Clustering With Constraints: Feasibility Issues and the k-Means Algorithm

Ian Davidson* S. S. Ravi†

Abstract

Recent work has looked at extending the k-Means algorithm to incorporate background information in the form of instance level must-link and cannot-link constraints. We introduce two ways of specifying additional background information in the form of δ and ϵ constraints that operate on all instances but which can be interpreted as conjunctions or disjunctions of instance level constraints and hence are easy to implement. We present complexity results for the *feasibility* of clustering under each type of constraint individually and several types together. A key finding is that determining whether there is a feasible solution satisfying *all* constraints is, in general, **NP**-complete. Thus, an iterative algorithm such as k-Means should not try to find a feasible partitioning at each iteration. This motivates our derivation of a new version of the k-Means algorithm that minimizes the *constrained vector quantization error* but at each iteration does not attempt to satisfy all constraints. Using standard UCI datasets, we find that using constraints improves accuracy as others have reported, but we also show that our algorithm reduces the number of iterations until convergence. Finally, we illustrate these benefits and our new constraint types on a complex real world object identification problem using the infra-red detector on an Aibo robot.

Keywords: k-Means clustering, constraints.

1 Introduction and Motivation

The k-Means clustering algorithm is a ubiquitous technique in data mining due to its simplicity and ease of use (see for example, [6, 10, 16]). It is well known that k-Means converges to a local minimum of the vector quantization error and hence must be restarted many times, a computationally very expensive task when dealing with the large data sets typically found in data mining problems.

Recent work has focused on the use of background information in the form of instance level must-link and cannot-link constraints. A must-link constraint enforces that two instances must be placed in the same cluster while a cannot-link constraint enforces that two instances must not be placed in the same cluster. We can divide previous work on clustering under constraints into two types: 1) Where the constraints help the algorithm learn a distortion/distance/objective function [4, 15] and 2) Where the constraints are used as "hints" to guide the algorithm to a useful solution [18, 19]. Philosophically, the first type of work makes the assumption that points surrounding a pair of must-link/cannot-link points should be close to/far from each other [15], while the second type just requires that the two points be in the same/different clusters. Our work falls into the second category. Recent examples of the second type of work include ensuring that constraints are satisfied at each iteration [19] and initializing algorithms so that constraints are satisfied [3]. The results of this type of work are quite encouraging; in particular, Wagstaff et al. [18, 19] illustrate that for simple classification tasks, k-Means with constraints obtains clusters with a significantly better purity (when measured on an extrinsic class label) than when not using constraints. Furthermore, Basu et al. [5] investigate determining the most informative set of constraints when the algorithm has access to an Oracle.

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

- We introduce two new constraint types which act upon *groups* of instances. Roughly speaking, the ϵ-constraint enforces that each instance x in a cluster must have another instance y in the same cluster such that the distance between x and y is at most ϵ. We shall see that this can be used to enforce prior information with respect to how the data was collected. The δ-constraint enforces that every instance in a cluster must be at a distance of at least δ from every instance in every other cluster. We can use this type of constraint to specify background information on the minimum distance between the clusters/objects we hope to discover.

- We show that these two new constraints can be eas-

*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.

ily represented as a disjunction and conjunction of must-link constraints, thus making their implementation easy.

- We present complexity results for the feasibility problem under each of the constraints individually and in combination. The polynomial algorithms developed in this context can be used for initializing the k-Means algorithm. We show that in many situations, the feasibility problem (i.e., determining whether there there is a solution that satisfies all the constraints) is **NP**-complete. Therefore, it is not advisable to attempt to find a solution that satisfies all the constraints at each iteration of a clustering algorithm.

- To overcome this difficulty, we illustrate that the k-Means clustering algorithm can be viewed as a two step algorithm with one step being to take the derivative of its error function and solving for the new centroid positions. With that in mind, we propose a new differentiable objective function that incorporates constraint violations and rederive a new constrained k-Means algorithm. This algorithm, like the original k-Means algorithm, is designed to monotonically decrease its error function.

We empirically illustrate our algorithm's performance on several standard UCI data sets and show that adding must-link and cannot-link constraints not only helps accuracy but can help improve convergence. Finally, we illustrate the use of δ and ϵ constraints in clustering distances obtained by an Aibo robot's infra-red detector for the purpose of object detection for path planning. For example, we can specify the width of the Aibo robot as δ; that is, we are only interested in clusters/objects that are more than δ apart, since the robot can only physically move between such objects.

The outline of the paper is as follows. We begin by considering the feasibility of clustering under all four types constraints individually. The next section builds upon the feasibility analysis for combinations of constraints. A summary of the results for feasibility can be found in Table 1. We then discuss the derivation of our constrained k-Means algorithm. Finally, we present our empirical results.

2 Definitions of Constraints and the Feasibility Problem

Throughout this paper, we use the terms "instances" and "points" interchangeably. Let $S = \{s_1, s_2, \ldots, s_n\}$ denote the given set of points which must be partitioned into K clusters, denoted by S_1, ..., S_K. 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 $d(s_i, s_j) = d(s_j, s_i)$. We consider the problem of clustering the set S under the following types of constraints.

(a) Must-Link Constraints: Each must-link constraint involves a pair of points s_i and s_j $(i \neq j)$. In any feasible clustering, points s_i and s_j must be in the same cluster.

(b) Cannot-Link Constraints: Each cannot-link constraint also involves a pair of distinct points s_i and s_j. In any feasible clustering, points s_i and s_j *must not* be in the same cluster.

(c) δ-Constraint (or Minimum Separation Constraint): This 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, for any pair of clusters S_i and S_j $(i \neq j)$, and any pair of points s_p and s_q such that $s_p \in S_i$ and $s_q \in S_j$, $d(s_p, s_q) \geq \delta$. Informally, this constraint requires that each pair of clusters must be well separated. As will be seen in Section 3.4, a δ-constraint can be represented by a conjunction of instance level must-link constraints.

(d) ϵ-Constraint: This 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_j containing two more more points, each point in S_j must have another point within a distance of at most ϵ. As will be seen in Section 3.5, an ϵ-constraint can be represented by a disjunction of instance level must-link constraints, with the proviso that when none of the must-link constraints is satisfied, the point is in a singleton cluster by itself. The ϵ-constraint is similar in principle to the ϵ+minpts criterion used in the DBSCAN algorithm [11]. However, in our work, the aim is to minimize the constrained vector quantization error subject to the ϵ-constraint, while in DB-SCAN, their criterion is central to defining a cluster and an outlier.

Although the constraints discussed above 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 [14]. For example, the clustering problem where the quality is measured by the maximum cluster diameter can be trans-

formed 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 [12, 13, 14]).

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 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 may admit trivial solutions. For instance, if we consider the feasibility problem for a collection of must-link constraints (or for a δ-constraint) and there is no lower bound on the number of clusters, a trivial feasible solution is obtained by having all the points in a single cluster. Likewise, when there is no upper bound on the number of clusters for the feasibility problem under cannot-link constraints (or an ϵ-constraint), a trivial feasible solution is to make each point into a separate cluster. Obviously, K_ℓ and K_u must satisfy the condition $1 \leq K_\ell \leq K_u \leq n$, where n denotes the total number of points.

For the remainder of this paper, 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 must-link or cannot-link 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 problems involving ϵ and δ constraints, it is assumed that the values of ϵ and δ are given. For convenience, the feasibility problems under constraints (a) through (d) defined above will be referred to as **ML-feasibility**, **CL-feasibility**, **δ-feasibility** and **ϵ-feasibility** respectively.

3 Complexity of Feasibility Problems

3.1 Overview In this section, we investigate the complexity of the feasibility problems by considering each type of constraint separately. In Section 4, we examine the complexity of feasibility problems for combinations of constraints. Our polynomial algorithms for feasibility problems do not require distances to satisfy the triangle inequality. Thus, they can also be used with nonmetric distances. On the other hand, the **NP**-completeness results show that the corresponding feasibility problems remain computationally intractable even when the set to be clustered consists of points in $\Re^2$.

The feasibility algorithms presented in Sections 3 and 4 focus on determining 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.

3.2 Feasibility Under Must-Link Constraints Klein et al. [15] showed that the ML-feasibility problem can be solved in polynomial time. 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.

As is well known, must-link constraints are transitive; that is, must-link constraints $\{s_i, s_j\}$ and $\{s_j, s_k\}$ imply the must-link constraint $\{s_i, s_k\}$. Thus, the two constraints can be combined into a single must-link constraint, namely $\{s_i, s_j, s_k\}$. Thus, a given collection C of must-link 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 must-link 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 sets C, and an edge between two nodes if the corresponding points appear together in a must-link 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 $O(m)$ edges. Therefore, its connected components can be found in $O(n+m)$ time [8]. 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 must-link constraints, the ML-feasibility problem can be solved in $O(n+m)$ time.* ∎

3.3 Feasibility Under Cannot-Link Constraints Klein et al. [15] mention that the CL-feasibility problem is **NP**-complete but omit the proof. Since we

Note: Whenever a feasible solution exists, the following algorithm outputs a collection of K_ℓ clusters satisfying all the must-link 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 must-link constraint.)
3. **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

   if $|S'| < K_\ell - r$ **then**
   Output "There is no solution."

   else
   (a) Let $t = K_\ell - r$. 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

wish to draw some additional conclusions from the proof, we have included it in the appendix. The proof uses a straightforward reduction from the GRAPH K-COLORABILITY problem (K-COLOR).

It is known that the K-COLOR problem is **NP**-complete even for graphs in which the number of edges is linear in the number of nodes [12]. This fact in conjunction with the proof in the appendix implies that the CL-feasibility problem is computationally intractable even when the number of constraints is linear in the number of points. Further, 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.

While **NP**-completeness result indicates that the CL-feasibility problem is at least as hard as the K-COLOR problem, it can also be shown that the feasibility problem is no harder than the K-COLOR problem as follows. For each point s_i, create a graph node v_i, and for each cannot-link constraint $\{s_i, s_j\}$, create the undirected edge $\{v_i, v_j\}$. It is easy to verify that the resulting graph is K-colorable iff there is a solution to the feasibility problem with at least 1 and at most K clusters. This reduction to the coloring problem 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, heuristics that work well in practice are known (see for example [1, 7]). The reduction also shows that when the upper bound on the number of clusters is two, the CL-feasibility problem can be solved in polynomial time. This is because the problem of determining whether an undirected graph can be colored using at most two colors can be solved efficiently [8].

1. **for** each point s_i **do**
   (a) Determine 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 must-link constraint $\{s_i, x_j\}$.
2. Let C denote the set of all the must-link 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 2: Algorithm for the δ-Feasibility Problem

3.4 Feasibility Under δ-Constraint In this section, we show that the δ-feasibility problem can be solved in polynomial time. The basic idea is simple: 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 must-link constraints and use the algorithm for the ML-feasibility problem. This shows that a δ-constraint can be replaced by a conjunction of must-link constraints. The steps of the resulting algorithm are shown in Figure 2.

The running time of the algorithm for δ-feasibility is dominated by the time needed to complete Step 1, that is, the time to compute the set of must-link constraints. Clearly, this step can be carried out in $O(n^2)$ time, and the number of must-link 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 3.2. *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. ∎

3.5 Feasibility Under 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

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

Note that a point is *not* an ϵ-neighbor of itself. Two distinct points s_p and s_q are 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 must-link 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 must-link constraints $\{s_i, s_{i_1}\}, \ldots, \{s_i, s_{i_r}\}$ are satisfied or the point s_i is in a singleton cluster by itself. In particular, any point in S which does not have an ϵ-neighbor must form a singleton cluster.

So, to determine feasibility under an ϵ-constraint for the set of points S, we first find the subset S_1 containing each point which does not have an ϵ-neighbor. Let $|S_1| = t$, and let C_1, C_2, …, C_t denote the singleton clusters formed from S_1. To cluster the points in $S_2 = S - S_1$ (i.e., the set of points each of which has an ϵ-neighbor), it is convenient to use an auxiliary undirected graph defined below.

DEFINITION 3.1. *Let a set of points S and a value $\epsilon > 0$ be given. Let $Q \subseteq S$ be a set of points such that for each point in Q, there is an ϵ-neighbor in Q. The **auxiliary graph** $G(V, E)$ **corresponding to** Q is constructed as follows.*

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

(b) For any two nodes v_p and v_q in V, the edge $\{v_p, v_q\}$ is in E if the points in Q corresponding to v_p and v_q are ϵ-neighbors.

Let $G(V, E)$ denote the auxiliary graph corresponding to S_2. Note that each connected component (CC) of G has at least two nodes. Thus, the CCs of G provide a way of clustering the points in S_2. Let X_i, $1 \leq i \leq r$, denote the cluster formed from the i^{th} CC of G. The t singleton clusters together with these r clusters would form a feasible solution, provided the upper and lower bounds on the number of clusters are satisfied. So, we focus on satisfying these bounds.

If $S_2 = \emptyset$, then the minimum number of clusters is $t = |S_1|$, since each point in S_1 must be in a separate singleton cluster. Otherwise, we need at least one additional cluster for the points in S_2; that is, the minimum number of clusters in that case is $t+1$. Thus, the minimum number of clusters, denoted by N^*, is given by $N^* = t + \min\{1, r\}$. If $K_u < N^*$, there is no feasible solution. We may therefore assume that $K_u \geq N^*$.

As mentioned above, there is a solution with $t + r$ clusters. If $t + r \geq K_u$, then we can get K_u clusters by simply merging (if necessary) an appropriate number of clusters from the collection X_1, X_2, …, X_r into a single cluster. Since each CC of G has at least two points, this merging step will not violate the ϵ-constraint.

The only remaining possibility is that the value $t+r$ is smaller than the lower bound K_ℓ. In this case, we can increase the number of clusters to K_ℓ by splitting some of the clusters X_1, X_2, …, X_r to form more clusters. One simple way to increase the number of clusters by one is to create a new singleton cluster by taking one point away from some cluster X_i with two or more points. To facilitate this, we construct a spanning tree for each CC of G. The advantage of having trees is that we can remove a leaf node from a tree and make the point corresponding to that node into a new singleton cluster. 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_ℓ.

The above discussion leads to the feasibility algorithm shown in Figure 3. It can be seen that the running time of the algorithm 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 3.3. *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.* ∎

4 Feasibility Under Combinations of Constraints

4.1 Overview In this section, we consider the feasibility problem under combinations of constraints. Since the CL-feasibility problem is **NP**-hard, the feasibility problem for any combination of constraints involving cannot-link constraints is, in general, computationally

1. Find the set $S_1 \subseteq S$ such that no point in S_1 has an ϵ-neighbor. Let $t = |S_1|$ and $S_2 = S - S_1$.
2. Construct the auxiliary graph $G(V, E)$ for S_2 (see Definition 3.1). Let G have r connected components (CCs) denoted by G_1, G_2, ..., G_r.
3. Let $N^* = t + \min\{1, r\}$. (**Note:** To satisfy the ϵ-constraint, at least N^* clusters must be used.)
4. **if** $N^* > K_u$ **then** Output "No feasible solution" and **stop**.
5. 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.
6. **if** $t + r \geq K_u$

   then /* We may have too many clusters. */
   (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, ..., $X_{K_u - t}$.

   else /* We have too few clusters. */
   (a) Let $N = t + r$. Construct spanning trees T_1, T_2, ..., T_r corresponding to the CCs of G.
   (b) **while** $(N < K_\ell)$ **do**
      (i) Find a tree T_i with at least two nodes. If no such tree exists, output "No feasible solution" and **stop**.
      (ii) Let v be a leaf in tree T_i. Delete v from T_i.
      (iii) Delete the point corresponding to v from cluster X_i and form a new singleton cluster X_{N+1} containing that point.
      (iv) $N = N + 1$.
   (c) Output the K_ℓ clusters C_1, C_2, ..., C_t, X_1, X_2, ..., $X_{K_\ell - t}$.

Figure 3: Algorithm for the ϵ-Feasibility Problem

intractable. So, we need to consider only the combinations of must-link, δ and ϵ constraints. We show that the feasibility problem remains efficiently solvable when both a must-link constraint and a δ constraint are considered together as well as when δ and ϵ constraints are considered together. When must-link constraints are considered together with an ϵ constraint, we show that the feasibility problem is **NP**-complete. This result points out that when must-link, δ and ϵ constraints are all considered together, the resulting feasibility problem is also **NP**-complete in general.

4.2 Combination of Must-Link and δ Constraints We begin by considering the combination of must-link constraints and a δ constraint. As mentioned in Section 3.4, the effect of the δ-constraint is to contribute a collection of must-link constraints. Thus, we can merge these must-link constraints with the given must-link constraints, and then ignore the δ-constraint. The resulting feasibility problem involves only must-link 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 $O(n^2)$ must-link constraints. Further, since each given must-link constraint involves two points, we may assume that the number of given must-link constraints is also $O(n^2)$. Thus, the total number of must-link constraints due to the combination is also $O(n^2)$. Thus, the following result is a direct consequence of Theorem 3.1.

THEOREM 4.1. *Given a set of n points, a value $\delta > 0$ and a collection C of must-link constraints, the feasibility problem for the combination of must-link and δ constraints can be solved in $O(n^2)$ time.* ∎

4.3 Combination of Must-Link and ϵ Constraints Here, we show that the feasibility problem for the combination of must-link and ϵ constraints is **NP**-complete. To prove this result, we use a reduction from the following problem which is known to be **NP**-complete [9].

PLANAR EXACT COVER BY 3-SETS (PX3C)

Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$, where $n = 3q$ for some positive integer q 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 $\{u, v\} \in E$ iff the element corresponding to u appears in the set corresponding to v, is planar.

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

For reasons of space, we have included only a sketch of this **NP**-completeness proof.

THEOREM 4.2. *The feasibility problem under the combination of must-link and ϵ constraints is **NP**-complete.*

Proof sketch: It is easy to verify the membership of the problem in **NP**. 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 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 [17]. The points and constraints for the feasibility problem are created from this embedding.

Using suitable scaling, assume that the each grid edge is of length 1. 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. Thus, for each edge e of G, this provides 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 a must-link constraint involving all the points in S'_e. We also introduce a must-link constraint involving all the points corresponding to the elements nodes of G. We choose the value of ϵ to be $1/2$. The lower bound on the number of clusters is set to $m - n/3 + 1$ and the upper bound is set to m. It can be shown that there is a solution to the PX3C problem instance if and only if there is a solution to the feasibility problem. ∎

4.4 Combination of δ and ϵ Constraints In this section, we show that the feasibility problem for the combination of δ and ϵ constraints can be solved in polynomial time. It is convenient to consider this problem under two cases, namely $\delta \leq \epsilon$ and $\delta > \epsilon$. For reasons of space, we will discuss the algorithm for the first case and mention the main idea for the second case.

For the case 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. This is done as follows. Construct the auxiliary undirected graph $G(V, E)$, where V is in one-to-one correspondence with the set of points and an edge $\{x, y\} \in E$ iff the distance between the points corresponding to nodes x and y is less than δ. Suppose C_1, C_2, ..., C_r denote the connected components (CCs) of G. Consider any CC, say C_i, with two or more nodes. By the above observation, the ϵ-constraint is satisfied for all the points corresponding to the nodes in C_i. Thus, we can "collapse" each such set of points into a single "super point". Let $S' = \{P_1, \ldots, P_r\}$ denote the new set of points, where P_i is a super point if the corresponding CC C_i has two or more nodes, and P_i is a single point otherwise. Given the distance function d for the original set of points S, we define a new distance function d' for S' as follows:

$$d'(P_i, P_j) = \min\{d(s_p, s_q) : s_p \in P_i, s_q \in P_j\}.$$

With this new distance function, we can ignore the δ constraint. We need only check whether there is a feasible clustering of the set S' under the ϵ-constraint using the new distance function d'. Thus, we obtain a polynomial time algorithm for the combination of ϵ and δ constraints when $\delta \leq \epsilon$. It is not hard to see that the resulting algorithm runs in $O(n^2)$ time.

For the case when $\delta > \epsilon$, any pair of ϵ-neighbors must be in the same cluster. Using this idea, it is possible to construct a collection of must-link constraints corresponding to the given δ and ϵ constraints, and solve the feasibility problem in $O(n^2)$ time.

The following theorem summarizes the above discussion.

THEOREM 4.3. *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.* ∎

The complexity results for various feasibility problems are summarized in Table 1. For each type of constraint, the table indicates whether the feasibility problem is in **P** (i.e., efficiently solvable) or **NP**-complete. Results for which no references are cited are from this paper.

5 A Derivation of a Constrained k-Means Algorithm

In this section we derive the k-Means algorithm and then derive a new constrained version of the algorithm

Constraint	Complexity
Must-Link	**P** [15]
Cannot-Link	**NP**-Complete [15]
δ-constraint	**P**
ϵ-constraint	**P**
Must-Link and δ	**P**
Must-Link and ϵ	**NP**-complete
δ and ϵ	**P**

Table 1: Results for Feasibility Problems

from first principles. Let C_j be the centroid of the j^{th} cluster and Q_j be the set of instances that are closest to the j^{th} cluster.

It is well known that the error function of the k-Means *problem* is the vector quantization error (also referred to as the distortion) given by the following equations.

$$VQE = \sum_{j=1}^{k} VQE_j \tag{5.1}$$

$$VQE_j = \frac{1}{2} \sum_{s_i \in Q_j} (C_j - s_i)^2 \tag{5.2}$$

The k-Means algorithm is an iterative algorithm which in every step attempts to further minimize the distortion. Given a set of cluster centroids, the algorithm assigns instances to their nearest centroid which of course minimizes the distortion. This is step 1 of the algorithm. Step 2 is to recalculate the cluster centroids so as to minimize the distortion. This can be achieved by taking the first order derivative of the error (Equation (5.2)) with respect to the j^{th} centroid and setting it to zero. A solution to the resulting equation gives us the k-Means centroid update rule as shown in Equation (5.3).

$$\frac{d(VQE_j)}{d(C_j)} = \sum_{s_i \in Q_j} (C_j - s_i) = 0 \tag{5.3}$$

$$C_j = \sum_{s_i \in Q_j} s_i / |Q_j| \tag{5.4}$$

Recall that Q_j is the set of points closest to the centroid of the j^{th} cluster.

Iterating through these two steps therefore monotonically decreases the distortion. However, the algorithm is prone to getting stuck in local minima. Nevertheless, good pragmatic results can be obtained, hence the algorithm's popularity. We acknowledge that a more complicated analysis of the algorithm exists, namely an interpretation of the algorithm as analogous to the Newton-Raphson algorithm [2]. Our derivation of a new constrained version of k-Means is not inconsistent with these other explanations.

The key step in deriving a constrained version of k-Means is to create a new *differentiable* error function which we call the *constrained* vector quantization error. Consider a a collection of r must-link constraints and s cannot-link constraints. We can represent the instances affected by these constraints by two functions $g(i)$ and $g'(i)$. These functions return the *cluster index* (i.e., a value in the range 1 through k) of the closest cluster to the 1^{st} and 2^{nd} instances governed by the i^{th} constraint. For clarity of notation, we assume that the instance associated with the function $g'(i)$ violates the constraint if at all. The new error function is shown in Equation (5.5).

$$CVQE_j = \frac{1}{2} \sum_{s_i \in Q_j} T_{j,1} + \frac{1}{2} \sum_{l=1, g(l)=j}^{s+r} (T_{j,2} \times T_{j,3}) \tag{5.5}$$

where

$$\begin{aligned} T_{j,1} &= (C_j - s_i)^2 \\ T_{j,2} &= \left[(C_j - C_{g'(l)})^2 \neg \Delta(g'(l), g(l))\right]^{m_l} \\ T_{j,3} &= \left[(C_j - C_{h(g'(l))})^2 \Delta(g(l), g'(l))\right]^{1-m_l}. \end{aligned}$$

Here $h(i)$ returns the index of the cluster (other than i) whose centroid is closest to cluster i's centroid and Δ is the Kronecker Delta Function defined by $\Delta(x, y) = 1$ if $x = y$ and 0 otherwise. We use $\neg\Delta$ to denote the negation of the Delta function.

The first part of the new error function is the regular distortion. The remaining terms are the errors associated with the must-link (m_l=1) and cannot-link (m_l=0) constraints. In future work we intend to allow the parameter m_l to take any value in $[0, 1]$ so that we can allow for a continuum between must-link and cannot-link constraints. We see that if a must-link constraint is violated then the cost is equal to the distance between the cluster centroids containing the two instances that should have been in the same cluster. Similarly, if a cannot-link constraint is violated the cost is the distance between the cluster centroid both instances are in and the nearest cluster centroid to one of the instances. Note that both violation costs are in units of distance, as is the regular distortion.

The first step of the constrained k-Means algorithm must minimize the new constrained vector quantization error. This is achieved by assigning instances so as to minimize the new error term. For instances that are not part of constraints, this involves as before, performing a nearest cluster centroid calculation. For pairs of instances in a constraint, for each possible combination of cluster assignments, the $CVQE$ is calculated and the instances are assigned to the clusters that minimally increases the $CVQE$.

The second step is to update the cluster centroids so as to minimize the constrained vector quantization error. To achieve this we take the first order derivative of the error, set to zero, and solve. By setting the appropriate values of m_l we can derive the update rules (Equation (5.6)) for the must-link and cannot-link constraint violations. The resulting equations are shown below.

$$\begin{aligned}\frac{d(CVQE_j)}{d(C_j)} &= \sum_{s_i \in Q_j} (C_j - s_i) + \\ &\quad \sum_{l=1, g(l)=j, \Delta(g(l), g'(l))=0}^{s} (C_j - C_{g'(l)}) + \\ &\quad \sum_{l=s+1, g(l)=j, \Delta(g(l), g'(l))=1}^{s+r} (C_j - C_{h(g'(l))}) \\ &= 0\end{aligned}$$

Solving for C_j, we get

$$C_j = Y_j / Z_j \tag{5.6}$$

where

$$\begin{aligned}Y_j &= \sum_{s_i \in Q_j} s_i + \sum_{l=1, g(l)=j, \Delta(g(l), g'(l))=0}^{s} C_{g'(l)} + \\ &\quad \sum_{l=s+1, g(l)=j, \Delta(g(l), g'(l))=1}^{s+r} C_{h(g'(l))}\end{aligned}$$

and

$$\begin{aligned}Z_j &= |Q_j| + \sum_{g(l)=j, l=1}^{s} (1 - \Delta(g(l), g'(l))) + \\ &\quad \sum_{g(l)=j, l=s+1}^{s+r} \Delta(g(l), g'(l)))\end{aligned}$$

The intuitive interpretation of the centroid update rule is that if a must-link constraint is violated, the cluster centroid is moved towards the other cluster containing the other point. Similarly, the interpretation of the update rule for a cannot-link constraint violation is that cluster centroid containing both constrained instances should be moved to the nearest cluster centroid so that one of the instances eventually gets assigned to it, thereby satisfying the constraint.

6 Experiments with Minimization of the Constrained Vector Quantization Error

In this section we compare the usefulness of our algorithm on three standard UCI data sets. We report the average over 100 random restarts (initial assignments of instances). As others have reported [19] for k=2 the addition of constraints improves the purity of the clusters with respect to an extrinsic binary class not given to the clustering algorithm. We observed similar behavior in our experiments. We are particularly interested in using our algorithm for more traditional unsupervised learning where $k > 2$. To this end we compare regular k-Means against constrained k-Means with respect to the number of iterations until convergence and the number of constraints violated. For the PIMA data set (Figure 4) which contains two extrinsic classes, we created a random combination of 100 must-link and 100 cannot-link constraints between the same and different classes respectively. Our results show that our algorithm converged on average in 25% fewer iterations while satisfying the vast majority of constraints.

Similar results were obtained for the BreastCancer data sets (Figure 6) where 25 must-link and 25 cannot-link constraints were used and Iris (Figure 8) where 13 must-link and 12 cannot-link constraints were used.

7 Experiments with the Sony Aibo Robot

In this section we describe an example with the Sony Aibo robot to illustrate the use of our newly proposed δ and ϵ constraints.

The Sony Aibo robot is effectively a walking computer with sensing devices such as a camera, microphone and an infra-red distance detector. The on-board processor has limited processing capability. The infra-red distance detector is connected to the end of the head and can be moved around to build a distance map of a scene. Consider the scene (taken from a position further back than the Aibo was for clarity) shown in Figure 5. We wish to cluster the distance information from the infra-red sensor to form objects/clusters (spatially adjacent points at a similar distance) which represent solid objects that must be navigated around.

Unconstrained clustering of this data with k=9 yields the set of significant (large) objects shown in Figure 7.

However, this result does not make use of important background information. Firstly, groups of clusters

Figure 4: Performance of regular and constrained k-Means on the UCI PIMA dataset

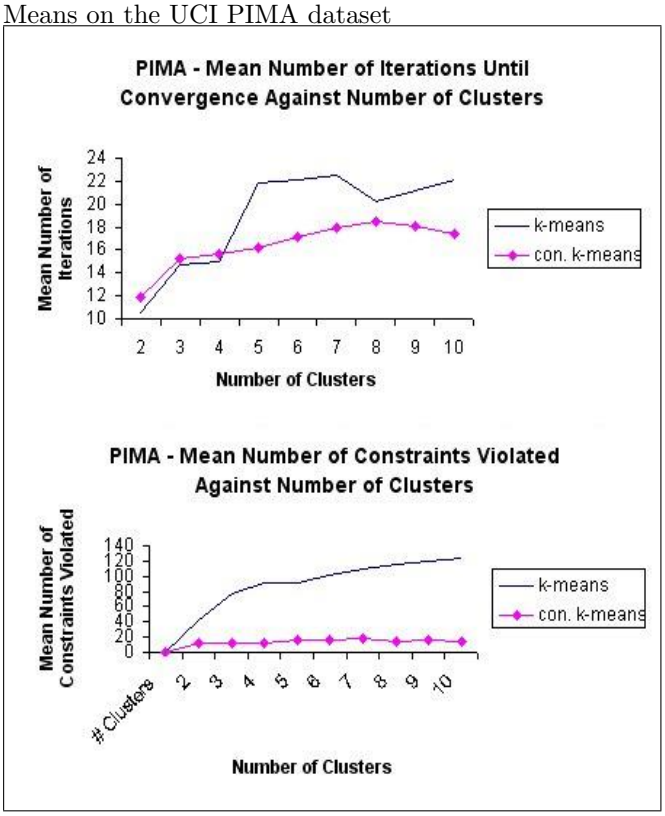


Figure 5: An image of the scene to navigate

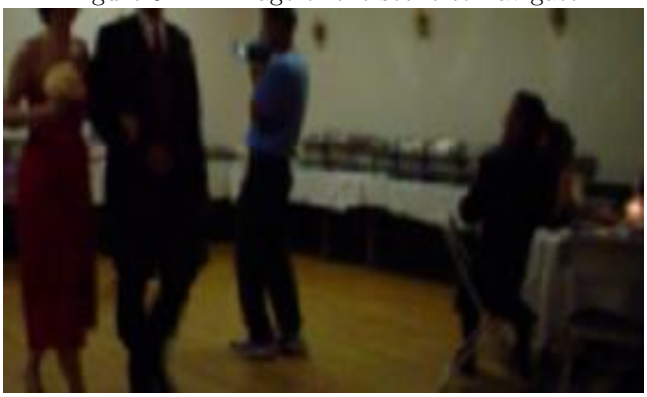


Figure 6: Performance of regular and constrained k-Means on the UCI Breast Cancer dataset

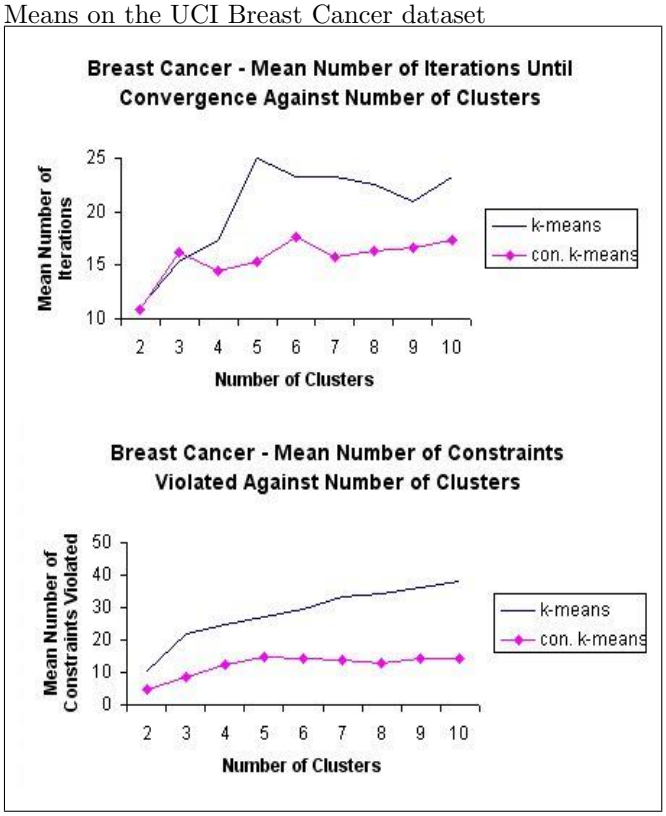


Figure 7: Unconstrained clustering of the distance map using k=9. The approximate locations of significant (large) clusters are shown by the ellipses.

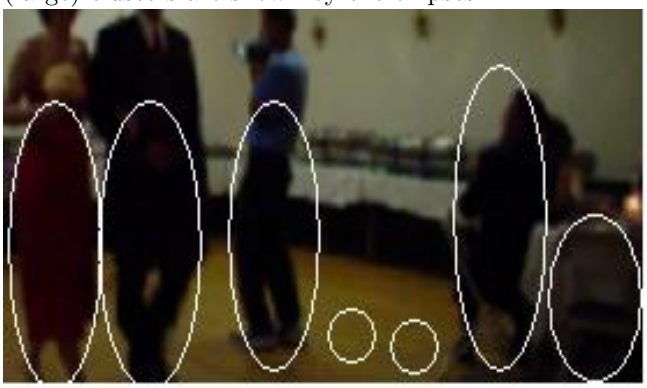
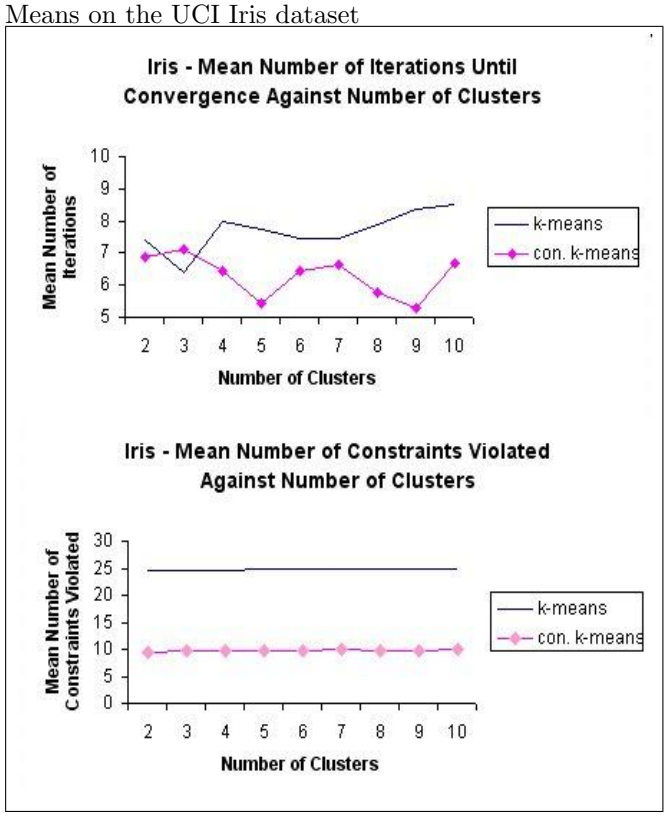


Figure 8: Performance of regular and constrained k-Means on the UCI Iris dataset

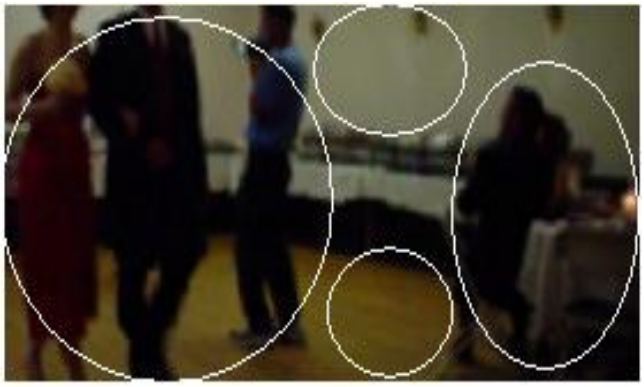


(such as the people) that are separated by a distance less than one foot can effectively be treated as one big cluster since the Aibo robot cannot easily walk through a gap smaller than one foot. Secondly, often one contiguous region is split into multiple objects due to errors in the infra-red sensor. These errors are due to poor reception of the incoming signal or the inability to reflect the outgoing signal which occurs in the wooden floor region of the scene. These errors are common in the inexpensive sensor on the Aibo robot.

Since we know the accuracy of the Aibo head movement we can determine the distance between the furthest (about three feet) adjacent readings which determines a lower bound for ϵ (namely, $3 \times \tan 1°$) so as to ensure a feasible solution. However, if we believe that it is likely that one but unlikely that two incorrect adjacent mis-readings occur, then we can set ϵ to be twice this lower bound to overcome noisy observations.

Using these values for our constraints we can cluster the data under our background information and obtain the clustering results shown in Figure 9. We see that the clustering result is now more useful for our intended purpose. The Aibo can now move towards the area/cluster that represents an open area.

Figure 9: Constrained clustering of the distance map using k=9. The approximate locations of significant (large) clusters are shown by the ellipses.



8 Conclusion and Future Work

Clustering with background prior knowledge offers much promise with contributions using must-link and cannot-link instance level constraints having already been published. We introduced two additional constraints: ϵ and δ. We studied the computational complexity of finding a *feasible* solution for these four constraint types individually and together. Our results show that in many situations, finding a feasible solution under a combination of constraints is **NP**-complete. Thus, an iterative algorithm should not try to find a feasible solution in each iteration.

We derived from first principles a constrained version of the k-Means algorithm that attempts to minimize the proposed constrained vector quantization error. We find that the use of constraints with our algorithm results in faster convergence and the satisfaction of a vast majority of constraints. When constraints are not satisfied it is because it is less costly to violate the constraint than to satisfy it by assigning two quite different (i.e. far apart in Euclidean space) instances to the same cluster in the case of must-link constraints. Future work will explore modifying hierarchical clustering algorithms to efficiently incorporate constraints.

Finally, we showed the benefit of our two newly proposed constraints in a simple infra-red distance clustering problem. The δ constraint allows us to specify the minimum distance between clusters and hence can encode prior knowledge regarding the spatial domain. The ϵ constraint allows us to specify background information with regard to sensor error and data collection.

References

[1] A. Hertz and D. de Werra, "Using Tabu Search Tech-

niques for Graph Coloring", *Computing*, Vol. 39, 1987, pp. 345–351.

[2] L. Bottou and Y. Bengio, "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.

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

[4] S. Basu, M. Bilenko and R. J. Mooney, "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.

[5] S. Basu, M. Bilenko and R. J. Mooney, "Active Semi-Supervision for Pairwise Constrained Clustering", *Proc. 4th SIAM Intl. Conf. on Data Mining* (SDM-2004).

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

[7] G. Campers, O. Henkes and J. P. Leclerq, "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.

[8] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, Cambridge, MA, 2001.

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

[10] I. Davidson and A. Satyanarayana, "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.

[11] M. Ester, H. Kriegel, J. Sander and X. Xu, "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.

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

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

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

[15] D. Klein, S. D. Kamvar and C. D. Manning, "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.

[16] D. Pelleg and A. Moore, "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.

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

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

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

9 Appendix

Here, we show that the feasibility problem for cannot-link constraints (CL-feasibility) is **NP**-complete using a reduction from the GRAPH K-COLORABILITY problem (K-COLOR) [12].

GRAPH K-COLORABILITY (K-COLOR)

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

Question: Can the nodes of G be colored using at most K colors so that for every pair of adjacent nodes u and v, the colors assigned to u and v are different?

THEOREM 9.1. *The CL-feasibility problem is **NP**-complete.*

Proof: It is easy to see that the CL-feasibility problem is in **NP**. To prove **NP**-hardness, we use a reduction from the K-COLOR problem. Let the given instance I of K-COLOR problem consist of 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 cannot-link 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. ■