Constraint-Driven Clustering

Rong Ge, Martin Ester, Wen Jin
School of Computing Science
Simon Fraser University, Burnaby, Canada
{rge,ester,wjin}@cs.sfu.ca

Ian Davidson
Department of Computer Science
State University of New York, Albany
davidson@cs.albany.edu

ABSTRACT

Clustering methods can be either data-driven or need-driven. Data-driven methods intend to discover the true structure of the underlying data while need-driven methods aims at organizing the true structure to meet certain application requirements. Thus, need-driven (e.g. constrained) clustering is able to find more useful and actionable clusters in applications such as energy aware sensor networks, privacy preservation, and market segmentation. However, the existing methods of constrained clustering require users to provide the number of clusters, which is often unknown in advance, but has a crucial impact on the clustering result. In this paper, we argue that a more natural way to generate actionable clusters is to let the application-specific constraints decide the number of clusters. For this purpose, we introduce a novel cluster model, Constraint-Driven Clustering (CDC), which finds an a priori unspecified number of compact clusters that satisfy all user-provided constraints. Two general types of constraints are considered, i.e. minimum significance constraints and minimum variance constraints, as well as combinations of these two types. We prove the NP-hardness of the CDC problem with different constraints. We propose a novel dynamic data structure, the CD-Tree, which organizes data points in leaf nodes such that each leaf node approximately satisfies the CDC constraints and minimizes the objective function. Based on CD-Trees, we develop an efficient algorithm to solve the new clustering problem. Our experimental evaluation on synthetic and real datasets demonstrates the quality of the generated clusters and the scalability of the algorithm.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

General Terms

Algorithms

Keywords

Constraints, Clustering, NP-hardness

1. INTRODUCTION

Clustering aims at grouping data objects into clusters in an effective and efficient manner. The generated clusters provide useful knowledge to support decision making in many applications. Depending on perspective, clustering methods can be either data-driven or need-driven [7]. The data-driven clustering methods intend to discover the true structure of the underlying data by grouping similar objects together while the need-driven clustering methods group objects based on not only similarity but also needs imposed by particular applications. Thus, the clusters generated by need-driven clustering methods are more *useful* and *actionable* to meet certain application requirements.

Need-driven or actionable data mining research has attracted significant interest recently. Kleinberg *et al.* [24] propose a framework to evaluate data mining results, in particular clustering results, by their utility in decision making. In this framework, the goal of generating useful clusters is achieved by a sophisticated objective function which is defined based on business needs. Alternatively, complex application needs can be captured by corresponding constraints. For example, in market segmentation, relatively balanced customer groups are more preferable so that the knowledge extracted from each group has equal significance and are thus easier to evaluate [19]. The special requirement of identifying balanced clusters can be effectively captured by imposing balancing constraints [9, 31, 7, 35].

These models enable us to find useful clusters. However, they require users to provide the number of clusters, K, which is often unknown in advance. Even if the most appropriate number of clusters determined from the data distribution is known, it may not suit the application needs [7]. Furthermore, an inappropriate number of clusters may result in generating distorted and less actionable clusters. We argue that a more natural way to generate actionable clusters is to let the constraints decide the number of clusters. In this paper, we propose a cluster model, *Constraint-Driven Clustering*, which aims at utilizing user-provided constraints to discover an arbitrary number of compact and balanced clusters. The compactness of a clustering is measured by the sum of the squared distances of all data objects to their corresponding cluster representatives.

For different application needs, various balancing constraints can be designed to restrict the generated clusters in order to make them actionable. Particularly, we are interested in constraint types, i.e., minimum significance constraints and minimum variance constraints. The minimum significance constraint specifies the minimum number of objects in a cluster. The minimum variance constraint poses a lower bound on the variance of a cluster. By imposing a minimum significance constraint or/and a minimum variance constraint, our model searches for clusters which are balanced in terms of cardinality or/and variance.

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

Energy Aware Sensor Networks [18, 20, 7]: Grouping sensors into clusters is an important problem in sensor networks since it can drastically affect the network's communication energy consumption [18]. Normally, a master node is chosen from sensors in each cluster or deployed to the central area of each cluster. Other sensors will communicate with the outside world through the closest master node. In this context, it is desirable to require each cluster to contain at least a certain number of sensors in order to balance the work load of master nodes. To prolong the lifetime of a sensor network, evenly distributing energy consumption among clusters is desired. Since the energy consumption of message transmissions increases quadratically with the distance between communicating sensors [10], the variance of a group of sensors corresponds to the amount of energy consumed by those sensors on average. The minimum variance constraint allows to group sensors into clusters which are balanced in terms of energy consumption. Moreover, in this application, it is natural to have the constraints decide the appropriate number of clusters instead of specifying a number in advance.

Privacy Preservation [28, 30]: In a privacy preservation application, we may want to release personal records to the public without a privacy breach. To achieve this, we can group records into small clusters and release the summary of each cluster to the public. In this context, the usability of a clustering is evaluated by how much privacy is preserved in the clustering. To preserve individual privacy, the k-anonymity model [30] requires that each cluster has to contain at least a certain number of individuals. However, these individuals could have very similar, even identical attribute values, allowing an adversary to accurately estimate their sensitive attribute values with high confidence from the summary. We argue, therefore, that the clusters to be published should also have a minimum variance which translates into the width of the confidence interval of the adversary estimate. In the context of privacy preservation, again, it is typically unreasonable to specify the number of groups in advance.

PPMicroCluster model [22] requires both minimum significance and minimum radius constraints to preserve privacy. Compared to this model, our Constraint-Driven Clustering model adopts a more practical constraint, i.e., minimum variance constraint, which describes the statistical properties of a cluster better and can be used for a wide range of applications. Besides, we systematically study the complexity of the Constraint-Driven Clustering problem and propose a dynamic algorithm which is shown to be more efficient than the static algorithm for solving the PPMicroCluster problem.

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

The contributions of this paper are as follows:

(1) We propose the Constraint-Driven Clustering problem, which incorporates a minimum significant constraint and a minimum variance constraint for discovering actionable clusters, but does not require an a priori specification of the number of clusters.

(2) We prove the NP-hardness of the proposed clustering problem with different constraints.

(3) We develop the efficient CD-Tree algorithm to generate clusters w.r.t the introduced constraints.

(4) We evaluate our CD-Tree algorithm on synthetic and real datasets and demonstrate the quality of the generated clusters and the efficiency of the algorithm.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 introduces the new cluster model and analyzes its complexity. Section 4 presents the CD-Tree algorithm. We report experimental results in Section 5 and conclude the paper in Section 6.

2. RELATED WORK

Actionable Clustering. Actionable clustering was first proposed by Kleinberg in [24], in which a clustering is evaluated by its utility in decision-making. By introducing a new objective function, the goal of clustering is shifted from identifying the true structure of the underlying distribution to discovering useful clusters. Ester *et al.*[16] extend the catalog segmentation problem [24] by introducing a new utility measured by the number of customers that have at least a specified minimum interest in the catalogs. A joint optimization approach is proposed in [21] to address two issues in market segmentation, i.e., segmenting customers into homogeneous groups and determining the optimal policy towards each segment.

Cluster-level Constraints. The research on clustering with constraints was introduced by [9] and systematically studied in [31]. Both clustering models aim at partitioning data points into k clusters while each cluster satisfies a significance constraint. Bradley *et al.* [9] proposed a constrained k-means algorithm and suggested to achieve a cluster assignment by solving a minimum cost network flow problem. Tung *et al.* [31] propose to solve this problem by starting with any valid clustering. The solution is repeatedly refined by moving some objects between clusters to reduce the clustering cost and maintaining the constraint satisfaction at the same time.

For the same problem, Banerjee *et al.* present an efficient three-step scheme [6, 7] which gives a very general methodology for scaling up balanced clustering algorithms. The same problem is converted to a graph partition problem in [29] to discover balanced clusterings. However, the complexity of the graph-based approach is higher than the one in [6]. Different from our proposed model, all these clustering models require the number of clusters as an input.

Instance-level Constraints. Work on instance level constraints and clustering were first presented to the machine learning and data mining communities by Wagstaff and Cardie [32]. In this line of work the constraints are on instances either forcing them to be in the same cluster (must-link) or different clusters (cannot-link). Though it is possible to specify cluster level constraints as instance level constraints this would require a large number of constraints for even moderately sized data sets. For example, specifying all clusters must have all their points more than δ distance apart can be achieved by must-linking all those points less than or equal to δ distance apart [13]. Specifying too many constraints is also problematic as algorithms that attempt to satisfy all constraints can quickly be over-constrained [12] so that efficient algorithms to find just a single solution cannot be found even though they exist.

Clustering methods in Sensor networks. Ghiasi *et al.* propose to cluster sensor nodes such that the number of sensors in each cluster (which has a master node) is in the range of $[\frac{n}{k}-\delta, \frac{n}{k}+\delta]$ and the total distance between sensor nodes and K master nodes is minimized [18]. The clustering problem presented in [18] is different from the Constraint-Driven Clustering in that it specifies the number of clusters. More practical protocols are studied in the sensor network literature to minimize the energy consumption or message transmissions by grouping sensors to clusters. Coyle *et al.*[5] proposed a randomized algorithm to find the optimal number of cluster heads by minimizing the total energy spent on communicating between sensors and the information-processing center through the cluster heads. Authors in [25] present a clustering method on self-organizing sensor networks, for the purpose of grouping sensors into the optimal number of clusters that minimize the number of message transmissions. Similar approaches on clustering in sensor networks also include [4, 8, 23] etc. However, these clustering methods focus on dealing with engineering constraints instead of systematically studying the properties of the proposed clustering models.

k-Anonymity. The k-Anonymity model [28, 30] was proposed for the purpose of protecting data privacy. The k-anonymity framework archives the goal by generalizing entries in a table with minimum cost such that every record becomes textually indistinguishable from $k-1$ other records in the table. [26] and [3] prove that k-Anonymity with Suppression is NP-hard and study approximation algorithms. The k-Anonymity model is defined on categorical data, and thus has different properties from our model which assumes a geometric space with the Euclidean distance. [2] introduces a k-nearest neighbor based algorithm to solve the k-anonymity problem for numerical data. Domingo-Ferrer *et al.* [14] study the optimal k-partition problem which can be considered as the k-Anonymity model in the Euclidean space. And it is a special case of our model where only significance constraints are allowed. But in [14] the complexity of the proposed problem is not analyzed. [17] considers privacy preservation as a problem of finding the minimum number of hyperspheres with a fixed radius to cover a dataset satisfying that each hypersphere covers at least a certain number of data objects. A similar model, named PPMicroCluster, is studied in [22] which requires both significance and radius constraints. Compared to the PPMicroCluster model, our model adopts a more practical constraint, i.e., minimum variance constraint which describes the statistical properties of a cluster better and can be used for a wide range of applications. Besides, [22] does not analyze the complexity and proposes a static algorithm.

3. PROBLEM DEFINITION AND COMPLEXITY ANALYSIS

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

3.1 The CDC Problem

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

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

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

DEFINITION 2. *For each cluster $P_i, 1 \leq i \leq m$,*

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

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

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

3.2 Complexity Analysis

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

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

DEFINITION 3. **(*sig-CDC*).** *Given a set of points P in d dimensional space, a constant $Sig > 1$ and a cost threshold W. Decide whether P can be partitioned into disjoint clusters $\{P_1, \cdots, P_m\}$, which satisfies the following conditions:*

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

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

Definition 4. *(***PLANAR X3C** *[15])*
Given a set Q with $|Q| = 3q$ and a set T of triples from $Q \times Q \times Q$ such that (1) every element of Q occurs in at most three triples and (2) the induced graph is planar. (The induced graph G contains a vertex for every element of Q and for every triple in T. There is an edge connecting a triple to an element if and only if the element is a member of the triple. Clearly, G is bipartite with vertex bipartition Q and T.) Decide whether there exists a subset of q triples in T which contains all the elements of Q.

Theorem 3.1. *sig-CDC is NP-complete for $Sig \geq 3$.*

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

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

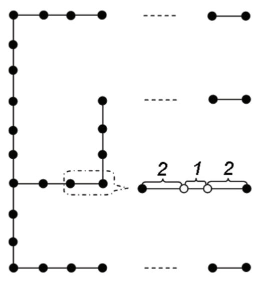

Figure 1: Rectilinear layout L.

2. Compute a rectilinear layout of G where each vertex v of G is mapped to a point p_v on the integer lattice. We further enlarge the layout by a factor of 1000 to ensure that every two distinct horizontal (vertical) line segments are far away enough from each other. Each edge $e = (q, t)$ of G is broken into a sequence of line segments of length 5 by placing points in the rectilinear layout (Figure 1). The resulting layout is denoted as L. [33] proposed a linear time algorithm to compute such a layout. A similar rectilinear layout is used in [13] to prove the NP-completeness of the feasibility problem for the Must-Link and ϵ constraints.

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

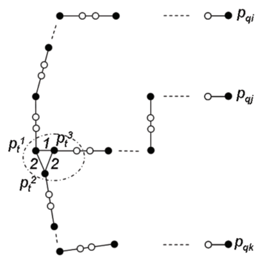

Figure 2: Final layout L'

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

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

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

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

Now assume that there is a feasible clustering with the total cost smaller than or equal to $5|P|/3$. We demonstrate how to obtain a subset of triples $C \subset T$ that covers every element in Q exactly once. First we prove that any feasible clustering consists of $|P|/3$ disjoint clusters each consist of 3 points. For any $i \geq 3$, let H_i denote the number of clusters consisting of i points. We have $|P| = \sum_i iH_i$ and the total number of clusters is $\sum_i H_i$. Note that the minimum cost of a cluster with three points is 5 by grouping three points connected through two consecutive edges and choosing the

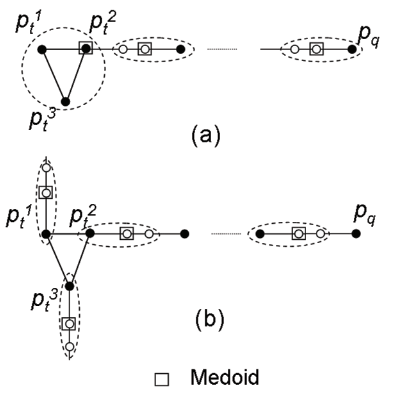

Figure 3: (a). Points in a triple set are grouped into one cluster. (b). Points in a triple set are grouped separately with auxiliary points.

middle point as the medoid. Furthermore, every additional node in a cluster with more than three points contributes at least 4 to the total cost (check the cluster containing four points in the dashed circle in Figure 2). Thus,

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

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

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

Next we demonstrate that the *sig-CDC* problem remains NP-Complete if using the mean vector (instead of the medoid) of a cluster as the representative. We refer to this model as *μ-sig-CDC*. Due to limited space, we only present the sketch of NP-hardness proof for the following problems.

THEOREM 3.2. *The μ-sig-CDC problem is NP-complete.*

PROOF. **(Sketch.)** The proof is by a reduction from PLANAR X3C that is similar to the one for Theorem 3.1. We first construct a rectilinear layout for a PLANAR X3C instance and replace each triple set by an isosceles triangle with two sides of length $\sqrt{37}$ and one side of length 2. See Figure 4 for the final layout. Next, we set the significance constraint $Sig = 3$ and set the total cost threshold $W = 26|P|/3$. Similar to the proof of Theorem 3.1, we can show that in a feasible clustering, there must be exactly $|P|/3$ disjoint clusters, each consists of three points

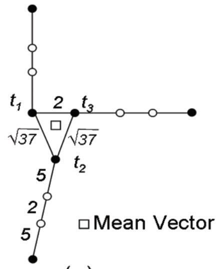

Figure 4: Transformed Layout for Theorem 3.2 and Theorem 3.3.

and has a cost of 26. Given this, we can show that there is a feasible clustering if and only if the corresponding instance of PLANAR X3C has a feasible solution. Hence, the *μ-sig-CDC* problem is NP-complete. □

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

THEOREM 3.3. *The μ-var-CDC problem is NP-complete.*

PROOF. (Sketch) To prove that the *μ-var-CDC* problem is NP-Complete, we use the same construction for proving Theorem 3.2. We set the variance constraint $Var = 26/3$ and set the total cost $W = 26|P|/3$. Let Cl be an arbitrary cluster and denote by $Cost(Cl)$ the total cost of Cl. Note that due to the minimum variance constraint, $Cost(Cl) \geq 26|Cl|/3$, and by a case analysis we can show that the equality holds only when $|Cl| = 3$. Hence, in order to have a total cost of $26|P|/3$, we must have $|P|/3$ disjoint clusters each of which consists of 3 points. The rest of the proof is similar to that of Theorem 3.2. □

4. ALGORITHM

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

We observe that a solution to the *CDC* problem has the following characteristics. (1) To minimize the objective function (the sum of squared distances), the generated clusters in an optimal solution should be balanced in terms of given constraints. For example, when only a significance constraint is provided, the generated clusters in an optimal solution should contain similar number of data points. (2) The membership assignment of any data point can be decided by considering its close neighbors. Thus, easily retrieving the local neighborhood of data points is critical to the design of a universal algorithm that can handle different constraints. Guided by these observations, we propose an algorithm based on a novel data structure, called the *CD-Tree*, which is similar to the *B*-Tree and *CF*-Tree [34].

4.1 The *CD*-Tree

The *CD*-Tree has two input parameters, i.e., a significance parameter S, a variance parameter V. Normally we set $S = Sig$ and $V = Var$, i.e., the parameters of the CDC problem. If one of the CDC constraints is trivial, methods for automatically determining appropriate parameter values are needed which are discussed in Section 4.4.

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

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

LEMMA 4.1. *In the μ-sig-CDC problem, there exists an optimal clustering s.t. the number of data points in any cluster is less than $2Sig$ and greater than or equal to Sig.*[1]

PROOF. By contradiction. Assume that in every optimal clustering, there is a cluster C containing l data points where $l \geq 2Sig$.

We arbitrarily split C into two clusters C_1 and C_2 where $|C_1| = Sig$ and $|C_2| = l - Sig$. Let $\overrightarrow{M}$ be the mean vector of the points in C. Similarly let $\overrightarrow{M_1}$ and $\overrightarrow{M_2}$ be the mean vectors of the points in C_1 and C_2 respectively. Note that

$$\begin{aligned}\overrightarrow{M} &= \frac{1}{l}\left(\sum_{p\in C_1}\overrightarrow{p} + \sum_{q\in C_2}\overrightarrow{q}\right) \\ &= \frac{1}{l}[Sig\overrightarrow{M_1} + (l-Sig)\overrightarrow{M_2}] \qquad (1)\end{aligned}$$

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

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

since $\sum_{p\in C}\overrightarrow{p} = l\overrightarrow{M}$. Similarly, $f(C_1) = \sum_{p\in C_1}\overrightarrow{p}^2 - Sig\overrightarrow{M_1}$ and $f(C_2) = \sum_{q\in C_2}\overrightarrow{q}^2 - (l-Sig)\overrightarrow{M_2}^2$.

Applying Equation 1 and straightforward algebra, we get

$$\begin{aligned}& f(C) - (f(C_1) + f(C_2)) \\ =\ & Sig\overrightarrow{M_1}^2 + (l-Sig)\overrightarrow{M_2}^2 - l\overrightarrow{M}^2 \\ =\ & \frac{Sig(l-Sig)}{l}(\overrightarrow{M_1}^2 + \overrightarrow{M_2}^2 - 2\overrightarrow{M_1}\overrightarrow{M_2}) \geq 0 \qquad (2)\end{aligned}$$

Thus, we could obtain a clustering whose objective value is smaller than or equal to the previous one by splitting C into C_1 and C_2, yielding a contradiction. □

[1]Note that a similar result is presented in [14].

In the CD-Tree, each entry of a leaf node represents an individual data point. The maximum capacity of a non-leaf node is set to Z which is a constant and can be set arbitrarily. Every entry of a non-leaf node corresponds to the subtree rooted at one of its child nodes. The entry stores a pointer to the child node, as well as the statistical information (the mean vector, linear sum and squared sum) of all the points in the corresponding subtree. Similar to the CF-Features [34], the statistical information is used to direct a new point along a path to the closest leaf node. Besides, all the leaf nodes are linked together for easy access of their neighborhood.

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

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

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

Split. In the construction of the CD-Tree, a split is invoked whenever the capacity of a node is exceeded. The proof of Lemma 4.1 provides an efficient way to evaluate the drop of the objective value during a split. Based on Equation 2, we design an efficient algorithm to split a group of $2S$ data points C into two clusters C_1 and C_2 (Algorithm 1). The algorithm proceeds in a greedy fashion. First we pick the point that is farthest from C's mean vector and add it to C_2. Then we iteratively add points that are closest to C_2's mean vector into C_2 until the objective value stops decreasing. And C_1 keeps all remaining points in C. Splitting non-leaf nodes can be done similarly by considering the mean vector saved in the entries of the non-leaf nodes.

Finally, after each split, we need to decide how to link the newly created leaf nodes with the existing nodes. Suppose we just split a leaf node C into C_1 and C_2. Let C_{prev} and C_{next} be the both neighbors of C before split. There are two ways to link C_1 and C_2. Let $\mu(C)$ be the mean vector of C. If $|\mu(C_{prev}) - \mu(C_1)| \leq |\mu(C_{prev}) - \mu(C_2)|$, we create links $C_{prev} \rightarrow C_1 \rightarrow C_2 \rightarrow C_{next}$, otherwise $C_{prev} \rightarrow C_2 \rightarrow C_1 \rightarrow C_{next}$.

4.2 Solving the CDC problem

After we construct a CD-Tree, the CDC problem can be solved by post-processing the leaf nodes of the CD-Tree. Similar to many other hierarchical tree structures, a CD-Tree has the property that data points located in the same

Algorithm 1 Splitting a group of $2S$ points.

```
1: Input: $C$ contains $2S$ points
2: Output: $C_1$ and $C_2$
3: $C_1 = C$; size = $|C_1|$;
4: Set $\overrightarrow{M_1}$ to the mean vector of points in $C_1$;
5: Pick the point $p$ in $C_1$ that is farthest from $\overrightarrow{M_1}$;
6: Remove $p$ from $C_1$ and update $\overrightarrow{M_1}$;
7: $C_2 = \{p\}$; $\overrightarrow{M_2} = \overrightarrow{p}$;
8: MaxObjDrop = $\frac{(size-1)}{size}(\overrightarrow{M_1}^2 + \overrightarrow{M_2}^2 - 2\overrightarrow{M_1}\overrightarrow{M_2})$;
9: for $i = 2$ to $S$ do
10:    Pick a point $p'$ from $C_1$ that is nearest to $\overrightarrow{M_2}$;
11:    Remove $p'$ from $C_1$ and update $\overrightarrow{M_1}$;
12:    Add $p'$ to $C_2$ and update $\overrightarrow{M_2}$;
13:    ObjDrop = $\frac{(size-i)}{size}(\overrightarrow{M_1}^2 + \overrightarrow{M_2}^2 - 2\overrightarrow{M_1}\overrightarrow{M_2})$;
14:    if ObjDrop $\geq$ MaxObjDrop then
15:       Continue;
16:    else
17:       Remove $p'$ from $C_2$ and add $p'$ back to $C_1$;
18:    end if
19: end for
```

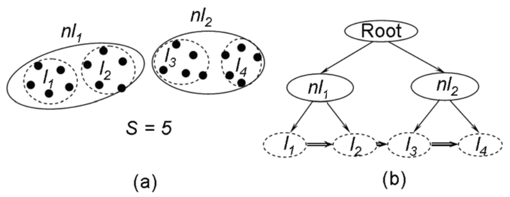

Figure 5: (a) Data Points. (b) Visualization of the CD-Tree.

subtree tend to be more similar than the points located in different subtrees. By linking tree nodes appropriately, we are able to retrieve the neighbors of any data point easily. For example, Figure 5 shows a small dataset and its corresponding CD-Tree. In this figure, dashed circles represent leaf nodes and solid circles correspond to non-leaf nodes of the CD-Tree. Since every new data point is inserted into the tree based on its distance to the mean vectors of subtrees, data points in l_1 are more similar to data points in l_2 than to data points in l_3 or l_4. This allows us to postprocess the neighboring leaf nodes to obtain constrained clusters.

We propose a sliding window approach for solving the CDC problem. A sliding window consists of exactly Z leaf nodes with the same parent node. Starting with the first leaf node in the window, we examine one leaf node at a time. Depending on the given significance constraint Sig and variance constraint Var, we distinguish between two types of a leaf node L. L is called *qualified* if L has at least Sig points and variance at least Var, otherwise *not-qualified*. For every qualified leaf node L, we output its "kernel", which is a subset of points of L that just satisfies the given constraints. Kernels can be easily calculated using a greedy approach (Case a of Algorithm 2). The remaining points in L are treated together with those under-qualified leaf nodes. We repeatedly absorb points from other leaf nodes in the same window to form clusters that satisfy the given constraints (Case b of Algorithm 2). Note that all the points which have been assigned to some clusters are not considered in case b. After looping through all the leaf nodes, a set of constrained clusters is generated.

Algorithm 2 Cluster leaf nodes

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

4.3 Runtime Analysis

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

An insertion of a single point into the CD-Tree involves two operations: locating the right leaf node to insert the point and splitting the leaf node if its capacity is exceeded. The time to locate the right leaf node is $O(n)$ since the height of the CD tree is $O(n)$. In a split (Algorithm 1), we create a new node starting with the farthest point from the mean of the old node (which can be found in $O(Sig)$ steps), and gradually absorb the closest point to the mean of the new node (each of the $O(Sig)$ absorptions takes $O(Sig)$ steps). Hence, the runtime of a split is $O(Sig^2)$. For building a CD-Tree with n points, the total runtime is $O(n^2 + Sig^2 n)$ in the worst case.

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

Yet, in practice, Sig is typically small compared to n. If the data distribution is not highly skewed, the height of the CD-Tree is usually small. In such cases, our algorithm is very efficient as demonstrated by the experimental evaluation in Section 5.

4.4 Discussion

How to handle a trivial variance constraint? If a variance constraint is trivial, i.e., $Var = 0$, we need to set the variance parameter V used for the CD-Tree construction automatically. A suitable threshold should allow Sig

closest points to be grouped together. Ideally, if we know the average Sig nearest neighbor distance of a dataset, this distance can be used to approximate the variance parameter. However, the exact computation of the average Sig nearest neighbor distance is expensive. Therefore, we propose to estimate the Sig nearest neighbor distance based on the following lemma.

LEMMA 4.2. *Given a dataset of n points that are uniformly distributed in a d dimensional space with volume Vol. Fix any point p, let R be the random variable indicating the radius of the smallest enclosing ball of the Sig nearest neighbors of p. Let $R^* = \pi^{-1/2}\sqrt[d]{Sig \cdot Vol\Gamma(d/2+1)n^{-1}}$ where Γ is the Gamma Function [1]. Then*

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

PROOF. We first show that $\Pr[R > 2^{2/d}R^*] \le n^{-2}$. Let H be the hypersphere with radius $2^{2/d}R^*$ centered at p. Let $vol(H)$ be the volume of H. Then, from [1], we have

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

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

Lemma 4.2 shows that we can estimate the Sig nearest neighbor distance with high probability if we know the volume Vol of a dataset for uniformly distributed data. In practice, if the volume is not known, we can draw a small set of samples to obtain a good estimation.

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

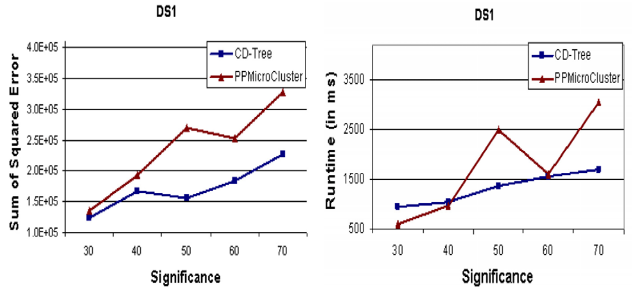

Figure 6: Results for Dataset DS1 (Only significance constraints are specified).

5. EXPERIMENTAL EVALUATION

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

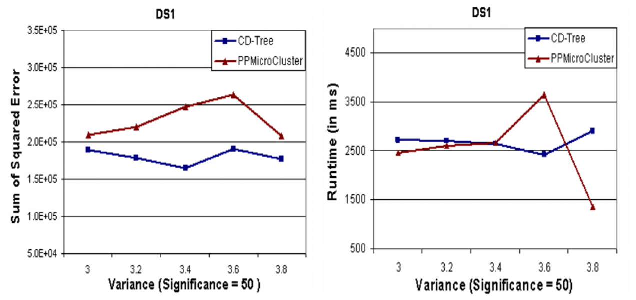

Figure 7: Results for Dataset DS1 (Both significance and variance constraints are specified).

5.1 Methodology

In order to evaluate the CD-Tree algorithm for sensor network applications, we generated a synthetic dataset (DS1) consisting of 5000 two dimensional data points which simulates a sensor network with 5000 sensors uniformly deployed in a two dimensional space. A similar simulation was used in [20] to evaluate clustering results for sensor networks. In addition, we evaluated the CD-Tree algorithm on two real datasets. The first one is the "Abalone" dataset and the second one is the "Letter" dataset. Both datasets are from UCI machine learning repository [27]. The "Abalone" dataset was also used by [2] for evaluating the quality of the condensation group approach for privacy preservation applications. The original abalone dataset contains 4177 data points and 9 attributes. We preprocessed the dataset and kept 7 out of total 8 continuous attributes since one of the attributes is the class label. The Letter dataset includes 20,000 instances and has 16 continuous attributes. Finally, we generated three large synthetic datasets, containing 0.5 million, 1 million, and 2 million three dimensional data points, to evaluate the scalability of the CD-Tree algorithm.

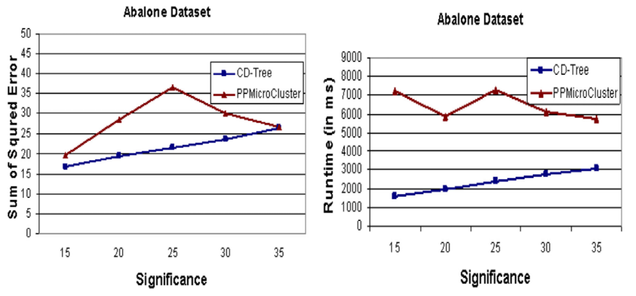

Figure 8: Results for Abalone Dataset (Only significance constraints are specified).

Two related approaches, the PPMicroCluster algorithm [22] and the condensation group approach [2], solve a constrained clustering problem similar to the CDC problem. We chose the PPMicroCluster algorithm as our comparison partner due to the following reasons. First, its problem definition is equivalent to the CDC problem except that is specifies a radius constraint instead of a variance constraint. Different from the variance, the radius of a cluster is the maximum distance between all points in a cluster to the cluster representative. We have adapted the PPMicroCluster algorithm to handle the variance constraint in

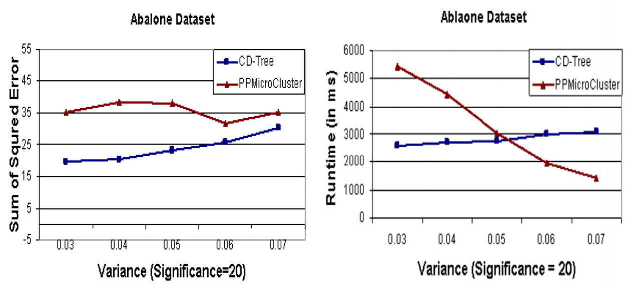

Figure 9: Results for Abalone Dataset (Both significance and variance constraints are specified).

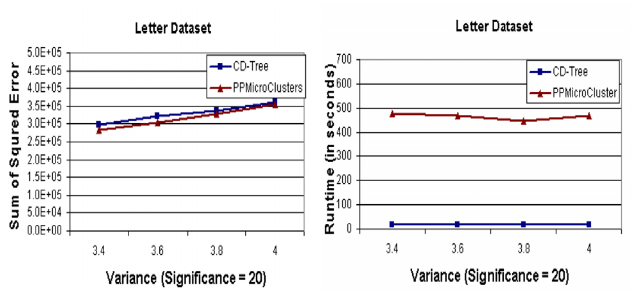

Figure 11: Results for the Letter Data set (Both significance and variance constraints are specified).

order to have a meaningful comparison. Second, the PPMicroClustering problem generalizes the condensation group approach (which has only the significance constraint), and the significance constraint is handled by the PPMicroCluster algorithm in the same style as by the condensation group approach. Note that the following experiments were conducted on the static phase of the PPMicroCluster algorithm since we did not evaluate the incremental updates of databases and the static phase of the PPMicroCluster algorithm is more effective due to the availability of the global knowledge for all data points.

5.2 Results

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

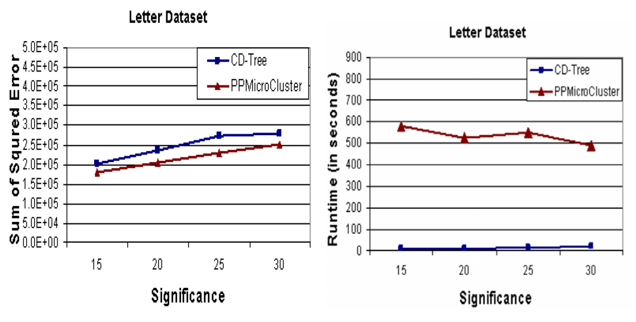

Figure 10: Results for the Letter Dataset (Only significance constraints are specified).

For dataset DS1, the results are presented in Figure 6 and 7 for significance constraints only and both constraints respectively. For both comparisons, the CD-Tree algorithm outperforms the PPMicroCluster algorithm in terms of clustering quality. Similar behavior is observed on the abalone dataset, for which the results are depicted in Figure 8 and 9. Due to the smaller size of the abalone dataset, both algorithms require comparable time.

For the larger Letter dataset (see Figure 10 and 11), we observe that the PPMicroCluster algorithm slightly outperforms the CD-Tree algorithm in terms of clustering quality. This behavior is expected since the PPMicroCluster algorithm relies on an index structure to maintain an accurate neighborhood relations among data points, while the tree structure built by the CD-Tree algorithm only keeps approximate neighborhood relations among data points. However, the CD-Tree algorithm runs more than 100 times faster than the PPMicroCluster algorithm. A small sacrifice of the clustering quality is reasonable for a dynamic algorithm like the CD-Tree algorithm.

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

Table 1: Scalability vs. Number of Points.

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

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

Table 2: Scalability vs. Different Constraints.

6. CONCLUSION

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

This paper suggests several interesting directions for future research. First, some issues related to our heuristic algorithm, such as how the maximum capacity of a non-leaf node influences the final partition and how effective is our heuristic algorithm on high dimensional data, can be further studied. Second, we want to evaluate the *CDC* model in real life applications. Third, the application needs may not always suggest exact values for the significance and variance constraints. Therefore, it is worthwhile to explore variants of the *CDC* model that allow the user to specify ranges instead of a fixed minimum value. Finally, we believe that the *CDC* framework and the CD-Tree algorithm can be generalized to include other constraint types, such as minimum separation constraints [11], to produce actionable clusters in an even broader category of applications.

7. REFERENCES

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

[2] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining. In *EDBT*, 2004.

[3] G. Aggarwal, T. F. K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, 2005.

[4] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *INFOCOM*, 2000.

[5] S. Bandyopadhyay and E. J. Coyle. An energy-efficient hierarchical clustering algorithm for wireless sensor networks. In *INFOCOM*, 2003.

[6] A. Banerjee and J. Ghosh. On scaling up balanced clustering algorithms. In *ICDM*, 2002.

[7] A. Banerjee and J. Ghosh. Scalable clustering algorithms with balancing constraints. *Data Mining Knowledge Discovery*, 13(3), 2006.

[8] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *INFOCOM*, 2001.

[9] P. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, MSR-TR-2000-65, Microsoft Research, 2000.

[10] J. Cartigny, D. Simplot, and I. Stojmenovic. Localized minimum-energy broadcasting in ad-hoc networks. In *INFOCOM*, 2003.

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

[12] I. Davidson and S. S. Ravi. Identifying and generating easy sets of constraints for clustering. In *AAAI*, 2006.

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

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

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

[16] M. Ester, R. Ge, W. Jin, and Z. Hu. A microeconomic data mining problem: customer-oriented catalog segmentation. In *KDD*, 2004.

[17] R. Ge, M. Ester, W. Jin, and Z. Hu. A disc-based approach to data summarization and privacy preservation. In *SSDBM*, 2006.

[18] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh. Optimal energy aware clustering in sensor network. *Sensor*, 2(7), 2002.

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

[20] G. Gupta and M. Younis. Load-balanced clustering of wireless sensor networks. *IEEE International Conference on Communications*, 2003.

[21] N. P. Jedid-Jah Jonkera and D. V. den Poel. Joint optimization of customer segmentation and marketing policy to maximize long-term profitability. *Expert Systems with Applications*, 27(2), 2004.

[22] W. Jin, R. Ge, and W. Qian. On robust and effective k-anonymity in large databases. In *PAKDD*, 2006.

[23] V. Kawadia and P. R. Kumar. Power control and clustering in ad hoc networks. In *INFOCOM*, 2003.

[24] J. Kleinberg, C. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. *J. Data Mining and Knowledge Discovery*, 1999.

[25] R. Krishnan and D. Starobinski. Efficient clustering algorithms for self-organizing wireless sensor networks. *Journal of Ad-Hoc Networks*, 2005.

[26] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *PODS*, 2004.

[27] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998.

[28] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, 1998.

[29] A. Strehl and J. Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *HiPC 2000*, 2000.

[30] L. Sweeney. *k*-anonymity: A model for protecting privacy. In *IJUFKS*, 2002.

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

[32] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *ICML*, 2000.

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

[34] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *SIGMOD*, 1996.

[35] S. Zhong and J. Ghosh. Scalable, balanced model-based clustering. In *SDM*, 2003.