A Declarative Framework for Constrained Clustering

Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain

Univ. Orléans, ENSI de Bourges, LIFO, EA 4022, F-45067, Orléans, France
{thi-bich-hanh.dao, khanh-chuong.duong, christel.vrain}@univ-orleans.fr

Abstract. In recent years, clustering has been extended to constrained clustering, so as to integrate knowledge on objects or on clusters, but adding such constraints generally requires to develop new algorithms. We propose a declarative and generic framework, based on Constraint Programming, which enables to design clustering tasks by specifying an optimization criterion and some constraints either on the clusters or on pairs of objects. In our framework, several classical optimization criteria are considered and they can be coupled with different kinds of constraints. Relying on Constraint Programming has two main advantages: the declarativity, which enables to easily add new constraints and the ability to find an optimal solution satisfying all the constraints (when there exists one). On the other hand, computation time depends on the constraints and on their ability to reduce the domain of variables, thus avoiding an exhaustive search.

1 Introduction

Clustering is an important task in Data Mining and many algorithms have been designed for it. It has been extended to semi-supervised clustering, so as to integrate previous knowledge on objects that must be or cannot be in the same cluster, and most algorithms have been adapted to handle such information. Other kinds of constraints could be specified by the user, as for instance the sizes of the clusters or their diameters, but classical frameworks are not designed to integrate different types of knowledge. Yet, in the context of an exploratory process, it would be important to be able to express constraints on the task at hand, tuning the model for getting finer solutions. Constrained clustering aims at integrating constraints in the clustering process, but the algorithms are usually developed for handling one kind of constraints. Developing general solvers with the ability of handling different kinds of constraints is therefore of high importance for Data Mining. We propose a declarative and generic framework, based on Constraint Programming, which enables to design a clustering task by specifying an optimization criterion and some constraints either on the clusters or on pairs of objects.

Relying on Constraint Programming (CP) has two main advantages: the declarativity, which enables to easily add new constraints and the ability to find an optimal solution satisfying all the constraints (when there exists one). Recent progress in CP have made this paradigm more powerful and several work [1–3] have already shown its interest for Data Mining.

In a recent work[4], we have proposed a CP model for constrained clustering, aiming at finding a partition of data that minimizes the maximal diameter of classes. In this paper, we generalize the model with more optimization criteria, namely maximizing the margin between clusters and minimizing the Within-Cluster Sums of Dissimilarities (WCSD). Clustering with WCSD criterion is NP-Hard, since one instance of this problem is the weighted max-cut problem, which is NP-Complete. Recent work [5] has addressed the problem of finding an exact optimum but the size of the database must be quite small, all the more when k is high. We have developed propagation algorithms for the WCSD problem, and experiments show that we are able of finding the optimal solution for small to medium databases. Moreover, adding constraints allows to reduce the computation time.

The main contribution of our paper is a general framework for Constrained Clustering, which integrates different kinds of optimization criteria and finds a global optimum. Moreover, we show that coupling optimization with some types of constraints allows to handle larger databases and can be interesting for the users.

The paper is organized as follows. In Section 2, we give background notions on clustering, constrained clustering and Constraint Programming. Section 3 presents related work. Section 4 is devoted to the model and Section 5 to experiments. A discussion on future work is given in Section 6.

2 Preliminaries

2.1 Clustering

Clustering is the process of grouping data into classes or clusters, so that objects within a cluster have high similarity but are very dissimilar to objects in other clusters. More formally, we consider a database of n objects $\mathcal{O} = \{o_1, \ldots, o_n\}$ and a dissimilarity measure $d(o_i, o_j)$ between two objects o_i and o_j of $\mathcal{O}$. Clustering is often seen as an optimization problem, i.e., finding a partition of the objects which optimizes a given criterion. Optimized criteria may be, among others: (the first four criterion must be minimized whereas the last one must be maximized)

• Within-Cluster Sum of Dissimilarities (WCSD) criterion:

$$E = \sum_{c=1}^{k} \sum_{o_i, o_j \in C_c} d(o_i, o_j)^2$$

• Within-Cluster Sum of Squares (WCSS) criterion, also called the least square criterion (m_c denotes the center of cluster C_c):

$$E = \sum_{c=1}^{k} \sum_{o_i \in C_c} d(m_c, o_i)^2$$

• Absolute-error criterion (r_c denotes a representative object of the cluster C_c):

$$E = \sum_{c=1}^{k} \sum_{o_i \in C_c} d(o_i, r_c)$$

• Diameter-based criterion: $E = max_{c \in [1,k], o_i, o_j \in C_c}(d(o_i, o_j))$. E represents the maximum diameter of the clusters, where the diameter of a cluster is the maximum distance between any two of its objects.

• Margin-based criterion: $E = min_{c<c' \in [1,k], o_i \in C_c, o_j \in C_{c'}}(d(o_i, o_j))$. E is the minimal margin between clusters, where the margin between two clusters $C_c, C_{c'}$ is the minimum value of the distances $d(o_i, o_j)$, with $o_i \in C_c$ and $o_j \in C_{c'}$.

We do not detail here well-known classical clustering algorithms, such as k-means that finds a local optimum of the WCSS criterion, or k-medoids for the absolute-error criterion. The FPF (Furthest Point First) method introduced in [6] is a very efficient method (complexity $O(kn)$) for finding a local optimum of the maximum diameter criterion. Moreover theoretical bounds are given and we show in Section 4 how such bounds can be used to reduce the complexity, when modeling the problem in CP.

Some algorithms do not rely on an optimization algorithm, as for instance DBSCAN [7], based on the notion of density. Parameters are needed to adjust the notion of density. Although our model does not currently allow to simulate the behavior of DBSCAN, the notion of density can be integrated as a constraint on the clustering task.

2.2 Constraint-based Clustering

Most clustering methods rely on an optimization criterion, and because of the inherent complexity search for a local optimum. Several optima may exist, some may be closer to the one expected by the user. In order to better model the task, but also in the hope of reducing the complexity, user-specified constraints are added, leading to Constraint-based Clustering that aims at finding clusters that satisfy user-specified constraints. User constraints can be classified into cluster-level constraints, specifying requirements on the clusters, or instance-level constraints, specifying requirements on pairs of objects.

Most of the attention has been put on instance-level constraints, first introduced in [8]. Commonly, two kinds of constraints are used. A must-link constraint specifies that two objects o_i and o_j have to appear in the same cluster: $\forall c \in [1,k],\ o_i \in C_c \Leftrightarrow o_j \in C_c$. A cannot-link constraint specifies that two objects must not be in the same cluster: $\forall c \in [1,k],\ \neg(o_i \in C_c \wedge o_j \in C_c)$.

Cluster-level constraints impose requirements on the clusters. We give some examples of such constraints that have been integrated to our model.

The minimum capacity constraint requires that each cluster has a number of objects greater than a given threshold α: $\forall c \in [1,k],\ |C_c| \geq \alpha$, whereas the maximum capacity constraint requires each cluster to have a number of objects inferior to a predefined threshold β: $\forall c \in [1,k],\ |C_c| \leq \beta$.

The maximum diameter constraint specifies an upper bound on the diameter of the clusters: $\forall c \in [1,k], \forall o_i, o_j \in C_c,\ d(o_i, o_j) \leq \gamma$ (γ is a given parameter). The minimum margin constraint, also called the δ-constraint in [9], requires the distance between any two points of different clusters to be superior to a given threshold δ: $\forall c \in [1,k], \forall c' \neq c, \forall o_i \in C_c, o_j \in C_{c'}, d(o_i, o_j) \geq \delta$.

The ϵ-constraint introduced in [9] requires for each point o_i to have in its neighborhood of radius ϵ at least another point of the same cluster:

$$\forall c \in [1,k], \forall o_i \in C_c, \exists o_j \in C_c, o_j \neq o_i \ and \ d(o_i, o_j) \leq \epsilon.$$

This constraint tries to capture the notion of density, introduced in DBSCAN. We propose a new density-based constraint, stronger than the ϵ-constraint: it requires that for each point o_i, its neighborhood of radius ϵ contains at least $MinPts$ points belonging to the same cluster as o_i.

In the last ten years, many works have been done to extend classical algorithms for handling must-link and cannot-link constraints, as for instance an extension of COBWEB [8], of k-means [10, 11], hierarchical non supervised clustering [12] or spectral clustering [13, 14], etc. This is achieved either by modifying the dissimilarity measure, or the objective function or the search strategy. However, to the best of our knowledge there is no general solution to extend traditional algorithms to different types of constraints. Our framework relying on Constraint Programming allows to add directly user-specified constraints.

2.3 Constraint Programming

Constraint Programming is a powerful paradigm to solve combinatorial problems, based on Artificial Intelligence or Operational Research methods. A *Constraint Satisfaction Problem (CSP)* is a triple $\langle X, D, C \rangle$ where $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables, $D = \{D_1, D_2, \ldots, D_n\}$ is a set of domains ($x_i \in D_i$), $C = \{C_1, C_2, ..., C_t\}$ is a set of constraints where each constraint C_i expresses a condition on a subset of X. A solution of a CSP is a complete assignment of values from D_i to each variable x_i that satisfies all the constraints of C. A *Constraint Optimization Problem (COP)* is a CSP with an objective function to be optimized. An optimal solution of a COP is a solution of the CSP that optimizes the objective function. In general, solving a CSP is NP-hard. Nevertheless, the methods used by the solvers enable to efficiently solve a large number of real applications. They rely on constraint propagation and search strategies.

Constraint propagation operates on a constraint c and removes all the values that cannot be part of a solution from the domains of the variables of c. A set of propagators is associated to each constraint, they depend on the kind of consistency required for this constraint (e.g. arc consistency removes all the inconsistent values, while bound consistency modifies only the bounds of the domain). Consistency is chosen by the programmer when the constraint is established. Let us notice that a formula or a mathematic relation can be a constraint in CP only if a set of propagators can be defined on it.

In a CP solver, two steps, constraint propagation and branching, are repeated until a solution is found. Constraints are propagated until a stable state, in which the domains of the variables are reduced as much as possible. If the domains of all the variables are reduced to singletons then a solution is found. If the domain of a variable becomes empty, then there exists no solution with the current partial assignment and the solver backtracks. In the other cases, the solver chooses a variable whose domain is not reduced to a singleton and splits its domain into different parts, thus leading to new branches in the search tree. The solver then explores each branch, activating constraint propagation since the domain of a variable has been modified.

The search strategy can be determined by the programmer. When using a depth-first strategy, the solver orders branches, following the order given by the programmer and explores in depth each branch. For an optimization problem, a branch-and-bound strategy can be integrated to depth-first search: each time a solution, i.e. a complete assignment of variables satisfying the constraints, is found, the value of the objective function for this solution is computed and a new constraint is added, expressing that a new solution must be better than this one. This implies that only the first best solution found is returned by the solver. The solver performs a complete search, pruning only branches that cannot lead to solutions and therefore finds an optimal solution. The choice of variables and of values at each branching is very important, since it may drastically reduce the search space and therefore computation time. For more details, see [15].

Example 1. Let us illustrate by the following COP: find an assignment of letters to digits such that $SEND + MOST = MONEY$, which maximizes $MONEY$. This problem can be modeled by a COP with eight variables S, E, N, D, M, O, T, Y, of the domain the set of digits $\{0, \ldots, 9\}$. Constraints for this problem are:

– the digits for S and M are different from 0: $S \neq 0,\ M \neq 0$
– the values of the variables are pairwise different: `alldifferent`(S, E, N, D, M, O, T, Y). Let us notice that instead of using a constraint $\neq$ for each pair of variables, the constraint *alldifferent* on a set of variables is used. This is a global constraint in CP, as the following linear constraint.
– $(1000 \times S + 100 \times E + 10 \times N + D) + (1000 \times M + 100 \times O + 10 \times S + T) = 10000 \times M + 1000 \times O + 100 \times N + 10 \times E + Y$
– maximize$(10000 \times M + 1000 \times O + 100 \times N + 10 \times E + Y)$.

The initial constraint propagation leads to a stable state, with the domains: $D_S = \{9\}$, $D_E = \{2, 3, 4, 5, 6, 7\}$, $D_M = \{1\}$, $D_O = \{0\}$, $D_N = \{3, 4, 5, 6, 7, 8\}$ and $D_D = D_T = D_Y = \{2, 3, 4, 5, 6, 7, 8\}$. Since some domains are not reduced to singletons, branching is then performed. At the end of the search, we get the optimal solution with the assignment $S = 9, E = 7, N = 8, D = 2, M = 1, O = 0, T = 4, Y = 6$, leading to $MONEY = 10876$.

Strategies specifying the way branching is performed are very important. When variables are chosen in the order S, E, N, D, M, O, T, Y and when values are chosen following an increasing order, the search tree is composed of 29 nodes and 7 intermediary solutions (solutions satisfying all the constraints, better than the previous ones found but not optimal). When variables are chosen in the order S, T, Y, N, D, E, M, O, the search tree has only 13 nodes and 2 intermediary solutions.

3 Related work

Recent work [16, 17] has proposed to use Constraint Programming for conceptual clustering. The problem is then formalized as the search of frequent, pairwise non overlapping k-patterns that cover the whole dataset. Several optimization criteria are considered as maximizing the minimal size of the clusters or minimizing the difference between the sizes of classes. These approaches can only be applied to qualitative databases, whereas our approach can handle all kinds of data, as soon as a dissimilarity measure is defined on data. Another approach is based on Integer Linear Programming [18, 19], where a set of candidate clusters must be known beforehand and the model searches for the best clustering among the subset of clusters. It has been experimented in the context of conceptual clustering, based on frequent patterns. This framework is less convenient for clustering in general since finding a good set of candidate clusters is difficult as the number of candidate clusters is exponential in the number of objects. A SAT framework [20] is proposed for constrained clustering, but only for a 2-class problem ($k = 2$). Several kinds of constraints are considered: must-link and cannot-link constraints on instances, constraints on cluster diameters and margins. Based on SAT, the algorithm allows to obtain a global optimum. Our approach is more general, since the number of classes is not limited to 2, and several optimization criteria as well as a larger class of constraints are considered.

Clustering with the presented criteria is NP-Hard, most algorithms are heuristics. For instance, k-means finds a local optimum for the WCSS criterion. There are few exact algorithms for the WCSD and WCSS criteria: they rely on lower bounds, which must be computed in a reasonable time and finding such bounds is a difficult subtask. The best known exact method for both WCSD and the maximum diameter criterion is a repetitive branch-and-bound algorithm [5]. This algorithm is efficient when the number k of groups is small; it solves the problem first with $k + 1$ objects, then with $k + 2$ objects and so on, until all n objects are considered. When solving large problems, smaller problems are solved for quickly calculating good lower bounds. The authors give the size n of the databases that can be handled: $n = 250$ for the minimum diameter criterion, $n = 220$ for the WCSS criterion, and only $n = 50$ with k up to 5 or 6 for the WCSD criterion. For the WCSS criterion, the best known exact method is a recent column generation algorithm [21]. The method solves problems with $n = 2300$, however, the number of objects per group (n/k) must be small, roughly equal to 10, in order to have a reasonable computation time. To the best of our knowledge, there exists no exact algorithm for WCSD or WCSS criterion that integrates user-constraints.

4 A CP framework for constrained clustering

We present a CP model for constrained clustering. As input, we have a dataset of n points and a dissimilarity measure between pairs of points, denoted by $d(i, j)$. Without loss of generality, we suppose that points are indexed and named by their index. The number of clusters is fixed by the user and we aim at finding a partition of data into k clusters, satisfying a set of constraints specified by the user and optimizing a given criterion.

4.1 A constraint-based model

Variables For each cluster $c \in [1,k]$, the point with the smallest index is considered as the representative point of the cluster[1]. An integer variable $I[c]$ is introduced, its value is the index of the representative point of c; the domain of $I[c]$ is therefore the interval $[1,n]$. Assigning a point to a cluster becomes assigning the point to the representative of the cluster. Therefore, for each point $i \in [1,n]$, an integer variable $G[i] \in [1,n]$ is introduced: $G[i]$ is the representative point of the cluster which contains the point i.

Let us for instance suppose that we have 7 points $o_1, \dots, o_7$ and that we have 2 clusters, the first one composed of o_1, o_2, o_4 and the second one composed of the remaining points. The points are denoted by their integer (o_1 is denoted by 1, o_2 by 2 and so on). Then $I[1] = 1$ and $I[2] = 3$ (since 1 is the smallest index among $\{1,2,4\}$ and 3 is the smallest index among $\{3,5,6,7\}$), $G[1] = G[2] = G[4] = 1$ (since 1 is the representative of the first cluster) and $G[3] = G[5] = G[6] = G[7] = 3$ (since 3 is the representative of the second cluster).

A variable is introduced for representing the optimization criterion. It is denoted by D for the maximal diameter, S for the minimal margin and V for the Within-Cluster Sum of Dissimilarities. It is a real-valued variable, since distance are real numbers. The domains of D and S are the interval whose lower (upper) bound is the minimal (maximal, resp.) distance between any two points. The domain of V is upper-bounded by the sum of the distances between all pairs of points. The clustering task is represented by the following constraints.

Constraints on the representation

- Each representative belongs to its cluster: $\forall c \in [1,k], G[I[c]] = I[c]$.
- Each point is assigned to a representative: $\forall i \in [1,n]$, $\bigvee_{c \in [1,k]} (G[i] = I[c])$. This relation can be expressed by a cardinality constraint in CP:
$$\forall i \in [1,n], \quad \#\{c \mid I[c] = G[i]\} = 1.$$
- The representative of a cluster is the point in this cluster with the minimal index; in other words, the index i of a point is greater or equal to the index of its representative given by $G[i]$: $\forall i \in [1,n], \quad G[i] \leq i$.

A set of clusters could be differently represented, depending on the order of clusters. For instance, in the previous example, we could have chosen $I[1] = 3$ and $I[2] = 1$, thus leading to another representation of the same set of clusters. To avoid this symmetry, the following constraints are added:

- Representatives are sorted in increasing order: $\forall c < c' \in [1,k], \ I[c] < I[c']$.
- The representative of the first cluster is the first point: $I[1] = 1$.

Modeling different objective criteria When minimizing the maximal diameter:

- Two points at a distance greater than the maximal diameter must be in different clusters: $\forall i < j \in [1,n], d(i,j) > D \rightarrow (G[i] \neq G[j])$. $(*)$

[1] It allows to have a single representation of a cluster. It must not be confused with the notion of representative in the medoid approach.

– The maximal diameter is minimized: minimize D.

When maximizing the minimal margin between clusters:

– Two points at a distance less than the minimal margin must be in the same cluster: $\forall i < j \in [1, n], d(i, j) < S \to G[i] = G[j]$.
– The minimal margin is maximized: maximize S.

When minimizing the Within-Cluster Sum of Dissimilarities (WCSD):

– $V = \sum_{i,j \in [1,n]} (G[i] == G[j]) d(i, j)^2$.   (**)
– The sum value is minimized: minimize V.

Modeling user-defined constraints All popular user-defined constraints may be straightforwardly integrated:

– Minimal size α of clusters: $\forall c \in [1, k], \#\{i \mid G[i] = I[c]\} \geq \alpha$.
– Maximal size β of clusters: $\forall c \in [1, k], \#\{i \mid G[i] = I[c]\} \leq \beta$.
– A δ-constraint expresses that the margin between two clusters must be at least δ. Therefore, for each $i < j \in [1, n]$ satisfying $d(i, j) < \delta$, we put the constraint: $G[i] = G[j]$.
– A diameter constraint expresses that the diameter of each cluster must be at most γ, therefore for each $i < j \in [1, n]$ such that $d(i, j) > \gamma$, we put the constraint: $G[i] \neq G[j]$.
– A density constraint that we have introduced expresses that each point must have in its neighborhood of radius ϵ, at least $MinPts$ points belonging to the same cluster as itself. So, for each $i \in [1, n]$, the set of points in its ϵ-neighborhood is computed and a constraint is put on its cardinality:
$$\#\{j \mid d(i, j) \leq \epsilon, G[j] = G[i]\} \geq MinPts.$$
– A must-link constraint on two points i and j is expressed by: $G[i] = G[j]$.
– A cannot-link constraint on i and j is expressed by: $G[i] \neq G[j]$.

Adding such constraints involves other constraints on D or S, as for instance $G[i] = G[j]$ implies $D \geq d(i, j)$ and $G[i] \neq G[j]$ implies $S \leq d(i, j)$.

Search strategy Let us recall that a solver iterates two steps: constraint propagation and branching when needed. In our model, variables $I[c]$ ($c \in [1, k]$) are instantiated before variables $G[i]$ ($i \in [1, n]$). This means that cluster representatives are first instantiated, allowing constraint propagation to assign some points to clusters; when all the $I[c]$ are instantiated, the variables $G[i]$ whose domains are not singletons are instantiated.

Variables $I[c]$ are chosen from $I[1]$ to $I[k]$. Since the representative is the one with the minimal index in the cluster, values for instantiating each $I[c]$ are chosen in an increasing order. Variables $G[i]$ are chosen so that the ones with the smallest remaining domain are chosen first. For instantiating $G[i]$, the index of the closest representative is chosen first.

4.2 Model improvement

Using constraint propagation to reduce the search space by deleting values in the domain of variables that cannot lead to a solution, CP solvers perform an exhaustive search, allowing to find an optimal solution. In order to improve the efficiency of the system, different aspects are considered.

Improvement by ordering the points. To instantiate the variables $I[c]$, the values (which are point indices) are chosen in an increasing order. The way points are indexed is therefore really important. Points are then ordered and indexed, so that points that are probably representatives have small index. In order to achieve this, we rely on FPF algorithm, introduced in Section 2. The algorithm is applied with $k = n$ (as many classes as points): a first point is chosen, the second point is the furthest from this point, the third one is the furthest from the two first and so on until all points have been chosen.

Improvement when minimizing the maximal diameter. Let us consider first the case where no user-defined constraints are put in the model. In [6], it is proven that if d_{FPF} represents the maximal diameter of the partition computed by FPF, then it satisfies $d_{opt} \leq d_{FPF} \leq 2d_{opt}$, with d_{opt} the maximal diameter of the optimal solution. This knowledge gives bounds on D: $D \in [d_{FPF}/2, d_{FPF}]$. Moreover, for each pair of points i, j :

- if $d(i,j) < d_{FPF}/2$, the reified constraint (*) on i, j is no longer put,
- if $d(i,j) > d_{FPF}$, the constraint (*) is replaced by: $G[i] \neq G[j]$.

Such a result allows to remove several reified constraints, without modifying the semantics of the model, and thus allows to improve the efficiency of the model, since handling reified constraints requires to introduce new variables.

In the case where user constraints are added, this result is no longer true, since the optimal diameter is in general greater than the optimal diameter d_{opt} obtained without user-constraints. The upper bound is no longer satisfied but the lower bound, namely $d_{FPF}/2$, still holds. Therefore for each pair of points i, j, if $d(i,j) < d_{FPF}/2$, the constraint (*) on i, j is not put.

Improvement when minimizing WCSD. Computing WCSD (**) requires to use a linear constraint on boolean variables ($G[i] == G[j]$). However, a partial assignment of points to clusters does not allow to filter the domain of the remaining values, thus leading to an inefficient constraint propagation. We have proposed a new method for propagating this constraint and filtering the domain of remaining variables. It is out of the scope of this paper, for more details, see [22]. Experiments in Section 5 show that it enables to handle databases that are out of reach of the most recent exact algorithms.

5 Experiments

5.1 Datasets and methodology

Eleven datasets are used in our experiments. They vary significantly in their size, number of attributes and number of clusters. Nine datasets are from the UCI repository [23]: Iris, Wine, Glass, Ionosphere, WDBC, Letter Recognition, Synthetic Control, Vehicle, Yeast. For the dataset Letter Recognition, only 600 objects are considered from the 20.000 objects of the initial data set, they are composed of the first 200 objects of each class. The datasets GR431 and GR666 are obtained from the library TSPLIB [24]; they contain coordinates of 431 and 666 European cities [25]. These two datasets do not contain information about the number of clusters and we choose $k = 3$ for the tests. Table 1 summarizes information about these datasets. There are few systems aiming at reaching a

Table 1. Properties of data sets used in the experimentation

Dataset	# Objects	# Attributes	# Clusters
Iris	150	4	3
Wine	178	13	3
Glass	214	9	7
Ionosphere	351	34	2
GR431	431	2	not available
GR666	666	2	not available
WDBC	569	30	2
Letter Recognition	600	16	3
Synthetic Control	600	60	6
Vehicle	846	18	4
Yeast	1484	8	10

global optimum. In Subsection 5.2, our model without user-constraints is compared to the Repetitive Branch-and-Bound Algorithm (RBBA) [5][2]. To the best of our knowledge, it is the best exact algorithm for the maximal diameter and WCSD criteria but it does not integrate user-constraints. The distance between objects is the Euclidean distance and the dissimilarity is measured as the squared Euclidean distance. As far as we know, there is no work optimizing the criteria presented in the paper and integrating user-constraints (with $k > 2$). In Subsections 5.3, 5.4 and 5.5, we show the ability of our model to handle different kinds of user-constraints.

Our model is implemented with the Gecode 4.0.0 library[3]. In this version released in 2013, float variables are supported. This property is important to obtain exact optimal value. All the experiments (our model and RBBA) are

[2] The program can be found in http://mailer.fsu.edu/~mbrusco/

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

performed on a PC Intel core i5 with 3.8 GHz and 8 GB of RAM. The time limit for each test is 2 hours.

5.2 Minimizing maximum diameter without user-constraint

Table 2 shows the results for the maximal diameter criterion. The first column gives the datasets, the second column reports the optimal values of the diameter. They are the same for both our model and the RBBA approach, since both approaches find the global optimal. The third and fourth columns give the total CPU times (in seconds) for each approach.

Table 2. Comparison of performance with the maximal diameter criterion

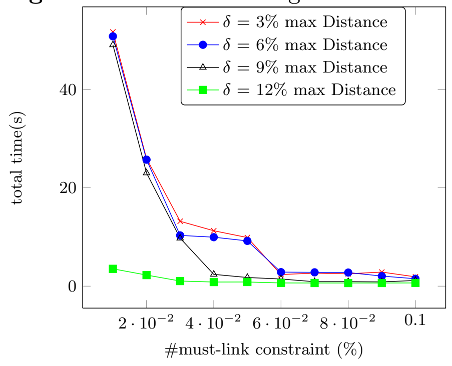
Dataset	Optimal Diameter	CP Framework	RBBA
Iris	2.58	0.03s	1.4s
Wine	458.13	0.3s	2.0s
Glass	4.97	0.9s	42.0s
Ionosphere	8.60	8.6s	$>$ 2 hours
GR431	141.15	0.6s	$>$ 2 hours
GR666	180.00	31.7s	$>$ 2 hours
WDBC	2377.96	0.7s	$>$ 2 hours
Letter Recognition	18.84	111.6s	$>$ 2 hours
Synthetic Control	109.36	56.1s	$>$ 2 hours
Vehicle	264.83	14.9s	$>$ 2 hours
Yeast	0.67	2389.9s	$>$ 2 hours

The results show that RBBA finds the optimal diameter only for the first three datasets. In [5], the authors mention that their algorithm is effective for databases with less than 250 objects. Table 2 shows that our model is able to find the optimal diameter for a data set with up to $n = 1484$ and $k = 10$. The performance does not only depend on the number of objects n and on the number of groups k, but also on the margin between objects and on the database features. The behavior of our model differs from classical models: for instance, when k increases, the search space is larger and in many approaches, solving such a problem takes more time. Indeed, since there are more clusters, the maximum diameter is smaller, and propagation of the diameter constraint is more effective, thus explaining that in some cases, it takes less computation time. As already mentioned, they may exist several partitions with the same optimal diameter; because of the search strategy of Constraint Optimization Problem in CP, our model finds only one partition with the optimal diameter.

5.3 Minimizing maximum diameter with user-constraints

Let us consider now the behavior of our system with user-constraints considering the Letter Recognition dataset. Figure 1 presents the results we obtain when must-link constraints (generated from the true classes of objects) and a margin constraint δ (the margin between two clusters must be at least δ) is used. The number of must-link constraints varies from 0.01% to 1% the total number of pairs of objects where δ ranges from 3% to 12% the maximum distance between two objects. Regarding to the results, both must-link and margin constraints boost the performance for this data set.

Fig. 1. Must-link and margin constraints



5.4 Minimizing Within-Cluster Sum of Dissimilarities

Minimizing the Within-Cluster Sum of Dissimilarities (WCSD) is a difficult task since the propagation of the sum constraint is less efficient than the propagation of the diameter constraint. Without users-constraints, both our model and the RBBA approach can find the optimal solutions only with the Iris dataset. Our model needs 4174s to complete the search whereas the RBBA takes 3249s. However, with appropriate user-constraints, the performance of our model can be significantly improved.

WCSD and the margin constraint. Let us add a margin constraint δ, where δ ranges from 0% (no constraint) to 12% of the maximum distance between two objects. Table 3 (left) reports the WCSD value of an optimal solution and the total computation time. It shows that when the margin constraint is weak, the optimal WCSD value does not change. But the computation time decreases significantly when this additional constraint becomes stronger. The reason is that the total number of feasible solutions decreases and the search space is reduced. When the margin constraint is weak, propagating this constraint is more time-consuming than its benefits.
WCSD and must-link constraints. Let us now add must-link constraints, where the number of must-link constraints, generated from the true classes of objects, varies from 0.2 to 1% of the total number of pairs. Results are expressed in Table

Table 3. margin and must-link constraint with dataset Iris

Margin Constraint	WCSD	Total time
no constraint	573.552	4174s
$\delta = 2\%$ max Dist	573.552	1452s
$\delta = 4\%$ max Dist	573.552	84.4s
$\delta = 6\%$ max Dist	573.552	0.3s
$\delta = 8\%$ max Dist	2169.21	0.1s
$\delta = 10\%$ max Dist	2412.43	0.04s
$\delta = 12\%$ max Dist	2451.32	0.04s

# must-link	WCSD	Total time
no constraint	573.552	4174s
0.2%	602.551	1275.1s
0.4%	602.551	35.6s
0.6%	617.012	16.1s
0.8%	622.5	3.5s
1%	622.5	1.6s
100%	646.045	0.04s

3 (right), giving the WCSD value and the total computation time. In fact, the optimal value of WCSD, with no information on classes, does not correspond to the WCSD found when considering the partition of this dataset into the 3 defined classes. The more must-link constraints, the less computation time is needed for finding the optimal value, and the closer to the value of WCSD, when considering the 3 initial classes. The reduction of computation time can be easily explained, since when an object is instantiated, objects that must be linked to it are immediately instantiated too. Furthermore, with any kind of additional constraint, the total number of feasible solutions is always equal or less than the case without constraint.

Table 4. Example of appropriate combinations of user-constraints

Data set	User constraints	WCSD	Total time
Wine	margin: $\delta = 1.5\%$ max Distance minimum capacity: $\beta = 30$	1.40×10^6	11.2s
GR666	margin: $\delta = 1.5\%$ max Distance diameter: $\gamma = 50\%$ max Distance	1.79×10^8	12.4s
Letter Recognition	# must-link constraints = 0.1% total pairs # cannot-link constraints = 0.1% total pairs margin: $\delta = 10\%$ max Distance	5.84×10^6	11.5s
Vehicle	margin: $\delta = 3\%$ max Distance diameter: $\gamma = 40\%$ max Distance	1.93×10^9	1.6s

WCSD and appropriate user-constraints. Finding an exact solution minimizing the WCSD is difficult. However, with appropriate combination of user-constraints, the performance can be boosted. Table 4 presents some examples where our model can get an optimal solution with different user-constraints, which reduce significantly the search space.

5.5 Interest of the model flexibility

Our system finds an optimal solution when there exists one; otherwise no solution is returned. Let us show the interest of combining different kinds of constraints. Figure 2 presents 3 datasets in 2 dimensions, similar to those used in [7].

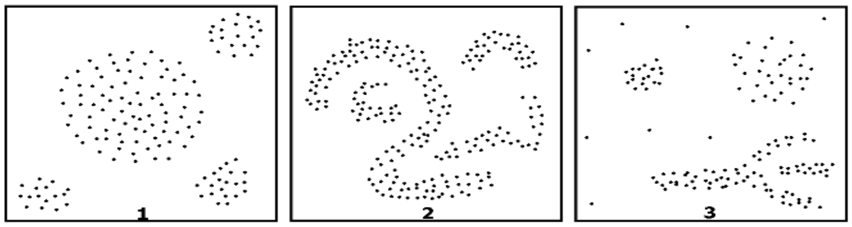


Fig. 2. Datasets

The first dataset is composed of 4 groups with different diameters. The second one is more difficult, since groups do not have the same shape. The third one contains outliers: outliers are not handled and are therefore integrated in classes.

When optimizing the maximal diameter, the solver tends to find rather homogeneous groups, as shown in Figure 3. Adding a min-margin constraint (with $\delta = 5\%$ of the maximum distance between pairs of points) improves the quality of the solution (see Figure 4). Let us notice that maximizing the minimum margin allows also to find this solution.

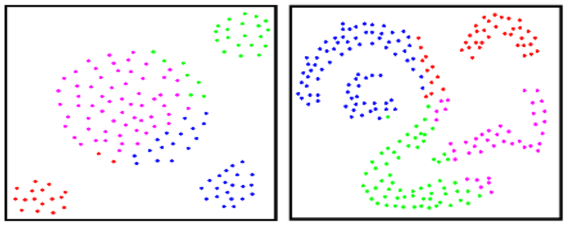


Fig. 3. Max-diameter optimization

Concerning the third dataset, minimizing the maximum diameter or maximizing the minimum margin do not allow finding a good solution (see Figure 5). The quality of the solution is improved when a density constraint is added with $MintPts = 4$ and $\epsilon = 25\%$ from the maximal distance between pairs of points.

6 Conclusion

We have proposed a declarative framework for Constrained Clustering based on Constraint Programming. It allows to choose among different optimization cri-

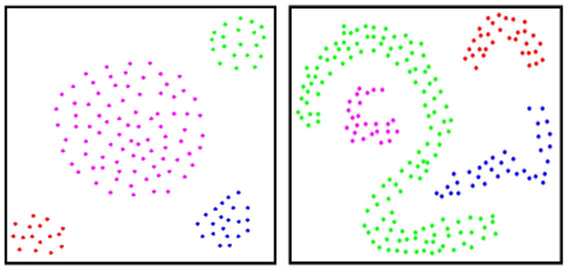


Fig. 4. maximal diameter optimization + margin constraint

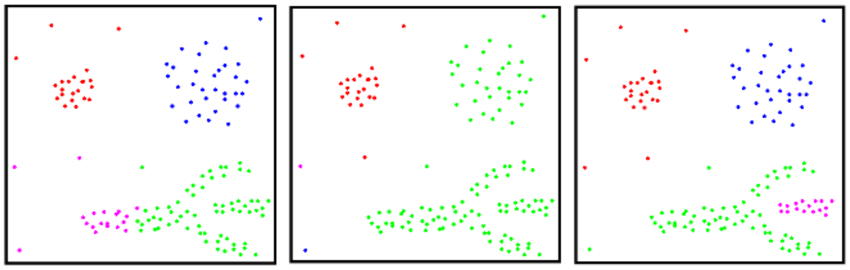


Fig. 5. Diameter opt. (left) Margin opt. (center) Margin opt. + density const. (right)

teria and to integrate various kinds of constraints. Besides the declarativity of CP, one of its advantage is that it allows to find an optimal solution, whereas most approaches find only local optima. On the other hand, complexity makes it difficult to handle very large databases. Nevertheless, integrating constraints enables to reduce the search space, depending on their ability to filter the domain of variables. Moreover, working on search strategies and on constraint propagation enables to increase the efficiency and to deal with larger problems. We plan to work on the search strategies and on the constraint propagators, thus being able to address larger databases. We do believe that global constraints adapted to the clustering tasks must be developed. From the Data Mining point of view, more optimization criteria should be added.

References

1. De Raedt, L., Guns, T., Nijssen, S.: Constraint programming for itemset mining. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2008) 204–212
2. De Raedt, L., Guns, T., Nijssen, S.: Constraint Programming for Data Mining and Machine Learning. In: Proc. of the 24th AAAI Conf. on Artificial Intelligence. (2010)
3. Boizumault, P., Crémilleux, B., Khiari, M., Loudni, S., Métivier, J.P.: Discovering Knowledge using a Constraint-based Language. CoRR **abs/1107.3407** (2011)

4. Dao, T.B.H., Duong, K.C., Vrain, C.: Une approche en PPC pour la classification non supervisée. In: 13e Conférence Francophone sur l'Extraction et la Gestion des Connaissances EGC. (2013)
5. Brusco, M., Stahl, S.: Branch-and-Bound Applications in Combinatorial Data Analysis (Statistics and Computing). 1 edn. Springer (July 2005)
6. Gonzalez, T.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science **38** (1985) 293–306
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Second International Conference on Knowledge Discovery and Data Mining. (1996) 226–231
8. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proc. of the 17th International Conference on Machine Learning. (2000) 1103–1110
9. Davidson, I., Ravi, S.S.: Clustering with Constraints: Feasibility Issues and the k-Means Algorithm. In: Proc. 5th SIAM Data Mining Conference. (2005)
10. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: Proceedings of the Eighteenth International Conference on Machine Learning. (2001) 577–584
11. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the Twenty-First International Conference on Machine Learning. (2004) 11–18
12. Davidson, I., Ravi, S.S.: Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. Proceedings of the 9th European Conf. on Principles and Practice of Knowledge Discovery in Databases (2005) 59–70
13. Lu, Z., Carreira-Perpinan, M.A.: Constrained spectral clustering through affinity propagation. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE (June 2008) 1–8
14. Wang, X., Davidson, I.: Flexible constrained spectral clustering. In: KDD '10: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2010) 563–572
15. Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Foundations of Artificial Intelligence. Elsevier B.V. (2006)
16. Guns, T., Nijssen, S., De Raedt, L.: k-Pattern set mining under constraints. IEEE Transactions on Knowledge and Data Engineering (2011)
17. Métivier, J.P., Boizumault, P., Crémilleux, B., Khiari, M., Loudni, S.: Constrained Clustering Using SAT. In: IDA 2012, LNCS 7619. (2012) 207–218
18. Mueller, M., Kramer, S.: Integer linear programming models for constrained clustering. In: Discovery Science. (2010) 159–173
19. Schmidt, J., Brändle, E.M., Kramer, S.: Clustering with attribute-level constraints. In: ICDM. (2011) 1206–1211
20. Davidson, I., Ravi, S.S., Shamis, L.: A SAT-based Framework for Efficient Constrained Clustering. In: SDM. (2010) 94–105
21. Aloise, D., Hansen, P., Liberti, L.: An improved column generation algorithm for minimum sum-of-squares clustering. Math. Program. **131**(1-2) (2012) 195–220
22. Dao, T.B.H., Duong, K.C., Vrain, C.: Constraint programming for constrained clustering. Technical Report 03, LIFO, Université d'Orléans (2013)
23. Bache, K., Lichman, M.: UCI machine learning repository (2013)
24. Reinelt, G.: TSPLIB - A t.s.p. library. Technical Report 250, Universität Augsburg, Institut für Mathematik, Augsburg (1990)
25. Grötschel, M., Holland, O.: Solution of large-scale symmetric travelling salesman problems. Math. Program. **51** (1991) 141–202