The Complexity of Non-Hierarchical Clustering With Instance and Cluster Level Constraints*

Ian Davidson† S. S. Ravi‡

Abstract

Recent work has looked at extending clustering algorithms with instance level must-link (ML) and cannot-link (CL) background information. Our work introduces δ and ϵ cluster level constraints that influence inter-cluster distances and cluster composition. The addition of background information, though useful at providing better clustering results, raises the important *feasibility* question: Given a collection of constraints and a set of data, does there exist at least one partition of the data set satisfying all the constraints? We study the complexity of the feasibility problem for each of the above constraints separately and also for combinations of constraints. Our results clearly delineate combinations of constraints for which the feasibility problem is computationally intractable (i.e., **NP**-complete) from those for which the problem is efficiently solvable (i.e., in the computational class **P**).

We also consider the ML and CL constraints in conjunctive and disjunctive normal forms (CNF and DNF respectively). We show that for ML constraints, the feasibility problem is intractable for CNF but efficiently solvable for DNF. Unfortunately, for CL constraints, the feasibility problem is intractable for both CNF and DNF. This effectively means that CL-constraints in a non-trivial form cannot be efficiently incorporated into clustering algorithms. To overcome this, we introduce the notion of a choice-set of constraints and prove that the feasibility problem for choice-sets is efficiently solvable for both ML and CL constraints. We also present empirical results which indicate that the feasibility problem occurs extensively in real world problems.

Keywords: Non-hierarchical clustering, Constraints, Complexity.

1 Introduction and Summary of Contributions

1.1 Motivation

Clustering is a ubiquitous technique in data mining and is viewed as a fundamental mining task [Bradley and Fayyad 1998, Pelleg and Moore 1999] along with classification, association rule mining and anomaly detection. However, non-hierarchical clustering algorithms such as k-Means are greedy algorithms that can converge to solutions that offer little insight; hence, in practice such algorithms are restarted many times with the hope that some execution will find a meaningful

*A conference version containing some of the results in this paper appeared as [Davidson and Ravi 2005a].

†Department of Computer Science, University at Albany - State University of New York, Albany, NY 12222. Email: davidson@cs.albany.edu.

‡Department of Computer Science, University at Albany - State University of New York, Albany, NY 12222. Email: ravi@cs.albany.edu.

solution. This is a computationally expensive task when dealing with the large data sets typically found in data mining problems.

Recent work has focused on the use of instance level must-link and cannot-link background information to enable clustering algorithms to converge to good clusterings that also have desirable characteristics. A must-link (ML) constraint enforces that two instances should be placed in the same cluster while a cannot-link (CL) constraint enforces that two instances should not be placed in the same cluster. We can divide previous work on clustering with instance level background information into two types:

(1) In the first type, the background information is used to learn a distortion/distance/objective function [Basu et' al 2004a, Klein et' al 2002].

(2) In the second type, the information is strictly enforced as constraints to guide the algorithm to a useful solution [Wagstaff and Cardie 2000, Wagstaff et' al 2001].

Philosophically, the first type of work makes the assumption that points surrounding a pair of points involved in an ML (CL) constraint should be close to (far from) each other [Klein et' al 2002], while the second type just requires that the two points involved in an ML (CL) constraint must be in the same cluster (different clusters). Recent examples of the second type of work include ensuring that constraints are satisfied at each iteration of an algorithm such as k-Means [Wagstaff et' al 2001] and initializing algorithms using the constraints [Basu et' al 2002]. The results of both types of work are quite encouraging; in particular, it is shown in [Wagstaff and Cardie 2000, Wagstaff et' al 2001] that k-Means with constraints obtains clusters with a significantly better purity (when measured on an extrinsic class label) than when not using constraints.

However, with the addition of background information comes the possibility of over-constraining and poorly specifying constraints to the point where there are no feasible solutions. Thus, the following *feasibility* problem must be addressed.

Definition 1.1 ***(Feasibility Problem)*** *Given a data set S, collection of constraints C, a lower bound K_ℓ and an upper bound K_u on the number of clusters, does there exist a partition of S into k blocks such that $K_\ell \leq k \leq K_u$ and all the constraints in C are satisfied?*

Considering a collection of just three constraints, namely $CL(x, y)$, $CL(x, z)$ and $CL(y, z)$, it is easy to see that there is no feasible clustering when $K_u < 3$. However, for a large number of constraints is the feasibility problem solvable in polynomial time? Is the feasibility problem under some types of constraints easy (i.e., in the computational class **P**) and for others intractable (i.e., **NP-complete**)? We believe that answers to these questions will play an important role in the design of algorithms for clustering under constraints, as have similar complexity analyses for belief networks inference [Cooper 1990] and association rule mining [Wijsen and Meersman 1998]. If the feasibility problem for a particular class of constraints is intractable, then clustering algorithms such as k-Means should not attempt to find a feasible solution at each iteration because doing so would be computationally expensive. In the absence of feasibility testing, each iteration of the k-Means

algorithm has a time complexity that is linear with respect to the number of points, attributes and clusters. However, if the feasibility problem is **NP**-complete, then the feasibility checking step will not run in polynomial time, unless $\mathbf{P} = \mathbf{NP}$. Similarly, learning a distance function for partitioning into k clusters under the assumption that the points involved in an ML-constraint are close together and that points involved in a CL-constraint are far apart is undesirable when no feasible partition satisfying the assumption exists.

1.2 Summary of Contributions

In this paper we carry out a formal analysis of clustering under constraints. Our work makes several pragmatic contributions.

1. We introduce two new constraint types which act upon *groups* of points. Roughly speaking, the ϵ-constraint enforces that each point x in a cluster must have another point y in the same cluster such that the distance between x and y is at most ϵ. This constraint can be used to enforce prior information with respect to how the data was collected [Davidson and Ravi 2005a]. The δ-constraint enforces that points in two different clusters are separated by a distance of at least δ. This constraint can be used to specify background information on the minimum distance between the clusters/objects we hope to discover.

2. We show that the ϵ and δ constraints can be easily represented using disjunctions and a single conjunction of ML constraints respectively, thus making their implementation easy.

3. We present complexity results for the feasibility problem for **conjunctions** of ML and CL constraints and for all combinations of ML, CL, δ and ϵ constraints. Our results show that in several situations, the feasibility problem is **NP**-complete. When a feasibility problem is in **P**, the corresponding algorithm can be used for initializing clustering algorithms.

4. We also consider more general forms of ML and CL constraints, namely constraints in conjunctive and disjunctive normal forms[1] (respectively CNF and DNF) and establish the complexity of the corresponding feasibility problem. In particular, we observe that the feasibility problem for CL constraints remains **NP**-complete for both forms.

5. The above results indicate that one cannot efficiently utilize CL constraints even though they have many potential uses. We identify a useful restricted CNF version of ML and CL constraints for which the feasibility problems can be solved efficiently. We call this the choice-set form of constraints.

6. We empirically illustrate that the feasibility problem is not a rare phenomenon and occurs extensively. This has been noted before [Wagstaff 2002], but not published in the literature. While our analytical results indicate the worst-case behavior of the feasibility problem, we provide and experimentally verify an explanation of which **instances** of the feasibility problem are difficult using results from graph coloring.

[1] Definitions of these forms appear in Section 7.

Constraint	Conjunction	CNF version	DNF version	Choice Set
Must-Link	**P**	**NP**-complete	**P**	**P**
Cannot-Link	**NP**-Complete	**NP**-complete	**NP**-Complete	**P**

Table 1: Complexity of Feasibility Problem for Instance-Level Constraints

Constraint	Complexity
δ-constraint	**P**
ϵ-constraint	**P**
Must-Link and δ	**P**
Must-Link and ϵ	**NP**-complete
δ and ϵ	**P**
Cannot-Link and any other constraint	**NP**-complete

Table 2: Complexity of Feasibility Problems for Cluster-Level and Combinations of Constraints

In the above list of results, Items 1, 2 and parts of 3 were presented in the conference version of this paper [Davidson and Ravi 2005a]. Here, we present detailed proofs of the results in Item 3. The results in Items 4, 5 and 6 are new.

The remainder of this paper is organized as follows. The next subsection is a short primer on complexity results and their interpretation. Those familiar with the area can skip this subsection. Next, we consider the complexity of the feasibility problem for conjunctions of instance level constraints and then empirically illustrate the issues arising in the context of algorithms that attempt to find feasible solutions at each iteration. Subsequent sections examine the feasibility problem for δ and ϵ constraints and combinations of the four types of constraints. A summary of our results for the feasibility problem can be found in Tables 1 and 2.

Our work in this paper is limited to the feasibility problem for non-hierarchical clustering. The reader is directed to [Davidson and Ravi 2005b] for results on hierarchical clustering.

1.3 A Short Primer on Complexity Results

A combinatorial problem is **efficiently solvable** if there is an algorithm for the problem with a running time that is polynomial in the size of the input. Examples of efficiently solvable problems include finding minimum spanning trees, finding shortest paths, etc. [Cormen et' al 2001]. Such problems are said to be in the computational class **P**, and their membership in **P** is proven by presenting an appropriate polynomial time algorithm. For some problems such as the Satisfiability problem (SAT) or the minimum vertex coloring problem, no efficient algorithms are known, and it is widely believed that no such algorithms exist [Garey and Johnson 1979]. These problems, which belong to the class of **NP-complete** problems, have the property that either all of them can be solved efficiently or none of them can be solved efficiently. All known algorithms for these problems have a worst-case running time which is exponential in the size of the input. Such problems are

commonly referred to as "computationally intractable" problems.

The theory of **NP**-completeness applies to *decision problems*, where the answer is either "Yes" or "No". (For example, the feasibility problem for clustering is a decision problem.) The major step in proving that a problem Π is **NP**-complete involves the development of a polynomial time *reduction* from a known **NP**-complete problem Π' to Π. Such a reduction shows how each instance I' of Π' can be efficiently transformed into an instance I of Π such that the answer to I' is "Yes" if and only if the answer to I is "Yes". Thus, such a reduction shows that the computationally intractable problem Π' is embedded in Π.

By their very nature, **NP**-completeness results are for the "worst-case"; they point out that among all instances of such a problem, there is a subset that is "hard". Thus, showing that a problem is **NP**-complete does *not* mean that *all* instances of the problem are hard to solve; rather, the result points out that one should not expect an efficient algorithm to correctly solve all instances of the problem. By imposing some restrictions on the problem, one can usually identify groups of instances that can be solved efficiently. We will illustrate this in the context of the graph coloring problem in Section 3.3.2.

2 Preliminary Definitions and Terminology

In the remainder of this paper, we use the term "points" to refer to the items[2] that must be grouped into clusters. In the **geometric** versions of clustering problems, the set S consists of points from Euclidean space and the distance between a pair of points is their Euclidean distance. In the **nongeometric** version, S consists of items (i.e., points without coordinates), and an explicit distance function specifies the distance between each pair of items. Let $S = \{s_1, s_2, \ldots, s_n\}$ denote the given set of points which must be partitioned into clusters. For any pair of points s_i and s_j in S, the distance between them is denoted by $d(s_i\,, s_j)$. The distance function is assumed to be symmetric so that $d(s_i\,, s_j) = d(s_j, s_i)$.

Although the ML and CL constraints mentioned earlier provide a useful way to specify background information to a clustering algorithm, it is natural to ask whether there is a feasible clustering that satisfies all the given constraints. The complexity of satisfying the constraints will determine how to incorporate them into existing clustering algorithms.

The feasibility problem has been studied for other types of constraints or measures of quality [Hansen and Jaumard 1997]. For example, the clustering problem where the quality is measured by the maximum cluster diameter can be transformed in an obvious way into a constrained clustering problem. Feasibility problems for such constraints have received a lot of attention in the literature (see for example [Garey and Johnson 1979, Gonzalez 1985, Hansen and Jaumard 1997]).

Typical specifications of clustering problems include an integer parameter k that gives the number of required clusters. We will consider a slightly more general version, where the problem

[2]In the clustering literature, the term "instances" is also used to refer to the items which must be grouped into clusters. In this paper, we use the term "instance" in the sense that is used in the literature on computational complexity. For example, an instance of the Graph K-Colorability problem consists of an undirected graph $G(V, E)$ and an integer $K \leq |V|$.

specification includes a lower bound K_ℓ and an upper bound K_u on the number of clusters rather than the exact number k. Without upper and lower bounds, some feasibility problems admit trivial solutions. For example, without a lower bound on the number of clusters, having all the points in a single cluster is a trivial solution to the feasibility problem for a collection of ML-constraints. Likewise, when there is no upper bound on the number of clusters, the feasibility problem under CL-constraints can be solved trivially by having each point in a separate cluster.

Obviously, the lower bound K_ℓ and the upper bound K_u must satisfy the condition $1 \leq K_\ell \leq K_u \leq n$, where n denotes the number of points. We will assume this condition throughout this paper. For the remainder of this paper, we use the following definition of a *feasible clustering*.

Definition 2.1 ***(Feasible Clustering)*** *A feasible clustering is one that satisfies all the given constraints and the upper and lower bounds on the number of clusters.*

For problems involving ML or CL constraints, it is assumed that the collection of constraints $C = \{C_1, C_2, \ldots, C_m\}$ containing the m constraints is given, where each constraint $C_j = (s_{j_1}, s_{j_2})$ specifies a pair of points. For convenience, the feasibility problems for ML and CL constraints will be referred to as **ML-feasibility** and **CL-feasibility** respectively.

3 Complexity of Feasibility for Instance Level Constraints

3.1 Overview

In this section, we investigate the complexity of the ML- and CL-feasibility problems. We show that the ML-feasibility problem can be solved efficiently while the CL-feasibility problem is **NP**-complete in general. Our algorithm for the ML-feasibility problem does not use distance information. So, the algorithm is applicable to inputs where distances do not satisfy the triangle inequality. Further, the **NP**-completeness result for CL-feasibility also does not rely on the coordinates of points or on distances. So, the CL-feasibility problem remains computationally intractable even when the set to be clustered consists of points in $\mathbb{R}^2$.

The goal of the feasibility algorithms presented here and in later sections is to determine whether there is a partition that satisfies all the given constraints; they do not attempt to optimize any objective. Thus, these algorithms may produce solutions with singleton clusters when the constraints under consideration permit such clusters. Whenever the answer to a feasibility question is "Yes", the corresponding algorithm also outputs a feasible solution.

3.2 Feasibility Under Must-Link Constraints

Klein et' al in [Klein et' al 2002], discuss a polynomial time algorithm for a geometric version of the ML-feasibility problem. They considered a more general version of the problem, where the goal is to obtain a new distance function that satisfies the triangle inequality when there is a feasible solution. In our definition of the ML-feasibility problem, no distances are involved. Therefore, a straightforward algorithm whose running time is linear in the number of points and constraints can be developed as discussed below.

Note: Whenever a feasible solution exists, the following algorithm outputs a collection of K_ℓ clusters satisfying all the ML-constraints.

1. Compute the transitive closure of the constraints in C. Let this computation result in r sets of points, denoted by M_1, M_2, ..., M_r.
2. Let $S' = S - \bigcup_{i=1}^{r} M_i$. ($S'$ denotes the subset of points that are not involved in any ML-constraint.)
3. **if** $K_\ell > |S'| + r$ **then** Output "No solution" and **stop**.
   else

   if $r \geq K_\ell$ **then**
   (a) Let $A = (\bigcup_{i=K_\ell}^{r} M_i) \cup S'$.
   (b) Output M_1, ..., $M_{K_\ell - 1}$, A.

   else /* Here, $r < K_\ell \leq r + |S'|$. */
   (a) Let $t = K_\ell - r$. (Note that $t \geq 1$.) Partition S' into t clusters A_1, ..., A_t arbitrarily.
   (b) Output M_1, ..., M_r, A_1, ..., A_t.

Figure 1: Algorithm for the ML-Feasibility Problem

As is well known, ML-constraints are *transitive*. For example, the constraints $ML(x, y)$ and $ML(y, z)$ together imply that the points x, y and z must all be the same cluster. In the formal definition of transitivity given below, we use the notation $ML(A)$, where A is a set of two or more points, to represent the constraint that all the points in A must be in the same cluster.

Definition 3.1 ***(Transitive Property of Must-Link constraints)*** *If S_1 and S_2 are subsets of points such that $S_1 \cap S_2 \neq \emptyset$ and the constraints $ML(S_1)$ and $ML(S_2)$ hold, then the constraint $ML(S_1 \cup S_2)$ must also hold.*

Using the transitivity property, a given collection C of ML constraints can be transformed into an equivalent collection $M = \{M_1, M_2, \ldots, M_r\}$ of constraints, by computing the transitive closure of C. The sets in M are pairwise disjoint and have the following interpretation: for each set M_i ($1 \leq i \leq r$), the points in M_i must all be in the same cluster in any feasible solution. For feasibility purposes, points which are not involved in any ML-constraint can be partitioned into clusters in an arbitrary manner. These facts allow us to obtain a straightforward algorithm for the ML-feasibility problem. The steps of the algorithm are shown in Figure 1. Whenever a feasible solution exists, the algorithm outputs a collection of K_ℓ clusters. The only situation in which the algorithm reports infeasibility is when the lower bound on the number of clusters is too high.

The transitive closure computation (Step 1 in Figure 1) in the algorithm can be carried out as follows. Construct an undirected graph G, with one node for each point appearing in the

constraint set C, and an edge between two nodes if the corresponding points appear together in a ML-constraint. Then, the connected components of G give the sets in the transitive closure. It can be seen that the graph G has n nodes and m edges. Therefore, its connected components can be found in $O(n+m)$ time [Cormen et' al 2001]. The remaining steps of the algorithm can be carried out in $O(n)$ time. The following theorem summarizes the above discussion.

Theorem 3.1 *Given a set of n points and m ML-constraints, the ML-feasibility problem can be solved in $O(n+m)$ time.* ∎

3.3 Feasibility Under Cannot-Link Constraints

3.3.1 NP-Completeness of CL-Feasibility

In general, the CL-feasibility problem is **NP**-complete. Klein et al. [Klein et' al 2002] mention this result, but omit the proof. Our proof uses a straightforward reduction from the GRAPH K-COLORABILITY problem (K-COLOR) defined below. This problem is known to be **NP**-complete [Garey and Johnson 1979].

GRAPH K-COLORABILITY (K-COLOR)

Instance: Undirected graph $G(V, E)$, integer $K \leq |V|$.

Question: Can a color be assigned to each node of G such that the number of colors used is at most K, and for every pair of nodes u and v for which $\{u, v\}$ is in E, the colors assigned to u and v are different?

Theorem 3.2 *The CL-feasibility problem is **NP**-complete.*

Proof: It is easy to see that the CL-feasibility problem is in **NP**, since one can guess a partition of the set S into at most K clusters and verify in polynomial time that the partition satisfies each of the given CL constraints. To prove **NP**-hardness, we use a reduction from the K-COLOR problem. Let the given instance I of K-COLOR problem consist of the undirected graph $G(V, E)$ and integer K. Let $n = |V|$ and $m = |E|$. We construct an instance I' of the CL-feasibility problem as follows. For each node v_i in V, we create a point s_i, $1 \leq i \leq n$. (The coordinates of the points are not specified as they play no role in the proof.) The set S of points is given by $S = \{s_1, s_2, \ldots, s_n\}$. For each edge $\{v_i, v_j\}$ in E, we create the CL-constraint (s_i, s_j). Thus, we create a total of m constraints. We set the lower and upper bound on the number clusters to 1 and K respectively. This completes the construction of the instance I'. It is obvious that the construction can be carried out in polynomial time. It is straightforward to verify that the CL-feasibility instance I' has a solution if and only if the K-COLOR instance I has a solution. ∎

The K-COLOR problem is known to be **NP**-complete for every *fixed* value of $K \geq 3$. From this fact, it follows that the CL-feasibility problem is also **NP**-complete when the lower bound on the number of clusters is 1 and the upper bound is fixed at any value ≥ 3.

Since the CL-feasibility problem is **NP**-complete, it follows that in general, the feasibility problem remains **NP**-complete even when the constraint set includes both ML and CL constraints.

3.3.2 Efficiently Solvable Special Cases of CL-Feasibility

Theorem 3.2 shows that the CL-feasibility problem is, in general, **NP**-complete. To identify some restricted versions of the CL-feasibility problem which can be solved efficiently, we now observe that the CL-feasibility problem can be reduced to the K-Color problem. Note that our previous reduction *from* the K-Color problem established the **NP**-completeness result for CL-feasibility. Let I denote the given instance of the CL-feasibility problem consisting of the set S of points, a collection C of CL constraints and integers K_ℓ and K_u. Construct an instance I' of the K-Color problem consisting of the graph $G(V, E)$ and integer K as follows. For each point s_i in S, create a node v_i in V, and for each CL-constraint (s_i, s_j) in C, create the undirected edge $\{v_i, v_j\}$ in E. Let the number of colors K be set to K_u. It is easy to verify that the resulting K-Color instance I' has a solution iff there is a solution to the CL-feasibility instance I.

This reduction to the coloring problem is useful for several reasons. First, it points out that in practice, one can use known heuristics for graph coloring in choosing the number of clusters. Although the coloring problem is known to be hard to approximate in the worst-case [Feige and Kilian 1998], heuristics that work well in practice are known ([Hertz and de Werra 1987, Campers' et al 1987]). In addition, the reduction also points out that the following two special cases of the CL-feasibility problem can be solved efficiently.

(a) When the upper bound on the number of clusters is two, the CL-feasibility problem corresponds to the problem of determining whether a graph is 2-colorable. Efficient algorithms are known for this problem [Cormen et' al 2001].

(b) Suppose the upper bound K_u on the number of clusters is at least $\Delta + 1$, where Δ is the maximum node degree of the graph constructed (as described above) from the given instance of the CL-feasibility problem. In this case, there is always a feasible solution. This follows from a well known result, called Brooks's Theorem, in graph theory [West 2001]. A statement of this theorem and an efficient algorithm that colors a graph using at most k colors, for any $k \geq \Delta + 1$, is given in Appendix A. In practice, this special case arises when the number of clusters exceeds the maximum number of CL-constraints involving the same point.

4 Is Feasibility Really a Problem? Experimental Results and a Theoretical Explanation

The **NP**-completeness of the CL-feasibility problem points out that when background knowledge is specified in terms of CL constraints, one should not expect to get a feasible solution in each iteration of an efficient algorithm such as k-Means. However, the **NP**-completeness result does not imply that all instances of the CL-feasibility problem are difficult (Section 1.3). Thus, it is of interest to investigate whether the intractability of the feasibility problem is an issue in practice. In this section, we examine experimentally when clustering under CL constraints by themselves and with ML constraints is difficult and when it is easy as a function of the number of constraints. (As shown in Section 3.2, clustering under ML constraints only is efficiently solvable.)

We examine a common method of extending k-Means to ensures that all ML and CL constraints are satisfied at each iteration. In the machine learning and data mining literature, the COP-k-Means algorithm [Wagstaff et' al 2001] is considered the seminal work on incorporating instance level constraints into the k-Means algorithm in a simple and elegant manner. The COP-k-Means algorithm handles ML constraints by first computing the transitive closure and then replacing all the points in each transitive closure set (connected component) with a single point which is the centroid of all the points in that set [Wagstaff 2005]. Since k is typically small, infeasibility due to ML constraints does not occur. The algorithm incorporates CL constraints by changing nearest-centroid assignment to *nearest-feasible-centroid* assignment. This is easily programmed by creating for each point a sorted list of centroids (in increasing order of distances) and progressing down the list until a feasible centroid is found to assign the point [Wagstaff et' al 2001, Basu et' al 2002]. For any point, if the algorithm reaches the bottom of this list and still cannot find a feasible assignment, the algorithm halts with an indication that it cannot find a feasible solution.

We used the following common method for generating ML and CL constraints. Two points from S_l (the set of labeled data points) are drawn randomly. If the labels of the two points agree, then an ML constraint is generated; otherwise, a CL constraint is generated. This method of constraint generation along with the COP-k-Means algorithm in its entirety are shown in Figure 2.

We implemented this algorithm and tried it on several UCI data sets. In our experiments, all records with missing values were removed and k was set to the number of extrinsic classes. We tried both intractable feasibility problem situations: clustering with ML and CL constraints and clustering with only CL constraints. As with all our experiments, the results were averaged over five hundred sets of randomly generated constraints. For each set of constraints, the clustering algorithm was run after a random choice of initial centroids.

Definition 4.1 *We say that a run of the COP-k-Means algorithm* ***converges*** *or succeeds if and only if a feasible solution is found in each iteration of the run.*

With ML and CL constraints, we note a quite unusual phenomenon as others have [Wagstaff 2002] in Figure 3. This result is quite counterintuitive. Initially for a small number of constraints the feasibility problem is "easy" and for the vast majority of the 500 trials, the algorithm converges. However, finding feasible solutions becomes more difficult as the number of constraints increases only to find it then becomes easy again! We believe this is a legitimate phenomenon for the clustering under constraints problems as both Wagstaff [Wagstaff 2002] and ourselves find the same "dipping" graph even though we used different initialization/termination schemes etc.

The algorithm in Figure 2 (and for that matter, any algorithm that attempts to satisfy all the constraints) may not converge for two reasons. Firstly, there maybe no feasible solution. For example, if $k = 2$ and there are three CL-constraints, namely $CL(x, y)$, $CL(y, z)$ and $CL(x, z)$, then there is no feasible solution. Secondly, even when there are feasible solutions, finding these solutions may be computationally intractable. Since we are generating constraints from a consistent source and setting k to be the number of extrinsic labels, infeasibility results are due to the latter reason. To explain the phenomenon observed in Figure 3, we will use Brooks's theorem (Appendix A) and

Input: S_u: unlabeled data, S_l: labeled data, k: the number of clusters to find, q: number of constraints to generate. (In this figure, ML and CL represent respectively the set of must-link and cannot-link constraints generated as part of the algorithm.)

Output: A partition of $S = S_u \cup S_l$ into k clusters so that all the constraints in $C = ML \cup CL$ are satisfied.

1. Initialize ML and CL to $\emptyset$.
2. **loop** q times **do**
   (a) Randomly choose two distinct points x and y from S_l.
   (b) **if** (Label(x) = Label(y)) **then** $ML = ML \cup \{(x, y)\}$ **else** $CL = CL \cup \{(x, y)\}$.
3. Compute the transitive closure of the set ML to obtain the connected components $CC_1, \ldots, CC_r$.
4. For each i, $1 \leq i \leq r$, replace data points in CC_i by their centroid.
5. Randomly generate cluster centroids C_1, ..., C_k.
6. **loop** until convergence **do**
   (a) **for** $i = 1$ **to** $|S|$ **do**
      (a.1) Assign s_i to the nearest feasible centroid.
   (b) Recalculate $C_1, \ldots, C_k$.

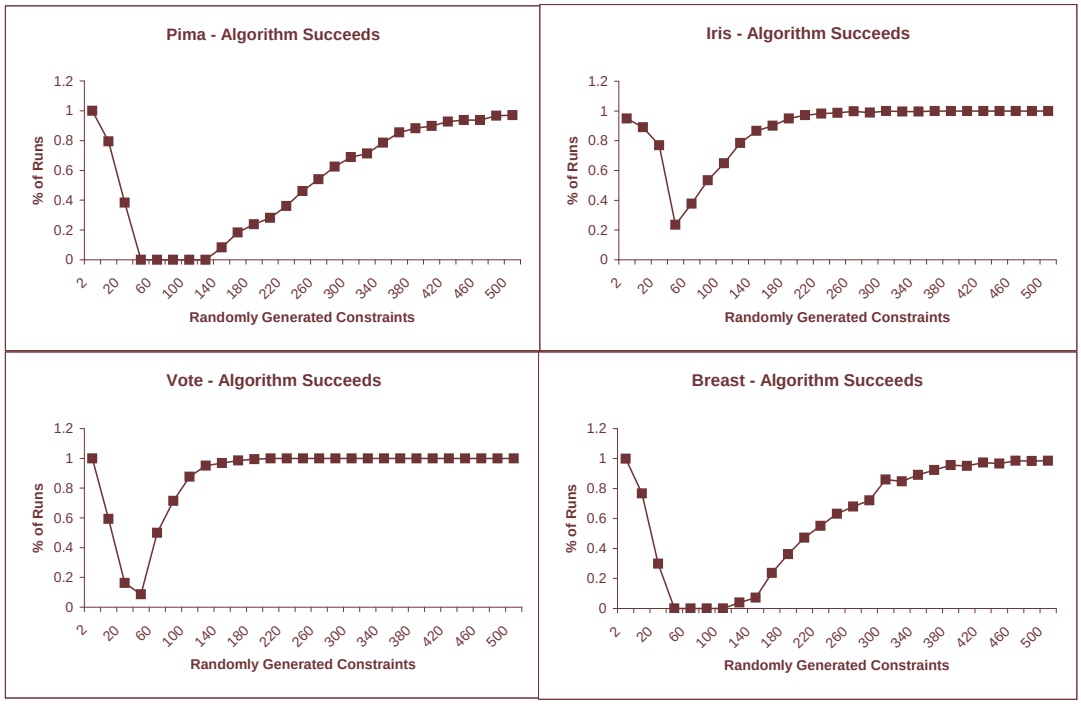
Figure 2: Generating Constraints and Clustering under Constraints Using COP-k-Means

the notion of *entailed constraints*. Brooks's theorem points out that if the maximum node degree of a graph G is Δ and $k \geq \Delta + 1$, then the coloring problem is easy; that is, graph G can always be colored using k colors and such a coloring can be obtained using a simple algorithm (Appendix A). The issue of entailed constraints was first discussed in [Basu et' al 2002], and it refers to non-explicitly provided constraints. For example, the transitive nature of ML constraints (Definition 3.1) leads to entailed constraints. As another example, consider the collection of three constraints $ML(a, b)$, $ML(c, d)$ and $CL(a, c)$. The additional entailed constraints from this collection are $CL(a, d)$, $CL(b, c)$ and $CL(b, d)$. This example can be generalized as follows.

Definition 4.2 ***(Entailed Constraint Property)*** *Let C be a collection of ML and CL constraints. Suppose the transitive closure of the ML constraints in C yields the sets $CC_1, \ldots, CC_r$. If there are points $x \in CC_i$, $y \in CC_j$, with $i \neq j$, such that the constraint $CL(x, y) \in C$, then $\forall a \in CC_j$ and $\forall b \in CC_j$, the entailed constraint $CL(a, b)$ exists.*

The clustering problem involving given and entailed constraints can be represented by a graph $G(V, E)$, where each node represents a transitive closure set or a point which was not part of any ML-constraint. (A node corresponding to a transitive closure set represents all the points in that set.) Each constraint CL(x, y) is represented by an edge which joins the two nodes representing

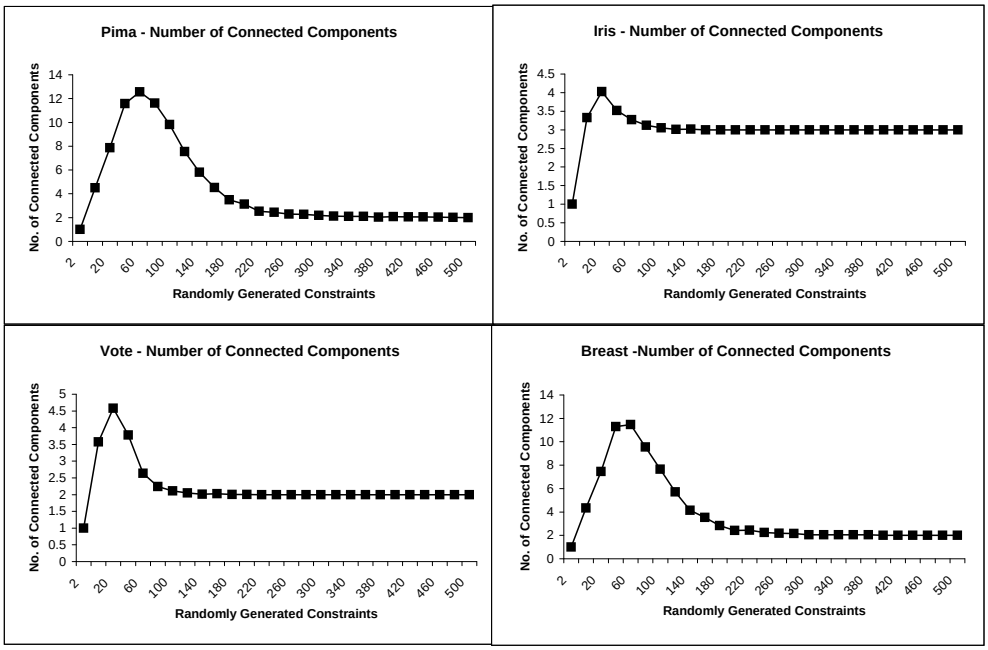
Figure 3: Plot of the proportion of times from 500 independent trials the algorithm in Figure 2 converges for various numbers of randomly chosen ML and CL constraints. Here, k = number of intrinsic classes: Iris (3), Pima (2), Breast (2) and Vote (2).



points x and y. When the maximum node degree Δ of this graph is at most $k-1$, the COP-k-Means algorithm often converges because the nearest-feasible-centroid assignment step can be considered as a coloring algorithm similar to the one based on Brooks's theorem (Appendix A). In difficult instances of the feasibility problem, the value of Δ is larger than $k-1$, and the coloring algorithm based on Brooks's theorem will fail in general. Informally, if the maximum number of CL-constraints involving the same point is at most $k-1$, then the feasibility problem is easy; otherwise, the feasibility problem will be difficult (i.e., the nearest-feasible-centroid assignment will fail in general). This insight can explain the results in Figure 3.

When the number of constraints is small, the number of connected components in the graph representing the clustering problem would also be small. So, the chance of any node being part of more than k CL constraints is small. Thus, the feasibility problem is easy. Conversely, when the number of constraints is large, the number of connected components will be typically the number of extrinsic labels (due to the transitive nature of ML-constraints; see Definition 3.1), and hence the maximum degree of the graph is $k-1$ (i.e., there is a CL-constraint between each pair of connected components). Thus, the maximum node degree Δ and k satisfy the condition $k \geq \Delta+1$, and again the problem is easy. However, when the number of constraints is not too large, the number of connected components is typically much greater than k. This is because many points with the same extrinsic label have not had the opportunity to combine and form a single connected component. Furthermore, the chance of a CL-constraint between two connected components is

Figure 4: Graph of the average number of connected components from 500 independent trials for various numbers of randomly chosen ML and CL constraints. Compare with Figure 3.



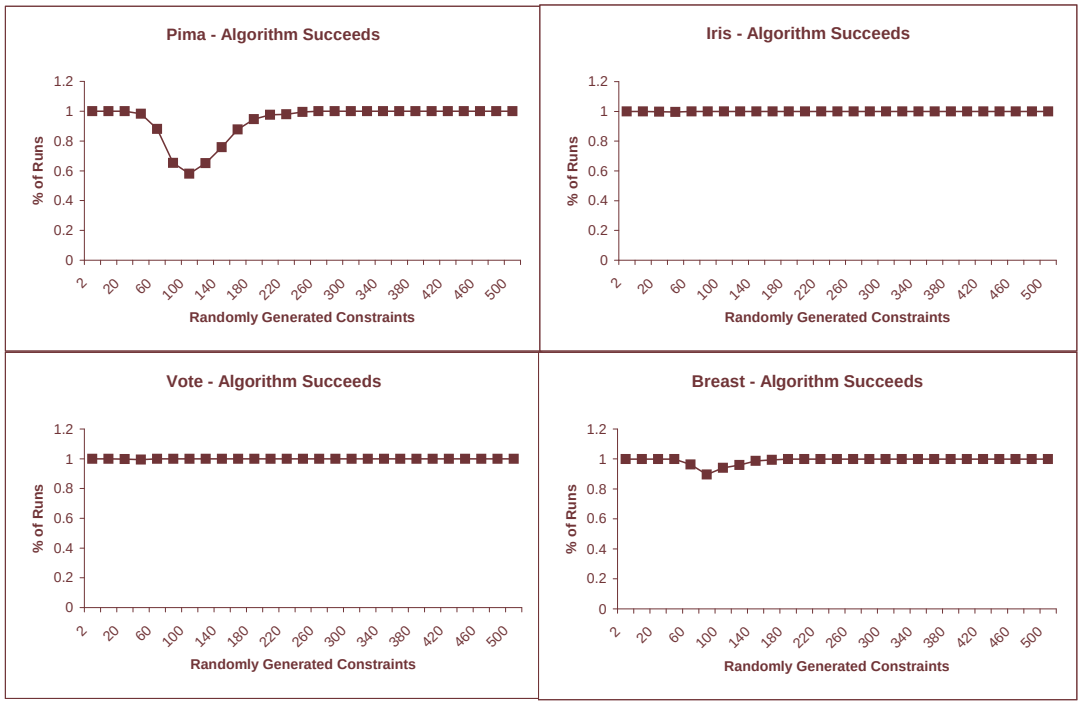
large due to the entailed constraint property. Hence, the maximum degree of the graph is typically larger than k. Figure 4 which shows the number of connected components provides an experimental confirmation of this explanation. Comparing Figures 4 and 3 we see that greatest proportion of times the algorithm could not find a feasible solution (and did not converge) corresponds to the situation when the number of connected components is a maximum. Furthermore, we see that when the number of constraints is large, the number of connected components equals the number of extrinsic labels. Since k is also equal to this value, the problem is easy.

The above discussion points out at least two ways of making the constrained clustering problem easy. Firstly, we can make a point part of only one ML or CL constraint. This, is achieved by sampling the points *without* replacement while generating constraints. Alternatively, we can make k larger than the number of extrinsic classes. Performing the same experiments as before, except with $k = 5$, produces the results shown in Figure 5 and feasibility problems become less pronounced.

Finally, we see that if there are no ML-constraints and only CL-constraints since the maximum graph degree can only increase, the problem gets progressively difficult as the number of constraints increases; see Figure 6.

We can therefore conclude that when there are ML and CL constraints, feasibility issues can occur for intermediate amounts of constraints. When clustering under only CL-constraints, feasibility problems occur for a large number of constraints.

Figure 5: Graph of the proportion of times from 500 independent trials the algorithm in Figure 2 converges for various numbers of randomly chosen ML and CL constraints, with $k = 5$. Compare with Figure 3.



5 Cluster Level Constraints and Feasibility Problems

5.1 Definitions of Constraints

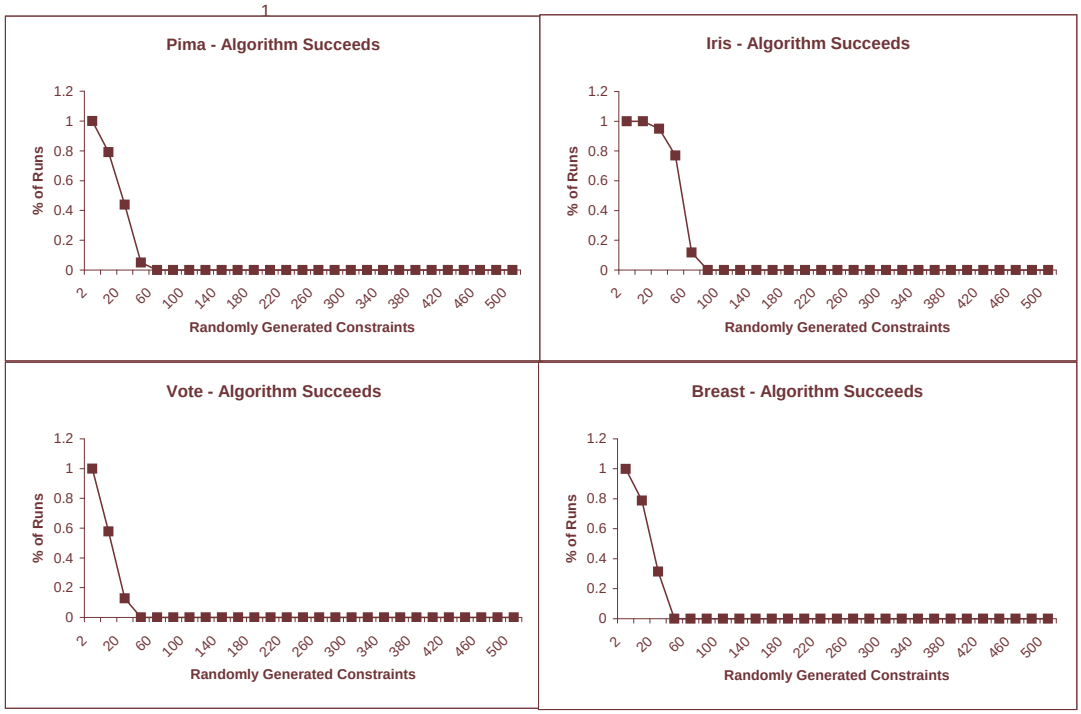
Here, we formally define two new cluster level constraints, called δ- and ϵ-constraints. We also consider an extension of the ϵ-constraint, called the ϵ-path-constraint. Subsequent sections will examine the complexity of the feasibility problems under these constraints.

The **δ-constraint** (or minimum separation constraint) specifies a value $\delta > 0$. In any solution satisfying this constraint, the distance between any pair of points which are in two different clusters must be at least δ. More formally, the δ-constraint requires that for any pair of clusters S_i and S_j $(i \neq j)$, and any pair of points s_p and s_q, where s_p is in S_i and s_q is in S_j, the distance $d(s_p, s_q)$ must be at least δ. Informally, this constraint requires that each pair of clusters must be well separated.

The **ϵ-constraint** specifies a value $\epsilon > 0$ and the feasibility requirement is the following: for any cluster S_i containing two or more points and for any point s_p in S_i, there must be another point s_q in S_i such that $d(s_p, s_q) \leq \epsilon$. Informally, this constraint requires that in any cluster S_i containing two more points, each point in S_i must have another point within a distance of at most ϵ. The ϵ-constraint is similar in principle to the ϵ+minpts criterion used in the DB-SCAN algorithm [Ester et' al 1996]. In their work, the ϵ+minpts criterion is central to the definition of a cluster and an outlier. The focus of our work is on the feasibility problem for the ϵ-constraint.

The **ϵ-path-constraint** [Ding 2005] is an extension of the ϵ-constraint, and the requirement is

Figure 6: Graph of the proportion of times from 500 independent trials the algorithm in Figure 2 converges for various numbers of randomly chosen CL only constraints.



the following: for any cluster S_i containing two or more points and for any pair of distinct points x and y in S_i, there is a sequence $\langle x, a_1, \ldots, a_r, y\rangle$ of points such that each point in the sequence is in S_i and the distance between any adjacent pair of points in the sequence is at most ϵ. It is easier to understand this constraint by considering an undirected graph, which we call the **auxiliary graph**, defined as follows.

Definition 5.1 *Let Q be a set of points and let a value $\epsilon > 0$ be given. The **auxiliary graph** $G(V_Q, E_Q)$ **corresponding to** Q is constructed as follows.*

(a) The node set V_Q has one node for each point in Q.

(b) For any two nodes v_i and v_j in V, the edge $\{v_i, v_j\}$ is in E_Q iff the distance between the points in Q corresponding to nodes v_i and v_j is at most ϵ.

With the above definition, it can be seen that a given partition of S into clusters S_1, S_2, ..., S_k satisfies the ϵ-path constraint if and only if for each cluster S_i with two or more points, the auxiliary graph corresponding to S_i is connected (i.e., consists of only one connected component). The notion of auxiliary graph plays an important role in our algorithms for the feasibility problems for both ϵ and ϵ-path constraints.

1. **for** each point s_i in S **do**

   (a) Find the set $X_i \subseteq S - \{s_i\}$ of points such that for each point x_j in X_i, $d(s_i, x_j) < \delta$.

   (b) For each point x_j in X_i, create the ML-constraint (s_i, x_j).

2. Let C denote the set of all the ML-constraints created in Step 1. Use the algorithm for the ML-feasibility problem (Figure 1) with point set S, constraint set C and the values K_ℓ and K_u.

Figure 7: Algorithm for the δ-Feasibility Problem

5.2 Feasibility Under the δ-Constraint

In this section, we show that the δ-feasibility problem can be solved in polynomial time. The basic idea is simple: if points in different clusters must be at least δ apart, then in any feasible solution, every pair of points s_i and s_j for which $d(s_i, s_j) < \delta$, must be in the same cluster. Thus, given the value of δ, we can create a collection of appropriate ML-constraints and use the algorithm for the ML-feasibility problem. This approach points out that a δ-constraint can be replaced by a conjunction of ML-constraints. The steps of the resulting algorithm are shown in Figure 7.

The running time of the algorithm is dominated by the time needed to complete Step 1, that is, the time to compute the set of ML-constraints. Clearly, this step can be carried out in $O(n^2)$ time, and the number of ML-constraints generated is also $O(n^2)$. Thus, the overall running time of the algorithm is $O(n^2)$. The following theorem summarizes the above discussion.

Theorem 5.1 *For any $\delta > 0$, the feasibility problem under the δ-constraint can be solved in $O(n^2)$ time, where n is the number of points to be clustered.* ∎

5.3 Feasibility Under the ϵ-Constraint

Let the set S of points and the value $\epsilon > 0$ be given. For any point s_p in S, the set Γ_p of **ϵ-neighbors** is given by

$$\Gamma_p \;=\; \{s_q \,:\, s_q \text{ in } S - \{s_p\} \;\text{ and }\; d(s_p, s_q) \leq \epsilon\}.$$

Note that a point is *not* an ϵ-neighbor of itself. Two distinct points s_p and s_q are ϵ-neighbors of each other if $d(s_p, s_q) \leq \epsilon$. The ϵ-constraint requires that in any cluster containing two or more points, each point in the cluster must have an ϵ-neighbor within the same cluster. This observation points out that an ϵ-constraint corresponds to a disjunction of ML-constraints. For example, if $\{s_{i_1}, \ldots, s_{i_r}\}$ denote the ϵ-neighbors of a point s_i, then satisfying the ϵ-constraint for point s_i means that either one or more of the ML-constraints $ML(s_i, s_{i_1})$, ..., $ML(s_i, s_{i_r})$ are satisfied or the point s_i forms a singleton cluster. In particular, any point in S which does not have an ϵ-neighbor must form a singleton cluster.

1. Find the set $S_1 = \{x \ : \ x \text{ in } S \text{ and } x \text{ does not have any } \epsilon\text{-neighbor in } S\}$. Let $t = |S_1|$ and $S_2 = S - S_1$.

2. Construct the auxiliary graph $G(V, E)$ for S_2 (see Definition 5.1). Let G have r connected components (CCs), denoted by G_1, G_2, ..., G_r.

3. Let C_1, C_2, ..., C_t denote the singleton clusters corresponding to points in S_1. Let X_1, X_2, ..., X_r denote the clusters corresponding to the CCs of G. Let $N^* = t + \min\{1, r\}$.

4. **if** $N^* > K_u$ **then** Output "No feasible solution" and **stop**.

5. Execute one of the following cases. (In each case, the output has K_u clusters.)

   Case 1: $K_u = t + r$: Output the $t + r$ clusters $C_1, \ldots, C_t, X_1, \ldots, X_r$.

   Case 2: $K_u < t + r$: Generate and output a solution with K_u clusters as indicated below.

   (a) Merge clusters $X_{K_u - t}$, $X_{K_u - t + 1}$, ..., X_r into a single new cluster $X_{K_u - t}$.
   (b) Output the K_u clusters C_1, C_2, ..., C_t, X_1, X_2, ..., $X_{K_u - t}$.

   Case 3: $K_u > t + r$: Generate $K_u - t - r$ additional singleton clusters from X_1, ..., X_r and output a solution with K_u clusters as indicated below.

   (a) Let $n_i = |X_i|$, $1 \le i \le r$ and let $N = K_u - t - r$.
   (b) Find the smallest integer j, $1 \le j \le r$ such that $\sum_{i=1}^{j}(n_i - 1) \ge N$. (Note that $n_i - 1$ represents the maximum number of additional clusters that can be generated from X_i.)
   (c) Make each point in $\cup_{i=1}^{j-1} X_i$ into a singleton cluster. Let $q_1 = \sum_{i=1}^{j-1} n_i$. Let Y_1, ..., Y_{q_1} denote the singleton clusters created in this step.
   (d) Let $q_2 = N - \sum_{i=1}^{j-1}(n_i - 1)$. (The value q_2 is the number of additional clusters to be created from X_j.) If $q_2 = n_j - 1$, then choose q_2 points (arbitrarily) from X_j and form a singleton cluster from each point. Otherwise (i.e., $q_2 < n_j - 1$), create q_2 new singleton clusters from X_j as follows.

      (i) Construct a spanning tree T_j for G_j.
      (ii) **for** $\ell = 1$ **to** q_2 **do**
         Delete a leaf node v from T_j and create a singleton cluster Z_ℓ from the point corresponding to v.

      Let Z_1, ..., Z_{q_2} denote the new singleton clusters created in Step (d).
   (e) Output the K_u clusters C_1, ..., C_t, Y_1, ..., Y_{q_1}, Z_1, ..., Z_{q_2}, X_j, ..., X_r.

Figure 8: Algorithm for the ϵ-Feasibility Problem

The steps of our algorithm for the ϵ-feasibility problem are shown in Figure 8. The key steps are finding the set S_1 of points which don't have any ϵ-neighbors and the construction of the auxiliary graph for the set $S_2 = S - S_1$. The following lemma establishes the correctness of the algorithm.

Lemma 5.1 *Let N^* be as defined in Step 3 of the algorithm in Figure 8.*

(a) If $N^* > K_u$, then there is no solution to the ϵ-feasibility problem.

(b) If $N^* \leq K_u$, then there is a solution to the ϵ-feasibility problem with K_u clusters.

Proof:

Part (a): Note that $N^* = t + \min\{1, r\}$, where t is the number of points without ϵ-neighbors and r is the number of connected components of the auxiliary graph for the set S_2. By the definition of ϵ-constraint, each point without any ϵ-neighbor must be in a singleton cluster. Thus, any feasible solution must have at least t clusters. Further, if $r \geq 1$, at least one additional cluster is needed for the points in S_2. Therefore, there is no solution to the ϵ-feasibility problem when $N^* > K_u$.

Part (b): Suppose $N^* \leq K_u$. We will show that there is a solution with K_u clusters by considering the following three cases.

Case 1: Suppose $K_u = t + r$.

In this case, there is a solution with $t + r$ clusters consisting of the clusters $C_1, \ldots, C_t, X_1, \ldots, X_r$. It is easy to verify that this solution satisfies the ϵ-constraint.

Case 2: Suppose $K_u < r + t$.

In this case, a solution with K_u clusters can be constructed by merging clusters $X_{K_u - t}, \ldots, X_r$ into a new cluster (say, $X_{K_u - t}$). The solution then consists of clusters $C_1, \ldots, C_t, X_1, \ldots, X_{K_u - t}$. Again, it is easy to verify that this solution satisfies the ϵ-constraint.

Case 3: Suppose $K_u > r + t$.

In this case, we can increase the number of clusters to K_ℓ by splitting some of the clusters $X_1, \ldots, X_r$ to form $K_u - t - r$ additional clusters. From each cluster X_i, we can create $|X_i| - 1$ *additional* (singleton) clusters. Thus, from clusters $X_1, \ldots, X_r$, we can create $\sum_{i=1}^{r}(|X_i| - 1) = n - t - r$ additional clusters. Since $K_u \leq n$, clusters $X_1, \ldots, X_r$ have enough points to create $K_u - t - r$ additional singleton clusters. One simple way to achieve this is to split all the clusters $X_1, \ldots X_{j-1}$, for an appropriate j, into singletons and then create the correct number of singleton clusters from X_j. To facilitate the creation of singleton clusters from X_j, we construct a spanning tree for the connected component G_j (corresponding to X_j). The advantage of a spanning tree is that we can remove a leaf node v from the tree and create a new singleton cluster containing the point corresponding to v. Since each tree has at least two leaves and removing a leaf will not disconnect a tree, this method will increase the number of clusters by exactly one at each step. Thus, by repeating the step an appropriate number of times, the number of clusters can be made equal to K_u. ∎

It can be seen that the running time of the algorithm in Figure 8 is dominated by the time needed for Steps 1 and 2. Step 1 can be implemented to run in $O(n^2)$ time by finding the ϵ-neighbor set for each point. Since the number of ϵ-neighbors for each point in S_2 is at most $n-1$, the construction of the auxiliary graph and finding its CCs (Step 2) can also be done in $O(n^2)$ time. So, the overall running time of the algorithm is $O(n^2)$. The following theorem summarizes the above discussion.

Theorem 5.2 *For any $\epsilon > 0$, the feasibility problem under the ϵ-constraint can be solved in $O(n^2)$ time, where n is the number of points to be clustered.* ∎

5.4 Feasibility Under the ϵ-Path-Constraint

A feasibility algorithm for the ϵ-path-constraint can be obtained by making the following minor modifications to our feasibility algorithm for ϵ-constraint shown in Figure 8.

1. In Step 3, N^*, the minimum number of clusters needed to satisfy the ϵ-path-constraint is given by $N^* = t + r$. This is because every point in S_1 must form a singleton cluster, and merging any pair of clusters X_i and X_j violates the ϵ-path-constraint.

2. With $N^* = t + r$, Step 4 correctly reports infeasibility when $K_u < t + r$. Thus, Case 2 of Step 5 cannot arise. Hence, Step 6 of the algorithm has only two cases, namely $K_u = t + r$ and $K_u > t + r$. For each of these cases, the computational steps are identical to those shown in Figure 8.

We thus conclude:

Theorem 5.3 *For any $\epsilon > 0$, the feasibility problem under the ϵ-path–constraint can be solved in $O(n^2)$ time, where n is the number of points to be clustered.* ∎

6 Feasibility Under Combinations of Constraints

6.1 Overview

In this section, we consider the feasibility problem under combinations of ML, CL, ϵ and δ constraints. Since the CL-feasibility problem is **NP**-hard, the feasibility problem for any combination of constraints involving CL-constraints is, in general, computationally intractable. So, our focus is on combinations of ML, δ and ϵ constraints. We show that the feasibility problem remains efficiently solvable when both ML-constraints and a δ constraint are considered together as well as when δ and ϵ constraints are considered together. Interestingly, when ML-constraints are considered together with an ϵ constraint, we show that the feasibility problem is **NP**-complete even though for each constraint type individually the feasibility problem is tractable. This result points out that when ML, δ and ϵ constraints are all considered together, the resulting feasibility problem is also **NP**-complete in general. Tables 1 and 2 and summarize our feasibility results.

6.2 Combination of Must-Link and δ Constraints

We begin by considering the combination of ML-constraints and a δ constraint. As mentioned in Section 5.2, the effect of the δ-constraint is to contribute a collection of ML-constraints. Thus, we can merge these ML-constraints with the given ML-constraints, and then ignore the δ-constraint. The resulting feasibility problem involves only ML-constraints. Hence, we can use the algorithm from Section 3.2 to solve the feasibility problem in polynomial time. For a set of n points, the δ constraint may contribute at most $O(n^2)$ ML-constraints. Further, since each given ML-constraint involves two points, the number of given ML-constraints is also $O(n^2)$. Thus, the total number of ML-constraints due to the combination is $O(n^2)$. Now, the following result is a direct consequence of Theorem 3.1.

Theorem 6.1 *Given a set of n points, a value $\delta > 0$ and a collection C of ML-constraints, the feasibility problem for the combination of ML and δ constraints can be solved in $O(n^2)$ time.* ∎

6.3 Combination of Must-Link and ϵ Constraints

As shown in the previous sections, the ML-feasibility and ϵ-feasibility problems are efficiently solvable. Here, we show that, in general, the combination of ML and ϵ constraints leads to a computationally intractable feasibility problem. For convenience, we will refer to this as the **ML-ϵ-feasibility** (MLEF) problem. The intractability of the MLEF problem is established by showing that the combination of ML and ϵ constraints has the power to model a set cover problem, which is known to be **NP**-complete.

We note that in the absence of the lower bound (K_ℓ) on the number of required clusters, the MLEF problem can be solved by the following simple algorithm.

1. Find the set S_1 of points which have no ϵ-neighbors (Step 1 of Figure 8).
2. If some point in S_1 is also involved in an ML constraint, then output "No solution" and stop.
3. Compute the quantity N^* as defined in Step 3 of Figure 8.
4. If $N^* > K_u$, then output "No solution". Otherwise, output the solution with N^* clusters. (This solution consists of the t singleton clusters and one cluster containing all the points in $S - S_1$.)

Thus, the computational intractability of the MLEF problem also relies on the specification of a suitable lower bound on the number of clusters.

6.3.1 The Nongeometric Case

For expository reasons, we first prove the **NP**-completeness result for the nongeometric case. (However, the specified distances between pairs of points do satisfy the triangle inequality.) The proof for the nongeometric case makes it easier to understand how a set cover problem can be modeled by the MLEF problem. Subsequently, by starting from an appropriate version of the set cover problem

and using a more complicated construction, we will show that the intractability result holds even when the set to be clustered consists of points from $\mathbb{R}^2$ and the distance between each pair of points is the Euclidean distance.

We use the following problem (called X3C) in our proofs. This problem is known to be **NP**-complete [Garey and Johnson 1979].

Exact Cover by 3-Sets (X3C)

Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$, where $n = 3t$ for some positive integer t and a collection $T = \{T_1, T_2, \ldots, T_m\}$ of subsets of X such that $|T_i| = 3$, $1 \leq i \leq m$.

Question: Does T contain a subcollection $T' = \{T_{i_1}, T_{i_2}, \ldots, T_{i_t}\}$ with t sets such that the union of the sets in T' is equal to X?

Note that $t < m$ and that the sets in T are not necessarily disjoint. Since $n = 3t$ and each subset in T contains exactly three elements, any solution T' to the X3C problem consists of a collection t sets that are pairwise disjoint. Thus, in such a solution, each element of X appears in exactly one of the subsets in T'. If element $x_i \in X$ appears in set $T_j \in T'$, we say that T_j **covers** x_i.

Theorem 6.2 *The nongeometric case of the ML-ϵ-feasibility (MLEF) problem is **NP**-complete.*

Proof: It is easy to see that the MLEF problem is in **NP** since one can guess a partition of S and verify that the partition satisfies the ML constraint, the ϵ-constraint and the bounds on the number of clusters. The proof of **NP**-hardness is by a reduction from the X3C problem. Consider an instance I of the X3C problem consisting of the set X and the collection T of 3-element subsets of X. We construct an instance I' of the MLEF problem as follows. For each element x_i in X, construct a point p_i, $1 \leq i \leq n$. For each set T_j in T, construct a point q_j, $1 \leq j \leq m$. The set S to be clustered has $n + m$ points and is given by $S = \{p_1, p_2, \ldots, p_n, q_1, q_2, \ldots, q_m\}$. The distances between pairs of points are chosen as follows.

$$\begin{array}{lcll} d(p_i, p_j) & = & 2 & \text{for all } i, j,\ i \neq j \\ d(q_i, q_j) & = & 2 & \text{for all } i, j,\ i \neq j \\ d(p_i, q_j) & = & 1 & \text{for all } i, j, \text{ if } x_i \text{ in } T_j \\ & = & 2 & \text{for all } i, j, \text{ if } x_i \text{ not in } T_j. \end{array}$$

Since each distance is either 1 or 2, the distances satisfy the triangle inequality.

The MLEF instance I' has the following $n - 1$ ML constraints: $\text{ML}(p_1, p_2)$, $\text{ML}(p_2, p_3)$, $\ldots$, $\text{ML}(p_{n-1}, p_n)$. Thus, to satisfy all of these ML constraints, the n points p_1, p_2, $\ldots$, p_n must be in the same cluster. The value of ϵ is chosen as 1. The lower bound K_ℓ and the upper bound K_u on the number of clusters are both chosen as $m - t + 1$. This completes the construction of the MLEF instance I'. Clearly, this construction can be done in polynomial time. We now show that there is a solution to the MLEF instance I' iff there is a solution to the X3C instance I.

Part 1: Suppose there is a solution T' to the X3C instance I given by $T' = \{T_{i_1}, T_{i_2}, \ldots, T_{i_t}\}$. We construct a solution consisting of $m - t + 1$ clusters to the MLEF instance I' as follows.

(a) Cluster C_1, which consists of $n+t$ points, is given by $C_1 = \{p_1, p_2, p_n, q_{i_1}, q_{i_2}, \ldots, q_{i_t}\}$.

(b) For each set T_j in $T - T'$, there is a singleton cluster consisting of the point q_j. There are $m-t$ such clusters, and they are denoted by $C_2, C_3, \ldots, C_{m-t+1}$.

Obviously, this solution to instance I' satisfies the bounds on the number of clusters. It also satisfies the ML constraints since all the points p_i, $1 \leq i \leq n$, are in the same cluster, namely C_1. We need to check the ϵ-constraint only for C_1, since all others are singleton clusters. To see that the ϵ-constraint holds for C_1, note that T' is a set cover for the X3C instance I. Thus, for each point p_i in C_1, there is some point q_j such that $d(p_i, q_j) = \epsilon = 1$. Likewise, since each set T_j covers some elements of the set X, for each point q_j in C_1, there is some point p_i in C_1 such that $d(p_i, q_j) = \epsilon = 1$. Therefore, the ϵ-constraint is satisfied for each of the points in C_1. Thus, we have a feasible solution to the MLEF instance I'.

Part 2: Suppose there is a solution to the MLEF instance I'. We have the following lemma.

Lemma 6.1 *Every solution to MLEF instance I' has $m-t$ singleton clusters and one cluster with $n+t$ points.*

Proof of lemma: Consider any solution to I'. Let $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, p_2, \ldots, q_m\}$. Because of the ML constraints, the set P must be entirely in some cluster, say C_1, in the solution. Since $d(p_i, p_j) = 2$ for $i \neq j$ and $\epsilon = 1$, no two points in P are ϵ-neighbors of each other. Thus, to satisfy the ϵ-constraint, C_1 must also contain some nonempty subset of Q. Further, each set T_j in T has only three elements; in other words, each point q_j can serve as an ϵ-neighbor for only three points of P. Therefore, at least $t = n/3$ points of Q must be in C_1 to satisfy the ϵ-constraint for the points in set P. Suppose there are more than t points of Q in C_1. Then, at most $m-t-1$ points of Q are not in C_1. Thus, including C_1, the maximum number of possible clusters is $m-t$. This contradicts the assumption that the solution contains $m-t+1$ clusters. Thus, C_1 contains all the n points of P and exactly t points of Q, for a total of $n+t$ points. To have $m-t+1$ clusters, each of the remaining $m-t$ points of Q must form a singleton cluster. ∎

To construct a solution to X3C instance I, consider the given solution to the MLEF instance I'. In that solution, let C_1 be the cluster with $n+t$ points. As argued above, C_1 has exactly $t = n/3$ points from the set Q. Let $Q' = \{q_{j_1}, q_{j_2}, \ldots, q_{j_t}\}$ denote the subset of Q that is contained in C_1. We claim that the collection $T' = \{T_{j_1}, T_{j_2}, \ldots, T_{j_t}\}$ is a solution to the X3C instance I. This can be seen from the fact that each point q_{j_ℓ} in Q', $1 \leq \ell \leq t$, serves as the ϵ-neighbor to exactly three points of P. Thus, by our construction, each set T_{j_ℓ}, $1 \leq \ell \leq t$, covers exactly three elements of X. We thus have a solution to the X3C instance I, and this completes the proof of Theorem 6.2. ∎

6.3.2 Extension to the Euclidean Case

We now extend the above proof to show that the MLEF problem remains **NP**-complete even when the points to be clustered are from $\mathbb{R}^2$ and the distances are Euclidean distances. To prove this

result, we use a reduction from the following version of the X3C problem, which is also known to be **NP**-complete [Dyer and Frieze 1986].

Planar Exact Cover by 3-Sets (PX3C)

Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$, where $n = 3t$ for some positive integer t and a collection $T = \{T_1, T_2, \ldots, T_m\}$ of subsets of X such that $|T_i| = 3$, $1 \leq i \leq m$. Each element x_i in X appears in at most three sets in T. Further, the bipartite graph $G(V_1, V_2, E)$, where V_1 and V_2 are in one-to-one correspondence with the elements of X and the 3-element sets in T respectively, and an edge $\{u, v\}$ is in E iff the element corresponding to u appears in the set corresponding to v, is planar.

Question: Does T contain a subcollection $T' = \{T_{i_1}, T_{i_2}, \ldots, T_{i_t}\}$ with q sets such that the union of the sets in T' is equal to X?

Theorem 6.3 *The ML-ϵ-feasibility problem is **NP**-complete even when the points to be clustered are from $\mathbb{R}^2$ and the distance metric is Euclidean.*

Proof: The membership in **NP** is obvious. To prove **NP**-hardness, we use a reduction from PX3C. Consider the planar (bipartite) graph G associated with the PX3C problem instance. From the specification of the problem, it follows that each node of G has a degree of at most three. Every planar graph with N nodes and maximum node degree three can be embedded on an orthogonal $N \times 2N$ grid such that the nodes of the graph are at grid points, each edge of the graph is a path along the grid, and no two edges share a grid point except for the grid points corresponding to the graph vertices. Moreover, such an embedding can be constructed in polynomial time [Tamassia and Tollis 1989]. The points and constraints for the feasibility problem are created from this embedding.

Assume that using suitable scaling, each grid edge is of length 1. Choose the bottom left corner of the grid as the origin $(0, 0)$. Thus, each grid point can be assigned x and y coordinates. Note that each edge e of G joins a set node to an element node. Consider the path along the grid for each edge of G. Introduce a new point in the middle of each grid edge in the path. (The x and y coordinates of such points will be multiples of 1/2.) For each edge e of G, this scheme yields the set S_e of points in the grid path corresponding to e, including the new middle point for each grid edge. The set of points in the feasibility instance is the union of the sets S_e, e in E. Let S'_e be obtained from S_e by deleting the point corresponding to the element node of the edge e. For each edge e, we introduce an ML-constraint involving all the points in S'_e. We also introduce an ML-constraint involving all the points corresponding to the elements nodes of G. We choose the value of ϵ to be 1/2. The lower and upper bound on the number of clusters are both set to $m - t + 1$.

By the above construction, each set T_j in the PX3C instance is represented by a set of points which is the union of the sets of points representing the three edges that join the node corresponding to T_j to the three nodes corresponding to the elements in T_j. Since $\epsilon = 1/2$, all the points in the set corresponding to each set T_j have ϵ-neighbors. The only points for which ϵ-neighbors need to be chosen are the points corresponding to the elements of the set X.

With the above correspondence between set T_j and the set of grid points representing T_j ($1 \leq j \leq m$), the proof that there is a solution to the PX3C instance I iff there is a solution to the MLEF instance I' is similar to that presented in the proof of Theorem 6.2. In particular, if $T' = \{T_{j_1}, \ldots, T_{j_t}\}$ is a solution to the PX3C instance I, then the solution to the MLEF instance consists of the following $m - t + 1$ clusters: one cluster contains all the points corresponding to the elements x_1, ..., x_n and all the points corresponding to the sets $\{T_{j_1}, \ldots, T_{j_t}\}$ and each of the remaining $m - t$ clusters contains the points corresponding to a set T_j in $T - T'$. Likewise, if there is a solution to the MLEF instance I', the chosen ML and ϵ constraints force such a solution to contain one cluster containing all the points corresponding to the elements and all the points corresponding to t of the subsets in T; those subsets together form a cover for the set X. ∎

6.4 Combination of δ and ϵ Constraints

We abbreviate the feasibility problem for the combination of δ and ϵ constraints as DEF. In this section, we show that the DEF problem can be solved in polynomial time. It is convenient to consider this problem under two cases, namely $\delta \leq \epsilon$ and $\delta > \epsilon$.

6.4.1 Algorithm for the case where $\delta \leq \epsilon$

When $\delta \leq \epsilon$, our algorithm is based on the following simple observation: any pair of points which are separated by a distance less than δ are also ϵ-neighbors of each other. This observation allows us to reduce the feasibility problem for the combination to one involving only the ϵ constraint as indicated below.

1. Using the δ-constraint, create a collection C of ML constraints as explained in Section 5.2.
2. Construct the transitive closure sets C_1, ..., C_r corresponding to the constraint set C.
3. For each set C_i, create a new point x_i, $1 \leq i \leq r$. (We don't need coordinates for the new points; a new distance function will be specified in Step 5.) Let $X = \{x_1, \ldots, x_r\}$.
4. Let $Y = \{y_1, \ldots, y_t\}$ denote the set of points of S which are not involved in any of the ML constraints in C.
5. Let $S' = X \cup Y$. Define a distance function d' for S' as follows. (Note that d represents the distance function for the original point set S.)

$$\begin{aligned} d'(x_i, x_j) &= \epsilon, \ 1 \leq i < j \leq r \\ d'(y_i, y_j) &= d(y_i, y_j), \ 1 \leq i < j \leq r \\ d'(x_i, y_j) &= \min\{d(s_p, y_j) : s_p \text{ in } C_i\}. \end{aligned}$$

6. Solve the ϵ-feasibility problem for the point set S' under the distance function d'.

The justification for Step 3 is the following. Since $|C_i| \geq 2$ and $\delta \leq \epsilon$, the ϵ-constraint is satisfied for all the points in C_i, $1 \leq i \leq r$. In other words, we may treat each C_i as a single point. After

Step 3, we need to focus only on satisfying the ϵ-constraint for points which were not involved in any of the constraints in C.

In Step 5, the reason for setting $d'(x_i, x_j) = \epsilon$ for all i and j is that any pair of clusters C_i and C_j can be merged without violating the ϵ-constraint. Further, we set $d'(x_i, y_j)$ to be the minimum of the distances between y_j and the points in C_i because as long as one of the points in C_i is within a distance of ϵ from y_j, we can merge y_j with C_i without violating the ϵ-constraint.

Thus, with the new distance function d', we can ignore the δ constraint, and the DEF problem reduces to the ϵ-feasibility problem for the set S' under the distance function d'.

We can estimate the running time of the above algorithm as follows. Steps 1, 2 and 3 run in $O(n^2)$ time since the dominant part is due to the creation of the constraint set C and the transitive closure computation. Step 4 needs only $O(n)$ time. Step 5 runs in time $O(n^2)$ since the number of distances to be created is $O(n^2)$. As explained earlier, the ϵ-feasibility test in Step 6 can be done in $O(n^2)$ time. Therefore, when $\delta \leq \epsilon$, the above algorithm for the DEF problem runs in $O(n^2)$ time.

6.4.2 Algorithm for the case where $\delta > \epsilon$

Now consider the DEF problem for the case when $\delta > \epsilon$. Suppose we create a collection C of ML constraints corresponding to the δ-constraint and construct the transitive closure sets $C_1, \ldots, C_r$ for the constraint set C. When $\delta > \epsilon$, we have the following observation.

Observation 6.1 *For any instance of the DEF problem, where $\delta > \epsilon$, the following statements hold.*

(a) For any point s_x in C_i, each ϵ-neighbor of s_x (if any) must also be in the same set C_i.

(b) Any point s_j of S which is not involved in any of the constraints in C has no ϵ-neighbor; thus, each such point must form a singleton cluster in any feasible solution. ■

Using the above observation, the steps of our DEF algorithm for the case when $\delta > \epsilon$ are as follows.

1. Create a collection C of ML constraints corresponding to the δ-constraint and construct the transitive closure sets $C_1, \ldots, C_r$.

2. If for some cluster C_i $(1 \leq i \leq r)$, there is a point s_x in C_i and s_x does not have an ϵ-neighbor in C_i, then output "No solution" and stop.

3. Let $Y = \{y_1, \ldots, y_t\}$ denote the set of points of S which are not involved in any of the ML constraints in C.

4. $N_{\min} = t + \min\{1, r\}$ and $N_{\max} = t + r$. If $K_\ell > N_{\max}$ or $K_u < N_{\min}$ then output "No solution" and stop.

5. Construct a solution with K_u clusters as follows. If $K_u \geq t + r$, output the $t + r$ clusters consisting of $\{y_1\}, \ldots, \{y_t\}, C_1, \ldots, C_r$. If $K_u < t + r$, then combine sets $C_{K_u - t - r - 1}, \ldots C_r$ into a single set $C_{K_u - t}$ and output the K_u clusters $\{y_1\}, \ldots, \{y_t\}, C_1, \ldots, C_{K_u - t}$.

In the above algorithm, the justification for Step 2 is Part (a) of Observation 6.1. In Step 3, the values $N_{\min}$ and $N_{\max}$ represent the largest and the smallest number of clusters in any feasible solution. The expressions for $N_{\min}$ and $N_{\max}$ rely on Part (b) of Observation 6.1 and the fact that splitting any of the sets C_i, $1 \leq i \leq r$, into two or more clusters would violate the δ-constraint. However, any pair of sets C_i and C_j can be merged into a single cluster. The other steps of the algorithm carry out straightforward checks to make sure that the number of clusters satisfies the bounds K_ℓ and K_u.

The running time of the above algorithm is dominated by the time to complete Steps 1 and 2. It is easy to verify that these steps can be implemented to run in $O(n^2)$ time. Therefore, the running time of the algorithm for the DEF problem for the case when $\delta > \epsilon$ is also $O(n^2)$. The following theorem summarizes the above discussion.

Theorem 6.4 *Given a set of n points and values $\delta > 0$ and $\epsilon > 0$, the feasibility problem for the combination of δ and ϵ constraints can be solved in $O(n^2)$ time.* ∎

7 More General Forms of Instance-Level Constraints

7.1 Definitions, Motivation and Overview

An ML or CL constraint set C can also be viewed as specifying a single constraint which is the conjunction of all the individual ML or CL constraints in C. This view suggests more general forms for ML and CL constraints. In particular, analogs of conjunctive and disjunctive normal forms (respectively CNF and DNF) for Boolean expressions (see for example [Garey and Johnson 1979]) can be defined for ML and CL constraints.

Definition 7.1 *The definitions of CNF and DNF versions of ML constraints are as follows.*

*(a) A **clause** C_i is a* disjunction *of one or more ML-constraints; that is,*

$$C_i = ML(x_1, y_1) \vee ML(x_2, y_2) \vee \ldots \vee ML(x_r, y_r).$$

*A **conjunctive normal form** (CNF) of ML constraints is a conjunction of clauses $C_1 \wedge \ldots \wedge C_t$.*

*(b) A **product term** P_i is a* conjunction *of one or more ML-constraints; that is,*

$$P_i = ML(x_1, y_1) \wedge ML(x_2, y_2) \wedge \ldots \wedge ML(x_r, y_r).$$

*A **disjunctive normal form** (DNF) of ML constraints is a disjunction of one or more product terms $P_1 \vee \ldots \vee P_t$.*

CNF and DNF versions of CL constraints are defined in a similar manner.

In this section, we consider the feasibility problems for CNF and DNF versions of ML and CL constraints. These problems will be referred to as CNF-ML, DNF-ML, CNF-CL and DNF-CL feasibility problems respectively.

The **NP**-completeness of the CL-feasibility problem (Section 3.3) showed that the problem is intractable even for a single conjunction of CL constraints. Since a single conjunction of CL constraints is in CNF as well as in DNF, it immediately follows that the CNF-CL and DNF-CL feasibility problems are both intractable.

In Section 3.2, it was shown that the feasibility problem for a single conjunction of ML constraints can be solved in polynomial time. This result can be used to derive an efficient algorithm for the DNF-ML feasibility problem as follows. Suppose the given DNF has p product terms. We execute the ML-feasibility algorithm in Figure 1 for each of the p product terms, treating each term as a collection of ML constraints. There is a solution to the DNF-ML feasibility problem if and only if at least one of these executions produces a feasible solution. Thus, the complexity of this algorithm is $O(p(m+n))$, where n is the number of points and m is the maximum number of ML constraints in a product term.

We will show in Section 7.2 that, in general, the feasibility problem for CNF-ML constraints is **NP**-complete. Moreover, this intractability holds even when each disjunction involves at most two ML-constraints. This should be contrasted with the result that the ML-feasibility problem (where each clause has only one ML-constraint) is efficiently solvable.

Special forms of CNF-ML and CNF-CL constraints arise when the constraints are specified through **choice-sets** for some of the points. In this scheme, for a point x, a choice-set $S_x = \{y_1, \ldots, y_r\}$ is given. In the context of ML-constraints, the choice-set S_x indicates that the cluster containing x must contain at least one of the points in S_x. In the context of CL-constraints, the choice-set S_x indicates that the cluster containing x must *not* contain at least one of the points in S_x. With this interpretation, the choice-set based ML constraint for the point x can be expressed as the following clause

$$ML(x, y_1) \ \vee \ ML(x, y_2) \ \vee \ \ldots \ \vee \ ML(x, y_r),$$

where each ML constraint includes the common point x. Similarly, the choice-set based CL constraint for the point x can be expressed as the following clause

$$CL(x, y_1) \ \vee \ CL(x, y_2) \ \vee \ \ldots \ \vee \ CL(x, y_r).$$

Since ML and CL constraints are inherently *symmetric* (i.e., $ML(x, y)$ is the same as $ML(y, x)$ and $CL(x, y)$ is the same as $CL(y, x)$), we will assume a corresponding symmetry condition for the choice-sets: if y appears in the choice-set for x, then x appears in the choice-set for y.

We note that the ϵ-constraint considered in Section 5.3 leads to choice-set based ML constraints, where the choice-set S_x for a point x is the set of all points within a distance of at most ϵ from x. However, to satisfy the ϵ-constraint, a point x may form a singleton cluster even when it has one or more ϵ-neighbors. To satisfy a choice-set based ML constraint, a point x with a nonempty choice-set cannot be in a singleton cluster.

We refer to the feasibility problems for choice-set based ML and CL constraints as **CSML-feasibility** and **CSCL-feasibility** respectively. We will show that both of these problems can be solved efficiently.

We believe that choice-set based ML and CL constraints will have much practical use. For example, in our empirical study section (Section 4) we saw that if we generate many ML-constraints from labeled data, then the number of connected components in the transitive closure always equaled the number of extrinsic labels (see Figure 4). Thus, we could not set k to be larger than the number of extrinsic labels. With the choice-set form of ML-constraints, one can do the following. Rather than overly constraining the problem by generating many ML constraints between the similarly labeled points, we can create a choice-set form of ML constraints among points with the same label. That is, each point with (say) label "+" must be linked with at least one other point with label "+", but *not necessarily with all* other points with label "+". A similar line of argument holds for using choice-set based CL-constraints. More importantly, this form allows us to utilize CL-constraints without making the feasibility problem difficult.

7.2 Complexity of Feasibility for CNF-ML Constraints

Here, we prove that the feasibility problem for CNF-ML constraints (CNF-ML-feasibility) is **NP**-complete, even when each clause has at most two ML constraints. Before presenting the proof, we review a graph theoretic definition. Given an undirected graph $G(V, E)$, a **vertex cover** for G is a subset V' of V such that for each edge $e = \{u, v\}$ in E, at least one of u and v is in V'. The definition allows us to state the following problem, which is known to be **NP**-complete [Garey and Johnson 1979].

Minimum Vertex Cover (Mvc)

Instance: An undirected graph $G(V, E)$ and an integer $\Gamma \leq |V|$.

Question: Is there a vertex cover V' for G such that $|V'| \leq \Gamma$?

Suppose $e = \{u, v\}$ is an edge in E and the chosen vertex cover V' includes u. We say that the edge e is **covered by** the node u. Note that e can be covered by choosing either u or v. In the **NP**-completeness proof below, this choice will be encoded as a disjunction of two ML-constraints.

Theorem 7.1 *The CNF-ML-feasibility problem is **NP**-complete.*

Proof: The membership in **NP** is obvious. To prove **NP**-hardness, we use a reduction from Mvc. Consider an instance I of the Mvc problem consisting of the graph $G(V, E)$, with $|V| = n$, $|E| = m$, and the integer $\Gamma \leq n$. We construct an instance I' of the CNF-ML-feasibility problem as follows. For each edge e_i in E, construct a point p_i, $1 \leq i \leq m$. For each node v_j in V, construct a point q_j, $1 \leq j \leq n$. The set S to be clustered has $m + n$ points and is given by $S = \{p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_n\}$.

The clauses in the constraint set C are chosen as follows. Consider any edge $e_i = \{v_x, v_y\}$. The clause A_i corresponding to e_i, $1 \leq i \leq m$, is given by

$$A_i = ML(p_i, q_x) \ \vee \ ML(p_i, q_y)$$

We also add $m - 1$ other clauses, denoted by $B_1, \ldots, B_{m-1}$, where each B_i has just one ML-constraint namely $ML(p_i, p_{i+1})$. Thus, each clause in A_i is the disjunction of two ML-constraints

while each clause in B_i has only one ML-constraint. In other words, each clause has at most two ML-constraints.

As mentioned earlier, the intuition behind the construction of clause A_i is that when a particular ML-constraint from A_i is satisfied, it corresponds to choosing a node to cover the edge e_i. For this reason, we will refer to the clauses A_1, ..., A_m as **covering constraints**. The clauses B_1, ..., B_{m-1} ensure that in any solution to the instance I' of the CNF-ML-feasibility problem, the set of points $\{p_1, \ldots, p_m\}$ is entirely in one cluster.

Thus, the constraint set $C = \{A_1, \ldots, A_m, B_1, \ldots, B_{m-1}\}$ has a total of $2m-1$ clauses. The lower and upper bounds on the number of clusters are set to $n - \Gamma + 1$ and n respectively. This completes the construction of the instance I'. It can be seen that the construction can be carried out in polynomial time. We now prove that the instance I' has a solution iff the instance I has a solution.

Part 1: Suppose the MVC instance I has a solution; that is, the graph G has a vertex cover of size $t \leq \Gamma$.

Let $V' = \{v_{j_1}, \ldots, v_{j_t}\}$ denote the given vertex cover. Construct a solution to I' consisting of $m - t + 1$ clusters as follows.

(a) Cluster C_1 consisting of $m + t$ points is given by $C_1 = \{p_1, \ldots, p_m, q_{j_1}, \ldots, q_{j_t}\}$.

(b) For each node v_x in $V - V'$, there is a singleton cluster consisting of the point q_x. There are $n - t$ such clusters, and they are denoted by C_2, C_3, ..., C_{n-t+1}.

The number of clusters is $n - t + 1 \geq n - \Gamma + 1$, since $t \leq \Gamma$. Also, the number of clusters is less than n. Thus, this solution to instance I' satisfies the bounds on the number of clusters. The solution satisfies all the clauses B_1, ..., B_{m-1}, since the set of points $\{p_1, \ldots, p_m\}$ is entirely in cluster C_1. We now show that the solution also satisfies the clauses A_1, ..., A_m.

Consider any clause A_i corresponding to edge $e_i = \{v_x, v_y\}$. Thus, clause A_i is given by $ML(p_i, q_x) \vee ML(p_i, q_y)$. Note that V' is a vertex cover for $G(V, E)$. Thus, for the edge $e_i = \{v_x, v_y\}$ in E, at least one of v_x and v_y is in V'. Without loss of generality, suppose v_x is in V'. The solution for I' constructed above includes the point q_x in C_1. Thus, the solution satisfies clause A_i. Therefore, we have a feasible solution to the instance I'.

Part 2: Suppose the CNF-ML-feasibility problem instance I' has a solution. Let $P = \{p_1, \ldots, p_m\}$ and $Q = \{q_1, \ldots, q_n\}$. Because of clauses B_1, ..., B_{m-1}, the set P must be entirely in some cluster, say C_1. Note that the solution to I' has at least $n - \Gamma + 1$ clusters. We have the following lemma.

Lemma 7.1 *The cluster C_1 contains at least one and at most Γ of the points q_1, ..., q_n.*

Proof of lemma: Note that the given solution also satisfies each of the clauses A_1, ..., A_m. Consider clause $A_i = ML(p_i, q_x) \vee ML(p_i, q_y)$ corresponding to point p_i. To satisfy this clause, at least one of q_x and q_y must be in C_1. Thus, C_1 has at least one point from Q.

Now, suppose C_1 has *more than* Γ points from the set Q. Thus, $|C_1| \geq m + \Gamma + 1$. Since $|S| = m + n$, at most $n - \Gamma - 1$ points of S are not in C_1. Thus, the maximum number of clusters, including C_1, that can be formed is at most $n - \Gamma$. This contradicts the assumption that the solution has at least $n - \Gamma + 1$ clusters. Therefore, C_1 has at most Γ points from the set Q. ∎

The above lemma enables us to construct a solution to the MVC instance I from the given solution to the CNF-ML-feasibility instance I'. In that solution, let C_1 be the cluster containing the set P. From the above lemma, C_1 has $t \leq \Gamma$ points from the set Q. Let $Q' = \{q_{j_1}, \ldots, q_{j_t}\}$ denote the subset of Q that is contained in C_1. We claim that the collection $V' = \{v_{j_1}, \ldots, v_{j_t}\}$ is a vertex cover for G. This can be seen from the fact that the covering constraint for each point p_i is satisfied by an appropriate point from Q'. Since $|V'| = t \leq \Gamma$, we have a solution to the MVC instance I, and this completes the proof of Theorem 7.1. ∎

7.3 Feasibility Problem for Choice-Set ML constraints

Here, we show that the feasibility problem for choice-set based ML constraints (the CSML-feasibility problem) is efficiently solvable. Our algorithm relies on some graph theoretic concepts and results which are reviewed below.

Let $G(V, E)$ be an undirected graph. Each node $v \in V$ whose degree is zero is called an **isolated** node. A subset $V' \subseteq V$ is an **independent set** if there is no edge between any pair of nodes in V'. A **matching** M in G is a subset of E such that no two edges in M share a node. Each node that appears in an edge in M is referred to as a **matched** node with respect to M. nodes of G that do not appear in M are referred to as **unmatched** nodes with respect to M. A **maximum matching** is a matching containing the largest number of edges. The following theorem lists some known results about maximum matchings in graphs [Cormen et' al 2001, West 2001].

Theorem 7.2 *Let $G(V, E)$ be an undirected graph.*

(a) A maximum matching of G can be found in $O(|E|\sqrt{|V|})$ time.

(b) Suppose M is maximum matching in G and U is the set of unmatched vertices with respect to M. Then, U is an independent set. ∎

The steps of our algorithm for the CSML-feasibility problem are shown in Figure 9. It can be seen that the running time of the algorithm is dominated by the maximum matching computation in Step 3. By Part (b) of Theorem 7.2, Step 3 can be implemented to run in $O(|E|\sqrt{|V|})$ time. Therefore, the algorithm runs in polynomial time. The correctness of the algorithm is established through the following lemmas.

Lemma 7.2 *Let G denote the graph constructed in Step 1 of the algorithm in Figure 9. Let r and t denote respectively the size of a maximum matching and the number of isolated nodes in G. If $K_\ell > r + t$, then there is no solution to the CSML-feasibility problem.*

Proof: The proof is by contradiction. So, assume that $K_\ell > r + t$ and that there is a feasible solution. Thus, the solution has at least $r + t + 1$ clusters. Because of the symmetric nature of the CSML-constraint and the construction of graph G in Step 1, the points for which the choice-set is empty are precisely those that correspond to the isolated nodes of G. In any feasible solution, every point x for which the choice-set S_x is nonempty must be in a cluster with at least one other point. Since the number of isolated nodes is t, at most t of the clusters in the assumed feasible solution may be singleton clusters. Thus, at least $r + 1$ clusters in the solution have points with nonempty choice-sets. Let C' denote this collection of nonsingleton clusters. Again by our construction, the subgraph of G corresponding to each of the clusters in C' contains at least one edge. Since these subgraphs are vertex disjoint, we can get a matching for G with $r + 1$ or more edges from these subgraphs. This contradicts the assumption that the size of the maximum matching in G is r, and the lemma follows. ∎

Lemma 7.3 *Let the parameters r and t be as defined in Lemma 7.2. If $K_\ell \leq r + t$, the algorithm in Figure 9 produces a feasible solution with K_ℓ clusters.*

Proof: We show that at the end of Step 6 of the algorithm, the number of clusters is $r + t$ and that this partition of the point set S satisfies the CSML-constraint. The lemma would follow from these results since Step 7 simply merges some clusters when $K_\ell < r + t$, and the CSML-constraint cannot be violated by merging clusters.

We first prove that the number of clusters at the end of Step 6 is equal to $r + t$. To see this, note that Step 2 creates t singleton clusters and that Step 5 produces one cluster for each of the r edges in the maximum matching M. Thus, there are $r + t$ clusters at the end of Step 5. As mentioned in the proof of Lemma 7.2, points which have nonempty choice-sets correspond to nodes with degree at least one in G. Each node with a degree of at least one is either a matched node or an unmatched node with respect to the maximum matching M. Step 6 adds each unmatched node to one of the clusters $C_1, \ldots, C_r$. By Part (b) of Theorem 7.2, the unmatched nodes with respect to M form an independent set. Therefore, the addition of unmatched nodes to clusters in Step 6 does not change the number of clusters. In other words, the number of clusters at the end of Step 6 is also $r + t$.

We now argue that at the end of Step 6, the CSML constraint is satisfied for each point in S. For each point corresponding to a matched node, the CSML-constraint is satisfied at the end of Step 5 itself because each edge in M gives rise to one of the clusters $C_1, \ldots, C_r$. So, we need to only show that the CSML-constraint is satisfied for unmatched nodes. Consider any such node v_x and let x denote the point corresponding to v_x. Again, by Part (b) of Theorem 7.2, the unmatched nodes with respect to M form an independent set. Therefore, all the nodes corresponding to the elements of the choice-set S_x of x are matched nodes. In other words, each node corresponding to a point in S_x appears in one of the clusters. Since Step 6 adds x to a cluster containing one of the points in S_x, the CSML-constraint is satisfied for the point x at the end of Step 6. This completes the proof of the lemma. ∎

The correctness of the algorithm is a direct consequence of the above two lemmas. We also observed that the algorithm runs in polynomial time. The following theorem summarizes the above results.

Theorem 7.3 *The CSML-feasibility problem can be solved in polynomial time.* ∎

7.4 Feasibility Problem for Choice-Set CL constraints

Here, we show that the feasibility problem for choice-set based CL constraints (CSCL-feasibility problem) can also be solved efficiently. Thus, unlike the CL-feasibility problem, the CSCL-feasibility problem is tractable. Assuming that the set S of points to be clustered has cardinality at least two and that at least one point[3] has a nonempty choice-set, it is obvious that there is no solution to CSCL-feasibility problem if $K_u = 1$. We will show that as long as $K_u \geq 2$, there is always a feasible solution with $\max\{2, K_\ell\}$ clusters.

Our algorithm for the CSCL-feasibility problem also relies on some simple graph theoretic concepts. Let $G(V, E)$ be an undirected graph. A **spanning forest** for G contains a spanning tree for each connected component of G. It is well known that any tree can be 2-colored in polynomial time [West 2001].

The steps of our algorithm for the CSCL-feasibility problem are shown in Figure 10. Since constructing a spanning forest and 2-coloring trees can all be done in polynomial time, the algorithm runs in polynomial time. The correctness of the algorithm is established through the following lemma.

Lemma 7.4 *The clusters C_1 and C_2 constructed at the end of Step 6 of the algorithm in Figure 10 satisfy all the CSCL constraints.*

Proof: The isolated nodes of G correspond to points with empty choice-sets. Therefore, they play no role in satisfying the CSCL constraints. Thus, we need to consider only points with nonempty choice-sets. Let x be any such point and let S_x denote its (nonempty) choice-set. By our construction of G, the node v_x corresponding to x has degree of at least 1. Therefore, in the spanning forest F, v_x is adjacent to at least one of the nodes corresponding to points in S_x. Let v_y be one such node and let y be the point corresponding to v_y. Because of the edge $\{v_x, v_y\}$ in F, the 2-coloring step assigns different colors to v_x and v_y. In other words, x and y appear in different clusters; that is, the CSCL constraint is satisfied for x. ∎

Step 7 of the algorithm in Figure 10 creates new singleton clusters from C_1 and C_2 (if necessary). Clearly, this step cannot violate any CSCL constraint. Thus, the correctness of our algorithm for the CSCL-feasibility problem is a consequence of Lemma 7.4. The following theorem summarizes the above results.

Theorem 7.4 *The CSCL-feasibility problem can be solved in polynomial time.* ∎

[3]Because of the assumed symmetry condition, this condition implies that at least two points have nonempty choice-sets.

8 Summary and Conclusions

A major limitation of clustering is that clustering algorithms that minimize an objective function such as vector quantization error do not always produce meaningful solutions to the practitioner. The area of clustering under constraints allows the practitioner to specify prior background information that can influence a clustering algorithm to find clusterings with specific properties. In effect, these constraints partition the space of all possible clusterings into feasible and infeasible clusterings. However, overly constraining or poorly specified constraints can make the set of feasible clusterings empty.

In this paper we formally studied the feasibility problem for clustering under constraints. We first developed (worst-case) complexity results for clustering under conjunctions of CL and ML constraints and our newly introduced ϵ and δ constraints individually. We also studied the problem for all combinations of these constraints. Our results indicate that for CL constraints and for several constraint combinations, the feasibility problem is **NP**-complete. We then studied CNF and DNF versions of ML and CL constraints. There also we found that the feasibility problem for CL constraints is **NP**-complete. Thus, one cannot hope to efficiently incorporate CL constraint checking into a clustering algorithm.

To allow the use of some general versions of CL and ML constraints, we introduced the notion of choice-sets, which lead to CNF versions of these constraints, with the restriction that each clause has a common point. We then showed that the feasibility problem for choice-set forms of ML and CL constraints is efficiently solvable, meaning that we can incorporate such forms of constraints efficiently into existing clustering algorithms. Our worst-case feasibility results are summarized in Tables 1 and 2.

Finally, we carried out an empirical study to determine whether the feasibility problem arises in practice. We found that for the same data set and the same constraint source but with varying number of constraints, the feasibility problem was sometimes easy and other times difficult. We then presented an approach that considers the maximum node degree of the graph constructed from the constraints to provide an explanation of which **instances** of the feasibility problem are easy and which are difficult.

Acknowledgments: We thank Dr. Kiri Wagstaff (JPL) for her detailed comments on the conference version of this paper.

References

[Bottou and Bengio 1995] Bottou, L. and Bengio, Y., "Convergence Properties of the K-Means Algorithms", *Advances in Neural Information Processing Systems*, Vol. 7, Edited by G. Tesauro and D. Touretzky and T. Leen, MIT Press, Cambridge, MA, 1995, pp. 585–592.

[Basu et' al 2002] Basu, S., Banerjee, A., and Mooney, R., "Semi-supervised Learning by Seeding", *Proc. 19th Intl. Conf. on Machine Learning* (ICML-2002), Sydney, Australia, July 2002, pp. 19–26.

[Basu et' al 2004a] Basu, S., Bilenko, M., and Mooney, R., "A Probabilistic Framework for Semi-Supervised Clustering", *Proc. 10th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining* (KDD-2004), Seattle, WA, August 2004, pp. 59–68.

[Basu et' al 2004b] Basu, S., Bilenko, M., and Mooney, R., "Active Semi-Supervision for Pairwise Constrained Clustering", *Proc. 4th SIAM Intl. Conf. on Data Mining* (SDM-2004), pp. 333–344.

[Bradley and Fayyad 1998] Bradley, P. and Fayyad, U., "Refining initial points for K-Means clustering", *Proc. 15th Intl. Conf. on Machine Learning* (ICML-1998), 1998, pp. 91–99.

[Campers' et al 1987] Campers, G., Henkes, O., and Leclerq, P., "Graph Coloring Heuristics: A Survey, Some New Propositions and Computational Experiences on Random and Leighton's Graphs", in *Proc. Operational Research '87*, Buenos Aires, 1987, pp. 917–932.

[Cooper 1990] Cooper, G. F., "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks", *Artificial Intelligence*, Vol. 42, No. 2-3, 1990, pp. 393–405.

[Cormen et' al 2001] Cormen, T., Leiserson, C., Rivest, R. and Stein, C., *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, Cambridge, MA, 2001.

[Dyer and Frieze 1986] Dyer, M. and Frieze, A., "Planar 3DM is NP-Complete", *J. Algorithms*, Vol. 7, 1986, pp. 174–184.

[Ding 2005] Ding, C., Personal communication, Apr. 2005.

[Davidson and Ravi 2005a] Davidson, I. and Ravi, S. S., "Clustering with Constraints: Feasibility Issues and the k-Means Algorithm", *Proc. 2005 SIAM International Conference on Data Mining* (SDM'05), Newport Beach, CA, Apr. 2005, pp. 138–149.

[Davidson and Ravi 2005b] Davidson, I. and Ravi, S. S., "Hierarchical Clustering with Constraints: Theory and Practice", *Proc. 9th European Principles and Practice of KDD* (PKDD'05), Porto, Portugal, Oct. 2005.

[Davidson and Satyanarayana 2003] Davidson, I. and Satyanarayana, A., "Speeding up K-Means Clustering Using Bootstrap Averaging", *Proc. IEEE ICDM 2003 Workshop on Clustering Large Data Sets*, Melbourne, FL, Nov. 2003, pp. 16–25.

[Ester et' al 1996] Ester, M., Kriegel, H., Sander, J., and Xu, X., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", *Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining* (KDD-96), Portland, OR, 1996, pp. 226–231.

[Feige and Kilian 1998] Feige, U. and Kilian, J., "Zero Knowledge and the Chromatic Number", *J. Computer and System Sciences*, Vol. 57, 1998, pp. 187–199.

[Garey and Johnson 1979] Garey, M. R. and Johnson, D. J., *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.

[Gonzalez 1985] Gonzalez, T., "Clustering to Minimize the Maximum Intercluster Distance", *Theoretical Computer Science*, Vol. 38, No. 2-3, June 1985, pp. 293–306.

[Hansen and Jaumard 1997] Hansen, P. and Jaumard, B., "Cluster Analysis and Mathematical Programming", *Mathematical Programming*, Vol. 79, Aug. 1997, pp. 191–215.

[Hertz and de Werra 1987] Hertz, A. and de Werra D., "Using Tabu Search Techniques for Graph Coloring", 1987 *Computing*, Vol. 39, pp. 345–351.

[Klein et' al 2002] Klein, D., Kamvar, S., and Manning, C., "From Instance-Level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering", *Proc. 19th Intl. Conf. on Machine Learning* (ICML 2002), Sydney, Australia, July 2002, pp. 307–314.

[Pelleg and Moore 1999] Pelleg, D. and Moore, A., "Accelerating Exact k-means Algorithms with Geometric Reasoning", *Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, Aug. 1999, pp. 277–281.

[Tamassia and Tollis 1989] Tamassia, R. and Tollis, I., "Planar Grid Embedding in Linear Time", *IEEE Trans. Circuits and Systems*, Vol. CAS-36, No. 9, Sept. 1989, pp. 1230–1234.

[Wagstaff and Cardie 2000] Wagstaff, K. and Cardie, C., "Clustering with Instance-Level Constraints", *Proc. 17th Intl. Conf. on Machine Learning* (ICML 2000), Stanford, CA, June–July 2000, pp. 1103–1110.

[Wagstaff et' al 2001] Wagstaff, K. Cardie, C., Rogers, S., and Schroedl, S., "Constrained K-means Clustering with Background Knowledge", *Proc. 18th Intl. Conf. on Machine Learning* (ICML 2001), Williamstown, MA, June–July 2001, pp. 577–584.

[Wagstaff 2002] Wagstaff, K., Ph.D. Thesis, *Personal Communication*, 2005.

[Wagstaff 2005] Wagstaff, K. (an Author of COP-K-Means), *Personal Communication*, 2005.

[West 2001] West, D. B., *Introduction to Graph Theory*, Second Edition, Prentice Hall, Inc., Englewood Cliffs, NJ, 2001.

[Wijsen and Meersman 1998] Wijsen, J. and Meersman, R., "On the Complexity of Mining Quantitative Association Rules", *Journal of Data Mining and Knowledge Discovery*, Vol. 2, No. 3, 1998, pp 263–281.

[Xing et' al] Xing, E., Ng, A., Jordan, M., and Russell, S., "Distance metric Learning, with Application to Clustering with Side-Information", *Advances in Neural Information Processing Systems* (NIPS) 15, 2003, pp. 505–512.

Note: Whenever a feasible solution exists, the following algorithm outputs a clustering with K_ℓ clusters.

1. Construct the following graph $G(V, E)$: The node set V is in one-to-one correspondence with the set of points S. For each point x with choice-set S_x, the set E contains edges that join the node corresponding to x to the nodes corresponding to the points in S_x.

2. Let $V_1 = \{v_{i_1}, \ldots, v_{i_t}\}$ denote the set of isolated nodes of G. (Thus, $t = |V_1|$.) Create t singleton clusters Y_1, ..., Y_t, where Y_j contains the point corresponding to node v_{i_j}, $1 \leq j \leq t$.

3. Find a maximum matching M in G. Let $M = \{e_1, \ldots, e_r\}$. (Thus, $r = |M|$.)

4. **if** $K_\ell > t + r$ **then** Output "No solution" and **stop**.

   Note: Here, $K_\ell \leq r + t$. The following steps construct a solution with K_ℓ clusters.

5. **for** each edge e_i in M **do**

   (a) Let $e_i = \{u, v\}$.

   (b) Create cluster C_i containing the two points corresponding to nodes u and v.

   Note: Step 5 created clusters C_1, C_2, ..., C_r.

6. **for** each point x whose choice-set S_x is not empty **do**

   if x is not in $C_1 \cup \ldots \cup C_r$ **then**

   (a) Choose an arbitrary point $y \in S_x$. (Note: The point y must be in one of the clusters C_1, ..., C_r.)

   (b) Add x to the cluster C_j which contains y.

   Note: At this point, there are $r + t$ clusters namely, C_1, ..., C_r, Y_1, ..., Y_t.

7. **if** $K_\ell = r + t$ **then** Output C_1, ..., C_r, Y_1, ..., Y_t.
   else

   (a) Combine the last $r + t - K_\ell + 1$ clusters in the list C_1, ..., C_r, Y_1, ..., Y_t into a single cluster X.

   (b) Output the first $K_\ell - 1$ clusters in the list C_1, ..., C_r, Y_1, ..., Y_t and the cluster X (for a total of K_ℓ clusters).

Figure 9: Algorithm for the CSML-Feasibility Problem

Note: It is assumed that $|S| \geq 2$ and that at least one point has a nonempty choice-set. Whenever a feasible solution exists, the following algorithm outputs a clustering with $\max\{2, K_\ell\}$ clusters.

1. **if** $K_u < 2$ **then** Output "No solution" and **stop**.
2. Construct the following graph $G(V, E)$: The node set V is in one-to-one correspondence with the set of points S. For each point x with choice-set S_x, the set E contains edges that join the node corresponding to x to the nodes corresponding to the points in S_x.
3. Let $V_1 = \{v_{i_1}, \ldots, v_{i_t}\}$ denote the set of isolated nodes of G.
4. Let r denote the number of connected components of G. Construct a spanning forest F of G consisting of trees T_1, ..., T_r.
5. Construct a 2-coloring of each tree in F using colors 1 and 2. Let n_1 and n_2 denote the number of nodes colored 1 and 2 respectively.
6. Construct cluster C_1 consisting of points corresponding to all the nodes in V_1 and those corresponding to nodes assigned color 1 in Step 5. Construct cluster C_2 containing points corresponding to nodes assigned color 2 in Step 5.
7. **if** $K_\ell \leq 2$ **then** output the clusters C_1 and C_2.
   else

   if $K_\ell \leq |C_1|$ **then**

   (a) Remove $K_\ell - 2$ points (arbitrarily) from C_1 and make each of them into a singleton cluster.

   (b) Output the singleton clusters constructed in (a) above, the remaining points in C_1 as a single cluster and C_2.

   else

   (a) Make each point in C_1 into a singleton cluster.

   (b) Remove $K_\ell - |C_1| - 1$ points (arbitrarily) from C_2 and make each of them into a singleton cluster.

   (c) Output the singleton clusters constructed in (a) and (b) above and the remaining points in C_2 as a single cluster.

Figure 10: Algorithm for the CSCL-Feasibility Problem

Appendix A: A Coloring Algorithm Based on Brooks's Theorem

Theorem 8.1 ***(Brooks's Theorem)*** *Let $G(V, E)$ be a graph in which the maximum node degree is Δ. Then G is $(\Delta + 1)$-colorable.*

Figure 11 gives an algorithm for coloring a graph using k colors, when $k \geq \Delta + 1$. The following lemma establishes the correctness of the algorithm. (The is the proof of Brooks's theorem given in [West 2001].)

Lemma 8.1 *Algorithm Brooks-Coloring outputs a valid coloring of G with each node receiving a color in the range $[1 \,..\, k]$.*

Proof: Consider the outline of Algorithm Brooks-Coloring shown in Figure 11. We need only show that for each node v, Step 2(b) of the algorithm always succeeds in finding a value for the variable α. To see this, note that the degree of v is at most Δ. Thus, $|Q_v| \leq \Delta$. Since $k \geq \Delta + 1$, it follows that there is at least one color that has not been used for any neighbor of v. In other words, there is at least one available value for α in each iteration of Step 2(b). ∎

Algorithm Brooks-Coloring (G, k)

Input: Undirected graph $G(V, E)$ with maximum node degree Δ, an integer $k \geq \Delta + 1$.

Output: A valid node coloring of G such that each node is given a color in $[1 \,..\, k]$.

1. Initially, no node has been assigned a color.
2. **for** each node $v \in V$ **do**
   (a) Let Q_v denote the set of colors used for the neighbors of v in G. (Q_v may be empty.)
   (b) Find an integer α such that $1 \leq \alpha \leq k$ and $\alpha \notin Q_v$.
   (c) Set Color(v) $=$ α.

Figure 11: Brooks-Coloring Algorithm
