ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-453

A Framework for Actionable Clustering Using Constraint Programming

Thi-Bich-Hanh Dao[1] and **Christel Vrain**[1] and **Khanh-Chuong Duong**[1] and **Ian Davidson**[2]

Abstract. Consider if you wish to cluster your ego network in Facebook so as to find several useful groups each of which you can invite to a different dinner party. You may require that each cluster must contain equal number of males and females, that the width of a cluster in terms of age is at most 10 and that each person in a cluster should have at least r other people with the same hobby. These are examples of cardinality, geometric and density requirements/constraints respectfully that can make the clustering useful for a given purpose. However existing formulations of constrained clustering were not designed to handle these constraints since they typically deal with low-level, instance-level constraints. We formulate a constraint programming (CP) languages formulation of clustering with these cluster-level styles of constraints which we call *actionable clustering*. Experimental results show the potential uses of this work to make clustering more actionable. We also show that these constraints can be used to improve the accuracy of semi-supervised clustering.

1 Introduction

Most clustering is unsupervised with a recent movement to adding constraints, an area generally known as constrained clustering [2]. Previous work is most suitable for the *semi-supervised* setting where a few instances are labeled and the instance-level `must-link` and `cannot-link` constraints can be generated from them [2]. The data is then clustered under small numbers of these constraints with the number of clusters equaling the number of different labels. Performance is typically measured in terms of prediction: how well the clustering found matches the ground truth clustering induced by the labels. However, in many domains experts can provide complex constraints that are not generated from a ground truth, rather they capture what makes the clustering useful (or not useful) in the domain. To emphasize this focus we term this *actionable* clustering.

Consider our motivating example of clustering an ego network so that each cluster can be invited to a separate dinner party. A clustering algorithm may find a useful grouping that results in a successful party but is unlikely to unless we somehow encode what is required. Further uses can be modeled in the semi-supervised clustering setting if we know something about the underlying label set. Suppose we have labels that correspond to gender. If we know that in our data set there are twice as many males as females we can constrain the cluster sizes accordingly. Similarly if we know the males are typically closer in age than the females we can create the requirement to enforce that the diameter of one cluster is smaller than the other.

Existing instance level constraints cannot be used to specify this type of guidance. Consider the constraint that the number of males and females in each cluster should be approximately equal. Since instance level constraints are specified *a priori* to the algorithm beginning execution they typically cannot constrain any property that dynamically changes during the algorithm's execution such as the cluster composition. Constraining cluster level properties has probably not been well studied as it is challenging to do so in procedural languages and mathematical programming but can be elegantly performed in constraint programming (CP) languages which we use in this work.

The use of CP means that any clustering algorithm implementation will not scale to data sets larger than 10,000 points as these methods find the global optimum. Though this will prevent our work's usage on big data problems, recently work has shown [15, 14] many interesting real world problems are small data problems involving small but difficult to understand data sets. One characteristic of these problems is the need to find the very best clustering (the global optima) which effectively precludes scalability due to the intractability of the underlying optimization problem. Our work can be considered as allowing actionable clustering for these small data problems but our work can still be applied to thousands of data points as we have done in our experiments.

Our contributions are as follows:

- We look at new types of constraints beyond instance level constraints that must be calculated dynamically allowing for a new style of cluster level constraints.
- We introduce cardinality, density and geometric styles of cluster level constraints and show they can be easily specified in CP formulations.
- We experimentally explore the uses of these constraints to enforce guidance to ensure clusters are actionable by showing they can find alternative clusterings.
- We show experimentally that these constraints can help in the semi-supervised setting in a number of ways. They can be used in addition to instance level constraints or can be used in lieu of instance level constraints to improve clustering accuracy.

Our paper is laid out as follows. In the next section we outline the actionable clustering problem and then related work. Next follows the type of constraints we can build actionable clusters on including geometric, density and cardinality constraints. Such constraints are challenging to encode in procedural or mathematical programming formulations so instead we use a CP framework that allows to encode a variety of objective functions. The benefits of modeling both

[1] Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, F-45067, Orléans, France, email: thi-bich-hanh.dao@univ-orleans.fr, khanh-chuong.duong@univ-orleans.fr, christel.vrain@univ-orleans.fr
[2] Department of Computer Science, University of California, Davis, email: davidson@cs.ucdavis.edu

the constraints and objective functions in CP include ease of solving but also allow specifying a framework which others can build upon by adding in more objectives and constraints. Next we describe our experiments including a variety of data set types after which we conclude.

2 The Actionable Clustering Problem

Consider the classic clustering problem. We are given a data set $\mathcal{X}$ where each instance $x \in \mathcal{X}$ is described by a vector of features f. Typical objective functions include minimizing the vector quantization error (k-means), minimizing graph cuts (spectral methods) if the data is represented as a graph and a similarity measure is used, and optimizing cluster properties such as minimizing the maximum cluster diameter. In our work we present the novel extension that each instance is further described by a set of properties from which the definitions of what is actionable/interesting is given. To separate our features and properties we use the notation: x_i^f and x_i^p to represent the features and properties of the i^{th} instance respectfully.

In our formulation the feature vectors $\mathcal{X}^f$ are used to calculate the clustering objective function value and the constraints are enforced on the property vectors $\mathcal{X}^p$. However, there is nothing stopping the same attribute of an instance being in both vectors and used as both a feature and property.

Formally the actionable clustering problem is formulated as:

$$\begin{aligned} & argmin_{\Pi}\ g(\mathcal{X}^f, \Pi) \qquad (1) \\ & s.t.\ \ \phi_c(\mathcal{X}^p, \Pi), \phi_d(\mathcal{X}^p, \Pi), \phi_g(\mathcal{X}^p, \Pi) \\ & where \ \ \Pi \text{ is a partition/clustering,} \\ & g \text{ is an objective function,} \\ & \phi_{c,d,g} \text{ models the cardinality, density and geometric constraints} \end{aligned}$$

The type of constraints we will explore in this paper can be divided into four categories: i) cardinality, ii) density, iii) geometric and iv) complex logical combination of these constraints which we now discuss in turn. They are not the only relevant ones but the most pragmatic in clustering: cardinality constraints are useful for categorical attributes, density constraints for relational information and geometric constraints for real value attributes. It is important to note that these constraints can be applied **simultaneously for multiple different properties** on multiple clusters.

i) Cardinality constraints place a requirement on a count of the elements in a cluster having a property. They may be as simple as each cluster should contain at least one female to more complex variations such as the number of males must be no greater than two times the number of females.

ii) Density constraints relate to a cardinality constraint in that it provides requirements on a count of a property except not for an entire cluster but rather a subset of instances in the cluster. For example, we may require each person have at least 10 people in his/her cluster sharing the same hobby.

iii) Geometric constraints place an upper or lower bound on some geometric property of a cluster or cluster combination. Examples include that the maximum diameter of a cluster with respect to the age property is 10 years. This would prevent clusters containing individuals with a wide range of ages.

iv) Complex logic constraints express logic combinations of constraints, which can be instance-level or cluster-level constraints. For instance, we may require that any cluster having more than 2 professors should have more than 10 PhD students.

Positive vs Negative Requirements. Up to this point we have discussed finding clusters which are actionable since they meet a particular set of requirements. However, it is also likely that a clustering is actionable because it *does not* contain a set of properties. This idea of using negative feedback was first explored in the alternative clustering literature [21, 3, 25, 19]. There the problem was given a good (according to the objective function value) clustering Π which is not actionable (perhaps because it is trivial or inappropriate) find an alternative clustering Π' such that Π' has a good objective function value but Π and Π' are different using some sort of measure such as the Rand index. However, that work is limited in that *how* Π' is different to Π is not controlled. Instead with actionable clustering if the existing clustering has undesirable properties such all females in one cluster, we can explicitly require that females be equally distributed (i.e. $\forall i, j$ $\texttt{CountFemale}(\pi_i) \approx \texttt{CountFemale}(\pi_j)$ where π_i is the i^{th} cluster).

3 Related Work

The style of constraints we explore in this paper to our knowledge has not been explored before. We briefly survey the related areas of: i) instance level constrained clustering, ii) relational clustering and iii) clustering using CP.

Instance level constrained clustering. As mentioned earlier most of this work is applicable to the semi-supervised setting where the `must-link` and `cannot-link` constraints must be given apriori. There have been a large variety of clustering algorithms to encode these constraints such as k-means, hierarchical, expectation maximization (EM) and spectral methods [2]. Our work differs since we explore more complex constraints beyond simple pairwise instance level constraints. Moreover, our framework finds a global optimum while classic clustering algorithms look for local optima.

Relational clustering. The area of relational clustering [20] aims to cluster data represented by both feature and simple pairwise relations such as `brother`, `son-of` and `married`. The technical challenge is to combine both feature and relational information in such a way that both can be used in the objective function of the clustering algorithm. Our work differs since we are looking at more complex guidance beyond relational information but also as we use extra information beyond features as constraints to be enforced not modeled as part of the objective function.

Clustering using CP. To our knowledge little work uses Constraint Programming for clustering. We proposed a CP model for distance-based clustering integrating several optimization criteria and different kinds of constraints [4, 6]. This work integrates only classic constraints and does not contain the complex constraints on properties we consider in this paper. A fundamentally different pattern mining approach based on CP [10, 17] has been generalized to k-pattern set mining [18] and applied to model conceptual clustering, where each cluster is characterized by a set of properties common to all the elements of the cluster.

4 A Quick Primer on CP

There are many benefits to formulate our work in a CP language. Not only does it allow elegant formulation as a constrained optimization problem but it guarantees to find a global optimum, a critical requirement for valued data that are small and collected only once (eg. fRMI data). Furthermore, relaxing the problem so it becomes just a satisfaction problem allows to explore all the possible solutions and even to determine if there exists a feasible one.

A *Constraint Satisfaction Problem* is modeled by (X, Dom, C), where X is a set of variables, each variable $x \in X$ is associated with a domain $Dom(x)$ and C is a set of constraints, each constraint expresses a condition on a subset of X. A solution is a complete assignment of value $v \in Dom(x)$ to each variable $x \in X$ that satisfies all the constraints of C. When an objective function is added, the problem becomes a *Constraint Optimization Problem* and the aim is to find a solution that optimizes the objective. The originality of CP solvers is to alternate two steps: propagation allowing to filter the variable domains and branching. Different strategies can be used to create and to order branches at each branching point. They can be standard search strategies defined by CP solvers or can be specifically developed. Moreover, many kinds of constraints are available: elementary constraints expressing arithmetic or logic relations, or global constraints, as for instance the cardinality constraint, denoted by # and allowing to count a set of objects and to place a constraint on the obtained number. Although equivalent to conjunctions of elementary constraints, global constraints usually benefit from more efficient filtering algorithms. Reified constraints are available, which allow to link a boolean variable to the truth value of a constraint.

5 A CP Formulation of Actionable Clustering

We begin by discussing how to find globally optimal clusterings in CP and then move onto discuss how to encode the new types of constraints we introduce in this work. As is typical in the field, we show how to encode these constraints in a generic language-free manner. To aid in reproducibility of results all code will be made available in the CP solver library Gecode[3].

5.1 Clustering in CP

For modeling clustering we rely on the CP clustering model proposed in our earlier work [6]. Clusters are identified by their index, varying from 1 to K (K denotes the number of clusters). Instances are also identified by their index, ranging from 1 to N. The assignment of instances to clusters is modeled by the integer variables G_i for $i \in [1, N]$, with $Dom(G_i) = [1, K]$ (the set of integers from 1 to K): $G_i = k$ means that the i^{th} instance is put into cluster number k. A complete assignment of the variables G_i therefore defines a clustering. However, several complete assignments can lead to the same composition of the clusters, where the only difference is the index of the clusters. In order to break this kind of symmetry, the CP constraint *precede*$([G_1, .., G_n], [1, .., K])$ is used. This constraint enforces that the instance number 1 is in the cluster number 1, and an instance number i can be in a cluster k if the cluster $k-1$ is not empty and contains an instance j with $j < i$.

The model in [6] integrates several optimization criteria: maximizing the minimal separation between clusters, minimizing the maximal diameter of the clusters, minimizing the within-cluster sum of dissimilarities, or minimizing the within-cluster sum of squares [5]. All these criteria are NP-hard: minimizing the diameter is polynomial for $K = 2$ but NP-hard for $K \geq 3$ [16], maximizing the separation is polynomial [11] but becomes NP-hard with user constraints [8], minimizing the sum of dissimilarities is NP-hard since the weighted max-cut problem, which is NP-complete [12], is a particular instance of this problem with $K = 2$ and the NP-hardness of minimizing the sum of squares is proven in [1]. These criteria can be used in our framework where the distances are computed from $\mathcal{X}^f$.

[3] http://www.gecode.org

We extend this previous work to integrate constraints put on the property $\mathcal{X}^p$. From the semantic point of view these constraints can be divided into four categories: cardinality constraints, density constraints, geometric constraints and complex logic constraints.

5.2 Cardinality Constraints

Cardinality constraints allow to express requirements on the number of instances that satisfy some conditions in each cluster. The condition can be for instance being more than 20 years old and the cardinality constraint can state that each cluster must have more than 30 persons being more than 20 years old. The minimal capacity constraint (also called minimum significant constraint in [13]) is then a special case of a cardinality constraint.

Given a condition, the set C of the instances that satisfy it can be computed and the number of instances of C that are in a cluster k can be captured using the CP cardinality constraint and a variable Y_k:

$$\#\{i \in C \mid G_i = k\} = Y_k \qquad (2)$$

The constraint $\sum_{k=1}^{K} Y_k = |C|$ enforces the link between the variables Y_k. Cardinality constraints are then expressed by arithmetic constraints on Y_k. Let us illustrate this by some examples.

- *Each cluster must have at least* 50 *females of age more than 20.* The set C is the set of instances i such that $female(i) = true$ and $age(i) > 20$. For $k \in [1, K]$, the constraint of Equation (2) is put as well as the constraint $Y_k \geq 50$. The number of new introduced variables is K and the number of constraints is $2K + 1$.
- *In each cluster, the number of teachers must be no less than half the number of students.* Let C_t and C_s be the sets of instances that are teachers and students, respectively. For $k \in [1, K]$, constraints similar to Equation (2) are put with the variables T_k and S_k to capture the number of teachers or students in the cluster k. These variables are linked by the constraint $2T_k \geq S_k$. The number of new variables is $2K$ and the number of constraints is $3K + 1$.

5.3 Density Constraints

Density constraints provide bounds on the occurrence of some properties on a subset of instances in each cluster. For instance, each person being more than 20 years old should have in his/her cluster more than 5 persons sharing the same hobby. Density constraints allow a more general form than the basic ϵ-ball count constraint [8]. To express this constraint, for each instance $i \in [1, N]$ which is eligible (eg. more than 20 years old), the set of neighborhood instances $NI(i)$ (eg. persons having the same hobby) is determined. The number of instances of $NI(i)$ in the same cluster as i can be captured using the variable Z_i and:

$$\#\{j \in NI(i) \mid G_j = G_i\} = Z_i \qquad (3)$$

Arithmetic conditions are then stated on Z_i to express density constraints. Let us take the following examples.

- *In the same cluster, each person should have at least* 5 *persons having the same hobby.* For each instance i, we compute the set $NI(i) = \{j \in [1, n] \mid hobby(i) = hobby(j)\}$. The fact that there must be at least 5 other persons of $NI(i)$ in the same cluster as i means the value of G_i must be taken at least 6 times by the elements in $NI(i)$. Therefore the constraint of Equation (3) is put as well as the constraint $Z_i \geq 6$.

- *Each person of age between 20 and 55 should have at least* 10% *persons with a difference in age less than 5 in the same cluster.* For each instance i such that $20 \leq age(i) \leq 55$, $NI(i)$ contains all the instances j such that $|age(i) - age(j)| \leq 5$. The constraint of Equation (3) is put as well as the constraint $Z_i \geq |NI(i)|/10$.

In these cases, the number of new introduced variables is N and the number of CP constraints is $2N$. Let us notice that the computation of the neighborhoods is done only once before putting CP constraints.

5.4 Geometric Constraints

Geometric constraints allow to set bounds on some geometric properties inside each cluster, or between the clusters.

- *Any two clusters must be separated by at least* 10 *on the age property.* Therefore any pair of instances i, j having $|age(i) - age(j)| < 10$ must be in the same cluster. For these pairs of instances the constraint $G_i = G_j$ is put.
- *Each cluster must have at most* 40 *difference on the age property.* That means any pair of instances i, j having $|age(i) - age(j)| > 40$ must be in different clusters. For them, $G_i \neq G_j$ is put.

In these cases, the number of CP constraints needed is at most quadratic compared to the number of instances. However, we have defined in our earlier work [6] the global constraints *split* and *diameter* that use a distance d. The constraint *split*(G, S, d) enforces that the minimal split (separation) between cluster is captured by a variable S, and *diameter*(G, D, d) enforces that the maximal diameter of the clusters is captured by a variable D. These global constraints are more efficient than expressing split or diameter constraints by must-link or cannot-link constraints. They can be used in our setting with d defined by $d(i, j) = |age(i) - age(j)|$, the first case is then expressed by two constraints *split*(G, S, d) and $S \geq 10$.

A geometric constraint can also place a bound on the sum of all the values on some properties inside each cluster, or a condition on the ranges of some properties of the clusters. For instance, age ranges of the clusters should or should not overlap, or the total sum of age in each cluster must not exceed some value.

- *The average age in each cluster must not exceed 50.* To express this constraint, for each instance i and each cluster k, we introduce a boolean variable $B_{ik} \in \{0, 1\}$ (0:$false$ and 1:*true*). A reified constraint [4] $B_{ik} \leftrightarrow (G_i = k)$ is put, ie. B_{ik} represents whether or not instance i is in the cluster k. For each $k \in [1, K]$, the sum of age S_k and the cardinality C_k are linked by:

$$\begin{aligned} \textstyle\sum_{i \in [1,N]} age(i) B_{ik} &= S_k \\ \#\{i \in [1, N] \mid G_i = k\} &= C_k \end{aligned}$$

  The bound is therefore expressed by: $S_k \leq 50C_k$. In this case, $N \times K$ boolean variables (B_{ik}), K float point value variables (S_k) and K integer value variable (C_k) are introduced. This case is expressed by $3K$ CP constraints.
- *Constraints on the property ranges of the clusters.* We consider constraints that state conditions on the ranges of the clusters on a property p. To capture the range of a cluster, for each cluster k, we introduce the variables Min_k and Max_k that represent the minimal and the maximal values on the property p of the elements in the cluster k. Let m_p be the maximal value of the property p for all the instances. The minimal and maximal values are linked by the following constraints:

$$\begin{aligned} Min_k &= \min_{i \in [1,n]} (p(i) B_{ik} + m_p(1 - B_{ik})) \\ Max_k &= \max_{i \in [1,n]} (p(i) B_{ik}) \end{aligned}$$

  The constraint that the ranges on p of the clusters should not overlap can be expressed by putting, for any two clusters k, k',

$$(Min_k > Max_{k'}) \vee (Min_{k'} > Max_k)$$

  A constraint stating that the range of a cluster k must be included in the range of another cluster k' can be expressed by:

$$Min_k \geq Min_{k'} \text{ and } Max_k \leq Max_{k'}$$

  This requires $N \times K$ boolean variables and $2K$ float point value variables. The number of constraints is linear on K.

[4] A reified constraint on a constraint c, stated by $B \leftrightarrow c$, links the truth value of a constraint c to a boolean variable B: B is 1 if the constraint c is satisfied, 0 if c cannot be satisfied, or $B \in \{0, 1\}$ if the satisfaction of c has not yet been determined.

5.5 Complex Logic Constraints

Complex logic constraints can be used to enhance the expressivity power of formulating knowledge. This can be done in CP using reified and Boolean constraints as shown by the following examples.

- *Two instances* $3, 9$ *are in the same cluster if the instances* $11, 15$ *are in different clusters.* Two Boolean variables B_1, B_2 are introduced with the constraints: $B_1 \leftrightarrow (G_{11} \neq G_{15})$, $B_2 \leftrightarrow (G_3 = G_9)$ and $B_1 \leq B_2$.
- *Any cluster having more than 5 professors must have at least 10 PhD students.* For each $k \in [1, K]$, let P_k and S_k be the variables that capture the number of professors and students in the cluster k, using cardinality constraints such as in Equation (2). Two Boolean variables BP_k and BS_k are introduced and linked by $BP_k \leftrightarrow (P_k \geq 5)$, $BS_k \leftrightarrow (S_k \geq 10)$, and $BP_k \leq BS_k$.

6 Experiments

We begin by outlining the underlying questions our experiments attempt to address and then move onto the experimental methodology and results. It is important to realize that many clustering papers will have experiments showing how well their algorithm performed by performance on some objective function value. These types of experiments are not applicable in our setting since the CP framework finds the global optima. Similarly experiments to verify our method finds a clustering that satisfies the constraints are not required since once the requirements are formulated by constraints, the CP framework is guaranteed to converge if a feasible clustering exists. Our experiments attempt to address the following questions:

1. In the semi-supervised clustering setting, can the quality of the solution be improved with constraints beyond instance-level constraints?
2. Can our constraints be used to find actionable clusterings that are alternatives to an existing clustering?
3. How do our constraints increase the run-time of the underlying clustering algorithm?

For the first question, we show that a cardinality constraint added to instance-level constraints yields better solutions than just using instance-level constraints (see Table 2). This is so as we can effectively control the resultant cluster sizes to better match the label populations, something instance level constraints cannot readily do. We also show for the setting where different labels are not well separated that a geometric constraint can increase performance where as previously no instance level constraints have been able to. Finally, we also show that cardinality constraints enforcing cluster size by themselves can improve semi-supervised clustering accuracy (see Figures 2 and 3). For the second question we show on a visually understandable digit data set that our constraints are capable of finding quite different clusterings (Figures 4, 5 and 6) with even simple constraints. Importantly this explores the novel direction of *guided* alternative clustering. For the last question we show that examples of cardinality, diameter geometric and logical constraints do increase run times compared to using no constraints (Tables 4, 6 and 7). This increase is only fractional for most of cases but can be more important for instance with a density constraint.

The framework is implemented using the CP solver library Gecode 4.3.3. The objective function in the experiments is to minimize the maximal diameter of the clusters. As it was done in [4, 6], the instances are reordered by the Furthest Point First algorithm [16], so that instances that are far from the others have small index. Concerning the search strategy, at each branching point, the solver chooses an unassigned variable G_i with the smallest remaining domain. All values $k \in Dom(G_i)$ are examined and the distance between instance i and cluster k is computed: it is equal to the smallest distance $d(i,j)$ such that G_j is instantiated to k, or 0 if cluster k is empty. The value k of the closest cluster to i is chosen and two branches are created with $G_i = k$ and $G_i \neq k$. All experiments are performed on a 3.4GHz Intel Core i5 processor with 8Gb of RAM under Ubuntu 14.04. The relevant code to reproduce the results is available on www.cp4clustering.com.

6.1 Improving Semi-Supervised Clustering Results

In the semi-supervised learning setting typically labels on a subset of instances are used to generate instance-level constraints such as must-link or cannot-link constraints. Here we first explore if the labels can be better exploited by inferring more complex constraints on the clusters. We illustrate this point by considering diverse UCI datasets, which vary on the number of instances and on the number of classes and that will be used in various conditions. They are described in Table 1. In these experiments all the objects of the datasets are considered and the number K of clusters is set to the true number of classes for each dataset. Performance is typically measured in terms of prediction: how well the found clustering matches the ground truth clustering. To measure the accuracy of a clustering P compared to the ground truth clustering P^*, we use the Rand Index [23] which is defined by $RI = (a+b)/(a+b+c+d)$, where a and b are the numbers of pairs of instances for which P and P^* are in agreement (a, or b, is the numbers of pairs of instances that are in the same class, or respectively in different classes, in both P and P^*), c and d are the numbers of pairs of instances for which P and P^* disagree (same class in P but different classes in P^* and vice versa). This index varies from 0 to 1 and the better the partitions are in agreement, the closer RI to 1.

Dataset	# objects	# classes
Iris	150	3
Wine	178	3
Glass	214	7
Ionosphere	351	2
Breast cancer	569	2
Synthetic control	600	6
Vehicle	846	4
Yeast	1484	10
Multiple feature morphology	2000	10
Image segmentation	2100	7

Table 1. Properties of datasets

6.1.1 *Adding A Single Cardinality Constraint to Instance Level Constraints*

Instance-level constraints can decrease the quality of the found clustering compared to the ground truth clustering, as was reported in our earlier work [9]. We consider the datasets Iris, Wine, Breast Cancer and Ionosphere in order to match the experiments in [9]. All the attributes are considered as features ($\mathcal{X}^f$) in order to compute pairwise Euclidean distance and they are also considered as properties ($\mathcal{X}^p$). We generate a number of randomly created (from labels) instance-level constraints as is standard [2]: two instances are randomly taken, whether their labels are the same or not a must-link or a cannot-link constraint is stated, and this is repeated until required the number of instance-level constraints is reached. On those same taken instances we generate a minimum cardinality constraint for all clusters as follows: let the whole dataset be of N instances, the labeled sample be of e instances, and the smallest cluster size observed on the labeled sample be m, then for the whole dataset, the smallest cluster size is set to $0.9\frac{m}{e}N$. This simulates a user guess at how big the smallest cluster should be based on the less frequent occurring label.

We compare two cases: first, clustering with a set of randomly generated instance-level constraints and second, clustering with the very same instance-level constraints and a minimum cardinality constraint as described above. We perform 1000 experiments and report the average Rand Index of all the experiments in Table 2. We can observe that for Iris, Wine and Breast Cancer, adding a cluster cardinality constraint helps to get better clusterings on average. For these datasets, we analyze the distribution over all experiments of the accuracy decrease or increase over not using any constraints. Figure 1 shows that adding instance-level constraints decreases the quality in a large number of cases which agrees with [9]. On the other hand, the improvement is more stable when using a minimum cardinality constraint along with instance-level constraints. This is most like because enforcing a cardinality constraint prevents skewed cluster sizes which can yield poor performance.

#	Iris		Wine		Breast Cancer	
0	0.8737	0.8737	0.6859	0.6859	0.5509	0.5509
10	0.8735	**0.8743**	0.6844	**0.6899**	0.5547	**0.7110**
20	0.8768	**0.8782**	0.6857	**0.6959**	0.5577	**0.6757**
30	0.8811	**0.8825**	0.6885	**0.7031**	0.5602	**0.6419**

Table 2. The average Rand Index vs number of instance-level constraints over 1000 runs for 2 cases: just instance-level constraints (left) and instance-level constraints with a minimal cluster cardinality constraint (right).

A Challenging Data Set. We observe from Table 3 what was earlier reported [9] for Ionosphere, instance level constraints do not improve results significantly. In that earlier paper of ours it was shown that the geometry of the data is quite different to the labels which means that many labels of different types are inter-mixed close together, that is there is not naturally a cluster for each class. Not surprisingly, adding a minimum cardinality constraint does not help either, as the clusterings found for this dataset with instance level constraints have quite balanced cluster sizes. Adding a minimum cardinality constraint in this case does not change therefore the result. We therefore experiment with a third case, where we force separate clusters as follows. From the subset of the labeled instances, we compute the approximate minimal split S between clusters, and infer that the clusters should be separated by at least $\frac{2}{3}S$. This enforces a cluster separation that would normally not be found with instance level constraints. This results in Table 3, which shows that this constraint yields better clustering.

#	Ionosphere		
0	0.4988	0.4988	0.4988
10	0.4992	0.4992	**0.5988**
20	0.4997	0.4997	**0.5584**
30	0.5002	0.5002	**0.5364**

Table 3. The average Rand Index over 1000 runs for 3 cases: (1) instance-level constraints, (2) instance-level constraints with a minimal cluster cardinality constraint, (3) only geometric minimal split constraint.

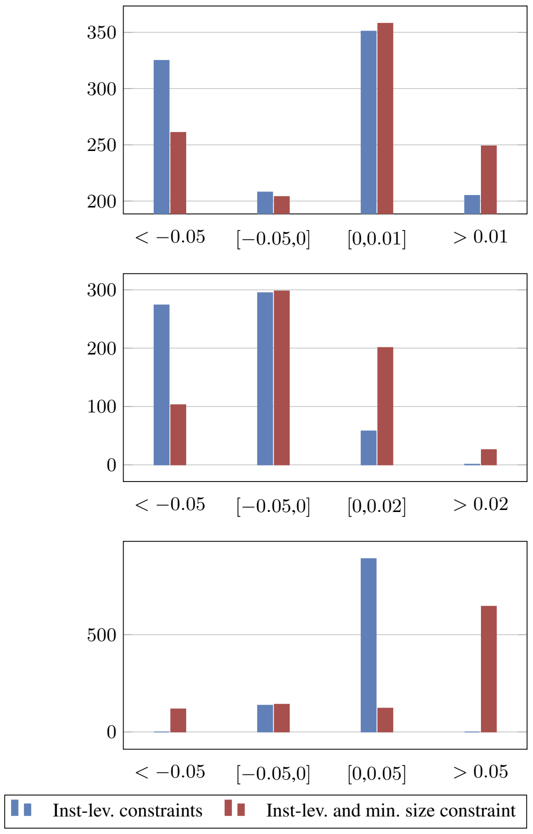

Figure 1. For the experiments reported in Table 2, the frequency distribution (y-axis) over the 1000 experiments by the amount of increase or decrease in accuracy over not using any constraints (x-axis). Ordering of figures are for Iris, Wine and Breast Cancer each one with 30 instance-level constraints. As we can see instance-level + cardinality constraints (red-bar) produces more increases and less decreases in accuracy.

6.1.2 *Adding Multiple Cardinality Constraints*

Here we consider the novel use of our work for semi-supervised clustering. Rather than using the labeled data to generate must-link and cannot-link constraints, we instead use the labels to provide upper and lower bound estimates on the cluster sizes. The datasets Glass, Breast cancer, Vehicle and Yeast, which have larger numbers of objects and of clusters are considered. We experiment four cases: (1) no constraint, (2) a constraint on minimal cluster size α is added, (3) a constraint on maximal cluster size β is added, and (4) a constraint on minimal cluster size α and a constraint on maximal cluster size β are added. The thresholds α and β are set depending on the known labels of the datasets: they are respectively 9 and 90 for Glass, 150 and 400 for Breast cancer, 150 and 250 for Vehicle and 10 and 500 for Yeast. The Rand Index is measured for each case and the results are presented in Figure 2. We can clearly observe that with a size constraint the Rand Index is improved compared to the unconstrained case. This behavior is the same for a small value of K (Breast cancer with $K = 2$) or with a large value of K (Yeast with $K = 10$). The improvement is very large for Breast cancer and convincingly shows that if the cluster sizes can be effectively estimated based on labels they can yield improved results over using no constraints.

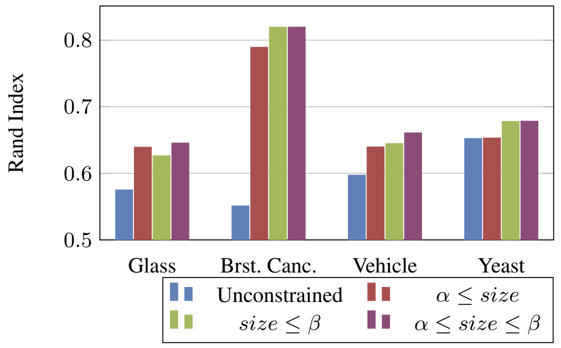

Figure 2. Rand Index in different cases: without constraint and with minimal and/or maximal size constraints

A related question is of course how do instance level and cardinality constraints interact. To explore this question we use a wide variety of data set properties. Figure 3 presents the obtained Rand Index for the datasets Breast cancer, Synthetic control, Multiple feature and Image segmentation with four other cases. In the first case, there is no user constraint. In the second case, 20 instance-level constraints are randomly generated and added. In the third case, the constraints requiring that the cluster size must be between α and β are added, where α and β are respectively 150 and 400 for Breast cancer, 50 and 150 for Synthetic control, 50 and 350 for Multiple feature and 200 and 400 for Image segmentation. In the fourth case both cluster size constraints and 20 instance-level constraints are added. For the second and the fourth cases, we make 100 runs and report the average Rand Index of all the runs. We can observe different behaviors with cluster size constraints here. While for Synthetic control, the constraints slightly decrease the Rand Index, for Multiple Feature, they bring slight improvement. The most significant improvement is observed for Image Segmentation. This dataset is composed by 2100 objects of 7 classes with 300 objects per class. In the unconstrained case, the clustering found has the Rand Index 0.226314 and is very unbalanced, with two clusters of 1 object and with a large cluster of 1972 objects. The situation is not improved with 20 random must-link or cannot-link constraints, the clusterings found are always unbalanced. Adding cluster size constraints such that the clusters must have their size between 200 and 400, the clustering found has the Rand Index grow to nearly 0.80. On Breast cancer we can see that with only size constraints, the Rand Index is over 0.80, while the same constraint together with 20 instance-level constraints gives the Rand Index only about 0.66.

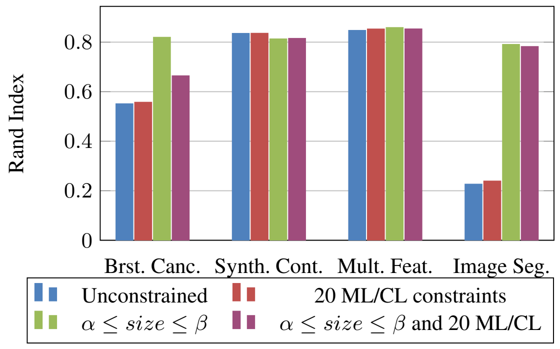

Figure 3. Rand Index in different cases: unconstrained, with instance-level constraints and/or minimal and maximal size constraints

6.2 Guided Alternative Clustering

In the area of alternative clustering one tries to find an equally good alternative to a given clustering [21, 3, 25, 19]. However, it is somewhat unrefined in that no guidance can be given to specify how the clustering is to differ from the given clustering. To address this issue we consider the UCI Pen Digit dataset where each instance corresponds to a single digit and has 16 attributes, which represent the 8 x, y positions of the pen as the digit is being written. All the 16 attributes are considered as features ($\mathcal{X}^f$) in order to compute pairwise Euclidean distances and are also considered as properties ($\mathcal{X}^p$). We use 1000 random instances of the dataset. We aim to find alternative ways that people write digits and we consider the simplest case where the number of clusters $k = 2$. This effectively forms a dichotomy of the two ways people in the data set write their digits.

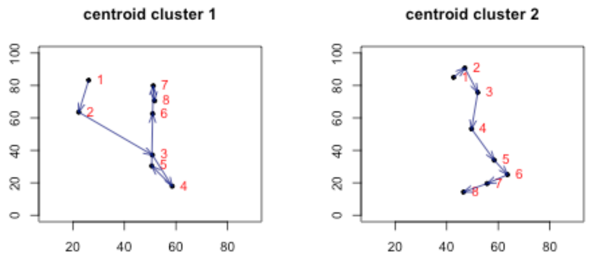

Figure 4. The centroids of the clustering found without any constraints. Time=1.16s, maximal cluster diameter=263.58. Arrows indicate pen movement.

For each cluster, the centroid is computed, which is considered as the representative of the cluster and can be easily visualized to show the underlying digit prototype the cluster represents. In the case of minimizing the maximal cluster diameter and without any constraints, the centroids of the found clusters, which represent two types of writing, are represented in Figure 4. The clustering found is the **global optimum** of the clustering algorithm objective function. However the centroids are not really meaningful unless of saying that in one way people write from up to down then up again, and in

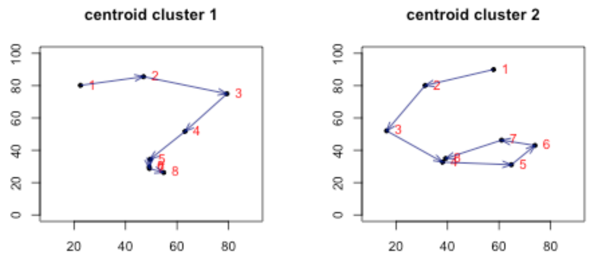

Figure 5. The centroids of the clustering found with a diameter constraint on the horizontal value of the third time step. Time=0.02s, maximal cluster diameter=291.50. Arrows indicate pen movement.

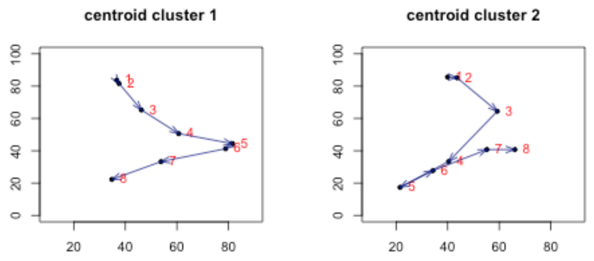

Figure 6. The centroids of the clustering found with a diameter constraint on the horizontal value of the fifth time step. Time=0.02s, maximal cluster diameter=269.68. Arrows indicate pen movement.

the other way only from up to down. Instead we wish to find an actionable alternative by adding a diameter constraint on the horizontal position of the pen at the 3^{rd} time step. This effectively means that all digits in the same cluster must have a similar horizontal pen location at the third time step. We obtain a different clustering whose centroids are shown in Figure 5 but whose quality, in term of the objective function, is comparable to the first clustering found in Figure 4. These centroids have the 3^{rd} positions respectively on the right and on the left. The centroids can give an interpretation such that in one way people write from left to right and in the other way like a spiral from right to left and in both ways from up to down. By adding a diameter constraint on the horizontal position of the pen at the 5^{th} time step, we obtain another very different clustering whose centroids are shown in Figure 6. Again the centroids have the 5^{th} position either on the right or on the left, and the quality of this clustering is comparable to the initial clustering in Figure 4. This is an example of guided alternative clustering which unlike ours and others earlier work [7, 3] did not find an arbitrary alternative clustering, rather we find one with specific properties. Adding constraints can deteriorate the quality in term of the objective function, however the flexibility of our framework allows to control the gap between the constrained case and the unconstrained case by means of constraints. For instance let the maximal diameter of the clusters in the unconstrained case be D_{opt}, one can require an actionable alternative clustering with both a diameter constraint on the horizontal position of the pen at the 3^{rd} time step and another constraint stating that the maximal diameter of the clusters does not exceed $1.2D_{opt}$.

6.3 Computational Effect of Constraints

To address the computational effect of constraints, we report times taken by different cases in Subsection 6.1. The total runtimes (stating constraints and search) are presented in Table 4, where +1800 means the solver did not complete the search after 30 minutes. Cluster size constraints can give large variations in runtime. One explanation is that the efficiency of a CP framework depends on the power of constraint propagation. For cardinality constraint filtering the domains of all variables to arc-consistency [22] is NP-hard. However, when setting an upper bound and a lower bound on the count variables, efficient filtering algorithms have been developed [24], which help pruning the search space.

Dataset	N	K	case	time (s)
Glass	214	7	unconstrained	0.02
			$9 \leq size \leq 90$	0.62
Breast cancer	569	2	unconstrained	0.30
			$150 \leq size \leq 400$	0.70
Vehicle	846	4	unconstrained	0.62
			$150 \leq size \leq 250$	5.57
Yeast	1484	10	unconstrained	3.15
			$10 \leq size \leq 500$	11.56
Synthetic Control	600	6	unconstrained	0.49
			$50 \leq size$	+1800
			$size \leq 150$	2.58
			$50 \leq size \leq 150$	3.02
Multiple Feature	2000	10	unconstrained	10.30
			$50 \leq size$	76.80
			$size \leq 350$	+1800
			$50 \leq size \leq 350$	71.04
Image Segmentation	2100	7	unconstrained	3.29
			$200 \leq size$	86.85
			$size \leq 400$	+1800
			$200 \leq size \leq 400$	92.00

Table 4. Total runtime in seconds for (un)constrained cases with cardinality requirements.

For other kinds of constraints we consider the census dataset available at UCI[5]. This dataset has 48,842 instances, each one is described by 14 attributes with 6 continuous and 8 symbolic. We choose 5 continuous attributes (age, capital-gain, capital-loss, hours-per-week and fnlwgt) as features ($\mathcal{X}^f$) to compute distances on. All of the 14 attributes are used as properties. We generate 5 samples each one of 1000 instances and for each sample we conduct experiments with 5 use-cases described in Table 5. In each use-case, the number of clusters is set to 2 and to 3. Tables 6 and 7 give average runtime across five samples for the use cases with $K = 2$ and $K = 3$ respectively.

1	*No constraints.*
2	*Cardinality constraint.* A cardinality constraint is added, which requires that in each cluster, the ratio of $\#female_c/\#male_c$ for the cluster c is between a half and twice the ratio of females and males in the sample.
3	*Density constraint.* A constraint is added, stating each person of age between 20 and 50 must have at least 10% of people with the same work occupation in the same cluster.
4	*Diameter Geometric constraint.* A constraint is added, which states that the difference in age in each cluster must not exceed $2(\max(age) - \min(age))/3$.
5	*Complex logic constraint.* A constraint is added, which states that a cluster having more than 20 persons younger than 20 should have more than 30 persons older than 45, and each cluster has at least 100 persons.

Table 5. The five use cases for testing the computation time effect of constraints.

5 https://archive.ics.uci.edu/ml/datasets/Adult

Sample	UC1	UC2	UC3	UC4	UC5
1	0.42	0.41	2.51	0.42	0.49
2	0.32	0.69	1.71	0.32	0.35
3	0.44	0.54	2.94	0.41	0.34
4	0.44	0.31	0.71	0.32	0.34
5	0.36	0.48	2.16	0.32	0.32

Table 6. The runtime (seconds) for use cases (see Table 5) across five samples, for $K = 2$.

Sample	UC1	UC2	UC3	UC4	UC5	UC3+4
1	0.32	1.93	+1800	0.40	0.36	4.14
2	0.44	2.79	6.82	0.44	0.45	2.72
3	0.50	2.23	+1800	0.48	0.40	+1800
4	0.44	0.33	+1800	0.86	0.34	1.19
5	0.33	2.30	+1800	0.36	0.32	2.91

Table 7. The runtime (seconds) for use cases (see Table 5) across five samples, for $K = 3$. Note how using UC3 and UC4 together mitigates the increases of just using UC3.

We can see from Table 6 that the run-time to find the best solution in the unconstrained setting is under 0.5 second (use case 1) for all the 5 samples. Use cases 2, 4 and 5 take comparable run-time. Use case 3, which is expressed by a large number of CP cardinality constraints, is the most difficult among all the use cases. This trend is confirmed with $K = 3$ (Table 7), for the solver does not complete the search after the timeout of 30 min for 4 of the 5 samples.

One explanation for this variety of run times is that some constraints when added help the solver to prune the search tree at the top levels which has a large effect. Some other constraints, for instance, the cardinality constraint, however are useful in pruning the search tree only in more deeper levels towards the leaf nodes reducing down the benefits of pruning. On the other hand, constraints such as the diameter geometric constraint are useful in general. We have combined the diameter geometric constraint of use case 4 with the constraint of use case 3. The run-times of the combination reported in the last column of Table 7 have almost dropped for most of the samples.

7 Conclusion

Clustering is ubiquitously used in AI as it can add structure to collections of images, documents and even songs. A recent progression has been the addition of instance level constraints to clustering. These constraints though useful are severely limited in the information they can encode. In particular they cannot constrain any dynamic property of the clustering and hence cannot be used to enforce complex constraints such as those on cardinality (have equal number of males and females) or density (each person in the cluster should have other q persons with the same hobby). In this work we introduce three new styles of complex constraints, geometric, cardinality and density as well as logical combinations of them. We show how CP is a natural vehicle to encode such constraints and has the added benefit of finding the global optima. Although this requirement precludes scaling our work to huge data sets in many settings finding the global optima is desirable and our method works with thousands but not tens of thousands of instances. We showed that our new constraints can improve semi-supervised clustering accuracy when added to instance level constraints, find guided alternative clusterings and finally does not significantly increase run time except for density constraints.

REFERENCES

[1] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat, 'NP-hardness of Euclidean Sum-of-squares Clustering', *Machine Learning*, **75**(2), 245–248, (2009).

[2] Sugato Basu, Ian Davidson, and Kiri Wagstaff, *Constrained clustering: Advances in algorithms, theory, and applications*, CRC Press, 2008.

[3] Xuan Hong Dang and James Bailey, 'A framework to uncover multiple alternative clusterings', *Machine Learning*, **98**(1-2), 7–30, (2015).

[4] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain, 'A Declarative Framework for Constrained Clustering', in *ECMLPKDD*, pp. 419–434, (2013).

[5] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain, 'Constrained minimum sum of squares clustering by constraint programming', in *Principles and Practice of Constraint Programming, CP 2015, Proceedings*, pp. 557–573, (2015).

[6] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain, 'Constrained clustering by constraint programming', *Artificial Intelligence*, (To appear).

[7] Ian Davidson and Zijie Qi, 'Finding alternative clusterings using constraints', in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 773–778. IEEE, (2008).

[8] Ian Davidson and S.S. Ravi, 'The complexity of non-hierarchical clustering with instance and cluster level constraints', *Data Min Knowl Disc*, **14**, 25–61, (2007).

[9] Ian Davidson, Kiri L. Wagstaff, and Sugato Basu, 'Measuring Constraint-Set Utility for Partitional Clustering Algoirthms', in *PKDD*, pp. 115–126, (2006).

[10] Luc De Raedt, Tias Guns, and Siegfried Nijssen, 'Constraint programming for itemset mining', in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 204–212, (2008).

[11] Michel Delattre and Pierre Hansen, 'Bicriterion Cluster Analysis', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (4), 277–291, (1980).

[12] Michael R. Garey and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.

[13] Rong Ge, Martin Ester, Wen Jin, and Ian Davidson, 'Constraint-driven clustering', in *KDD*, pp. 320–329, (2007).

[14] Sean Gilpin and Ian Davidson, 'A flexible ilp formulation for hierarchical clustering', *Artificial Intelligence*, (2015).

[15] Sean Gilpin, Siegried Nijssen, and Ian N Davidson, 'Formalizing hierarchical clustering as integer linear programming.', in *AAAI*, (2013).

[16] T. Gonzalez, 'Clustering to minimize the maximum intercluster distance', *Theoretical Computer Science*, **38**, 293–306, (1985).

[17] Tias Guns, Siegfried Nijssen, and Luc De Raedt, 'Itemset mining: A constraint programming perspective', *Artificial Intelligence*, **175**, 1951–1983, (2011).

[18] Tias Guns, Siegfried Nijssen, and Luc De Raedt, 'k-Pattern set mining under constraints', *IEEE Transactions on Knowledge and Data Engineering*, **25**(2), 402–418, (2013).

[19] Kleanthis-Nikolaos Kontonasios and Tijl De Bie, 'Subjectively interesting alternative clusterings', *Machine Learning*, **98**(1-2), 31–56, (2015).

[20] Bo Long, Zhongfei Mark Zhang, and Philip S Yu, 'A probabilistic framework for relational clustering', in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 470–479. ACM, (2007).

[21] ZiJie Qi and Ian Davidson, 'A principled and flexible framework for finding alternative clusterings', in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 717–726. ACM, (2009).

[22] Claude-Guy Quimper, Alejandro López-Ortiz, Peter van Beek, and Alexander Golynski, 'Improved algorithms for the global cardinality constraint', in *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, Proceedings*, pp. 542–556, (2004).

[23] William M. Rand, 'Objective Criteria for the Evaluation of Clustering Methods', *Journal of the American Statistical Association*, **66**(336), 846–850, (1971).

[24] Jean-Charles Régin, 'Generalized arc consistency for global cardinality constraint', in *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1.*, pp. 209–215, (1996).

[25] Duy Tin Truong and Roberto Battiti, 'A flexible cluster-oriented alternative clustering algorithm for choosing from the pareto front of solutions', *Machine Learning*, **98**(1-2), 57–91, (2015).