A Framework for Minimal Clustering Modification via Constraint Programming

Chia-Tung Kuo
University of California, Davis
tomkuo@ucdavis.edu

S. S. Ravi
University at Albany
sravi@albany.edu

Thi-Bich-Hanh Dao
University of Orleans
thi-bich-hanh.dao@univ-orleans.fr

Christel Vrain
University of Orleans
Christel.Vrain@univ-orleans.fr

Ian Davidson
University of California, Davis
davidson@cs.ucdavis.edu

Abstract

Consider the situation where your favorite clustering algorithm applied to a data set returns a good clustering but there are a few undesirable properties. One adhoc way to fix this is to re-run the clustering algorithm and hope to find a better variation. Instead, we propose to not run the algorithm again but **minimally** modify the existing clustering to remove the undesirable properties. We formulate the minimal clustering modification problem where we are given an initial clustering produced from *any* algorithm. The clustering is then modified to: i) remove the undesirable properties and ii) be minimally different to the given clustering. We show the underlying feasibility sub-problem can be intractable and demonstrate the flexibility of our constraint programming formulation. We empirically validate its usefulness through experiments on social network and medical imaging data sets.

Introduction

Consider the situation where you wish to cluster your ego-network (those people you have a direct friendship link to) into k groups and invite each group to a separate dinner party. For each person you know their interests, location, gender and age. After applying your favorite clustering algorithm you have k very cohesive clusters except perhaps the range of ages for some cluster is too large or one cluster has too many females compared to males. Heuristically trying to move points from one cluster to another until a satisfactory result is found is not a viable approach as we show the intractability of re-clustering data to reduce cluster diameter (see Theorems (1) and (2)). Simply removing data points to get desirable clusters undermines the intended use of clustering in the first place. For example we can't just leave out some friends from the dinner parties. A more principled approach is to add constraints and reapply the clustering algorithm with the hope that the resultant clustering is similar to the previous yet free of the undesirable properties. However, this approach has several issues: 1) most constrained clustering algorithms (Basu, Davidson, and Wagstaff 2008) deal with simple instance level constraints, such as MUST-LINK and CANNOT-LINK, which cannot be used to balance cardinality of males/females and 2) there is no guarantee the resultant clustering found will be similar to the original.

An alternative we explore in this paper is to start with the initial clustering and minimally modify it whilst removing the undesirable properties. Though in this work the initial clustering is given by an algorithm, in practice it may be generated from an existing solution to a clustering problem or a set partition induced in any manner.

We can view this proposed work as being related to but quite different to constrained clustering. Constrained clustering allows domain experts to inject human guidance into clustering *a priori* before the clustering algorithm begins. This work instead allows providing guidance *a posteriori* after the clustering is found. This has the advantage of allowing feedback to be injected for **any** clustering algorithm. In short, we provide a principled way to generate a new minimally modified clustering whilst retaining most of the original solution.

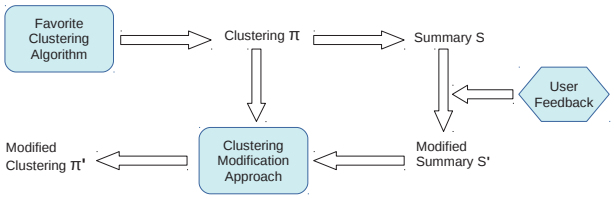


Figure 1: A schematic workflow for our clustering modification setting. The initial clustering Π can be produced in any manner.

Our approach is as follows and shown in Figure 1. We start with an initial clustering, Π, the output of any clustering algorithm or any other process that produces a set partition; this initial clustering has a corresponding clustering summary of properties S. Note this summary need not be with respect to the features used in the clustering algorithm. For example, in our experimental section we cluster Facebook data based only on the friend-network topology and summarize it based on user profile information. The user then modifies S, leading to a modified summary S'. Our approach then looks for a Π' that satisfies the modified summary S' but is

also a minimal modification of Π. The general problem is:

Problem 1. ***Minimal Clustering Modification (MCM).***
Input: Initial clustering Π of k blocks and its k-part summary S. The user modifies parts of S to obtain a user-desired summary S'.
Output: A modified clustering Π' of k blocks similar to Π but also satisfying summary S', formally:

$$\begin{array}{ll} \underset{\Pi'}{\textit{minimize}} & d(\Pi, \Pi') \\ \textit{subject to} & \Pi' \textit{ satisfies } S' \end{array} \quad (1)$$

where $d(\Pi, \Pi')$ is a distance measure between 2 clusterings.

The contribution of this paper is summarized as follows.

- We introduce the minimal clustering modification (MCM) problem and a discrete optimization problem formulation.
- We show intractability results for an underlying feasibility sub-problem.
- We formulate the MCM problem using constraint programming as the modeling platform.
- We describe how to encode different feedback in our formulation and provide a complexity analysis.
- Our empirical experiments show our method scales approximately **linearly** with respect to the number of instances to re-cluster (see Figure 8).

We next discuss related work and then types of human feedback of interest to a user. Complexity results and their implications are presented afterwards. We then formulate our problem under a constraint programming model and then demonstrate the usefulness of our approach on real data sets and discuss the results after which we conclude.

Related Work

We discuss related work below and address how our current problem is related yet different from it.

Constrained Clustering allows the injection of guidance **a priori** and is dependent on the chosen clustering algorithm (Basu, Davidson, and Wagstaff 2008) with guidance suitable for semi-supervised clustering setting where the guidance is given from labels. In contrast, our proposed work is algorithm independent, applied after the algorithm converges and allows high level feedback from a domain expert.

Alternative Clustering (Qi and Davidson 2009; Dang and Bailey 2010) addresses the problem of finding a clustering that is **different** from the given one yet whose quality is comparable to the original one. Our setting is different in that the user finds the given clustering acceptable yet only wants it to be minimally modified to satisfy some additional criteria as opposed to targeting a completely distinctive one.

Constraint Programming (CP) is a declarative paradigm for discrete optimization (Freuder 1997; Barták 2001). Some prior work introduced declarative framework such as the SAT problem in modeling constraints in clustering (Gilpin and Davidson 2011; Davidson, Ravi, and Shamis 2010). More recent work formulated clustering and constrained variations as a CP framework and proposed several efficient propagators to detect invalid partial solutions early in the search (Dao, Duong, and Vrain 2013; 2015).

Types of Human Feedback

Below we list some examples of common summaries of intra-cluster and inter-cluster properties which provide feedback that can be modified. We present how to encode most of these in Table 1 but only experiment with those in the intra-cluster level feedback category due to space restrictions.

- **Intra-Cluster Level Summaries for Feedback**:
  - Shrink diameter: e.g. reduce the diameter of cluster i with respect to feature `age` to 10 years.
  - Balance categorical feature values: e.g. make the number of females and males equal in cluster i.
  - Upper/lower bounds on feature values per cluster: e.g. every cluster should contain at least 10% female and no more than 90% males.
- **Inter-Cluster Level Summaries for Feedback**:
  - Widen/shrink the distance between two clusters based on a feature: e.g. the distance between any two instances in cluster i and cluster j should be at least 5 years with respect to age.

These modifications could all be encoded as constraints in our CP model we introduce below and there are possibly many other modifications a user may desire. CP is particularly suited in this context owing to its flexibility in allowing multiple criteria optimization, complex constraints and auxiliary variables. As mentioned earlier in this work our main focus will be on intra-cluster level feedback, however we do list the encodings of other feedback as well in later sections to show CP's flexibility.

Complexity of Reclustering Under Diameter Reduction

In this section we establish intractability results on the general re-clustering problem under one particular modification: namely, cluster diameters. Such results support our choice of CP as a modeling platform as one cannot expect to find a solution efficiently with any particular algorithm.

A general statement of the **diameter-based reclustering problem** is as follows.

Given: A set $P = \{p_1, p_2, \ldots, p_n\}$ of n objects, an integer k $(1 \leq k \leq n)$, a k-clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ of P, a list of $\ell \geq 1$ attributes, with respect to which the diameter of each cluster is computed, and numbers δ_i, $1 \leq i \leq \ell$.

Requirement: Is there another clustering $\mathcal{C}_1$ of P, also with k clusters, such that the maximum diameter of any cluster in $\mathcal{C}_1$ along attribute i is at most δ_i, $1 \leq i \leq \ell$? If so, find one such clustering.

The remainder of this section presents complexity results for two versions of this reclustering problem. The first version shows the problem is NP-complete even if the diameter needs to be reduced along just two dimensions. The second version shows when the number of dimensions along which the diameter needs to be reduced is not a constant, the problem is NP-complete even for just three clusters.

Diameter reduction along two dimensions. Suppose the goal of reclustering is to reduce the diameters along $\ell = 2$

dimensions; that is, along the two chosen dimensions, the maximum diameter of any cluster must be at most δ_1 and δ_2 respectively. The following theorem shows that the reclustering problem is computationally intractable.

Theorem (1)**.** *The reclustering problem where the maximum diameter must be reduced along two dimensions is NP-complete.*

Diameter reduction along many dimensions. When the number of dimensions ℓ along which the diameter must be reduced is large, we can show that the reclustering problem is NP-complete even when the number of clusters is just three. This result is shown below.

Theorem (2)**.** *Suppose the number of dimensions along which the maximum diameter must be reduced is ℓ. Let δ_i denote the bound on the diameter along dimension i, $1 \leq i \leq \ell$. The reclustering problem is NP-complete for any $k \geq 3$.*

See Appendix[1] for proofs of the two theorems.

Constraint Programming Formulation

In this section we describe how to encode the MCM problem (Problem 1) as a CP model through auxiliary variables and constraints, It is followed by a model complexity analysis in terms of numbers of variables, constraints needed and typical run time in our experiments. We provide enough details to reproduce our results and our code is made available[2].

Variables and Constants Used in Our Model

Clustering. We represent a k-way clustering Π (and Π') as a list of n cluster indices in $[1, \ldots, k]$, one for each data point. The difference between two clusterings Π and Π', $d(\Pi, \Pi')$, is measured by the number of positions in which the two lists differ; formally $d(\Pi, \Pi') = \sum_{i=1}^{n} \mathbb{I}[\Pi[i] \neq \Pi'[i]]$ where $\Pi[i]$ is the i-th entry in Π and $\mathbb{I}[\cdot]$ is the indicator function. Such a choice eliminates the ambiguity of permutations of the cluster indices. It is important to note that since the given clustering Π is desirable, we wish to minimally change its composition. Having an objective focused on some measure of **clustering quality** difference can result in a fundamentally different clustering.

Clustering summary. We focus our discussion here on numerical features but will list some encoding of feedback on binary features in latter section. The feature-wise **diameters** for the clusters are represented as a $k \times f$ matrix $\mathcal{D}$ where $\mathcal{D}[i, j]$ records the diameter for the i-th cluster with respect to the j-th feature. Analogously $\mathcal{D}'$ is defined as the diameters the user desires on the modified clustering Π'.

For convenience we pre-compute **all pairwise distances** with respect to each feature and denote them by a 3-dimensional array D where $D[t, i, j]$ is the distance between the i-th instance and the j-th instance with respect to the t-th feature. Finally let X denote the $n \times f$ feature matrix.

[1]https://sites.google.com/site/chiatungkuo/publication

[2]https://sites.google.com/site/chiatungkuo/publication

Encoding Our Model: Objective and Summary

Here we describe how to encode the objective and the constraints using auxiliary variables and simple constructs found in most popular CP platforms. Note that we chose to implement our model in the CP language Numberjack (Hebrard, O'Mahony, and O'Sullivan 2010) due to its simple interface and its use of state-of-the-art integer linear program (ILP) solvers. ILP solvers such as Gurobi (Inc. 2015) (used in our experiments) can easily exploit multi-core architectures. Other more sophisticated and extensible CP languages exist (e.g. (Gecode Team 2006)).

Objective. The number of instances moved from the initial clustering can be encoded straightforwardly with auxiliary variables, z, recording where Π and Π' disagree.

$$\forall i = 1, \ldots, n,\ z[i] = \mathbb{I}[\Pi'[i] \neq \Pi[i]] \tag{2}$$

It then follows the objective is

$$\texttt{minimize} \sum_{i=1}^{n} z[i] \tag{3}$$

Diameter summary. In order to succinctly measure the modified clusterings (Π') diameters, we define a *cluster membership* matrix as a $k \times n$ binary matrix C, where each row indicates the membership of the corresponding cluster. This is enforced by the following constraints.

$$\forall c = 1, \ldots, k,\ \forall i = 1, \ldots, n,\ C[c, i] = \mathbb{I}[\Pi'[i] = c] \tag{4}$$

Now we describe how we encode constraints to enforce the desired diameters.

First attempt. A straightforward encoding follows the definition of diameter: we require each pair of instances in the same cluster to have (feature-wise) distance smaller than or equal to the specified diameter, shown as follows.

$$\begin{aligned} &\forall c = 1, \ldots, k,\ \forall t = 1, \ldots, f, \\ &\max_{i,j=1,\ldots,n} \{C[c, i]C[c, j]D[t, i, j]\} \leq \mathcal{D}'[c, t] \end{aligned} \tag{5}$$

Note $C[c, i] = 1$ if the i-th instance is in cluster c. Thus $C[c, i]C[c, j] = 1$ if and only if the i-th instance and the j-th instance are both in cluster c. One significant drawback of this encoding, however, is that `max` is taken over n^2 variables. This makes the encoding and solving very inefficient (both memory and CPU) when n is large.

More efficient encoding. Here we describe our encoding of the same diameter summary where each constraint involves at most n variables. The crucial observation is that these diameters are defined feature-wise, as opposed to the classical notion where a single diameter encompasses all dimensions. Accordingly, instead of requiring each pair in the same cluster to obey this cluster's diameter, we require just the difference between the maximum and the minimum value of the feature in a cluster to obey such diameter. Specifically we pre-compute the feature-wise minimums and maximums of our data as follows.

$$\begin{aligned} M_l[t] &\leftarrow \min_{i=1,\ldots,n} \{X[i, t]\}\ \forall t = 1, \ldots, f \\ M_u[t] &\leftarrow \max_{i=1,\ldots,n} \{X[i, t]\}\ \forall t = 1, \ldots, f \end{aligned} \tag{6}$$

Then the last sets of constraints in the complete optimization encoding in Figure 2 (last 4 lines) enforce the diameter constraints. Note $L[c,t]$ and $H[c,t]$ are the lowest and highest values in cluster c for feature t respectively. The multiplication by $C[c,i]$ either keeps a value (if the i-th instance is in cluster c) or zeros it out (otherwise). The additional subtractions and additions of the pre-computed minimums (M_l) and maximums (M_u) of features ensure that maximizing (or minimizing) over extra 0's does not affect the results.

$$\begin{aligned}
&\underset{z,C,L,H}{\texttt{minimize}} \quad \sum_{i=1}^{n} z[i] \\
&\texttt{subject to} \\
&\quad \forall c = 1,\ldots,k,\ \forall i = 1,\ldots,n,\ C[c,i] = \mathbb{I}[\Pi'[i] = c] \\
&\quad \forall i = 1,\ldots,n,\ z[i] = \mathbb{I}[\Pi'[i] \neq \Pi[i]] \\
&\quad \forall c = 1,\ldots,k,\ \forall t = 1,\ldots,f, \\
&\qquad L[c,t] = \underset{i=1,\ldots,n}{\texttt{min}} \{C[c,i](X[i,t] - M_u[t])\} + M_u[t] \\
&\qquad H[c,t] = \underset{i=1,\ldots,n}{\texttt{max}} \{C[c,i](X[i,t] - M_l[t])\} + M_l[t] \\
&\qquad H[c,t] - L[c,t] \leq \mathcal{D}'[c,t]
\end{aligned}$$

Figure 2: CP optimization encoding where the user provides a set of desired (feature-wise) diameters $\mathcal{D}'$ as feedback.

Encoding other feedback/summary. As mentioned earlier CP is flexible in encoding other types of feedback as constraints. A list of common constraints conforming to feedback introduced earlier and their encodings are presented in Table 1. It is also worth mentioning that these constraints need not apply to all features or clusters. The ranges of the indices for the constraints in Table 1 (i.e. c, t, etc) could be determined at the user's discretion.

Complexity and Implementation Issues

Here we present the complexity of our model and show that the numbers of variables and constraints are linear in the number of instances n, the number of clusters k and the number of features f. In addition the variables also have rather small domains. This is important since a CP solver, in the worst case, might need to search through all possible combinations of variable assignments (typically much fewer though). Therefore in general smaller domain sizes lead to shorter run times. Consequently our experiments on the Facebook data ($n = 4039, k = 4, f = 2$) and fMRI data ($n = 1730, k = 4, f = 2$) **each take less than 2 minutes to finish on a 12-core workstation.**

Figure 3 provides a tabulation of the numbers of variables, their domain sizes and the associated constraints used to encode our model in Figure 2. Note the domain size r for variables L and H arises from the discretization of continuous values and the choice of r typically involves a tradeoff between precision and model complexity as is the case in all other discretization problems.

Vars.	Number	Domain size
Π'	n	k
z	n	2
C	nk	2
L	nk	r
H	nk	r

(a) Numbers and domain sizes of variables used in our model.

Constraints	Number	#. Vars. involved
Bind z	n	2
Bind C	nk	2
Bind L	kf	$n+1$
Bind H	kf	$n+1$
$H - L \leq \mathcal{D}'$	kf	2

(b) Number of constraints and the numbers of variables involved in each constraint.

Figure 3: Complexity of encoding Figure 2 model.

Empirical Evaluation

We experiment with two real world data sets (social network and medical imaging) to explore the benefits of using modification and also UCI data sets to explore scalability issues.

Experiment #1: Social Network Modification

We apply our proposed approach to a network data set: **Facebook-egonets** from Stanford SNAP Data sets (Leskovec and Krevl 2014). This data set consists of 4039 Facebook users where the friendships among them are known and for each person a list of binarized categorical features such as gender[3]. We run (normalized) spectral clustering algorithm (von Luxburg 2007) on this graph to find an initial 4-way clustering; a hard clustering is then obtained from the best of 10 runs of k-means on the spectral embedding. Note that spectral clustering *only utilizes the friendship graph topology, but not the node features*. The clustering found is of very low cut cost but a summary of the initial clustering shown in Figure 4(a) shows a widely differing composition compared to the population averages.

Our aim now is to minimally modify the original clustering to correct for gender and language imbalance by constraining them to be close to the population averages. We choose the upper and lower bounds according to the averages in the initial summary and set bounds $[0.36, 0.4]$ for gender and $[0.13, 0.15]$ for language so that these two features are "balanced" across clusters. We find a minimum of 69 nodes need to be moved between clusters and the summary for the resulting modified clustering is presented in Figure 4(b).

An important comparison is against another clustering satisfying the same summary of "balanced" features but without enforcing the objective of "minimal modification". This simulates re-running the clustering algorithm from the beginning and enforcing the balancing constraints. One often found clustering simply puts most instances in one cluster, resulting in 4015 instances in cluster 1 and 8 instances

[3]The feature values are anonymized so, for example, it is unknown if 1 is male or female.

Constraints	Encoding
Diameters	$\forall c = 1, \ldots, k,\ \forall t = 1, \ldots, f,\ H[c,t] - L[c,t] \leq \mathcal{D}'[c,t]$
Splits	$\forall c_1, c_2 = 1, \ldots, k$ where $c_1 \neq c_2, \forall t = 1, \ldots, f, \min_{i,j=1,\ldots,n} \{C[c_1,i]C[c_2,j]D[t,i,j]\} \geq \mathcal{S}'[c_1,c_2,t]$
Bound cluster size	$l_c \leq \sum_{i=1}^{n} C[c,i] \leq u_c$
Keep a cluster	$\sum_i C[c,i] \geq 1$ for cluster c to be kept
Merge clusters	$\sum_i C[c,i] = 0$ for cluster c to be merged (i.e. effectively empty a cluster)
Balance binary features	$\forall c = 1, \ldots, k,\ \forall t = 1, \ldots, f,\ p_l \sum_{i=1}^{n} C[c,i] \leq \sum_{i=1}^{n} C[c,i]X[i,t] \leq p_u \sum_{i=1}^{n} C[c,i]$

Table 1: Common constraints and their encodings. Note we assume H and L are properly encoded auxiliary variables as in Figure 2. $\mathcal{S}'$ is a user-desired $k \times k \times f$ matrix of the desired splits between clusters; p_l and p_u are the user-desired lower and upper bounds on the counts of 1's (True) in binary features for each cluster (see our Experiment #1).

in each of clusters 2, 3 and 4, leading to a total of 1074 swaps across clusters from the initial clustering. We also report the normalized cut costs (the objective of normalized spectral clustering) on the three clusterings: *initial*, *satisfying summary+minimally modified*, *satisfying just summary*. Their cut costs are, respectively, 0.97, 1.34 and 3.04. Note a constraint on the cut cost could be additionally included if it was desired to keep it below a bound.

	Initial clustering				
	C1	C2	C3	C4	Population
Gender	1096 (0.37)	37 (0.54)	169 (0.49)	230 (0.36)	1532 (0.38)
Language	402 (0.13)	5 (0.07)	64 (0.19)	78 (0.12)	549 (0.14)
Size	2988	69	345	637	4039

(a) Initial clustering summary

	Modified clustering				
	C1	C2	C3	C4	Population
Gender	1124 (0.37)	22 (0.39)	117 (0.40)	269 (0.40)	1532 (0.38)
Language	408 (0.14)	7 (0.13)	43 (0.15)	91 (0.13)	549 (0.14)
Size	3014	56	293	676	4039

(b) Modified clustering summary

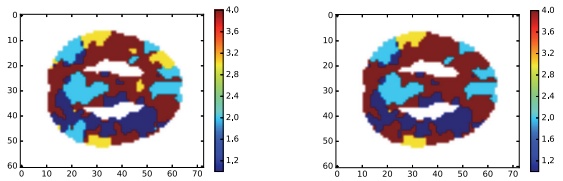
Figure 4: Summaries for the initial and modified clusterings. "Gender" and "Language" record the numbers of instances that have this feature being 1; the numbers in the brackets give the ratios. Size is the number of instances in the cluster.

Experiment #2: Spatial Region Modification

In this experiment, we apply our approach to a fMRI brain imaging data which allows exploring modification based on spatial information. We work on one particular slice in the mid-brain so that each scan consists of 2D snapshots over time and each slice has a total of 1730 voxels/nodes whose blood oxidation levels are measured at an interval of 3ms over 200+ time steps.

We start off by constructing a 1730 node completely connected graph where the edge weights are the absolute value of Pearson correlations between the voxels measured over time. Such correlation measure has been widely used in the neuroscience community (Friston 2011). As before we create an initial clustering by running normalized spectral clustering on this graph and then selecting the best result from 10 runs of k-means on the spectral embedding. The initial clustering is shown in Figure 5(a).

Often in practice we like clusters to represent compact regions in the brain; however this initial clustering is generated based on correlations and does not take into account **any spatial coordinates** in the brain. Accordingly we look



(a) Initial (b) Modified

Figure 5: Initial and modified clusterings on the fMRI scan. The modification asks for the yellow cluster's x-diameter and the cyan cluster's y-diameter to be shrunk. The color-coded cluster numbers match the numbering in Figure 6.

for a modified clustering with tighter diameters in x-y spatial coordinates, specifically, x-diameter ≤ 15 for cluster 3 and y-diameter ≤ 30 for cluster 2 (yellow and cyan in Figure 5) while keeping the diameters for the other clusters. Our CP model returns a new clustering that moves a total of 109 voxels where 36 voxels were moved from cluster 2 to cluster 4 and 63 voxels were moved from cluster 3 to cluster 4. We present the summaries of the initial and the modified clusterings in Figure 6. Note this is a **globally** optimal solution.

Cluster index	C1 (Blue)	C2 (Cyan)	C3 (Yellow)	C4 (Red)
x-diameter	46	49	48	52
y-diameter	37	41	45	44
size	390	369	152	819

(a) Initial summary

Cluster index	C1 (Blue)	C2 (Cyan)	C3 (Yellow)	C4 (Red)
x-diameter	46	49	**15**	52
y-diameter	37	**26**	45	44
size	390	323	89	928

(b) Modified summary

Figure 6: Summaries for the initial and modified clusterings for the fMRI scan experiment. Size is the number of voxels in the cluster.

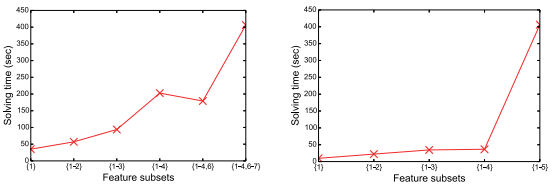
Experiment #3: Run time analysis

Earlier we presented the modeling complexity. Here we empirically observe the run time and evaluate the quality of the modified clustering in terms of the algorithm's objective, compared to the initial clustering. We test our formulation on two data sets from UCI repository, **Yeast** and **Pima**

(Lichman 2013). **Yeast** consists of 1484 data instances with 6 numerical features (No. 1-4, 7 and 8). **Pima** has 768 data instances with 8 numerical features. Note that we only consider enforcing diameters on the numerical features as diameters are not suited for categorical features.

Run time v.s. number of features. Here we test the run times of our model in Figure 2 when the diameter constraints are enforced on different subsets of features. For each data set, we compute an initial clustering Π using k-means ($k = 4$ for Yeast and $k = 3$ for Pima) on the feature-normalized data (again we pick the best out of 10 runs). We then compute the summary of Π as a matrix of diameters, $\mathcal{D}$, as shown earlier. Now we tighten the first dimension of the first cluster with modified diameters $\mathcal{D}'$ defined as $\mathcal{D}'[1,1] = 0.8\mathcal{D}[1,1]$ and $\mathcal{D}'[i,j] = \mathcal{D}[i,j]$ everywhere else. The results are shown in Figure 7 where we gradually enforce the diameter constraints on more and more dimensions. Figure 7(b) only includes feature subsets up to 5 features (No. 1-5) as the model becomes infeasible at this point. It is worth noting the jump in run time when the model is infeasible due to that the solver potentially needs to search exhaustively through all branches to conclude infeasibility.

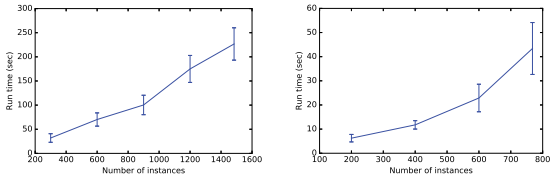
Run time v.s. number of instances. Here we report the run times when the feature set is fixed but we vary the number of instances, n. For each n we randomly sample a subset of instances and create an initial clustering similarly as above. Then we compute its summary and shrink the first diameter of the first cluster as above and solve for it. We draw 5 such random samples for each n and report the averages and standard deviations in Figure 8. The results suggest the run time scales roughly linearly in the number of instances when we shrink one diameter in one cluster. Further results (not shown) indicate a similar linear relationship when more than 1 diameter is shrunk but as our intractability results indicate the feasibility problem then becomes intractable and the solver does not always find a solution.



(a) Yeast (b) Pima

Figure 7: Run time v.s. feature subsets. The x-axis specifies the subsets of features for which diameter summaries are enforced (i.e. changing the index set of t in Figure 2).

Quality Here we report the sums of within-cluster (Euclidean) distances (SWCD) for the initial and modified clusterings from above. Figure 9 records our objectives (i.e. # of points moved) and SWCD for corresponding experiments in Figure 7. Notice that for solutions with the same optimal objective value, their SWCDs can still be different since one could possibly move an instance to different other clusters. It should be noted, however, that since SWCD is calculated based on all features at once whereas our diameter summary is feature-wise, in general our formulation need not necessarily lead to small changes in SWCD. But since the initial clustering was generated using k-means and our features were normalized, we believe a small change in SWCD is expected in this case.



(a) Yeast (b) Pima

Figure 8: Run time v.s. number of instances. The error bars are the standard deviations out of 5 random samples.

Dim. Red.	Initial	{1}	{1-2}	{1-3}	{1-4}	{1-4,6}	{1-4,6-7}
Objective	0	3	3	3	3	3	16
SWCD	9277	9328	9328	9328	9324	9342	9473

(a) Yeast

Dim. Red.	Initial	{1}	{1-2}	{1-3}	{1-4}	{1-5}
Objective	0	3	3	3	6	infeasible
SWCD	5829	5849	5849	5849	5843	N/A

(b) Pima

Figure 9: Quality of the initial and the minimally modified clusterings for varying number of dimensions reduced.

Conclusions and Future Work

In this paper we introduce the problem of minimal clustering modification and formulate it under the CP model. Our approach provides a high level summary of a given initial clustering and then allows the user to modify the summary. We then find the globally minimal number of changes needed to generate a new clustering that satisfies the modified summary. We provide intractability results showing some particular problem cases are difficult and empirically evaluate our approach on several benchmark and real data sets to demonstrate its usefulness in applications and assess its efficiency.

We briefly discuss several possible extensions here. Our work could easily be extended to consider a resource budget model where moving a particular instance is associated with a particular cost with the aim to minimize the total cost. This is useful in settings where there are real costs associated with modifying a clustering. We mentioned, but did not explore, inter-cluster level modifications such as splitting a cluster and merging two clusters. These types of modifications involve changing the numbers of clusters between the initial and final clusterings. We aim to explore such modifications in future work.

Acknowledgment

We gratefully acknowledge support of this work from NSF IIS:1422218, Functional Network Discovery for Brain Connectivity.

References

Barták, R. 2001. Theory and practice of constraint propagation. In *Proceedings of the 3rd Workshop on Constraint Programming in Decision and Control*, volume 50.

Basu, S.; Davidson, I.; and Wagstaff, K. 2008. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press.

Dang, X. H., and Bailey, J. 2010. Generation of alternative clusterings using the cami approach. In *SDM*, volume 10, 118–129. SIAM.

Dao, T.-B.-H.; Duong, K.-C.; and Vrain, C. 2013. A Declarative Framework for Constrained Clustering. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 419–434.

Dao, T.-B.-H.; Duong, K.-C.; and Vrain, C. 2015. Constrained clustering by constraint programming. *Artificial Intelligence*.

Davidson, I.; Ravi, S.; and Shamis, L. 2010. A sat-based framework for efficient constrained clustering. In *SDM*, 94–105. SIAM.

Freuder, E. C. 1997. In pursuit of the holy grail. *Constraints* 2(1):57–61.

Friston, K. J. 2011. Functional and effective connectivity: a review. *Brain connectivity* 1(1):13–36.

Gecode Team. 2006. Gecode: Generic constraint development environment. Available from `http://www.gecode.org`.

Gilpin, S., and Davidson, I. 2011. Incorporating sat solvers into hierarchical clustering algorithms: an efficient and flexible approach. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1136–1144. ACM.

Hebrard, E.; O'Mahony, E.; and O'Sullivan, B. 2010. Constraint Programming and Combinatorial Optimisation in Numberjack. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010*, 181–185.

Inc., G. O. 2015. Gurobi optimizer reference manual.

Leskovec, J., and Krevl, A. 2014. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.

Lichman, M. 2013. UCI machine learning repository.

Qi, Z., and Davidson, I. 2009. A principled and flexible framework for finding alternative clusterings. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 717–726. ACM.

von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17(4):395–416.