center) problem, for constant $k \geq 2$ and $d \geq 1$, where d is the dimensionality.

DEFINITION 3.1. (CkC problem) *Given k as the number of centers, a radius constraint $r \in \mathcal{R}^+$, a distance function $|| \cdot ||$ and a graph $G = (V, E)$ where every node in V is associated with a coordinate vector $w : V \to \mathcal{R}^d$. Decide whether there exist disjoint partitions $\{V_1, \ldots, V_k\}$ of V, i. e. , $V = V_1 \cup \ldots \cup V_k$ and $\forall 1 \leq i < j \leq k, V_i \cap V_j = \phi$, which satisfy the following two conditions:*

1. *The induced subgraphs $G[V_1], \ldots, G[V_k]$ are connected.* (internal connectedness constraint)

2. *$\forall 1 \leq i \leq k$, there exists a center node $c_i \in V_i$, such that $\forall v \in V_i, ||w(v) - w(c_i)|| \leq r$.* (radius constraint)

Intuitively, the problem is to check whether the input graph can be divided into k connected components, such that every 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 assume the given graph is connected which is reasonable for many application scenarios, such as social networks which are normally considered to be connected. Even if the entire graph is not connected, the problem can be applied to its different connected components.

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 after attaching a constraint of internal connectedness?* To answer this question, we analyze the complexity of the CkC problem.

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 combinations of k centers, and for each combination, assign remaining nodes to the centers such that both the radius constraint and the internal connectedness constraint are satisfied. However, we note that there may exist some "articulation" node v which itself can connect to two or more centers within radius r, and v is critical (the only choice) to connect some other nodes to their corresponding centers. In a legal partitioning, every articulation node must be assigned to a unique center. If there are many such articulation nodes, it is difficult to assign each of them to the right center in order to maintain the linkages for others. Therefore the naive algorithm may fail to determine the specified clustering. Hence, intuitively the CkC problem is hard even for a constant k. In the following, we prove a hardness result for the CkC problem, by a reduction from 3CNF-SAT. For completeness, we define the 3CNF-SAT problem as follows:

DEFINITION 3.2. (3CNF-SAT) *Given a set $U = \{u_1, \ldots, u_n\}$ of variables, let $C = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be an instance of 3CNF-SAT. Each C_i contains three literals*

$$C_i = l_i^1 \vee l_i^2 \vee l_i^3,$$

where each literal l_i^x, $x = 1, 2, 3$, is a variable or negated variable. Decide whether there is a truth assignment of U which satisfies every clauses of C.

THEOREM 3.1. *For any $k \geq 2$ and $d \geq 1$, the CkC problem is NP-Complete.*

Proof. We only show the case for $k = 2$ and $d = 1$, the same proof can be easily extended to larger k and d. First we prove $C2C$ is in NP. We can nondeterministically guess a partitioning of a graph G and pick a node as center from each partition. For each partition, we can deterministically check if it is valid by traversing the corresponding subgraph to verify both the connectedness and radius constraints.

Next, we perform a reduction from 3CNF-SAT to show the NP-hardness. Denote $L = \{u_1, \overline{u}_1, \ldots, u_n, \overline{u}_n\}$ to be the set of literals. For any 3CNF-SAT instance

$C = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, we construct an instance of C2C $f(I) = (G, w, r)$, where $G = (V, E)$ is a graph, $w : V \rightarrow R$ is a function which assigns a coordinate vector to each node, $r \in \mathcal{R}^+$ is the radius constraint, by the following procedure:

1. First, we create a set of nodes $V = P \cup L \cup C \cup A \cup B$. $P = \{p_0, p_1\}$ where p_0, p_1 are two center nodes; L, 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 reduction. In the following, we shall refer to the cluster including node p_0 (or p_1) as V_0 (or V_1, resp.).

2. Next, we link the nodes created in step 1. First, we connect p_0 and p_1 with every node in L. Second, for every literal $l \in L$ and clause $C_i \in C$, we link l and C_i if $l \in C_i$. Finally, $\forall i \in \{1, 2, \ldots, n\}$, we link each of a_i and b_i with each of u_i and $\overline{u}_i$. Refer to Figure 2 for a visualization of graph G. Note that every node in A, B, C can only connect to the center nodes p_0 and p_1 via some node in L, hence the nodes in L are *articulation* nodes.

   Now, we assign each node in V a carefully chosen coordinate such that each node of A, B, C is within distance r to one unique center node p_0 or p_1. Note in order to have a legal partitioning, every articulation 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 articulation nodes can then be transferred back to a truth assignment for the input 3CNF-SAT instance I. Besides, we need to guarantee that the truth assignment we get 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.

3. Finally, we simply set an arbitrary positive value to r and assign each node $v \in V$ a coordinate as follows:

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

Figure 3 illustrates the deployment of nodes on the line.

Clearly the above reduction is polynomial. Next, we show that I is satisfiable if and only if $f(I) = (G, w, r)$ has a legal partitioning.
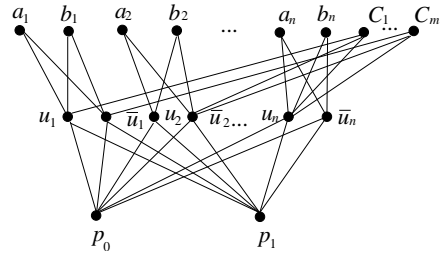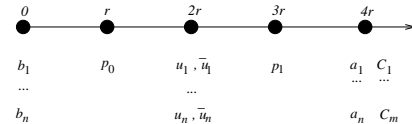

Figure 2: The constructed graph G.

Figure 3: Deployment of nodes on the line.

"$\Leftarrow$": If $f(I) = (G, w, r)$ has a legal partitioning, we can have the following simple observations:

OBSERVATION 3.1.

1. Both p_0 and p_1 must be selected as centers, otherwise some node can not be reached within 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 $i \in \{1, \ldots, n\}$, node u_i and $\overline{u}_i$ can not be in the same cluster. If node u_i and $\overline{u}_i$ are both assigned to cluster V_0 (or V_1), some node in A (or B) is not 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 3.1.3, u_i and $\overline{u}_i$ are always assigned with different values, hence the assignment is proper. Moreover, the assignment satisfies I since, by Observation 3.1.4, all the clauses are satisfied.

"$\Rightarrow$": 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 solution is valid since every node in V is within distance r from one center node, p_0 or p_1.

Finally, we note that the above proof can be easily extended to bigger 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 $d-1$ dimensions with identical value to the coordinate vector w. ∎

The internal connectedness constraint poses new challenges to the traditional k-center problem. Table 1 compares the hardness of these two problems in different settings.

	Traditional k Center	CkC
k is a constant	Polynomial Solvable	NP-complete
k is an input, $d = 1$	Polynomial Solvable	NP-complete
k is an input, $d > 1$	NP-complete	NP-complete

Table 1: Complexity results.

Remarks:

1. Theorem 3.1 also implies that the following sub-problem is NP-hard: Given a set of k centers, assign each node to one particular center to obtain k clusters which are all internally connected and the maximum cluster radius is minimized.

2. Similar to the CkC problem, one can define the connected k-median and k-means problems. In fact, the proof of Theorem 3.1 can be extended to these problems to show their NP-Completeness.

4 Approximation Algorithm

In this section we consider the min-CkC problem, the corresponding optimization version of CkC and present an approximation algorithm.

DEFINITION 4.1. (*min-CkC problem*) *Given k as the number of centers, a distance function $||\cdot||$, and a graph $G = (V, E)$ where every node in V is associated with a coordinate vector $w : V \to \mathcal{R}^d$. Find the minimum radius $r \in \mathcal{R}^+$, such that there exist disjoint partitions $\{V_1, \ldots, V_k\}$ of V, i. e. , $V = V_1 \cup \ldots \cup V_k$ and $\forall 1 \le i < j \le k, V_i \cap V_j = \phi$, which satisfy the internal connectedness constraint and radius constraint defined in Definition 3.1.*

In the following we prove an inapproximability result for the min-CkC problem, which can be viewed as a corollary of Theorem 3.1.

THEOREM 4.1. *For any $k \ge 2$, $\epsilon > 0$, the min-CkC problem is not approximable within $2-\epsilon$ unless $P = NP$.*

Proof. We only study the case for $k = 2$. We show, if there is a polynomial algorithm $\mathcal{A}$ which is guaranteed to find a feasible solution within $(2 - \epsilon)opt$, it can actually be used to solve the 3CNF-SAT problem. The reduction is similar to the proof of Theorem 3.1. First, for a given 3CNF-SAT instance I, we construct a $C2C$ instance $f(I) = (G, w, r)$ by the same procedure as in the proof of Theorem 3.1. Later, we invoke Algorithm $\mathcal{A}$ on the input (G, w, r).

Since the coordinates of all nodes are multiples of r, the optimal radius must also be a multiple of r. Hence, 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 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 min-CkC problem can not be approximated within $2 - \epsilon$. ∎

Let *opt* be the optimal radius of the above min-CkC problem. In the following we propose a polynomial algorithm which is guaranteed to find a solution within $3opt$. To prove this result, first we remove the constraint of min-CkC that each point must belong to exactly one of the clusters, and define a relaxed problem which we will refer to as min-CkC':

DEFINITION 4.2. (*min-CkC' problem*) *Given k as the number of clusters, a distance function $||\cdot||$, and a graph $G = (V, E)$ where every node in V is associated with a coordinate vector $w : V \to \mathcal{R}^d$. Find the minimum radius constraint $r \in \mathcal{R}^+$, such that there exist node sets $V_1, \ldots, V_k \subseteq V$, $V = V_1 \cup \ldots \cup V_k$, which satisfy the internal connectedness constraint and radius constraint defined in Definition 3.1.*

We first propose a polynomial procedure to solve the min-CkC' problem. Let opt' be the optimal radius of min-CkC'. Clearly $opt' \le opt$ since opt is also a feasible solution of min-CkC'. Next we present a procedure to transfer the optimal solution of min-CkC' to a solution of min-CkC with radius at most $3opt'$. Combining the two procedures we obtain an algorithm for min-CkC with approximation ratio 3.

4.1 Solving the min-CkC' problem. To solve the min-CkC' problem, we need the following notion of *reachability*:

DEFINITION 4.3. *Given a graph $G = (V, E)$, for $u, v \in V$, v is* reachable *from u (w.r.t. r) if there exists a path $u \to s_1 \to \ldots \to s_k \to v$, such that $s_1, \ldots, s_k \in V$, $(s_i, s_{i+1}) \in E$ and $\forall 1 \le i \le k, ||w(u) - w(s_i)|| \le r$.*

Intuitively, v is reachable from u w.r.t. r, if and only if v can be included in the cluster with center u

Algorithm 1 A polynomial algorithm for the min-CkC' problem.

1: Calculate all pairwise distances of V and store those distance values in set D;
2: Sort D in decreasing order;
3: **for** every value $d \in D$ **do**
4:     Enumerate all possible k centers;
5:     **for** every set of k centers $\{c_1, \ldots, c_k\} \subseteq V$ **do**
6:         Perform BFS from each center c_i and mark all nodes which are reachable from c_i w.r.t. d;
7:         **if** all vertices are marked **then**
8:             Return d and k clusters;
9:         **end if**
10:     **end for**
11: **end for**

and radius r. Clearly it can be decided in polynomial time by performing a breadth first search (BFS) for node v from node u. Based on this, we propose a simple algorithm, Algorithm 1, to solve the min-CkC' problem.

Runtime complexity: Note $|D| = \binom{n}{2}$ and we enumerate all possible k centers, hence the search space is polynomial, i.e., n^{k+2}. Besides, BFS takes time at most $O(n^2)$. Thus Algorithm 1 must terminate in $O(n^{k+4})$ steps.

4.2 Back to the min-CkC problem. Let $sol' = \{V_1', \ldots, V_k'\}$ be the clustering found by Algorithm 1 where $V_i' \subseteq V$. Note $V_1', \ldots, V_k'$ may not be disjoint. We propose the following procedure (Algorithm 2) to determine k disjoint clusters for the min-CkC problem, denoted by $sol = \{V_1, \ldots, V_k\}$. Let $c_1, \ldots, c_k$ be the centers of $V_1, \ldots, V_k$. Note that $c_1, \ldots, c_k$ are also the centers of $V_1', \ldots, V_k'$.

Algorithm 2 3-approximation algorithm for the min-CkC problem.

1: **for** i from 1 to k **do**
2:     $V_i = \phi$, $c_i \leftarrow c_i'$;
3:     Add all 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 nodes reachable w.r.t. r from v in $G[V_i']$ to the cluster of v (by performing a breadth first search from v in $G[V_i']$);
6:     **end for**
7: **end for**
8: Output clusters $V_1, \ldots, V_k$;

Algorithm 2 assigns every node in V to a unique cluster V_i for $1 \le i \le k$. For each iteration $1 \le i \le k$, line 3 allocates nodes in V_i' which have not been assigned to any previous clusters $V_1, \ldots, V_{i-1}$ to V_i. Afterwards, there may still be some unassigned nodes in V_i'. In line 5, we allocate the unassigned nodes to one of the clusters $V_1, \ldots, V_{i-1}$ from which they are reachable w.r.t. r.
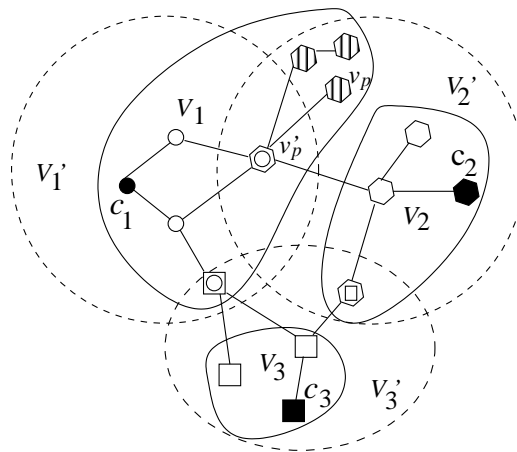

Figure 4: Demonstration of Algorithm 2.

Algorithm 2 is illustrated in Figure 4. The circles with dashed lines represent three initial clusters (with overlapping) V_1', V_2' and V_3' generated by Algorithm 1. Applying Algorithm 2, we obtain three new clusters V_1, V_2 and V_3 that are disjoint (The center nodes do not move).

LEMMA 4.1. *Algorithm 2 is guaranteed to find a solution of the min-CkC problem with maximum radius at most $3opt'$.*

Proof. First we show that Algorithm 2 assigns every node $u \in V$ to a unique cluster. There are two cases. In case 1, u can be reached via a path from the 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 2. 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 2, 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 opt'. In case 2, observe that the maximum distance between u and v is at most $2opt'$ due to the triangle inequality and the fact 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 opt'. Therefore, again by the triangle inequality, the distance between u and its corresponding center node is at most $3opt'$. ∎

THEOREM 4.2. *Combining algorithm 1 and 2 gives a polynomial 3-approximation algorithm for the min-CkC problem.*

Remark: Note the approximation results rely on the triangle inequality; thus, to make the approximation results valid, the distance function for min-CkC has to be metric. However, for the NP-completeness proof, the distance function can be non-metric.

5 Heuristic Algorithm

The complexity analysis has demonstrated the hardness of the CkC problem. Moreover, Theorem 3.1 implies that even the assignment step alone, i.e., given k centers finding the optimal assignment of the remaining nodes to minimize the radius, is NP-hard. While providing an algorithm with guaranteed performance is important theoretically, the expensive enumeration operation prevents it from being practical in large datasets. In this section, we propose **NetScan**, an efficient heuristic algorithm.

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

- Step I: Randomly pick k initial cluster centers.

- Step II: Assign all 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 from a group of n nodes requires $O(n^2)$ time. To make it more scalable for large datasets, 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 steps look similar to the k-means algorithm. However, the complexity analysis tells us that given k centers, finding an optimal assignment requires a search through an exponential space, which is unacceptable in practice. Thus, the major challenge of NetScan is finding a good membership assignment, i.e., step II.

From the design principle of the approximation algorithm, we observe that the BFS-based approach provides an efficient way to generate clusters without violating the internal connectedness constraint. Inspired by this observation, we start the membership assignment from the center. Neighboring nodes (directly connected by an edge of the graph) of already assigned nodes are gradually absorbed to the cluster. The search

starts from cluster centers with respect to an initial radius threshold R_0. Nodes are tested and assigned to the first cluster where their distances to the center are no larger than the current radius threshold R_i. If all the centers have been processed with R_i and not all nodes have been assigned, the assignment is resumed in the next *radius increment round* with a larger radius threshold R_{i+1}. The pseudo code of step II is provided in Algorithm 3, and more detailed aspects of NetScan are discussed later. A running example is illustrated in Figure 5 with the assignment sequence given in Table 2.

Algorithm 3 Step II of NetScan.

1: Empty working queue Q;
2: **for** every center c_j of cluster C_j **do**
3:     Append all unassigned neighbors of c_j to Q;
4:     **while** Q is not empty **do**
5:         Pop the first element q from Q;
6:         **if** $||q - c_j|| \leq R_i$ **then**
7:             **if** q is a potential articulation node **then**
8:                 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;
9:             **else**
10:                Assign q to C_j and append q's unassigned neighbors to Q;
11:             **end if**
12:         **end if**
13:     **end while**
14: **end for**
15: **if** all nodes are assigned to some C_j **then**
16:     Stop;
17: **else**
18:     Increase R_i and goto 1;
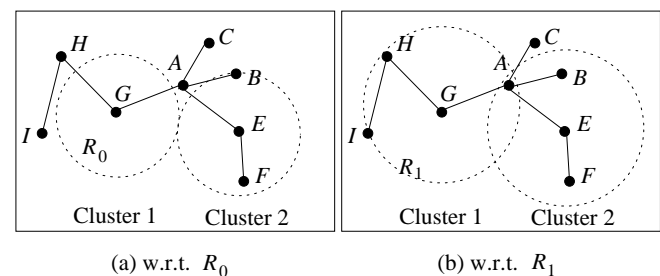19: **end if**

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

Figure 5: NetScan demonstration.

Why gradually increment R_i? The radius threshold R_i plays an important role in minimizing the maximum

	Cluster 1	Cluster 2
R_0	$\{G\}$	$\{E, F\}$
R_1	$\{G, H, I\}$	$\{E, F, A, B, C\}$

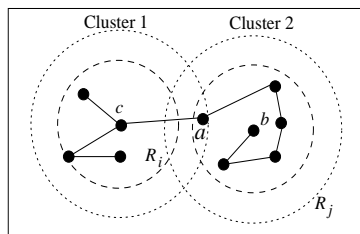Table 2: Node assignment w.r.t. R_0 and R_1.
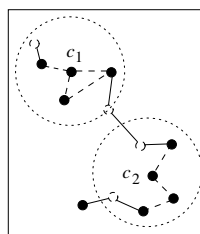

Figure 6: Radius increment.   Figure 7:   Runtime.

radius of resulting clusters. Figure 6 demonstrates an example where a larger radius threshold R_j allows node a to be assigned to cluster 1, leading to a larger radius of cluster 1. Instead, if we have a smaller radius 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 radii, which may greatly increase the runtime. As a trade-off, we propose the increment to be the average pairwise distance of nodes.

How to choose R_i? Algorithm 2 shows that the nodes located in the overlapping area of two clusters w.r.t. a given radius constraint are the source of the difficulty in the assignment. Thus, to start with, we choose R_0 to be half of the smallest distance among all pairs of cluster centers. This chosen R_0 does not create overlap that introduces any ambiguity in the node assignments, thus reducing the problem size.

Most likely the initial radius threshold R_0 can not make all nodes assigned. We need to increase the radius threshold to allow the BFS continue until all nodes are assigned. 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 R_{i+1} makes it likely that at least some further nodes can be assigned in the next round to each of the cluster. $\overline{D}$ can be obtained efficiently by drawing a small set of samples and calculating the average distance of the samples.

How to assign nodes to clusters? Most nodes are assigned based solely on their distance 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. Similar to the concept of articulation nodes introduced in Section 3, we call these nodes *potential articulation nodes* in NetScan. We assign potential articulation nodes not only based on their distances to the different cluster centers, but also their neighborhood situations. For example, in Figure 5 (b), A is a potential articulation node and its assignment affects 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 an articulation node depends on two factors: 1) the node has neighbors who have been assigned membership and those neighbors are from more than one cluster, e.g., C_i, C_j. 2) the node is within R_i distance from both centers of C_i and C_j.

We propose the following look-ahead approach for the cluster assignment of potential articulation nodes. For the sake of efficiency, for each articulation node, we only check its unassigned neighbors (if any) which have a degree of 1, the *unary* neighbors. The membership assignment decision is made mainly based on the unary neighbors. An articulation node is assigned to its closest center unless the node has a direct unary neighbor which is closer to other centers. 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 as well. However, this would significantly increase the runtime without guaranteed 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, giving up few customers to meet most customers' preference is acceptable. Hence, we propose an optional step, which utilizes a graphical approach to eliminate "outliers" from the solution of the CkC problem. Every node remembers the radius threshold (called assignment radius) at which it is assigned. We sort all nodes by their assignment radius and filter out the node (and its following nodes) which causes a sudden increase of the radius. The "cutoff" point can be determined from the assignment radius chart, either by manual inspection or automatic detection. Figure 8 illustrates an example where part (a) shows an input graph and part (b) depicts its corresponding assignment radius chart. In this case, only f would be removed as an outlier.

Runtime complexity: In each iteration of step II and III, the NetScan algorithm generates k clusters
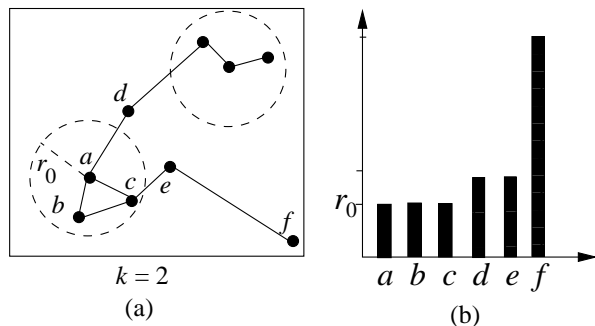
Figure 8: Outlier identification.

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. As in the k-means algorithm, we assume the NetScan algorithm converges in 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. The worst case rarely happens, in which all the edges are connected to the frontier nodes. For example, the dashed edges in Figure 7 do not have to be considered in the next radius increment round. In the figure, the frontier nodes are dashed. In real cases, the number of edges visited in one iteration can be reasonably assumed to be $O(|E|)$.

6 Experimental Results

In this section, we demonstrate the meaningfulness of our CkC clustering model on a small real dataset and show the efficiency of the NetScan algorithm using synthetic datasets.

6.1 Real dataset. The real dataset includes 50 professors from three major computer science communities: theory, databases and machine learning. The attributes of each professor were collected from his/her homepage representing the keyword frequencies of his/her research interests. The relationship data is a connected subgraph

extracted from the DBLP [4] coauthorship network. 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 professor'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.
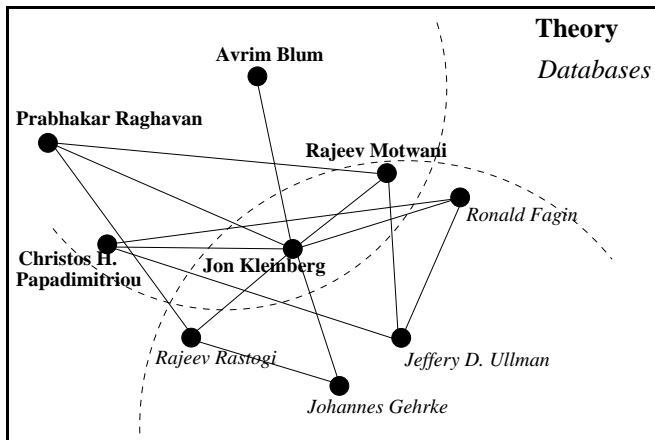

Figure 9: NetScan on real data.

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) [15] for the traditional k-center problem, both with $k = 3$. Table 3 reports the clustering results averaged over 20 runs for both algorithms, recording the number of correctly identified professors 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 significantly 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
Total	50	27	36
Accuracy		54%	72%

Table 3: Comparison of NetScan and Greedy k-center.
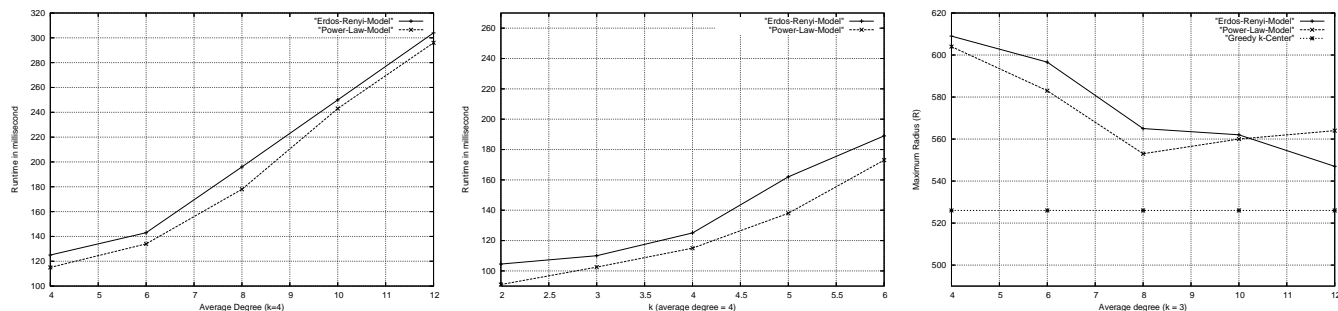
The main reason why NetScan significantly outper-

Figure 10: NetScan on synthetic data. (a) runtime vs. average degree. (b) runtime vs. k. (c) maximum radius vs. average degree.

forms Greedy k-center is that both relationship and attribute data make contributions in the clustering process, and considering only one data type can mislead the clustering algorithm. For example, Jon 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 (see Figure 9), which is a better match for his overall research profile. On the other hand, Jeffery D. Ullman has broad coauthorship connections, which alone cannot be used to confidently identify his community membership. However, he claims his research interest in databases exclusively, and NetScan clustered him into the database community as expected.

6.2   Synthetic datasets. Due to the lack of publicly available databases with both attribute and relationship data, we used synthetic data for the efficiency evaluation. Based on the complexity analysis in Section 5, the runtime is directly related to the number of edges instead of the number of nodes, thus we can fix the number of nodes and vary the degree of each node to evaluate the efficiency of NetScan. We took the UCI PIMA dataset [5], which is also used in [8] to evaluate the quality of k-means algorithm. Since the PIMA dataset contains only numeric attribute data, we automatically generated relationship data based on two random graph models, Erdös-Rényi model [9] and Power-law graph model [2]. In the Erdös-Rényi model, every pair of nodes is connected with the same probability. In the Power-law graph model, 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 op-

erating system.

We study the effect of average number of edges on runtime. Figure 10 (a) shows the results on the average runtime over 50 restarts for both models. The average degree was set from 4 since with smaller degree the data generator often failed to generate a connected network. Although the worst case runtime complexity analysis of the NetScan algorithm demonstrates that the runtime is related to k, $|E|$ and the number of iterations, our results shows that the runtime is far from proportional to both k and $|E|$ on average.

We also evaluate the effect of k on runtime, which is shown in Figure 10 (b). As we expected, the increase of k does not cause a proportional increase in the runtime.

In addition, it is interesting to evaluate how the input graph is related to the maximum radius R of the clustering results. Intuitively, the more edges exist in the graph, the smaller R would be. The traditional k-center problem can be considered as a special case of the CkC problem with a complete input graph. The optimal R for traditional k-center is smaller than the optimal R for CkC. The results in Figure 10 (c) show that R decreases with the increase of average degree.

7   Conclusion

Existing cluster analysis methods are either targeting attribute data or relationship data. However, in scenarios where these two data types contain orthogonal information, a joint cluster analysis of both promises to achieve more accurate results. In this paper, we introduced the novel Connected k-Center problem, which takes into account attribute data as well as relationship data. We proved the NP-completeness of this problem and presented a corresponding 3-approximation algorithm. To improve the scalability, we also developed an efficient heuristic algorithm NetScan. Our experimental evaluation using a real dataset for community identification demonstrated the meaningfulness of the

NetScan results and the accuracy gain compared to the classical k-center clustering algorithm. Tests on synthetic datasets showed the efficiency of our algorithm.

Under the framework of joint cluster analysis of attribute data and relationship data, the CkC problem can be extended in many ways. Firstly, to better model real applications with varied requirement subtleties, more practical clustering criteria (such as k-means) can be considered other than the k-center criterion, which allowed us to provide theoretical analysis in this paper. Secondly, similarly motivated, the internal connectedness constraint can as well be replaced by specifications on any property or combination of properties of graphs; e.g., length of paths, degree of vertices, and connectivity etc. Thirdly, the relations can be non-binary and the edges can be weighted to indicate the degree of relationship; e.g., friendship can go from intimate and close to just nodding acquaintanceship. Also, the relations can be non-symmetric; e.g., citation relations between documents. Finally, we believe that with the increasing availability of attribute data and relationship data, data analysis in general, not only cluster analysis, will benefit from the combined consideration of both data types.

Acknowledgement

References

[1] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. In *SIGMOD*, 1999.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] S. Basu, M. Bilenko, and R. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, 2004.

[4] C. S. Bibliography. *DBLP. http://www.informatik.uni-trier.de/ ley/db/index.html*. 2005.

[5] C. Blake and C. Merz. UCI repository of machine learning databases., 1998.

[6] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *11th Europ. Symp. Algorithms*, pages 568–579, 2003.

[7] P. Brucker. *On the complexity of clustering problems, in Optimization and Operations Research*. Springer-Verlag, 1977.

[8] I. Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *SDM*, 2005.

[9] P. Erdos and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[11] S. Guha, R. Rastogi, and K. Shim. Rock: a robust clustering algorithm for categorical attributes. In *ICDE*, 1999.

[12] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2001.

[13] R. A. Hanneman and M. Riddle. *Introduction to social network methods*. http://faculty.ucr.edu/ hanneman/, 2005.

[14] E. Hartuv and R. Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76:175–181, 2000.

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

[16] D. Iacobucci. *Networks in marketing*. Sage Publications, 1996.

[17] A. Jain and R. Dubes. *Algorithms for clustering data*. Prentice Hall, 1988.

[18] G. Karypis, E. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE COMPUTER*, 32:68–75, 1999.

[19] L. Kaufman and P. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 1990.

[20] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. prob.*, pages 281–297, 1967.

[21] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13(1):182–196, 1984.

[22] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, 1994.

[23] J. Scott. *Social Network Analysis: A handbook*. Sage, London, 2000.

[24] C. Toregas, R. Swan, C. Revelle, and L. Bergman. The location of emergency service facilities. *Oper. Res.*, 19:1363–1373, 1971.

[25] A. K. H. Tung, R. T. Ng, L. V. S. Lakshmanan, and J. Han. Constraint-based clustering in large databases. In *ICDT*, pages 405–419, 2001.

[26] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge Univ. Press, 1994.

[27] C. Webster and P. Morrison. Network analysis in marketing. *Australasian Marketing Journal*, 12(2):8–18, 2004.