Joint Cluster Analysis of Attribute Data and Relationship Data: The Connected k-Center Problem, Algorithms and Applications

RONG GE, MARTIN ESTER, BYRON J. GAO, ZENGJIAN HU,
and BINAY BHATTACHARYA
Simon Fraser University
and
BOAZ BEN-MOSHE
Ariel University Center

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 article, we introduce the novel Connected k-Center (CkC) problem, a clustering model taking into account attribute data as well as relationship data. We analyze the complexity of the problem and prove its NP-hardness. Therefore, we analyze the approximability of the problem and also present a constant factor approximation algorithm. For the special case of the CkC problem where the relationship data form a tree structure, we propose a dynamic programming method giving an optimal solution in polynomial time. We further present NetScan, a heuristic algorithm that is efficient and effective for large real databases. Our extensive experimental evaluation on real datasets demonstrates the meaningfulness and accuracy of the NetScan results.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Algorithms, Management, Performance

A short version of this article appeared in *Proceedings of the 6th SIAM International Conference on Data Mining*. SIAM, Philadelphia, PA, 2006.
This work was supported in part by NSERC Discovery grant 250960-06.
This work was done while B. Ben-Moshe was at Simon Fraser University as a post doctor fellow.
Authors' addresses: R. Ge, M. Ester (contact author), B. J. Gao, Z. Hu, and B. Bhattacharya, School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC, V5A 1S6, Canada; B. Ben-Moshe, Department of Computer Science, Ariel University Center, Ariel, 44837 Israel; contact email: ester@cs.sfu.ca.

Additional Key Words and Phrases: Attribute data, relationship data, joint cluster analysis, NP-hardness, approximation algorithms, community identification, document clustering, market segmentation

ACM Reference Format:

Ge, R., Ester, M., Gao, B. J., Hu, Z., Bhattacharya, B., and Ben-Moshe, B. 2008. Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. ACM Trans. Knowl. Discov. Data. 2, 2, Article 7 (July 2008), 35 pages. DOI = 10.1145/1376815.1376816 http://doi.acm.org/10.1145/1376815.1376816

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. 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 [Wasserman and Faust 1994; Scott 2000; Webster and Morrison 2004] has been gaining popularity in the study of marketing, community identification, epidemiology, 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 entities 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 deduced from or closely approximated by the other, a separate analysis on either is sufficient. For example, the classical facility location problem [Toregas et al. 1971] only manipulates locations (attributes) since the Euclidean distance between A and B can be used to well approximate their reachability via the road network (relations).

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 [Wasserman and Faust 1994; Scott 2000].

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 article, 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, that is, 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 centers such that the maximum distance of any entity to its closest cluster center, the radius of the cluster, is minimized.

1.1 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 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 [Iacobucci 1996]. 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. 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 the task [Wasserman and Faust 1994]. 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 [Scott 2000; Hanneman and Riddle 2005]. 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 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.

1.2 Contributions and Outline

This article makes the following contributions:

(1) We advocate joint cluster analysis of attribute data and relationship data and introduce the novel *CkC* clustering problem.
(2) We analyze the complexity of the *CkC* problem, and prove its NP-hardness.
(3) We analyze the approximability of the *CkC* problem and propose a constant factor approximation algorithm.
(4) 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.
(5) 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.
(6) We perform an extensive experimental evaluation on three real life datasets, demonstrating the meaningfulness and accuracy of the NetScan results.

The rest of the article 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 provide a constant factor approximation algorithm for the *CkC* problem. Section 5 studies the *CkC* problem for the special case of tree-structured relationship data. To achieve better scalability, in Section 6, we present the efficient heuristic algorithm NetScan. We report experimental results in Section 7 and conclude the article in Section 8.

2. RELATED WORK

Clustering has been widely studied in the mathematics, statistics, and computer science literatures for decades.

2.1 Theory and Algorithms

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 problem [Dyer and Frieze 1985; Hochbaum and Shmoys 1985; Gonzalez 1985; Feder and Greene 1988; Agarwal and Procopiuc 2002], the k-Median problem [Kariv and Hakimi 1979; Lin and Vitter 1992; Charikar et al. 1999; Jain and Vazirani 2001], the min-diameter problem (pairwise clustering) [Brucker 1977], the min-sum problem [Bartal et al. 2001; Guttman-Beck and Hassin 1998], the min-sum of diameters (or radii) problem [Doddi et al. 2000; Charikar and Panigrahy 2004], and the k-Means problem [Steinhaus 1956; Lloyd 1982]. These problems minimize 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 points to cluster centers), respectively.

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-Complete for d (dimensionality) ≥ 2 and arbitrary k [Megiddo and Supowit 1984]. In the case of $d = 1$, the Euclidean k-Center problem is polynomially solvable using dynamic programming techniques [Megiddo et al. 1981; Frederickson and Johnson 1979]. For $d \geq 2$ and fixed k, 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 $d = 1$. In this sense, the CkC problem is harder than the Euclidean k-Center problem. It is NP-hard to approximate the k-Center problem for $d \geq 2$ within a factor smaller than 2 even under the L_∞ metric [Feder and Greene 1988]. Hochbaum and Shmoys [1985] give a 2-approximation greedy algorithm for the k-Center problem in any metric Feder and Greene [1988] also give a 2-approximation algorithm but improve the running time to $O(n \log k)$.

Many of the above-mentioned clustering models are closely related to the general facility location problem [Toregas et al. 1971], which has been extensively studied in the operations research literature. Given a set of facilities and a set of customers, the problem is to decide which facilities should be opened and which customers should be served from which facilities so as to minimize the total cost of serving all the customers. Note that the recently studied Connected k-Median problem [Swamy and Kumar 2004] is not 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 the theoretical perspective, they have no intention to perform joint cluster analysis, and they do not require clusters to be cohesive with respect to both attribute data and relationship data.

2.2 Data Mining

In the data mining community, clustering research emphasizes more on real life applications and development of efficient and scalable algorithms. Existing methods roughly fall into several categories, including partitioning methods such as k-Means [MacQueen 1967], k-Medoids [Kaufman and Rousseeuw 1990], and CLARANS [Ng and Han 1994], hierarchical methods such as AGNES and DIANA [Kaufman and Rousseeuw 1990], and density-based methods such as DBSCAN [Ester et al. 1996] and OPTICS [Ankerst et al. 1999]. These clustering methods, in general, take only attribute data into consideration.

While almost all clustering algorithms 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. [2001] present 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 paper, 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.

We have introduced the Connected k-Center problem in Ester et al. [2006] to jointly analyze attribute data and relationship data. The current contribution is a major extension of this study, including new theoretical results and algorithms for tree-structured relationship data. In addition, we improve the NetScan algorithm and experiment with several larger real-life datasets. In Moser et al. [2007], we have investigated the Connected X Cluster (CXC) problem, as an extension to the CkC model, for joint cluster analysis without a-priori specification of the number of clusters.

2.3 Social Network Analysis and Graph Clustering

Recently, the increasing availability of relationship data has stimulated research on network analysis [Wasserman and Faust 1994; Scott 2000; Hanneman and Riddle 2005]. Clustering methods for network analysis are mostly graph-based, separating sparsely connected dense subgraphs from each other as in Brandes et al. [2003]. 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, for example, *normalized cut* [Shi and Malik 2000] and *ratio cut* [Chan et al. 1994], have been well studied. Those graph clustering problems can be effectively solved by spectral methods that make use of eigenvectors. Recently, Dhillon et al. [2007] 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 Dhillon et al. [2007] 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 [Shi and Malik 2000]. 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 matrixes, which are derived from attribute data. A similarity graph can be a complete graph as in the agglomerative hierarchical clustering algorithms, for example, single-link, complete link, and average link [Jain and Dubes 1988], or incomplete retaining those edges whose corresponding similarity is above a threshold [Guha et al. 1999; Hartuv and Shamir 2000]. CHAMELEON [Karypis et al. 1999] 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 attribute data of two connected nodes to a single, relative distance value. In CkC, attribute data are used directly, avoiding information loss.

2.4 Constraint-Based and Semi-Supervised Clustering

Joint cluster analysis 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 [Tung et al. 2001]. 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 belong to the same cluster, those with a cannot-link should be assigned to different clusters. Basu et al. [2004] propose a probabilistic framework based on Hidden Markov Random Fields(HMRF), incorporating supervision into k-clustering algorithms. Kulis et al. [2005] show that the objective function of the HMRF-based semi-supervised clustering model, as well as which of some graph clustering models, can be expressed as special cases of weighted kernel k-Means objective. Based on these theoretical connections, a unified algorithm is proposed to perform semi-supervised clustering on data given either as vectors or as a graph. Davidson and Ravi [2005] consider 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 approach under the framework of constraint-based 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.

2.5 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. [2002] propose 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. [2003] introduce 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 [2007] present 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, that is, not every gene needs to be assigned to a cluster.

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.

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 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.

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 article, 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 3.1 (*CkC Problem*). Given an integer k, a graph $G = (V, E)$, a function $w : V \to \mathcal{R}^d$ 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, that is, $V_1 \cup \cdots \cup V_k = V$ and $\forall 1 \leq i < j \leq k, V_i \cap V_j = \phi$, such that the partitions satisfy the *internal connectedness constraint*, that is, 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, for example, 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.

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 section of complexity analysis, the names of the problems refer to their decision versions.

Definition 3.2 (*CkC Problem, Decision Version*). Given an integer k, a graph $G = (V, E)$, a function $w : V \to \mathcal{R}^d$ 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, that is, $V_1 \cup \cdots \cup V_k = V$ 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*, that is, $\forall 1 \leq i \leq k$, 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, that is, 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 that 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. By intuition, 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 3.3 (*3SAT Problem*). Given a set $U = \{u_1, \ldots, u_n\}$ of variables, a boolean formula $I = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \cdots \wedge \mathcal{C}_m$ where each clause $\mathcal{C}_i = l_i^1 \vee l_i^2 \vee l_i^3$ 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 3.4. *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 $I = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \cdots \wedge \mathcal{C}_m$, we construct a $C2C$ instance $f(I) = (G, w, r)$, where $G = (V, E)$ is the underlying graph, $w : V \to R$ 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 $V = P \cup L \cup \mathcal{C} \cup A \cup B$. $P = \{p_0, p_1\}$ where p_0 and p_1 are two center nodes. L and $\mathcal{C}$ 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.
(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 $\mathcal{C}_i \in \mathcal{C}$, we link l to $\mathcal{C}_i$ if $l \in \mathcal{C}_i$. For each $i \in \{1, 2, \ldots, n\}$, we link a_i and b_i to both u_i and $\overline{u}_i$.

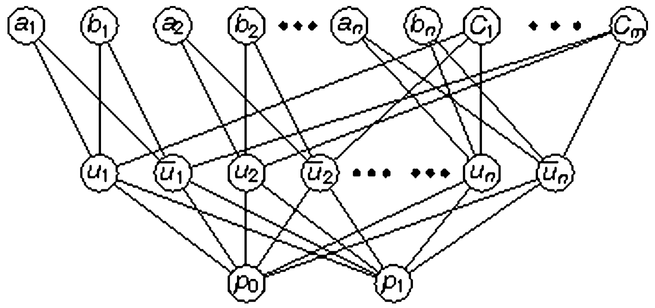

Fig. 1. Constructed graph G. © 2006 SIAM (Reprinted with permission)

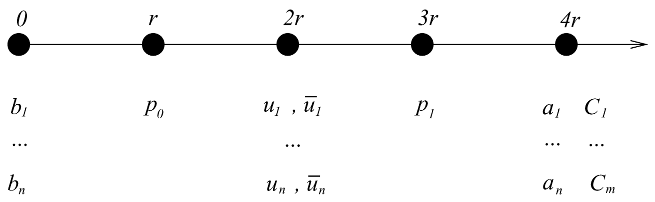

Fig. 2. Deployment of nodes on the line. © 2006 SIAM (Reprinted with permission)

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

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

Steps (1) and (2) construct the underlying graph G. A visual explanation of the construction method is provided in Figure 1. Note that every node in $A, B, \mathcal{C}$ 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 $A, B, \mathcal{C}$ is within distance r to one unique center node p_0 or p_1. Figure 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, that is, $\forall i \in \{1, 2, \ldots, n\}$, 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.

Table I. Complexity Results

	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

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 can not be reached within distance r.
(2) For the same reason, each node in A and $\mathcal{C}$ 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 $\overline{u}_i$ can not 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 $\mathcal{C}_i \in \mathcal{C}$, there must be at least one literal assigned to cluster V_1, otherwise $\mathcal{C}_i$ 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:

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

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. □

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

Remarks

(1) Theorem 3.4 implies the CkC problem defined in Definition 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 3.4 can be extended to these problems to show their NP-Completeness.

4. APPROXIMATION ALGORITHMS

In this section, we study the CkC problem defined in Definition 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, that is, 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 can not be closed within 2.64.

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 3.4.

THEOREM 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 3.4.

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 3.4. 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 3.4. 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 3.4, 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 $2-\epsilon$. □

4.2 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 3 remains valid even for nonmetric 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 essentially 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.

Algorithm 1: 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 $low \leq high$
5:      $middle = (low + high)/2$;
6:      $r = R[middle]$;
7:      for each set of $k$ centers $\{c_1, \ldots, c_k\} \subseteq V$
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
10:          if $low = high$
11:               Return $r$ and the $k$ clusters;
12:          else
13:               $high = middle - 1$;
14:     else
15:           $low = middle + 1$;
```

Fig. 3. Polynomial exact algorithm for CkC'. © 2006 SIAM (Reprinted with permission)

Definition 4.2 (*CkC' Problem*). Given an integer k, a graph $G = (V, E)$, a function $w : V \to \mathcal{R}^d$ 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 $V_1 \cup \cdots \cup V_k = V$, such that the node sets satisfy the internal connectedness constraint and the maximum radius defined on $||\cdot||$ is minimized.

Complexity of CkC'. If k is treated as part of the input, CkC' is NP-hard as it is an extension of the traditional k-center problem. On the contrary, if k is fixed as a constant, CkC' is in P (justification follows).

4.2.1 *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 4.3. Let $G = (V, E)$, for $u, v \in V$, v is *reachable* from u with respect to r, $r \in \mathcal{R}^+$, if there exists a path $p : \{u = s_0 \to s_1 \to \cdots \to s_l \to s_{l+1} = v\}$, $s_1, \ldots, s_l \in V$, 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 with respect to 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 1 (Figure 3), which returns the optimal solution for CkC' in polynomial time.

Runtime Complexity. Algorithm 1 (Figure 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 1 is $O(n^{k+2} \log n)$.

Algorithm 2: Algorithm converting a solution of CkC' to a solution of CkC

```
1: for $i$ from 1 to $k$
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 $v \in \left(\cup_{j=1}^{i-1} V_j\right) \cap V'_i$
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 $V_1, \ldots, V_k$;
```

Fig. 4. Algorithm converting a solution of CkC' to a solution of CkC. © 2006 SIAM (Reprinted with permission)

4.2.2 *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 4.4. Let G, u, v, p be defined as in Definition 4.3. The distance between u and v with respect to 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, that is, $\forall u, v \in V, dist(u, v) = dist(v, u)$. It also satisfies the triangle inequality, that is, $\forall u, v, s \in V, dist(u, s) \leq dist(u, v) + dist(v, s)$. 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 Hochbaum and Shmoys [1985] 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.

4.2.3 *Back to CkC.* In Algorithm 2 (Figure 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 2 gives approximations for the CkC problem.

Let $\{V'_1, \ldots, V'_k\}$ be a clustering returned by Algorithm 1 (Figure 3) or the approximation algorithm specified in Section 4.2.2 where $V'_i \subseteq V$ and the node sets (clusters) $V'_1, \ldots, V'_k$ may not be disjoint. Algorithm 2 (Figure 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 2 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 5 provides an illustration for Algorithm 2. The circles with dashed lines represent the three initial (overlapping) clusters V'_1, V'_2 and V'_3 generated

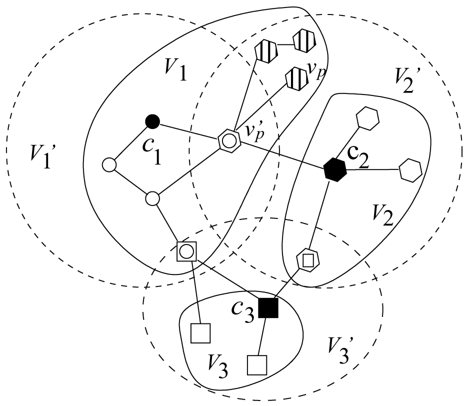

Fig. 5. Illustration of Algorithm 2. © 2006 SIAM (Reprinted with permission)

by Algorithm 1 (Figure 3). Applying Algorithm 2 (Figure 4), we obtain three new disjoint clusters V_1, V_2 and V_3. The center nodes were not changed.

LEMMA 4.5. *Let r be the maximum radius associated with a feasible solution for the CkC' problem. Algorithm 2 (Figure 4) is guaranteed to find a feasible solution for the CkC problem with maximum radius at most $3r$.*

PROOF. First we show that Algorithm 2 (Figure 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 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 r. In case 2, observe that the distance between u and v is at most $2r$ 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 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 4.6. *Combining Algorithm 1 (Figure 3) and Algorithm 2 (Figure 4) gives a polynomial 3-approximation for the CkC problem for fixed k.*

THEOREM 4.7. *Combining the approach proposed in 4.2.2 and Algorithm 2 (Figure 4) gives a polynomial 6-approximation for the CkC problem for arbitrary k.*

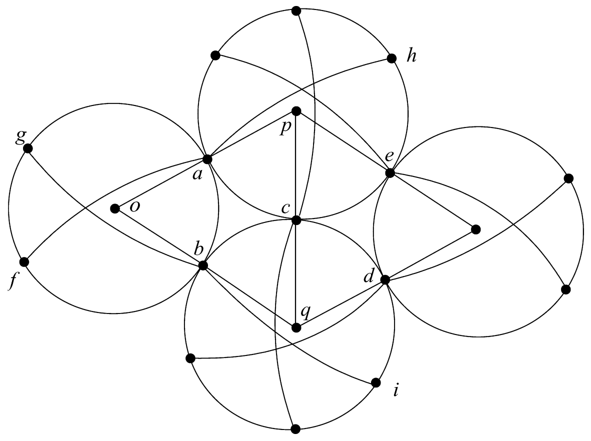

Fig. 6. Lower bound for the gap between CkC and CkC'.

Table II. Approximability Results

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

4.2.4 *Lower Bound for the Gap between CkC and CkC′.* In Lemma 4.5, 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 6, 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 $\sqrt{7}r \approx 2.64r$. We omit the mathematical details.

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

5. EXACT ALGORITHM FOR *CkC* ON TREES

As shown in Theorem 3.4, 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, for example, tree graphs. Tree structure is exhibited in common *organization charts*, which graphically illustrate how authority and responsibility are distributed within organizations. Although organization charts

Algorithm 3: 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 $low \leq high$
5:     $middle = (low + high)/2$;
6:     $r = R[middle]$;
7:     Call the dynamic programming algorithm specified in section 5.2 on
       the decision version of $CkC$ with $r$ as the radius threshold;
8:     if the dynamic programming algorithm returns "yes"
9:         if $low = high$
10:            Construct the $k$-partitioning from the dynamic programming table
               and return $r$;
11:        else
12:            $high = middle - 1$;
13:    else
14:        $low = middle + 1$;
```

Fig. 7. Polynomial exact algorithm for CkC on trees.

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.

5.1 Polynomial Exact Algorithm for CkC on Trees

Algorithm 3 (Figure 7) 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 in V and sorts them in increasing order. Then we perform a binary search to find the smallest distance that is feasible. The feasibility can be decided by invoking a dynamic programming procedure presented as follows.

5.2 Dynamic Programming Algorithm

The dynamic programming algorithm solves the decision version of the CkC problem defined in Definition 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 Tamir [1996]. 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

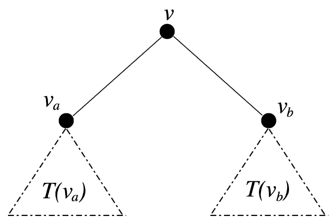

Fig. 8. The tree structure.

number of clusters in a feasible clustering of $T(v_i)$ with respect to 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.

Fix some arbitrary node v where the f values of all of its descendants have been computed, Figure 8 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 can not 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 3 is $O(n^2 \log n)$.

6. HEURISTIC ALGORITHM

The complexity analysis has demonstrated the hardness of the general case of the *CkC* problem. Meanwhile, Algorithm 2 provides a way to guarantee the performance. Moreover, Theorem 3.4 implies that even the assignment step alone, that is, 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.

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 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 3.2. Thus, the major challenge of NetScan is finding a good membership assignment, that is, 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

Algorithm 4: Step II of NetScan

```
1: $R_i = R_0$;
2: for every center $c_j$ of cluster $C_j$
3:     Empty working queue $Q$;
4:     Append all unassigned nodes that are directly reachable from nodes in $C_j$ to $Q$;
5:     while $Q$ is not empty
6:         Pop the first element $q$ from $Q$;
7:         if $||q - c_j|| \leq R_i$
8:             if $q$ is a potential bridge node
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 nodes are assigned to some $C_j$
13:     Stop;
14: else
15:     Increment $R_i$ and goto 2
```

Fig. 9. Step II of NetScan. © 2006 SIAM (Reprinted with permission)

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 10. In the example, 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, b, and c are assigned to cluster 2. The pseudocode of Step II is given in Algorithm 4 (Figure 9), and more details of NetScan will be discussed shortly.

6.2 More Details on NetScan

6.2.1 *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, that is, *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, that is, the 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.

6.2.2 *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. 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 2 (Figure 4) suggests that the nodes located in the overlapping area of two clusters with respect to 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.

6.2.3 *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 with respect to R_i. Inspired by the concept of bridge nodes introduced in Section 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 10, 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.

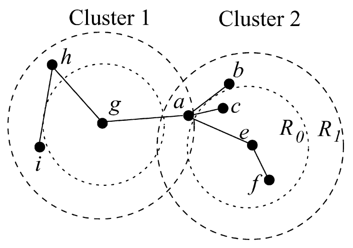

Fig. 10. Node assignment in NetScan.

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, for example, C_i and C_j, (2) the node is within R_i distance from both centers of C_i and C_j, and (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 mainly based on the unary neighbors. A potential bridge node is assigned to its closest center unless the node has a direct unary neighbor that 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.

6.2.4 *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 11. 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

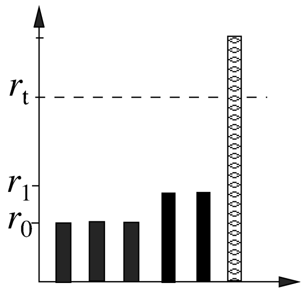

Fig. 11. Outlier elimination.

the amount of radius increment, the total number of radius increases in one iteration is a constant, $\frac{diam}{D}$, 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 $O(k|E|\frac{diam}{D})$, 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 $O(tk|E|\frac{diam}{D})$.

However, in each iteration, we only need to consider those edges connecting to the nodes in the *frontier*, that is, a set of unassigned nodes that are direct neighbors of the assigned nodes. The worst case rarely happens, in which all the edges are connected to the frontier nodes. 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.

6.3 Adaptation of NetScan to the Connected k-Means Problem

As we have discussed in related work (Section 2), various clustering problems can be formulated depending on different objectives. The well-known k-Means problem [Steinhaus 1956; Lloyd 1982] minimizes the compactness, that is, the sum of squared distances from data points to their corresponding cluster centers. The corresponding k-Means algorithm [MacQueen 1967] 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.

Table III. Overview of the Real Datasets

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

7. EXPERIMENTAL RESULTS

In this section, we demonstrate the meaningfulness and accuracy of our joint cluster analysis model on three real life datasets in applications for community identification, document clustering and gene clustering.

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 [Ulitsky and Shamir 2007] 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 [DBLP] 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 [Spellman et al. 1998], while the relationship data was extracted from the protein interaction network of Saccharomyces Cerevisiae, which was downloaded from the BioGRID database [Stark et al. 2006]. 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 III.

Comparison Partners. Related work, as discussed in Section 2, can be categorized as partitioning vs. overlapping, distance-based vs. probabilistic, and unsupervised vs. 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,[1] 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 [Dhillon et al. 2007] on an important graph clustering problem, the normalized cut [Shi and Malik 2000], which has been shown to be effective for discovering meaningful clusters in several real life applications [Dhillon et al. 2007].

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 [Steinbach et al. 2000] utilizing only the attribute data. We applied our adapted NetScan (for the Connected k-Means problem, see Section 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 [Ulitsky and Shamir 2007]. 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.

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 [DBLP] 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) [Hochbaum and Shmoys 1985] 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 IV 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

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

Table IV. Comparison of NetScan and Greedy k-Center on Dataset DBLP I

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%

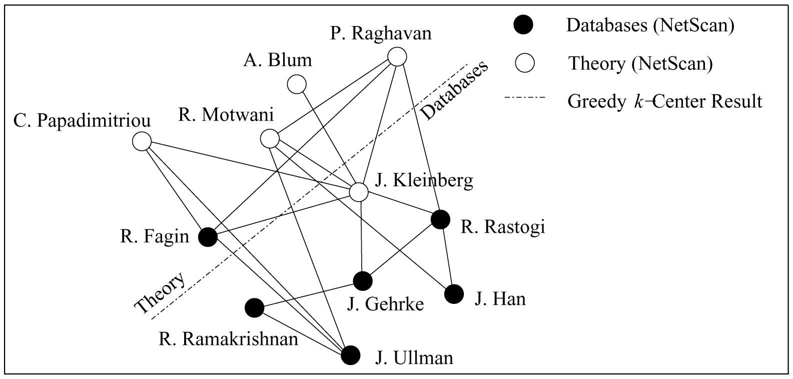

Fig. 12. Partial clustering results on Dataset DBLP I.

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.

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 12 illustrates the differences 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.

Table V. Summaries of Dataset DBLP II

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

Table VI. Comparison of NetScan, GraClus and k-Means on Dataset DBLP II

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%

7.3 DBLP Dataset II: Clustering Papers

The second real dataset[2] was constructed for document clustering. As summarized in Table V, the dataset includes 1436 selected papers from DBLP [DBLP] 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 [2006]. 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 [DBLP]. 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 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 and Malik [2000]. In order to measure the accuracy, a cluster is labeled by a majority vote of its members. Table VI 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.

To illustrate the benefits of joint cluster analysis, we present in Figure 13 a small portion of the DBLP II dataset and the clusterings produced by k-Means

[2]The dataset can be downloaded from http://www.cs.sfu.ca/~ester/datasets/.

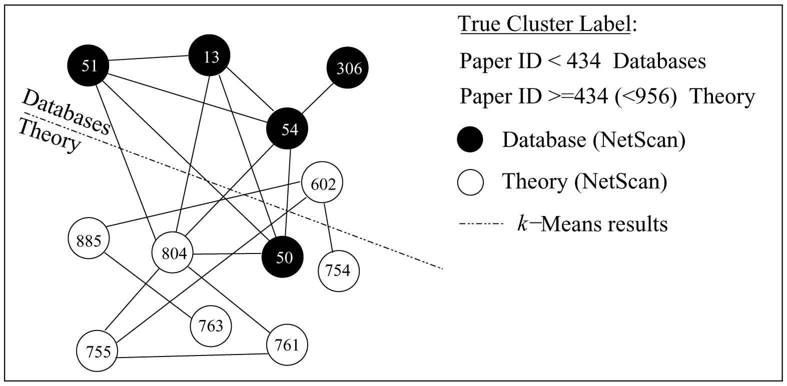

Fig. 13. Partial clustering results on dataset DBLP II.

and by NetScan. Table VII lists the details of the chosen papers, which allows readers to verify and compare the clustering results shown in Figure 13. 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, for example, "Query", "Search", "Framework", and "Algorithm". From this attribute information alone, the paper would 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 [Shi and Malik 2000]. 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 [Barabasi et al. 2002], the CkC model is expected to be able to handle many real life applications such as community identification and market segmentation, in which internal connectivity is requested.

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 [Spellman et al. 1998] 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

Table VII. Papers Used in Figure 13

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

downloaded from the BioGRID database [Stark et al. 2006]. 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 [Berriz et al. 2003]. FuncAssociate measures the over-representation of GO[3] (Gene

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

Table VIII. Comparison of NetScan, GraClus and k-Means on the Adapted Spellman Dataset

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

Ontology) terms within a cluster using negative log P-values, a standard cluster validation method in the bioinformatics literature [Ulitsky and Shamir 2007]. The P-value measures the probability that a certain over-representation of GO terms appears by chance, that is, the smaller the P-value (the larger the negative log P-value), the more significant the cluster.

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 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 VIII.

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 and Shamir [2007], identifying connected subnetworks in the interaction data helps to ensure that clusters are biologically relevant.

8. CONCLUSION

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 article, we introduced the novel Connected k-Center (CkC) problem, a clustering model that takes into account attribute data as well as relationship data. We 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 proposed a dynamic programming method giving an optimal solution in polynomial time. To improve the scalability for large real datasets, we 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.

The framework of joint cluster analysis of attribute and relationship data suggests several interesting directions for future research. First, 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. Second, the relationships can be nonbinary, 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 nonsymmetric 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.

ACKNOWLEDGMENTS

We would like to thank Dr. Petra Berenbrink for the valuable discussions in the early stage of this study. We also thank Recep Colak for preparing the gene dataset.

REFERENCES

Agarwal, P. and Procopiuc, C. M. 2002. Exact and approximation algorithms for clustering. *Algorithmica 33,* 2, 201–226.

Ankerst, M., Breunig, M. M., Kriegel, H., and Sander, J. 1999. Optics: Ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Philadelphia, PA). ACM, New York, 49–60.

Barabasi, A. L., Jeong, H., Neda, Z., Ravasz, E., Schubert, A., and Vicseks, T. 2002. Evolution of the social network of scientific collaborations. *Physica A 311,* 3–4, 590–614.

Bartal, Y., Charikar, M., and Raz, D. 2001. Approximating min-sum k-clustering in metric spaces. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing* (Hersonissos, Greece). ACM, New York, 11–20.

Basu, S., Bilenko, M., and Mooney, R. J. 2004. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Seattle, WA). ACM, New York, 59–68.

Berriz, G. F., King, O. D., Bryant, B., Sander, C., and Roth, F. P. 2003. Characterizing gene sets with funcassociate. *Bioinformatics 19,* 18, 2502–2504.

Brandes, U., Gaertler, M., and Wagner, D. 2003. Experiments on graph clustering algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms* (Budapest, Hungary). Springer-Verlag, Berlin/Heidelberg, Germany, 568–579.

Brucker, P. 1977. On the complexity of clustering problems. In *Optimization and Operations Research*, R. Hehn, B. Korte, and W. Oettli, Eds. Springer-Verlag, Berlin, Germany, 45–54.

Chan, P. K., Schlag, M. D. F., and Zien, J. Y. 1994. Spectral k-way ratio-cut partitioning and clustering. *IEEE Trans. Computer-Aided Desi. Integ. Circ. Syst. 13,* 9, 1088–1096.

Charikar, M., Guha, S., Tardos, É., and Shmoys, D. 1999. A constant factor approximation algorithm for the k-median problem. *J. Comput. Syst. Sci. 65,* 1, 129–149.

Charikar, M. and Panigrahy, R. 2004. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci. 68,* 2, 417–441.

CiteSeer. 2006. Scientific literature digital library. http://citeseer.ist.psu.edu/.

Davidson, I. and Ravi, S. S. 2005. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 5th SIAM International Conference on Data Mining* (Newport Beach, CA). Society for Industrial and Applied Mathematics, Philadelphia, PA, 138–149.

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

Dhillon, I. S., Guan, Y., and Kulis, B. 2007. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29,* 11, 1944–1957.

Doddi, S., Marathe, M. V., Ravi, S. S., Taylor, D., and Widmayer, P. 2000. Approximation algorithms for clustering to minimize the sum of diameters. In *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory* (Bergen, Norway). Springer-Verlag, Berlin/Heidelberg, Germany, 237–250.

Dyer, M. and Frieze, A. M. 1985. A simple heuristic for the p-center problem. *Oper. Res. Lett. 3*, 285–288.

Ester, M., Ge, R., Gao, B. J., Hu, Z., and Ben-Moshe, B. 2006. Joint cluster analysis of attribute data and relationship data: the connected k-center problem. In *Proceedings of the 6th SIAM Conference on Data Mining* (Bethesda, MD). Society for Industrial and Applied Mathematics, Philadelphia, PA, 246–257.

Ester, M., Kriegel, H., Sander, J., and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (Portland, OR). AAAI Press, 226–231.

Feder, T. and Greene, D. H. 1988. Optimal algorithms for approximate clustering. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (Chicago, IL). ACM, New York, 434–444.

Frederickson, G. N. and Johnson, D. B. 1979. 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*. The Johns Hopkins Univ., Baltimore, MD, 47–52.

Gonzalez, T. 1985. Clustering to minimize the maximum inter-cluster distance. *Theoret. Comput. Sci. 38,* 2–3, 293–306.

Guha, S., Rastogi, R., and Shim, K. 1999. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering* (Sydney, Austrialia). IEEE Computer Society, Los Alamitos, CA, 512–521.

Guttman-Beck, N. and Hassin, R. 1998. Approximation algorithms for min-sum p-clustering. *Disc. Appl. Math. 89,* 1–3, 125–142.

Hanisch, D., Zien, A., Zimmer, R., and Lengauer, T. 2002. Co-clustering of biological networks and gene expression data. *Bioinformatics 18*, S145–S154.

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

Hartuv, E. and Shamir, R. 2000. A clustering algorithm based on graph connectivity. *Inf. Proc. Lett. 76,* 4–6, 175–181.

Hochbaum, D. and Shmoys, D. 1985. A best possible heuristic for the k-center problem. *Math. Oper. Res. 10*, 180–184.

Iacobucci, D. 1996. *Networks in Marketing*. Sage Publications, Thousand Oaks, CA.

Jain, A. and Dubes, R. 1988. *Algorithms for clustering data*. Prentice-Hall, Englewood Cliffs, NJ.

JAIN, K. AND VAZIRANI, V. 2001. Approximation algorithms for metric facility location and k-median problems using the primal-dual scheme and lagrangian relaxation. *J. ACM 48,* 2, 274–296.

KARIV, O. AND HAKIMI, S. L. 1979. An algorithmic approach to network location problems, Part II: p-medians. *SIAM J. Appl. Math. 37*, 539–560.

KARYPIS, G., HAN, E., AND KUMAR, V. 1999. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Comput. 32,* 8, 68–75.

KAUFMAN, L. AND ROUSSEEUW, P. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York.

KULIS, B., BASU, S., DHILLON, I. S., AND MOONEY, R. 2005. Semi-supervised graph clustering: A kernel approach. In *Proceedings of the 22nd International Conference on Machine Learning* (Bonn, Germany). ACM, New York, 457–464.

LIN, J. AND VITTER, J. 1992. Approximation algorithms for geometric median problems. *Inf. Proc. Lett. 44,* 5, 245–249.

LLOYD, S. 1982. Least squares quantization in pcm. *IEEE Trans. Inf. Theory 28,* 2, 129–136.

MACQUEEN, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematics, Statistics and Probability*. 281–297.

MEGIDDO, N. AND SUPOWIT, K. J. 1984. On the complexity of some common geometric location problems. *SIAM Journal on Computing 13,* 1, 182–196.

MEGIDDO, N., TAMIR, A., ZEMEL, E., AND CHANDRASEKARAN, R. 1981. An $o(n \log^2 n)$ algorithm for the k-th longest path in a tree with applications to location problems. *SIAM J. Comput. 10,* 2, 328–337.

MOSER, F., GE, R., AND ESTER, M. 2007. Joint cluster analysis of attribute and relationship data without a-priori specification of the number of clusters. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Jose, CA). ACM, New York.

NG, R. T. AND HAN, J. 1994. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases* (Santiago de Chile, Chile). Morgan Kaufmann, San Francisco, CA, 144–155.

SCOTT, J. 2000. *Social Network Analysis: A handbook*. Sage Publications, Thousand Oaks, CA.

SEGAL, E., WANG, H., AND KOLLER, D. 2003. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics (Suppl. 1) 19*, 264–272.

SHI, J. AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. Patt. Analysis and Machine Intelligence 22,* 8, 888–905.

SPELLMAN, P. T., SHERLOCK, G., ZHANG, M. Q., IYER, V. R., ANDERS, K., EISEN, M. B., BROWN, P. O., BOTSTEIN, D., AND FUTCHER, B. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molec. Biol. Cell 9,* 12, 3273–3297.

STARK, C., BREITKREUTZ, B., REGULY, T., BOUCHER, L., BREITKREUTZ, A., AND TYERS, M. 2006. Biogrid: A general repository for interaction datasets. *Nucleic Acids Res. 34*, D535–D539.

STEINBACH, M., KARYPIS, G., AND KUMAR, V. 2000. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*.

STEINHAUS, H. 1956. Sur la division des corp materiels en parties. *Bulletin L'Acadmie Polonaise des Science C1. III, IV*, 801–804.

SWAMY, C., AND KUMAR, A. 2004. Primal-dual algorithms for connected facility location problems. *Algorithmica 40,* 4, 245–269.

TAMIR, A. 1996. An $o(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters 19*, 59–64.

TASKAR, B., SEGAL, E., AND KOLLER, D. 2001. Probabilistic classification and clustering in relational data. In *Proceedings of 17th International Joint Conference on Artificial Intelligence* (Seattle, WA). Morgan Kaufmann, San Francisco, CA, 870–878.

TOREGAS, C., SWAN, R., REVELLE, C., AND BERGMAN, L. 1971. The location of emergency service facilities. *Oper. Res. 19*, 1363–1373.

TUNG, A. K. H., NG, R. T., LAKSHMANAN, L. V. S., AND HAN, J. 2001. Constraint-based clustering in large databases. In *Proceedings of the 8th International Conference on Database Theory* (London, UK). Springer-Verlag, New York, 405–419.

Ulitsky, I. and Shamir, R. 2007. Identification of functional modules using network topology and high-throughput data. *BMC System Biology 1,* 8.

Wasserman, S. and Faust, K. 1994. *Social Network Analysis*. Cambridge University Press, Cambridge, UK.

Webster, C. and Morrison, P. 2004. Network analysis in marketing. *Australasian Market. J. 12,* 2, 8–18.

Received December 2006; revised July 2007; accepted December 2007