

A SAT-based Framework for Efficient Constrained Clustering

Ian Davidson *

S. S. Ravi †

Leonid Shamis ‡

Abstract

The area of clustering under constraints has recently received much attention in the data mining community. However, most work involves adding constraints to **existing** algorithms which, although being quite pragmatic, raises several difficulties. Examples of these difficulties include creating intractable constraint satisfaction sub-problems and constrained clustering algorithms that are easily over-constrained so they may not converge or converge to a poor clustering solution. In this paper we show how both instance and cluster-level constraints can be expressed as instances of the 2SAT problem and how multiple calls to a 2SAT solver can be used to construct algorithms that are guaranteed to satisfy all the constraints and converge to a global optimum for a number of intuitive objective functions. Our approach provides two additional advantages. Firstly, it leads to polynomial time algorithms for the $k = 2$ case for several objective functions. Secondly, one can specify large sets of constraints without fear of over-constraining the problem: if one or more solutions satisfying all constraints exist, our algorithm is guaranteed to find a best such solution. We present experimental results to show that our approach outperforms several popular algorithms particularly for large constraint sets, where these algorithms are over-constrained and fair poorly.

1 Introduction and Motivation

Wagstaff and Cardie [15, 16, 17] first introduced constraints to the machine learning and data mining communities. The introduction of constraints addresses an important problem elegantly: the clustering algorithm's objective function need not capture all the domain expert's requirements, but user specified constraints can help guide the algorithm to a desirable set partition. Wagstaff and Cardie introduced simple instance-level constraints that were termed must-link (two instances must be in the same cluster) and cannot-link (two in-

stances must be in different clusters). They added these constraints to the k -means and COB-WEB algorithms with others adding these constraints to EM [1, 3], hierarchical [8] and spectral clustering algorithms [4], just to name a few. Most algorithms make use of these two types of constraints and typically for the situation $k = 2$ since constraints are often derived from two-class labeled instances [2].

However, the addition of constraints to pre-existing clustering algorithms is not without its difficulties some of which we list below.

1. The addition of constraints is typically in a form that is **incompatible** with the objective function. This makes it difficult to trade-off satisfying most constraints with minimizing the objective function. For example, k -means minimizes the vector quantization error (a function of pairwise distances) but one cannot easily add another part to the objective function to guarantee satisfying all constraints (a decision problem).
2. The addition of cannot-link constraints can lead to a **computationally intractable feasibility sub-problem** [6]; that is, unless $\mathbf{P} = \mathbf{NP}$, there is no efficient algorithm for finding **any** clustering that satisfies all the constraints, let alone a best such clustering. Workarounds such as trying to minimally prune instance-level constraints have also been shown to be computationally intractable [7].
3. Even if a feasible clustering satisfying all constraints exists, algorithms that attempt to satisfy all constraints are easily **over-constrained** so that they do not converge [6]. This is so since they typically build up a clustering incrementally (without back-tracking) to satisfy the constraints.
4. Finally, the whole idea behind constrained clustering is to allow the user to guide the algorithm to a desired set partition. However, due to the complex interaction between constraints and the objective functions used by existing algorithms, this is often not the case. This is starkly illustrated in our earlier work [9] where for a variety of algorithms and

*Department of Computer Science, University of California - Davis, Davis, CA 95616, davidson@cs.ucdavis.edu

†Department of Computer Science, University at Albany - Albany, NY 12222, ravi@cs.albany.edu

‡Department of Computer Science, University of California - Davis, CA 95616, lshamis@ucdavis.edu

Data	Algorithm			
	CKM [17]	PKM [3]	MKM [3]	MPKM [3]
Glass	31%	7%	16%	3%
Ionosphere	26%	68%	3%	80%
Iris	27%	23%	31%	41%
Wine	35%	38%	82%	69%

Table 1: Fraction of 1000 randomly selected 25-constraint sets that caused a drop in accuracy, compared to an unconstrained run with the same centroid initialization for a variety of constrained clustering algorithms.

constraint sets generated from the **ground truth labels**, the quality (as measured by the cluster purity using those same labels) of the resulting clustering is often **worse** compared to using no constraints. This result is reproduced in Table 1.

To overcome these issues, we consider formulating **both** the constraints and the objective function in the same underlying language, namely propositional formulas in conjunctive normal form (CNF). In this paper we focus on the first but pragmatic step of this research by considering the $k = 2$ case, that is, partitioning a given set of points into two clusters. This may be considered as limiting but it should be noted that $k = 2$ is an often studied problem [2] in the constrained clustering literature given the sources of constraints are often two-class labels. Furthermore, the $k = 2$ case has recently been shown to be intractable for popular algorithms such as k -means and self organized maps (SOM) that attempt to optimize the vector quantization error [10].

We observe that the resulting constrained optimization problems can be solved efficiently using a subroutine for solving 2SAT (i.e., CNF formulas in which each clause has at most two literals). The number of calls to the 2SAT subroutine is typically $O(\log n)$, where n is the number of points to be clustered.

Contributions. We make several contributions to the field of constrained clustering as summarized below.

- We show how instance-level constraints (must-link and cannot-link) and complex cluster-level constraints such as maximum diameter and minimum separation can be represented in the language of a CNF expression in Section 2. A long term goal is to have a language of constraints based on a first order logic.
- Using the fact that every optimization problem has a corresponding decision problem, we show how the

cluster-level constraints can be used as a framework for objective functions such as minimizing the maximum diameter, minimizing the difference in diameters and even multi-objectives such as minimizing the maximum diameter while maximizing the minimum separation in Section 3 and Proposition 3.1.

- Unlike most existing work [5] we present a setting for constrained clustering that is not easily over-constrained and is guaranteed to find a global optimum (provided there are solutions satisfying all the constraints).
- All our algorithms run in polynomial time. Further, in Section 4 we show if a near-optimal solution¹ is acceptable, our approach can also lead to asymptotically faster approximation schemes (see Theorem 4.1).
- We show how SAT solvers can be used to globally optimize a constrained clustering problem directly, rather than solving a relaxed version of the problem (as done in spectral clustering) or just finding local minima (as done by the k -means algorithm).

2 Expressing Constrained Clustering Problems as CNF Formulas

For the remainder of this paper, we assume that a given set $S = \{s_1, s_2, \dots, s_n\}$ with n points is to be partitioned into two clusters; that is, $k = 2$. We also assume that an $n \times n$ distance matrix $D = [D_{ij}]$, where D_{ij} represents the distance between s_i and s_j , is given.

We assume that the reader is familiar with the satisfiability problem (SAT) for formulas in conjunctive normal form. Throughout this paper, we refer to the 2SAT problem, which is a special case of SAT in which each clause has at most two literals. A proof of the following theorem can be found in [13].

THEOREM 2.1. *Given a 2SAT formula F with n variables and m clauses, the satisfiability of F can be determined in time $O(n + m)$. If F is satisfiable, then a satisfying assignment for F can also be constructed in $O(n + m)$ time. \square*

To express the clustering problem as a CNF formula, each point s_i is represented by a propositional variable X_i , $i = 1, 2, \dots, n$. Since $k = 2$, we may use indices 0 and 1 to denote the two clusters. We use the convention that if X_i is assigned the value TRUE (FALSE), point s_i belongs to cluster 1 (0).

¹By “near optimal”, we mean that the solution is within a factor of $(1 + \epsilon)$ of the optimal solution value, for any given $\epsilon > 0$.

2.1 Instance-Level Constraints We start by observing how must-link (ML) and cannot-link (CL) constraints can be expressed as CNF formulas. The notation $ML(i, j)$ means that points s_i and s_j must be in the same cluster. Similarly, the notation $CL(i, j)$ means that points s_i and s_j must *not* be in the same cluster. In all cases, we assume that the set C of clauses to be constructed is initially empty and that clauses resulting from various constraints are added to C .

DEFINITION 1. $ML(i, j)$

The constraint $ML(i, j)$ is logically equivalent to the formula $(X_i \Leftrightarrow X_j)$, which is, in turn, logically equivalent to the CNF formula $(\neg X_i \vee X_j) \wedge (X_i \vee \neg X_j)$. Therefore, the constraint $ML(i, j)$ can be accommodated by setting $C = C \cup \{(\neg X_i \vee X_j), (X_i \vee \neg X_j)\}$.

DEFINITION 2. $CL(i, j)$

The constraint $CL(i, j)$ is logically equivalent to the formula $(X_i \Leftrightarrow \neg X_j)$, which is, in turn, logically equivalent to the CNF formula $(X_i \vee X_j) \wedge (\neg X_i \vee \neg X_j)$. Therefore, the constraint $CL(i, j)$ can be accommodated by setting $C = C \cup \{(X_i \vee X_j), (\neg X_i \vee \neg X_j)\}$.

We note that both ML and CL constraints add clauses with two literals to C . Thus, the resulting set C of clauses represents an instance of 2SAT.

2.2 Cluster-Level Constraints We now indicate how certain cluster-level constraints can also be modeled using 2SAT. We recall that the **diameter** of a cluster Q is the *maximum* distance between a pair of points in Q . Further, the **separation** between two clusters S_1 and S_2 is the *minimum* distance between a pair of points, one from S_1 and the other from S_2 .

Cluster-Level Maximum Diameter. This cluster-level constraint requires that the diameter of any cluster be at most a given value α . To achieve this, we must ensure that any pair of points s_i and s_j with $D_{ij} > \alpha$ are in different clusters; that is, we need the constraint $CL(i, j)$. This is captured by the following definition (which uses the clauses for a CL constraint introduced in Definition 2).

DEFINITION 3. $Maximum-Diameter(\alpha)$

This constraint can be handled by adding to the set C the two clauses $(X_i \vee X_j)$ and $(\neg X_i \vee \neg X_j)$ for every pair of points s_i and s_j for which $D_{ij} > \alpha$, $1 \leq i < j \leq n$.

Cluster-Level Minimum Separation. This constraint requires that the separation between the two clusters be at least a given value β . To achieve this, we must ensure that each pair of points s_i and s_j with

$D_{ij} < \beta$ are in the same cluster; that is, we need the constraint $ML(i, j)$. This is captured by the following definition (which uses the clauses for an ML constraint introduced in Definition 1).

DEFINITION 4. $Minimum-Separation(\beta)$

This constraint can be handled by adding to the set C the two clauses $(\neg X_i \vee X_j)$ and $(\neg X_i \vee X_j)$ for every pair of points s_i and s_j for which $D_{ij} < \beta$, $1 \leq i < j \leq n$.

We note that the maximum diameter and minimum separation constraints also introduce only clauses with two literals; that is, the formula C continues to be an instance of 2SAT.

3 Extending Cluster-Level Constraints to a Family of Objective Functions

Here, we make use of the fact that for every optimization problem there is a corresponding decision problem. We begin with the case where there is a single objective function and then consider a multi-objective function.

3.1 Optimizing a Single Objective

We show here how repeated calls to 2SAT instances generated from the constraints can be used to develop an optimization procedure for the maximum diameter and minimum separation objectives. An important observation is that the optimum value for either of these objectives must be one of the pairwise distance values in the matrix D . Therefore, as long as a solution satisfying all the constraints exists, the optimum value can be found by carrying out a binary search over the distance values in D . The corresponding algorithm shown in Figure 1. We will explain how the algorithm works for minimizing the maximum diameter. A similar explanation can be given for maximizing the minimum separation.

The algorithm begins by sorting the pairwise distances and chooses the median value as a candidate solution to the diameter problem. A 2SAT instance is constructed where clauses are created so that any pair of points whose distance is greater than the median value are required to be in different clusters. (These are in addition to the clauses for the given ML and CL constraints.) If the 2SAT solver returns a solution (clustering) for this problem, then we set the median as the new upper bound on the diameter (we are minimizing the objective function), select the new median value and construct another 2SAT instance. If the 2SAT solver fails, then we know this median value is below the optimum value and we set it as the new lower bound, recalculate the median and construct another 2SAT instance as before. This is repeated until the algorithm converges to the smallest legal cluster diameter that also satisfies all ML and CL constraints.

Algorithm SingleFunctionOptimize

Input:

- D : Matrix of pairwise distances.
 C : The set of ML and CL constraints expressed as clauses.
 f : Objective: {Minimize Maximum-Diameter, Maximize Minimum-Separation}.

Output: The optimal value γ for the objective function and a corresponding 2-clustering.

1. Create a sorted array $L[1 .. t]$ in **increasing** order of all distinct pairwise distances in D . Let $\ell = 1$ and $h = t$.
2. **while** ($\ell \neq h$) **do**
3. Let $m = \lceil(\ell + h)/2\rceil$ and $\gamma = L[m]$.
4. Create a 2SAT instance P using C , γ and Definition 3 or 4 depending on f .
5. **if** P is satisfiable
6. **if** f is Minimize-Diameter, let $h = m$.
7. **if** f is Maximize-Separation, let $\ell = m$.
8. **else** /* P is not satisfiable */
9. **if** f is Minimize-Diameter, let $\ell = m + 1$.
10. **if** f is Maximize-Separation, let $h = m - 1$.
11. **end while**
12. **return** $\gamma = L[\ell]$ and the corresponding 2-clustering (based on the solution to P).

Figure 1: Optimizing a Single Objective Function

In specifying the algorithm in Figure 1, we assume that there is a solution that satisfies all the given ML and CL constraints² in the set Ψ . Note that this can be determined using just one call to the algorithm for 2SAT.

It can be seen from the above discussion that given a set of points S and a collection Ψ of ML and CL constraints, the (decision) problem of determining whether there is a 2-clustering of S such that all the constraints in Ψ are satisfied and the maximum cluster diameter is α can be solved efficiently by constructing the corresponding 2SAT instance and testing whether it is satisfiable. In particular, since the number of possible distinct constraints (and hence the number of clauses in the 2SAT instance) is $O(n^2)$, the running time of the 2SAT algorithm would be $O(n^2)$.

We will use the notation $\text{DIA-TEST}(S, \alpha)$ to denote a function that returns “Yes” if there is a partition of S into two clusters such that a given set Ψ of ML and CL constraints is satisfied and each cluster has a diameter of at most α . To avoid clutter, we think of Ψ as an implicit set of constraints and do not include Ψ in the notation $\text{DIA-TEST}(S, \alpha)$. Note that the function $\text{DIA-TEST}(S, \alpha)$ can be implemented by one call to the algorithm for 2SAT. Table 2 includes the definitions of $\text{DIA-TEST}(S, \alpha)$ and other similar functions which will be used in the remainder of this paper. In each case, whenever the answer is “Yes”, we can also obtain a 2-clustering from the satisfying assignment returned by the algorithm for 2SAT.

3.2 Other Single Objectives

Minimizing the Difference Between Diameters.

One can also exploit the relationship to 2SAT to optimize other functions of diameters. For example, one may be interested in obtaining a 2-clustering such that the *difference* between the diameters of the two clusters is minimized. Intuitively, this corresponds to finding two clusters **which have nearly equal diameters**. This problem can be solved efficiently by considering the following generalization of the diameter problem: Given two positive values α_1 and α_2 , where $\alpha_1 \leq \alpha_2$, is there a partition into two clusters S_1 and S_2 such that the diameter of S_i is at most α_i for $i = 1, 2$? This problem can also be reduced to 2SAT as follows.

As before, let X_i denote the Boolean variable corresponding to point s_i , $1 \leq i \leq n$. We will continue to use the convention that after solving the 2SAT problem, all points corresponding to variables set

²Note that these constraints are separate from the constraints due to the objective function.

Function	Interpretation
DIA-TEST(S, α)	Returns “Yes” if there is a partition of S into two clusters such that each cluster has a diameter of at most α ; otherwise, returns “No”.
SEP-TEST(S, β)	Returns “Yes” if there is a partition of S into two clusters such that the separation between the two clusters is at least β ; otherwise, returns “No”.
DIA-SEP-TEST(S, α, β)	Returns “Yes” if there is a partition of S into two clusters such that each cluster has a diameter of at most α and the separation between the two clusters is at least β ; otherwise, returns “No”.

Table 2: List of functions used in subsequent sections. (Each function can be implemented using one call to the algorithm for 2SAT. In each case, there may be additional implicit set Ψ of ML and CL constraints.)

to TRUE (FALSE) are in cluster S_1 (S_2). For each pair of points s_i and s_j ($i \neq j$), we do the following.

1. If $D_{ij} > \alpha_2$, then we add the two clauses $(X_i \vee X_j)$ and $(\neg X_i \vee \neg X_j)$. (This is equivalent to the constraint CL(i, j)).
2. If $\alpha_1 < D_{ij} \leq \alpha_2$, we introduce the clause $(\neg X_i \vee \neg X_j)$. (This is to ensure that the two points don’t appear together in S_1 , the cluster with the smaller diameter α_1).

It is easy to verify that the resulting instance of 2SAT has a solution if and only if there is a 2-clustering where the diameter of S_i is at most α_i , $i = 1, 2$.

Let TWO-DIA-TEST(S, α_1, α_2) denote the function corresponding to the above generalized version of the diameter problem. It is a simple matter to use this function to solve the problem of minimizing the difference between the diameters. One can try each possible pair (α_1, α_2) of diameter values, where $\alpha_1 \leq \alpha_2$. For each value of α_1 , the smallest value of α_2 such that TWO-DIA-TEST(S, α_1, α_2) returns “Yes” can be found using a binary search on the possible values of α_2 . Thus, for each value of α_1 , the number of calls of the form TWO-DIA-TEST(S, α_1, α_2), each of which results in a call to 2SAT, is $O(\log n)$. Since the number of possible

values of α_1 is $O(n^2)$, the total number of calls to 2SAT to get the smallest difference between the diameter values is $O(n^2 \log n)$. We note that here also, a set Ψ of ML and CL constraints can be accommodated without increasing the number of calls to 2SAT.

Minimizing Sum of Diameters. In the previous section, we considered the objective of minimizing the maximum diameter of a cluster. Given a partition into two clusters, this objective corresponds to taking the *maximum* of the diameters. Instead of maximum, one can consider taking the *sum* of the diameters of the two clusters. This leads to the problem of finding a 2-clustering where the goal is to minimize the sum of the two diameters. For this problem, an algorithm with a running time of $O(n^3 \log n)$ was presented in [11]. An improved algorithm with a running time of $O(n^3)$ was presented in [14]. Both of these algorithms rely on efficient algorithms for 2SAT. We note that the use of 2SAT also allows us to solve the minimum sum of diameters problem subject to ML and CL constraints.

3.3 Time Complexity of Single Objective Algorithm

This section refers to the time complexity of the algorithm shown in Figure 1 which is used to produce our experimental results in Section 5. This algorithm is guaranteed to converge to the global optimum. In Section 4 we describe an approximation scheme that is guaranteed to converge within $1 - \epsilon$ of the true optimum which has a time complexity of $O(\log \log \frac{D_{max}}{D_{min}})$ where D_{max} and D_{min} are the maximum and minimum values in the distance matrix $[D_{ij}]$.

Since the binary search is over an array of size $O(n^2)$, the number of calls to 2SAT used by the algorithm in Figure 1 is $O(\log(n^2)) = O(\log n)$. Since each call to the 2SAT solver has complexity $O(n^2)$ in the worst case the overall worst case complexity of our approach is $O(n^2 \log n)$ where n is the number of data points. We note that this worst case analysis assumes the number of clauses is $\Theta(n^2)$; in practice, the number of clauses would be far less.

3.4 Multiple Objectives

So far we have limited ourselves to single objective functions. Our framework also lends itself to combining these objective functions though optimizing multiple objectives is non-trivial. Possible methods for optimizing multiple objectives include the following.

1. A bi-criteria optimization approach.
2. Optimizing a ratio of single objective functions.
3. Seeking a Pareto optimum.

In this section we explore the first two approaches; the third approach is left for future work.

A Bi-criteria Optimization Approach. Suppose we want to consider the minimum diameter and maximum separation objectives together. One way to handle both of these objectives is to use one of them as a constraint and optimize the other subject to that constraint. For example, constraining the cluster diameter to be no more than a chosen value α , one can focus on maximizing the separation subject to this constraint. This can also be done by a binary search similar to the one shown in Figure 1. The steps of this procedure are as follows.

1. Construct the set C of clauses using the given constraint set Ψ and the diameter bound α .
2. Carry out a binary search over the distances in D to find the largest separation β satisfying Ψ and the diameter bound α . (Each iteration of the binary search uses a call to the $\text{SEP-TEST}(S, \beta)$ function defined in Table 2.)

Thus, we can find the maximum separation value for each specified diameter bound α using $O(\log n)$ calls to 2SAT. In a similar manner, we can also find the minimum diameter for each specified separation bound β using $O(\log n)$ calls to 2SAT.

In practice, when both the diameter and separation objectives are considered, one can save some calls to the 2SAT algorithm using the following result.

PROPOSITION 3.1. *Suppose α and β denote the required upper bound on the diameter and the required lower bound on the separation respectively, with $\alpha < \beta$. If there are points s_i and s_j such that $\alpha < D_{ij} < \beta$, then no feasible 2-clustering satisfying both constraints exists.*

Proof. From Definitions 3 and 4, it can be seen that the diameter and separation constraints on s_i and s_j lead to the following clauses:

$$\begin{array}{ll} (1) (X_i \vee X_j) & (2) (\neg X_i \vee \neg X_j) \\ (3) (\neg X_i \vee X_j) & (4) (X_i \vee \neg X_j). \end{array}$$

Resolving (1) and (3) produces X_j and resolving (2) and (4) produces the contradiction $\neg X_j$. Hence, the set of clauses is not satisfiable.

Minimizing A Ratio of Objective Functions. In the last section, we considered the two objectives, namely minimizing the diameter and maximizing the separation, separately. They can also be combined into a single objective as follows: find a 2-clustering that minimizes the ratio of diameter to separation. This problem can also be solved efficiently using 2SAT as

follows. Recall that given a diameter value α and separation value β , the problem of determining whether there is a 2-clustering such that the maximum diameter is at most α and the minimum separation is at least β can be solved using 2SAT. Let $\text{DIA-SEP-TEST}(S, \alpha, \beta)$ denote a procedure which solves this problem as indicated in Table 2. Note that this procedure can be implemented by one call to an appropriately constructed instance of 2SAT. To solve the problem of minimizing the diameter to separation ratio, we proceed as follows. (The method is similar to the one for minimizing the difference between the diameters.) For each value of α , the largest value of β such that $\text{DIA-SEP-TEST}(S, \alpha, \beta)$ returns “Yes” can be found using a binary search on the possible values of β . Thus, for each value of α , the number of calls of the form $\text{DIA-SEP-TEST}(S, \alpha, \beta)$, each of which results in a call to 2SAT, is $O(\log n)$. Since the number of possible values of α is $O(n^2)$, the total number of calls to 2SAT to get the smallest diameter to separation ratio is $O(n^2 \log n)$. As before, a set Ψ of ML and CL constraints can be also accommodated without increasing the number of calls to 2SAT.

3.5 Extensions to Incorporate Dimension-Level Constraints.

To extend the above constraints to the dimension-level counterparts requires that for each of the d dimensions, pairwise distances be stored in matrices $D^{(r)}$, $r = 1, \dots, d$. Then dimension-level constraints are the same as above except the test is on a specific dimension r , that is, matrix $D^{(r)}$ rather than D . (The extension to a diameter-level constraint on a subset of the d dimensions is straightforward.) Of course, this may decrease the probability that a feasible clustering exists, but fortunately checking for this situation can be done efficiently because of Theorem 2.1.

4 Reducing the Number of Calls to 2SAT: Approximation Schemes

Basic Idea. In the previous section, it was shown that for $k = 2$, a clustering which minimizes the maximum diameter (or maximizes the minimum separation) can be obtained in polynomial time using a subroutine for 2SAT. With n points to be clustered, the number of calls to the 2SAT subroutine was shown to be $O(\log n)$. In this section, we observe that the number of calls to 2SAT can be asymptotically smaller if one is willing to accept an approximate solution which is provably within a factor $(1 + \epsilon)$ of the optimal value, for any $\epsilon > 0$. We will present the details for minimizing the maximum diameter and indicate the necessary modifications for maximizing the minimum separation.

Let D_{\min} and D_{\max} denote the smallest and largest distance values in the input. The number of calls to

-
1. Let $\ell = 0$ and $h = q$.
 2. **while** ($\ell \neq h$) **do**
 - (a) Let $m = \lceil(\ell + h)/2\rceil$.
 - (b) **if** DIA-TEST($S, D_{\min}(1 + \epsilon)^m$) returns “Yes” **then** $h = m$ **else** $\ell = m + 1$.
 3. **return** ℓ .
-

Figure 2: Steps of the Binary Search used in the Approximation Scheme

the 2SAT subroutine used by the approximation scheme presented in this section is³ $O(\log \log (D_{\max}/D_{\min}))$. Thus, the number of calls used by the approximation algorithm can be asymptotically smaller than that used by the optimization algorithm when the ratio D_{\max}/D_{\min} is significantly smaller than 2^n . Such approximation schemes are also known for other optimization problems (see for example [12]).

Steps of the Approximation Scheme for Diameter. Recall that for any $\alpha > 0$, the function DIA-TEST(S, α) defined in Table 2 can be implemented using one call to the 2SAT subroutine. Further, the function DIA-TEST(S, α) also returns the two clusters S_1 and S_2 .

Given any $\epsilon > 0$, our approximation scheme produces a solution which is at most $(1 + \epsilon)$ times the optimal diameter. The steps of this approximation scheme are as follows.

1. Let D_{\min} and D_{\max} denote the smallest and largest distances between points in S . Compute the smallest integer q such that $D_{\min}(1 + \epsilon)^q \geq D_{\max}$. Note that $q = O(\log (D_{\max}/D_{\min}))$.
2. Perform a binary search over the set of indices $\{0, 1, 2, \dots, q\}$ to find the *smallest* index j such that DIA-TEST($S, D_{\min}(1 + \epsilon)^j$) returns the answer “Yes”; that is, there is a partition of S into two clusters S_1 and S_2 such that each of the clusters has a diameter of at most $D_{\min}(1 + \epsilon)^j$. The steps of this binary search procedure are shown in Figure 2.
3. Return the partition (S_1, S_2) found in Step 2 above as the approximate solution.

³We use $\log \log (x)$ to denote $\log (\log (x))$.

We now establish the performance guarantee of the above approximation scheme and also bound the number of calls to 2SAT.

THEOREM 4.1. *The above approximation algorithm satisfies the following two properties.*

- (a) Let D denote the diameter returned by the approximation algorithm and let D^* denote the optimal diameter. Then, $D \leq (1 + \epsilon) D^*$.
- (b) The number of calls to 2SAT used by the algorithm is $O(\log \log (D_{\max}/D_{\min}))$.

Proof:

Part (a): Since D_{\min} is the smallest distance between a pair of points in S , we have $D^* \geq D_{\min}$. Let j be the smallest index in $\{0, 1, \dots, q\}$ such that DIA-TEST($S, D_{\min}(1 + \epsilon)^j$) returns “Yes”. If $j = 0$, then the diameter of both S_1 and S_2 is at most D_{\min} ; that is, we have an optimal solution. So, we may assume that $j > 0$. Since DIA-TEST($S, D_{\min}(1 + \epsilon)^j$) returns “Yes”, we have

$$(4.1) \quad D \leq D_{\min}(1 + \epsilon)^j.$$

Since $j > 0$ is the smallest index satisfying the above condition, DIA-TEST($S, D_{\min}(1 + \epsilon)^{j-1}$) returns “No”. In other words, there is no solution where each of the clusters has a diameter of at most $D_{\min}(1 + \epsilon)^{j-1}$. Therefore,

$$(4.2) \quad D^* > D_{\min}(1 + \epsilon)^{j-1}.$$

Using Equations (4.1) and (4.2), it follows that $D \leq (1 + \epsilon) D^*$.

Part (b): As mentioned in the description of the approximation algorithm, $q = O(\log (D_{\max}/D_{\min}))$; in other words, the size t of the set over which binary search is carried out is $O(\log (D_{\max}/D_{\min}))$. The number of indices tried by the binary search procedure is $O(\log t) = O(\log \log (D_{\max}/D_{\min}))$. For each such index, there is one call to DIA-TEST(S, α) or equivalently, one call to 2SAT. Hence the number of calls to 2SAT made by the approximation algorithm is $O(\log \log (D_{\max}/D_{\min}))$. \square

Modifications for Maximizing Minimum Separation. The outline of the approximation scheme and for maximizing the minimum separation and its analysis are similar to those for minimizing the maximum diameter. The key differences between the two approximation schemes are as follows.

- (a) Recall that for any β , the function SEP-TEST(S, β) defined in Table 2 can be implemented using one

call to the 2SAT subroutine. In the binary search procedure for approximating the minimum separation, we must use SEP-TEST(S, β) instead of DIA-TEST(S, α).

- (b) In the approximation scheme, we need to find the *smallest* index j such that the call to SEP-TEST($S, D_{\max}/(1 + \epsilon)^j$) returns “Yes”.

Let β^* denote the optimal separation. We note that $\beta^* \leq D_{\max}$. Carrying out an analysis similar to that for the diameter problem, we can conclude that the approximation scheme produces a separation β which satisfies the condition $\beta \geq \beta^*/(1 + \epsilon)$ for any chosen $\epsilon > 0$. The number of calls to 2SAT made by the approximation algorithm remains $O(\log \log (D_{\max}/D_{\min}))$.

5 Experimental Results

Our previous sections have shown that our algorithms are guaranteed to find the global optimum. Hence our experimental section is then not to test this claim (we already proved it), but rather ask other useful questions. We focus on our simplest objective functions, namely maximizing the minimum separation (min-sep) and minimizing the maximum diameter (max-dia).

We aim to answer several key questions in this section:

1. How does the quality of the constrained clustering found by our new algorithms compare with existing popular algorithms in the literature?
2. The performance of previous constrained clustering algorithms was often worse when compared to the case where no constraints were used (see Table 1). Do our algorithms exhibit the same behavior?

The answer to Question (1) is most important since it not only validates the usefulness of our objective functions but also the interaction of the constraints with these objective functions. We investigate four data sets, namely Ionosphere, Iris, Hepatitis and Heart Spectroscopy, for this purpose. These were chosen primarily since they are small enough to do many thousands of repetitive experiments so as to clearly understand the effects of constraints but also have **a large variety of properties**. For example, Iris can be considered as a two class problem since one class can be removed (as it is perfectly isolated) but the remaining two classes highly overlap in lower dimensional space. Ionosphere is a thirty-four dimensional continuous data set while the Heart spectroscopy data set is a nineteen dimensional Boolean dataset. In all experiments, constraints are generated in the standard way of sampling the labels of the points and if they agree generating a ML

Data	Algorithm			
	CKM	MPKM	Max-S	Min-D
Heart	0.50	0.55	0.66	0.66
Hepatitis	0.73	0.72	0.76	0.78
Ionosphere	0.59	0.59	0.64	0.66
Iris	0.83	0.82	0.87	0.88

Table 3: Average (over 1000 constraint sets) of the RAND index for a variety of algorithms using 24 randomly chosen constraints. The seminal COP- k -Means algorithm (CKM), metric learning conditional random field approach (MPKM) and our algorithms Max-S (maximize cluster separation) and Min-D (minimize cluster diameter). Our algorithms converge to global minima and hence were run only once per constraint set. CKM and MPKM were run ten times and the best results were chosen. Best performers (in bold) are statistically significant at the 95% confidence level using a pairwise student T test.

constraint otherwise a CL constraint. The success of a constrained clustering algorithm is then measured using the RAND index between the clustering found by the algorithm and the set partition induced by the labels.

Table 3 shows the average RAND index with respect to the labels (over 1000 constraint sets) for the four datasets previously mentioned. Note the COP- k -Means algorithm is a seminal work in the field and the MPKM algorithm is a combination of the popular PKM and MKM algorithms [3] that won the best paper award at KDD 2004. Since the COP- k -Means and MPKM algorithms converge to local minima, they were run 10 times from random restarts and the best results were chosen. The total run-time of our algorithms was typically far less than these other algorithms since they are run only once.

The base (unconstrained) RAND index results for maximizing minimum separation is 0.46 (Iris), 0.53 (Ionosphere), 0.72 (Hepatitis) and 0.49 (Heart) and for minimizing the maximum diameter is 0.70 (Iris), 0.58 (Ionosphere), 0.707 (Hepatitis) and 0.48 (Heart). Our algorithms perform better than the COP- k -Means and MPKM approaches but not due to the objective function (unconstrained k -means obtains better RAND index results than above), but rather due to the addition of constraints. An explanation of our better results can be found later where we describe how algorithms that attempt to iteratively construct a feasible set partition can become over-constrained.

An important property to investigate is whether the RAND index increases as a function of the number of constraints. Our results can be seen in Figures 3, 4, 5

and 6 where we report the average RAND index over fifty constraint sets for 0 (unconstrained), 10, 24, 50 and 100 constraints randomly generated from the data with an equal number of ML and CL constraints. For CKM and MPKM results are averaged over ten random restarts. We see two interesting trends. Firstly, even though our unconstrained objective functions (whose results correspond to zero sized constrained sets in the figures) are typically as good as (Heart and Ionosphere) or worse (Iris and Hepatitis) as k -means with the addition of constraints they quickly exceed the performance of other algorithms. This is most likely since our algorithms find a global optimum of their objective function rather than a local optimum.

Secondly, the RAND index does improve as the number of constraints increases, but we also see that it **continues** to rise as the number of constraints goes above fifty. The COP- k -means algorithm gets over-constrained and does not converge for **any of the runs when more than 50 constraints are generated for any dataset**. The reason is that we can view COP- k -means as being a greedy search algorithm that attempts to find the best clustering while attempting to satisfy all constraints **without backtracking** [6]. Consider a simple illustrative problem with just five instances a, b, c, d, e and the constraints $CL(a, b), CL(b, c), CL(c, d)$ and $CL(d, e)$. Clearly, this set of points can be clustered to satisfy all the constraints for $k = 2$. However, if the order in which instances are assigned to clusters is a, c, e, b, d and if a and c are assigned to clusters 1 and 2 respectively, then the COP- k -means algorithm cannot converge. Even though MPKM converges, it still suffers from this problem since it too incrementally constructs the set partition. We see that our algorithm works best when there are 50+ constraints.

Finally, to address Question (2) we break down the entries in Table 3 and see what proportion of the 1000 constraint sets for each of our algorithms resulted in a decrease of the RAND index compared to the unconstrained algorithm. These results are shown in Table 4 and by comparing it with Table 1 we see that the number of constraints set that lead to poorer results than using no constraints is still non-zero, but far less than comparable algorithms. We believe this is so since our algorithms are not easily over-constrained.

6 Conclusions and Future Work

In this paper we have taken the first step towards completely constraint driven clustering algorithms by showing how to formulate a variety of constraints as instances of the 2SAT problem. By repeated calls to a 2SAT solver, we showed how optimization goals such as min-

Data	Algorithm	
	Min-Sep	Max-Diam
Heart	8%	6%
Hepatitis	11%	5%
Ionosphere	11%	7%
Iris	8%	12%

Table 4: Fraction of 1000 randomly selected constraint sets of size 24 that caused a drop in accuracy, compared to the counterpart unconstrained algorithm. Compare with Table 1.

imizing the maximum cluster diameter or maximizing the minimum cluster separation can be achieved. We illustrated how optimizing these objective functions can be performed using $O(\log n)$ calls to a 2SAT solver, thus obtaining efficient algorithms. It was also shown in Section 4 how the number of calls can be further reduced if a near-optimal solution (factor $1 + \epsilon$ approximation) is acceptable resulting in $O(\log \log(\frac{D_{max}}{D_{min}}))$ calls. Finally, we illustrated how to perform efficient search with multi-objective functions. In all cases, the algorithms are guaranteed to not only converge to the global optima, but also to never become over-constrained.

Our experimental results in Table 3 show that our algorithms on average outperform other popular constrained clustering algorithms and (c.f. Table 1 and Table 4) are less likely to produce a result worse than using no constraints. The stability of our algorithms in this respect is of practical significance since it means constraint sets are unlikely to adversely affect the algorithm's performance. Figures 3, 4, 5 and 6 show that the performance of our algorithms increases with more constraints and are not over-constrained. This is an important property if constraints are plentiful or we have multiple sources of constraints.

The source code for our algorithms will be made available on our site www.constrained-clustering.org. Future work will address the following issues: (a) extension to $k > 2$, (b) using approximation results for weighted MAX-SAT to allow some constraints to be ignored and (c) the trade-off between the value of ϵ and the quality of solution in generating near-optimal solutions (Section 4).

Acknowledgments: The authors thanks the anonymous reviewers for their excellent comments and the NSF, ONR and Google for support of this work via NSF GRANT IIS-0801528 CAREER:Knowledge Enhanced Clustering with Constraints, ONR - Award N000140910712 P00001 and a Google Research Award.

References

- [1] A. Bar-Hillel, T. Hertz, N. Shental and D. Weinshall, “Learning Distance Functions Using Equivalence Relations”, *Proc. Intl. Conference on Machine Learning (ICML 2003)*, Washington, DC, Aug. 2003, pp. 11–18.
- [2] S. Basu, I. Davidson, and K. Wagstaff, (editors) *Constrained Clustering: Theory, Algorithms and Applications*, CRC Prentice Hall, 2008.
- [3] 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, Aug. 2004, pp. 59–68.
- [4] T. Coleman, J. Saunderson and A. Wirth, “Spectral Clustering with Inconsistent Advice”, *Proc. Intl. Conference on Machine Learning (ICML 2008)*, Helsinki, Finland, June 2008, pp. 152–159.
- [5] I. Davidson and S. S. Ravi, “Identifying and Generating Easy Sets of Constraints For Clustering”, *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 2006)*, Boston, MA, July 2006, 6 pages.
- [6] I. Davidson and S. S. Ravi, “The Complexity of Non-Hierarchical Clustering with Constraints”, *J. Knowledge Discovery and Data Mining (DMKD)*, Vol. 14, No. 1, 2007, pp. 25–61.
- [7] I. Davidson and S. S. Ravi, “Intractability and Clustering with Constraints”, *Proc. Intl. Conference on Machine Learning (ICML 2007)*, Corvallis, OR, June 2007, pp. 201–208.
- [8] I. Davidson and S. S. Ravi, “Using Instance-Level Constraints in Hierarchical Agglomerative Clustering: Theoretical and Empirical Results”, *Data Mining and Knowledge Discovery*, Vol. 18, No. 2, Apr. 2009, pp. 257–282.
- [9] I. Davidson, K. Wagstaff and S. Basu, “Measuring Constraint-Set Utility for Partitional Clustering Algorithms”, *Proc. Intl. Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)*, Berlin, Germany, Sept. 2006, pp. 115–126.
- [10] P. Drineas, R. Kannan, A. Frieze, S. Vempala, and V. Vinay, Clustering of large graphs via the singular value decomposition, *Machine Learning* (56), pp. 9–33, 2004.
- [11] P. Hansen and B. Jaumard, “Minimum Sum of Diameters Clustering”, *J. Classification*, Vol. 4, 1987, pp. 215–226.
- [12] E. L. Lloyd, R. Liu, M. V. Marathe, R. Ramanathan and S. S. Ravi, “Algorithmic Aspects of Topology Control Problems for Ad Hoc Networks”, *Mobile Networks and Applications (MONET)*, Vol. 10, Issue 1-2, Feb.–Apr. 2005, pp. 19–34.
- [13] C. H. Papadimitriou, *Computational Complexity*, Addison Wesley, Reading, MA, 1994.
- [14] R. Sarnath, “Dynamic Digraph Connectivity Has- tens Minimum-Sum-of-Diameters Clustering”, *SIAM J. Discrete Mathematics*, Vol. 18, No. 2, Oct. 2004, pp. 272–286.
- [15] K. Wagstaff, *Intelligent Clustering with Instance-Level Constraints*, Ph.D. Dissertation, Department of Computer Science, Cornell University, Ithaca, NY, 2002.
- [16] 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.
- [17] 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.

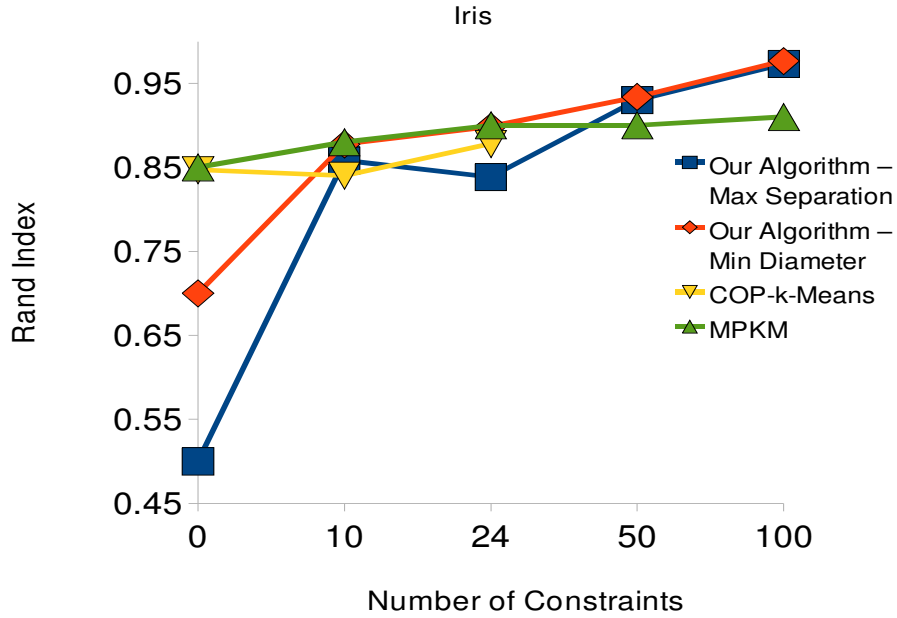


Figure 3: Iris: For a variety of algorithms the RAND index averaged over fifty constraint sets against the size of the constraint set.

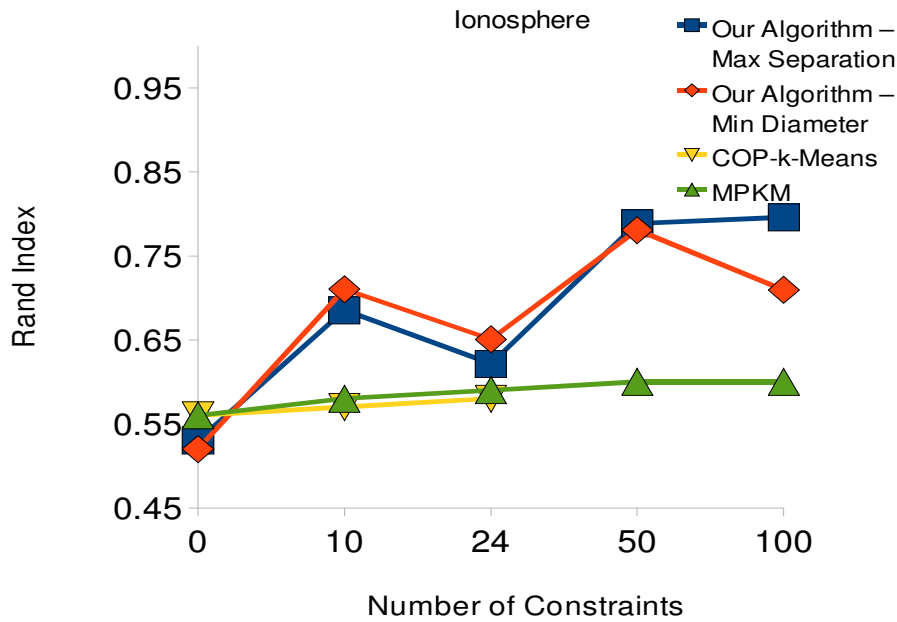


Figure 4: Ionosphere: For a variety of algorithms the RAND index averaged over fifty constraint sets against the size of the constraint set.

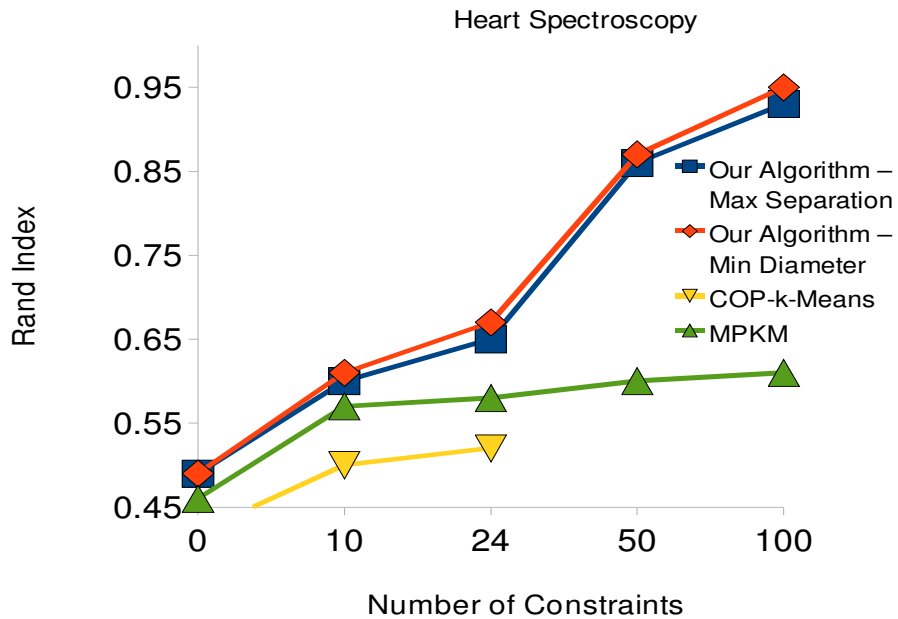


Figure 5: Heart Spectroscopy: For a variety of algorithms the RAND index averaged over fifty constraint sets against the size of the constraint set.

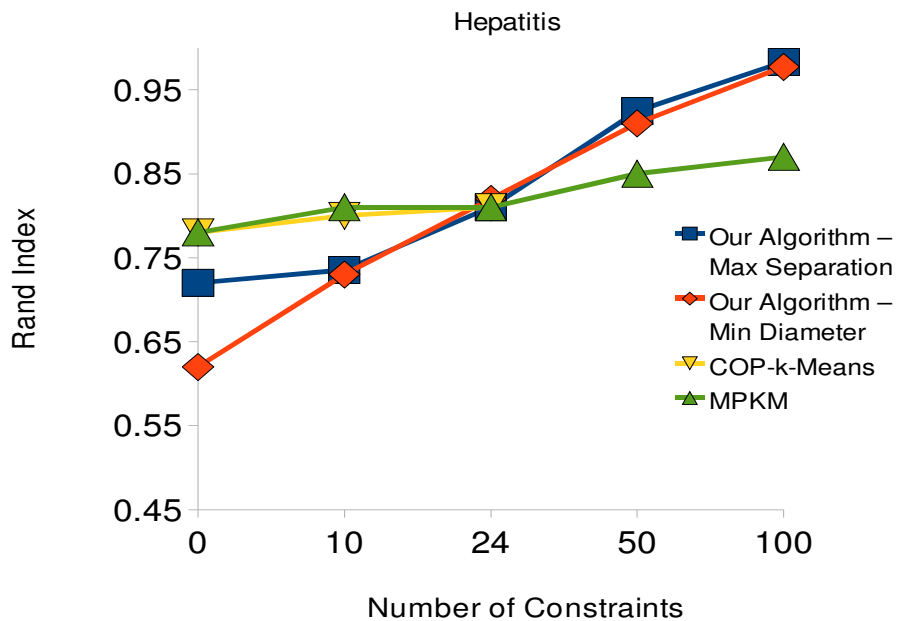


Figure 6: Hepatitis: For a variety of algorithms the RAND index averaged over fifty constraint sets against the size of the constraint set.