LIFO - Bâtiment 3IA
Rue Léonard de Vinci
BP 6759
45067 Orléans Cedex 2, France

Rapport de Recherche

Modeling Constrained Clustering Tasks with Bounded Number of Clusters

Thi-Bich-Hanh Dao, Khanh-Chuong Duong, Christel Vrain
LIFO, Université d'Orléans

Rapport n^o **RR-2013-08**

Modeling Constrained Clustering Tasks with Bounded Number of Clusters

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. Constrained clustering is an important task in Data Mining. In the last ten years, many works have been done to extend classical clustering algorithms to handle user-defined constraints, but in general to handle one kind of constraints. In our previous work [1], we have proposed a declarative and generic framework, based on Constraint Programming, which enables to design a clustering task by specifying an optimization criterion and different kinds of user-constraints. The model is designed for a clustering task which divides data in exactly k clusters. In this paper, we present a new model for constrained clustering tasks where the number of clusters is only upper and lower bounded. The new model is simpler and first experiments show that it has a better performance for the diameter based criterion.

1 Introduction

Constrained clustering is an important Data Mining task, allowing to better model a clustering task by integrating user-constraints. Several kinds of constraints can be considered, they can be statements on the clusters, as for instance on their diameters or on pairs of objects that must be or cannot be in the same cluster. Nevertheless designing an algorithm able to handle different kinds of constraints is difficult and most classical approaches have been extended to semi-supervised clustering, taking only into account one kind of constraints. In a previous work [1], we have proposed a declarative and generic framework, based on Constraint Programming, which enables to design a clustering task by specifying an optimization criterion and different kinds of user-constraints either on the clusters or on pairs of objects. This model is based on a two-level representation: a set of variables assigning a representative point to each cluster and a set of variables assigning a representative point to each object. It requires the number of clusters to be fixed beforehand. In this paper, we propose a simpler framework composed only of a set of variables assigning to each object the index of the cluster it belongs to. It allows to fix an upper bound k_{max} and a lower bound k_{min} on the number of clusters, thus leading to more flexibility when designing the clustering task. The model will find a clustering of k clusters such that $k_{min} \leq k \leq k_{max}$ where all user-constraints are satisfied and the specified criterion is globally optimized, if there exists a solution.

Relying on Constraint Programming (CP) has two main advantages: the declarativity, which enables to easily add new constraints and the ability to find a global optimal solution satisfying all the constraints while there exists solution. Recent progresses in CP have made this paradigm more powerful and several works [2–4] have already shown its interest for Data Mining. It is well known that the efficiency of a model based on Constraint Programming depends strongly on the choice of the variables, which represent the elements of the problem, on the ability to break symmetries (different representations of the same solutions), and on the search strategy. In our previous model, the use of cluster representatives is motivated by breaking symmetries between different representations of a same cluster set. In this new model, symmetry breaking is expressed by a more detailed study on partitioning. The result is promising since while the model represents more flexibility, it is simpler and eases the search strategy studies. First experiments comparing the two models on basic benchmarks with the diameter criterion show that the new model has a better performance.

The paper is organized as follows. In Section 2, we give background notions on clustering, constrained clustering and constraint programming. Our previous CP model with cluster representatives is presented in Section 3. Section 4 is devoted to the presentation of the new 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:

- Minimizing Within-Cluster Sum of Dissimilarities (WCSD)

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

  where o_i, o_j are objects in the cluster C_c.
- Minimizing Within-Cluster Sum of Squares (WCSS) (also called the least square criterion):

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

  where m_c is the center of cluster C_c.
- Minimizing absolute-error:

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

  where r_c is a representative object of the cluster C_c.
- Minimizing the maximal diameter:

$$E = max_{c \in [1,k], o_i, o_j \in C_c}(d(o_i, o_j))$$

  E is the maximum diameter of the clusters, where the diameter of a cluster is the maximum distance between any two of its objects.
- Maximizing the minimal margin:

$$E = min_{c < c' \in [1,k], o_i \in C_c, o_j \in C_{c'}}(d(o_i, o_j))$$

  E is the minimal margin between clusters, where the minimal margin between 2 clusters $C_c, C_{c'}$ is the minimum value of the distances $d(o_i, o_j)$, with $o_i \in C_c$ and $o_j \in C_{c'}$.

Clustering task with these criteria is NP-Hard, well-known algorithms use heuristics and usually find a local optimum. Different algorithms are developed, as for instance k-means for the WCSS criterion, k-medoids for the absolute-error criterion or FPF (Furthest Point First) [5] for the maximum diameter criterion.

Some algorithms do not rely on an optimization algorithm, as for instance DBSCAN [6], based on the notion of density. Two input parameters are needed, a radius ϵ and a threshold $MinPts$, which allows to adjust the notion of density.

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-defined 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 or instance-level constraints.

Most of the attention has been put on instance-level constraints, first introduced in [7]. Commonly, two kinds of constraints are used: must-link and cannot-link. 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. The minimum (maximum) capacity constraint requires that each cluster has at least (at most) a given number α (β) of objects: $\forall c \in [1,k],\ |C_c| \geq \alpha$ or $\forall c \in [1,k],\ |C_c| \leq \beta$. The maximum diameter constraint specifies an upper bound γ on each cluster diameter:

$$\forall c \in [1,k], \forall o_i, o_j \in C_c,\ d(o_i,o_j) \leq \gamma$$

The minimum margin constraint, also called the δ-constraint in [8], specifies a lower bound δ on the margin between clusters:

$$\forall c \in [1,k], \forall c' \neq c, \forall o_i \in C_c, o_j \in C_{c'}, d(o_i,o_j) \geq \delta$$

The ϵ-constraint introduced in [8] requires for each point o_i to have in its neighborhood of radius ϵ at least another point of the same cluster:

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

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 with radius ϵ contains at least $MinPts$ 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 [7], of k-means [9, 10], hierarchical non supervised clustering [11] or spectral clustering [12, 13], ... 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 (CP) 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 $a_i \in D_i$ 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 (in general, arc consistency removes all the inconsistent values, while bound consistency modifies only the bounds of the domain). Consistency for each constraint is chosen by the programmer. Let us notice that not all formulae or mathematic relations can be a constraint, constraints in CP are only those for which we can design a propagation algorithm. Arithmetic and logic relations are available as constraints, as well as more complex relations expressed by global constraints.

In the solver, 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 singleton 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 singleton and splits its domain into two parts, thus leading to two 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 all 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 really important, since it may drastically reduce the search space and therefore computation time. For more details, see [14].

Example 1. Let us illustrate the behavior of a solver by the following COP: find an assignment of letters to numbers so that $SEND + MOST = MONEY$ and maximizing the value of $MONEY$. This problem can be modeled by a Constraint Optimization Problem with eight variables S, E, N, D, M, O, T, Y, the domain of which is 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: $\texttt{alldifferent}(S, E, N, D, M, O, T, Y)$. 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.
- $(1000S + 100E + 10N + D) + (1000M + 100O + 10S + T) = 10000M + 1000O + 100N + 10E + Y$
- Maximize$(10000M + 1000O + 100N + 10E + Y)$.

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 all the domains are not reduced to singleton, 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 is really important. When variables are chosen in the order S, E, N, D, M, O, T, Y and for each variable, values are chosen following an increasing order, the search tree is composed of 29 nodes and 7 intermediate solutions (solutions satisfying all the constraints, better than the previous ones 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 intermediate solutions. The two search trees are given in Figure 1, they have been generated by the environment Gist of the Gecode solver[1]. A blue circle is a stable state that is not a solution, a red square

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

denotes a failure (no solution possible with the current partial assignment) and a green rhombus is an intermediate solution and the orange one is the optimal solution. Each stable state which is not a solution leads to a branching: in the left branch, an assignment is made on a variable with a value of its domain and in the right branch, the chosen value is removed from the domain of the variable.

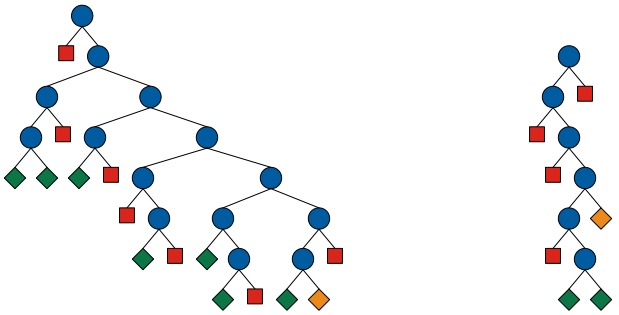


Fig. 1: Search tree.

3 CP model with cluster representatives

In [1], we have presented a Constraint Programming model for constrained clustering tasks. 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 k is fixed by the user and the model aims at finding a partition of data into k clusters, satisfying a set of constraints specified by the user and optimizing a given criterion.

3.1 Variables

For each cluster $c \in [1, k]$, the point with the smallest index is considered as the representative point of the cluster[2]. 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 $o_1, \ldots, o_7$ 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.

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

3.2 Constraints

The constraints allow to model a partitioning with cluster representatives, different optimization criteria and user-defined constraints.

Constraints of partitioning Theses constraints express the relation between points and their clusters (representatives).

- 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 "count" constraint in CP:

$$\forall i \in [1,n], \quad \#\{c \mid I[c]{=}G[i]\} = 1. \tag{1}$$

  For each $i \in [1,n]$, this constraint counts the number of times the value taken by $G[i]$ appears in the array $I[1],\ldots,I[k]$ and puts the condition that this number must be equal to 1. This means $G[i]$ must have a value among all the values taken by the variables $I[1],\ldots,I[k]$. For instance, with $n = 7$ and $k = 2$, if $I[1] = 1$ and $I[2] = 3$, then the propagation of this constraint will reduce the domain of each $G[i]$ into the set $\{1, 3\}$.
- 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: $\forall i < j \in [1,n]$, $d(i,j) > D \rightarrow (G[i] \neq G[j])$.
- 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: $\forall i < j \in [1,n]$, $d(i,j) < S \rightarrow G[i] = G[j]$.
- 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 $S \leq d(i,j)$.

3.3 Search strategy

The variables $I[c]$ ($c \in [1, k]$) are instantiated before the 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 domain is not a singleton are instantiated. It is worth to notice that when all the variables $I[c]$ are instantiated, by the constraint (1) among the partitioning constraints, the remaining domains of $G[i]$ contain only the indices of the points which are representatives. For the details of search strategy and the improvements of this model, we refer the reader to [15].

4 A new CP model for constrained clustering

In this section we present a new CP model for constrained clustering, where cluster representatives are no longer used. This new model aims at finding a clustering which is composed of at least k_{min} clusters and at most k_{max} clusters, where k_{min} and k_{max} are known parameters, which satisfies all user-constraints and optimizes the specified criterion.

4.1 Variable choice

The new model makes use of integer valued variables $G[1], \ldots, G[n]$ whose domain is the set of integers in $[1, k_{max}]$. These variables represent the assignment of points to clusters. An assignment $G[i] = c$ means that the point i is assigned to the cluster having the index c. The same name for these variables is kept with respect to the previous model, since these variables have the same meaning, but they do not have the same domain. The other variables D, S and V are the same with respect to the previous model.

4.2 Constraints

We can see that in the previous model all the constraints except the partitioning constraints deal only with the variables $G[i]$. These variables in the two models have the same meaning, they represent the assignment of points to clusters. In the new model therefore only the constraints of partitioning need to be reformulated. In this model, clusters are not identified by their representatives, but by their index (their number). A set of clusters could have different representations by permutations on the cluster indices. In order to break symmetries, clusters will be numbered such that a new number c is used for a new cluster if and only if all the other numbers $c' < c$ are already used. The first cluster must contain the first point. If we need to put a point in another cluster (the second), then the second cluster must have number 2. And if there are already 2 used clusters and we need to put a point in a new cluster, the new cluster must have number 3, and so on. Partitioning is expressed by the following constraints.

– The first point belongs to the first cluster:

$$G[1] = 1.$$

– There must be at most k_{max} clusters: this condition is straightforwardly satisfied, since the domain of each variable $G[i]$ is the set of integers in $[1, k_{max}]$.
– There must be at least k_{min} clusters: that means, all the number from 1 to k_{min} must be used for clusters. This means for all $c \in [1, k_{min}]$, the value c must be taken at least once by the variables $G[1], \ldots, G[n]$. This is expressed by a "count" constraint: $\forall c \in [1, k_{min}]$,

$$\#\{i \mid G[i] = c\} \geq 1.$$

In case the users need exactly k clusters, the parameters $k_{min} = k_{max} = k$, so any value $c \in [1, k]$ must appear at least once in the array $G[1], \ldots, G[n]$.
– A number c of cluster is used if and only if all the numbers $c' < c$ are already used: $\forall i \in [1, n]$

$$G[i] \leq \max_{j \in [1, i-1]} (G[j]) + 1.$$

This constraint imposes that each point i must be either in a same cluster as another precedent point, or in a new cluster of index $c = c' + 1$, where c' is the maximal index of clusters already used.

4.3 Search strategy

Points are reordered previously in such a way that the first k_{max} points are more likely to be in different clusters. We use FPF (Furthest Point First) heuristic [5] to reorder points. The first picked point is the furthest point and all the other points have as a head this point. At each step, the point which is the furthest from its head is picked, and the unpicked points which are closer to this point than to their head change their head to this point. Steps are repeated until all points are picked. The order of picking points is the order of points.

Branching is realized on the variables in the array G. The choice of the variables G depends on the optimized criterion. For the WCSD criterion, the branching on uninstantiated variables in G finds a variable $G[i]$ and a value c in the domain of $G[i]$ and makes two alternatives: $G[i] = c$ and $G[i] \neq c$. The variable $G[i]$ is selected among those which have the smallest domain size. In this model as in the previous model, a mixed strategy is used. Because an upper bound is necessary for the constraint which computes the WCSD criterion, a greedy search is used first to find quickly a solution. In this step, $G[i]$ and c are selected to make sure that the value of V will increase as little as possible. The solution found in general is quite good. After first solution, the search strategy is changed to a "first-fail" search, which tends to cause the failure early. In this strategy, the branching will try to make alternatives on frontier points, *i.e.* those that make most changes on V.

For the maximal diameter criterion, the variable $G[i]$ with the smallest remaining domain is chosen first. When a variable $G[i]$ is chosen, all values c in the domain of $G[i]$ are examined and we search for the 'closest' class c_0 to point i. The distance between point i and a class c is defined as the maximum distance between i to all points j such that $G[j]$ has been instantiated and $G[j] = c$. If a class c is empty (there does not exist any instantiated point $G[j]$ such that $G[j] = c$), the distance between i and this class is zero. The class c_0 which is closest to the point i is then determined based on these distances. The branching on $G[i]$ makes then two alternatives $G[i] = c_0$ and $G[i] \neq c_0$. With this strategy, the branching will assign point to empty group first when possible. This strategy is different from the one used with the previous model, where the branching depends on the distance between point i to each cluster representative.

5 Experiments

5.1 Datasets and methodology

The same 11 datasets as in [1] are used for experiments. They vary significantly in their size, number of attributes and number of clusters. Nine datasets are from the UCI repository [16]: Iris,

Wine, Glass, Ionosphere, WDBC, Letter Recognition, Synthetic Control, Vehicle, Yeast. For the dataset Letter Recognition, only 600 objects of 3 classes are considered from the 20.000 objects in the original dataset, they are composed of the first 200 objects of each class. The data sets GR431 and GR666 are obtained from the library TSPLIB [17]; they contain the coordinates of 431 and 666 European cities [18]. These two datasets do not contain the information about the number of clusters k and we choose $k = 3$ for the tests. Table 1 summarizes informations about these datasets.

Table 1: Properties of datasets used in the experiments

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

The two models are implemented with the Gecode library version 4.0.0. In this version released in April 2013, float variables are supported. Experiments are all performed on a PC Intel core i5 with 3.8 GHz and 8 GB of RAM. Basic benchmarks on two models with the maximal diameter criterion are studied.

5.2 Comparison of performances of two models

Table 2 shows the results for minimizing the maximal diameter criterion without user-constraint. The first column gives the datasets, while the second column reports the optimal values of the diameter. The third and fourth columns give the total CPU times (in seconds) for each model. For the previous model, the number of classes is fixed, whereas in the new model, a lower bound on the number of classes is used (k_{min} is set to 1). However, since there is no user-constraint, the new model always finds the optimal solution with maximum number of classes allowed because the more number of classes, the less so the better maximal diameter. Finally, both models find the same optimal diameter and the same number of classes, but the solutions may be different as there may exist several partitions with the same optimal diameter and both models find only one (the first) partition with the optimal diameter.

The results show that, with new search strategy, the performance of new model is significantly better. The reason is: with a better search strategy, the search space is smaller. Although new search strategy requires more computations, the overall performance of the new model is better in most of tests.

These are first experiments on the new model. More studies on the effectiveness of is model with user-constraints and other criteria are planned for the near future.

5.3 Interest of this new model

The framework finds an optimal solution when there exists one, otherwise no solution is returned. With the new model, users can find a clustering without specifying the exact number of clusters. With the maximal diameter criterion and without user-constraints, in general, the solver always

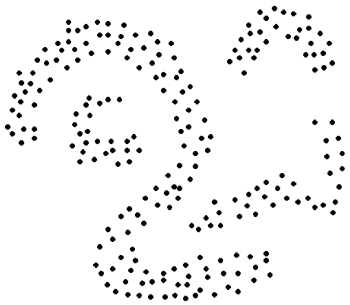
Table 2: Comparison of two models with maximal diameter criterion

Dataset	Optimal Diameter	Previous Model	New Model
Iris	2.58	0.03	0.03
Wine	458.13	0.3	0.06
Glass	4.97	0.9	0.4
Ionosphere	8.6	8.6	0.5
GR431	141.15	0.6	0.5
GR666	180	31.7	5.3
WDBC	2377.96	0.7	1
Letter Recognition	18.84	111.6	86.4
Synthetic Control	109.36	56.1	24.4
Vehicle	264.83	14.9	12.9
Yeast	0.67	2389.9	592.5

tries to find the best solution with a maximum number of clusters. However, with user-constraints, the optimal solution may have less clusters.

Let us illustrate the interest of having the flexibility on the number of clusters with a dataset similar to dataset used in [6]. In this dataset given in Figure 2, there are 4 groups of different shapes. The maximal diameter criterion is chosen and the number of classes is between $k_{min} = 1$ and $k_{max} = 6$. Without user-constraints, the model finds the optimal solution with 6 groups. However, these groups do not reflect the real partition as the solver tends to rather find homogeneous groups. Adding a min-margin constraint may help to improve the quality of the solution. With the min-margin constraint with $\delta = 5\%$ of the maximum distance between pairs of points, it is not possible to get a solution with 5 or 6 groups. In this case, the solver found an optimal solution with 4 groups. Let us notice that the previous model can find the same solution but user have to set exactly $k = 4$. When the min-margin constraint is more strict ($\delta \geq 10\%$), the solver finds the optimal solution with only 2 or 3 groups. Solutions of different cases are expressed in table 3.

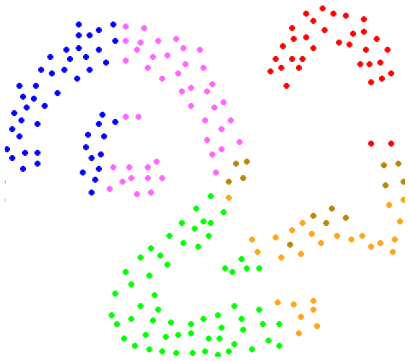
Fig. 2: Dataset



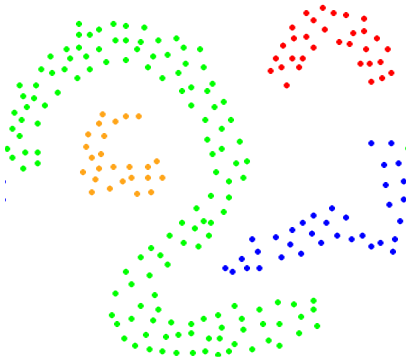
6 Conclusion

In this paper we present a model based on Constraint Programming to address Constrained Clustering tasks. This model has the ability to handle an optimization criterion and different kinds of user-constraints. Moreover, it allows the users not to fix an exact number of clusters, but only to define an upper and a lower bound on the desired number of clusters. With the diameter-based criterion, first experiments show that the model has a better performance. This could be a result by this simpler model and by a more appropriated search strategy. We plan to study search strategies in order to improve better the efficiency of our models.

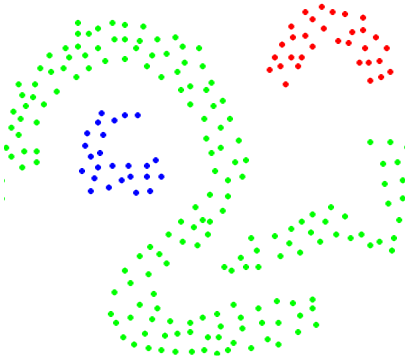
Table 3: Optimal solution with constraint of separation



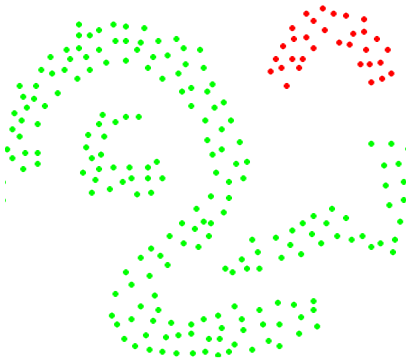
No user-constraint



$\delta = 5\%$ max Distance



$\delta = 10\%$ max Distance



$\delta = 13\%$ max Distance

References

1. Dao, T.B.H., Duong, K.C., Vrain, C.: A declarative framework for constrained clustering. In: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ECML/PKDD. (2013)
2. 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
3. De Raedt, L., Guns, T., Nijssen, S.: Constraint Programming for Data Mining and Machine Learning. In: Proc. of the 24th AAAI Conference on Artificial Intelligence. (2010)
4. Boizumault, P., Crémilleux, B., Khiari, M., Loudni, S., Métivier, J.P.: Discovering Knowledge using a Constraint-based Language. CoRR **abs/1107.3407** (2011)
5. Gonzalez, T.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science **38** (1985) 293–306
6. 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
7. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proceedings of the Seventeenth International Conference on Machine Learning. (2000) 1103–1110
8. Davidson, I., Ravi, S.S.: Clustering with Constraints: Feasibility Issues and the k-Means Algorithm. In: Proc. 5th SIAM Data Mining Conference. (2005)
9. 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
10. 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
11. 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
12. 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
13. 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
14. Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Foundations of Artificial Intelligence. Elsevier B.V., Amsterdam, Netherlands (August 2006)
15. Dao, T.B.H., Duong, K.C., Vrain, C.: Constraint programming for constrained clustering. Technical Report 03, LIFO, Université d'Orléans (2013)
16. Bache, K., Lichman, M.: UCI machine learning repository (2013)
17. Reinelt, G.: TSPLIB - A t.s.p. library. Technical Report 250, Universität Augsburg, Institut für Mathematik, Augsburg (1990)
18. Grötschel, M., Holland, O.: Solution of large-scale symmetric travelling salesman problems. Math. Program. **51** (1991) 141–202