A Filtering Algorithm for Constrained Clustering with Within-Cluster Sum of Dissimilarities Criterion

Thi-Bich-Hanh Dao, Khanh-Chuong Duong, Christel Vrain
Univ. Orléans, ENSI de Bourges, LIFO, EA 4022, F-45067, Orléans, France
Email: {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 restricted to handle one kind of user-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. One of the criteria is the within-cluster sum of dissimilarities, which is represented by a sum constraint and reified equality constraints $V = \sum_{1 \le i < j \le n}(G[i] == G[j])a_{ij}$. A direct implementation using predefined constraints is not effective as the propagation of theses constraints is weak. In this paper, we consider this criterion as a global constraint and develop a filtering algorithm for it. This filtering helps to improve significantly the model performance. Experiments on classical databases show the interest of our approach.**

***Keywords*-Constrained clustering, modeling, filtering algorithm**

I. Introduction

Cluster analysis 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 or requirements on clusters. Since its introduction [2], various kinds of user-constraints have been integrated to clustering methods [3]. Nevertheless, dedicated algorithms need to be developed for each kind of criterion and user-constraints. Developing a general framework with the ability of handling different kinds of constraints is still of high importance.

Relying on the declarativity inherent to Constraint Programming (CP), several works [4], [5], [6] have investigated the use of CP for modeling Data Mining tasks. 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. One of the optimization criteria which can be handled by the framework is the within-cluster sum of dissimilarities (WCSD) criterion. The clustering task with the WCSD criterion is NP-Hard since the weighted max-cut problem, which is NP-Complete [7], is a particular instance of this problem with two clusters. Algorithms for clustering in general use heuristics and find a local optimum, and when adapted to handle user-constraints, usually need to make compromise between the optimized criterion and the constraints to be satisfied [3]. Our CP model guarantees to find a global optimum satisfying all user-constraints, if there exists a solution.

One of the key constraints of our model is a combination of a sum constraint with reified equality constraints $V = \sum_{1 \le i < j \le n}(G[i] == G[j])a_{ij}$, with V a variable, G an array of variables and a_{ij} constants. A direct implementation of the model using predefined constraints is not effective as the propagation of these constraints is weak. The main contribution of this paper is a filtering algorithm for this constraint in case of a partitioning problem. This algorithm takes benefit not only on the instantiated variables, but also on the unassigned variables. It improves significantly the efficiency of the model, makes it comparable to the best known exact approach for WCSD criterion [8] without user-constraints. Moreover, our model can handle and get benefits from user-constraints.

The paper is organized as follows. After some preliminaries on constrained clustering with the WCSD criterion in Section II, Section III presents the CP model and Section IV presents the filtering algorithm. Experiments are presented in Section V, and conclusion and future works are given in Section VI.

II. Preliminaries

A. 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 points into k clusters $C_1, \ldots, C_k$ optimizing a criterion E. Optimized criteria may be, among others:

- Within-Cluster Sum of Squares (WCSS) criterion:

$$E = \sum_{c=1}^{k} \sum_{o_i \in C_c} ||m_c - o_i||^2$$

where m_c is the center of cluster C_c.

- Within-Cluster Sums of Dissimilarities (WCSD) (also called the least square criterion):

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

where o_i, o_j are objects in the cluster C_c. When squared Euclidean distance is used as the dissimilarity measure, this criterion once standardized via the division by the size of each group, $\sum_{c=1}^{k}(\sum_{o_i,o_j \in C_c} d(o_i, o_j))/|C_c|$, is mathematically equivalent to the Within-Cluster Sum of Square criterion. Nevertheless, optimizing these criteria requires different techniques.
- Absolute-error criterion:

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

where r_c is a representative object of the cluster C_c.
- Diameter-based criterion:

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

E is the maximum diameter of the clusters, where the diameter of a cluster is the maximum distance between any two of its objects.

The clustering task with one of these criteria is NP-Hard. Well-known algorithms such as k-means, k-medoids use heuristics and usually find a local optimum. Exact algorithms are much less numerous. For the WCSS criterion, the best known exact method is a column generation algorithm [9], which solves problems up to $n = 2300$. However, the ratio n/k should be small, roughly equal to 10, in order to have a reasonable time. For the WCSD criterion, the best known exact method is a repetitive branch-and-bound algorithm (RBBA) [8], where the authors claim to solve problems up to $n = 50$ and $k = 6$.

B. Constrained-Clustering

Most clustering methods rely on an optimization criterion, and because of the inherent complexity, search for a local optimum. Several local 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-constraints can be added, leading to Constrained-Clustering that aims at finding clusters that satisfy user-specified constraints. Adding user-constraints to a clustering task allows to have a solution closer to the desired one. For example, the well-known k-means method tends to find homogeneous clusters and it cannot find a useful partition for databases with clusters of different forms (e.g. Figure 1). Adding user-constraints may guide the search to a good partition.

User-constraints can be classified into cluster-level constraints and instance-level constraints. Most of work on user-constraints has been put on instance-level constraints, first introduced in [2], which state that two points must or must not be in a same cluster. Cluster-level constraints may be conditions on the maximal or minimal size of clusters, on the diameter of clusters, on the minimal distance between clusters or on the density of clusters.

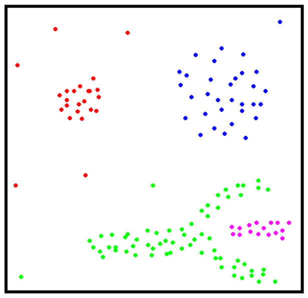
Figure 1. Clusters of different forms and sizes

In the last ten years, many classical algorithms have been extended for handling must-link and cannot-link constraints, as for instance an extension of COBWEB [2], 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, algorithms to handle user-constraints are usually developed for a certain type of constraints. To the best of our knowledge there is no general solution to extend traditional algorithms to different types of constraints.

Approaches for finding a global optimum with user-constraints are less numerous and are based on Constraint Programming [5], SAT [6], [15] or Integer Linear Programming [16].

C. Related work

L. De Raedt *et al.* present in [5] a framework in Constraint Programming for k-patterns set mining and show how it can be applied to conceptual clustering. The problem consists in finding k-patterns that do not overlap and cover all the transactions. Given a set of transactions, each transaction can be seen as an object and two objects are similar if they share the same pattern. Constraints are used to express the relationship of covering, non-overlapping, etc. Additional constraints can be added to express the criterion function maximizing the minimum size of clusters or minimizing the difference between cluster sizes.

P. Boizumault *et al.* present in [6] and [15] a constraint-based language expressing queries to discover patterns in data mining. The constrained conceptual clustering problem can be expressed in this language. The language elements are translated into a set of clauses and a SAT solver is used to solve the clustering task.

Mueller *et al.* propose in [16] an approach to constrained clustering based on Integer Linear Programming. This approach takes a set of candidate clusters as input and constructs a clustering by selecting a suitable subset. It allows constraints on the degree of completeness of a clustering, on the overlapping of clusters and it supports set-level constraints which restrict the combinations of admissible clusters. This approach is different as it takes into account constraints on candidate clusters, but no constraints on individual objects. It has different objective functions optimizing the minimum, the mean or the median of the individual cluster qualities in a clustering. Their framework is flexible and guarantees to find a global optimum but requires a set of candidate clusters. This condition makes the framework less convenient for clustering in general, since finding a good set of candidate clusters is a difficult task and the number of candidate clusters is exponential compared to the number of points. This approach is more suitable for conceptual clustering where objects are itemsets and candidate clusters might be created from frequent itemsets.

To the best of our knowledge, there is no exact algorithm for clustering tasks with WCSD criterion when apply any kind of user-constraints. In most cases, it is difficult to adjust an approach developed without user-constraints to take into account user-constraints. For example, in order to find an optimal solution with n points, RBBA finds optimal solution for problems with $k+1$ to $n-1$ points and uses those optimal values to calculate a lower bound for next step. However, user-constraints are defined for the full dataset of n points, not for $k+1$ to $n-1$ points. In our opinion, there is no easy way to integrate any kind of user-constraints to RBBA or other exact algorithms.

In [1], we present a CP model, which is general for modeling a clustering task with different types of user-constraints and different optimization criteria. Our approach considers the clustering problem in its original form and is capable to proceed on qualitative and quantitative databases. In this paper, we focus on the WCSD criterion. After the presentation of our model (Section III), we present the filtering algorithm, which helps to improve significantly the model performance with this criterion (Section IV).

III. Constraint programming model

A. Model

Variables: A solution is an assignment of each point to a cluster. Therefore, for each point we need at least a variable to represent this assignment. However, simply assigning a point to a cluster index may cause symmetrical solutions that affect significantly the performance. To break symmetries, for each cluster $c \in [1, k]$, the point with the smallest index is considered as the representative of the cluster[1]. An integer variable $I[c]$ is introduced, its value is the index of the representative point of cluster c; the domain of $I[c]$ is therefore the set of integers in $[1, n]$. 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.

To represent the value of the criterion, we introduce a float variable V. The domain of V is upper-bounded by the sum of the square distances between all pairs of points.

Modeling clustering task: Relations between points and their cluster:

- Each representative is its representative: for $c \in [1, k]$, $G[I[c]] = I[c]$.
- Each point must be assigned to a representative: for each i, the value of $G[i]$ must appear in the array $I[1], \ldots, I[k]$. This can be expressed by n "count" constraints: for $i \in [1, n]$,
$$\#\{c \mid I[c] = G[i]\} = 1 \qquad (1)$$
- A cluster representative must have the smallest index among those in the cluster, so any point must have its index greater or equal to its representative: for $i \in [1, n]$, $G[i] \leq i$.
- To avoid symmetries, cluster representatives are in an increasing order: for $c < c'$, $I[c] < I[c']$, and the first representative is the first point $I[1] = 1$.
- The WCSD criterion V is minimized, where
$$V = \sum_{1 \leq i < j \leq n} (G[i] == G[j]) d(i, j)^2. \qquad (2)$$

Modeling user-constraints: User-constraints can be formulated directly. For instance-level constraints, a must-link constraint on i, j is expressed by $G[i] = G[j]$ and a cannot-link constraint by $G[i] \neq G[j]$. For cluster-level constraints:

- A minimal (maximal) capacity constraint states that each cluster must have at least (at most) a number α (or β) of points: for $c \in [1, k]$, $\#\{i \mid G[i] = I[c]\} \geq \alpha$ (or $\#\{i \mid G[i] = I[c]\} \leq \beta$).
- A separation constraint states that any two clusters must be separated by at least δ, so any two points at a distance less than a parameter δ must be in the same cluster: for $i < j \in [1, n]$ such that $d(i, j) < \delta$, $G[i] = G[j]$.
- A maximal diameter constraint states that the diameter of each cluster must be at most γ, so any two points at a distance greater than γ must be in different clusters: for $i < j \in [1, n]$ such that $d(i, j) > \gamma$, $G[i] \neq G[j]$.
- A density constraint states that any point, within a radius of ϵ around itself, must have at least m points in the same cluster. So, for any point i, a set S of points with distance from i less than ϵ is calculated, on which this constraint imposes: $\#\{j \in S \mid G[j] = G[i]\} \geq m$.

B. Branching

Variables are chosen first those in I, then those in G, which means the cluster representatives are first identified,

[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.

points are then assigned to clusters. Since a cluster representative $I[c]$ must have the smallest index among those in the cluster, values for $I[c]$ are chosen in increasing order. Point indices are then really important, by consequent, points are ordered previously in such a way that those which are more likely to be representative have small indices. We use FPF (Furthest Point First) heuristic [17] 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 points which are picked is the order of points used in our model.

After instantiating all variables $I[1], \dots, I[k]$, the constraints (1) allow to reduce the domain of each variable $G[i]$ to the set of indices of the representative points. The branching on uninstanciated 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. The value c is always the representative of the closest group to point i. In our model, a mixed strategy is used. Because an upper bound is necessary for the constraint (2) to be effective, a greedy strategy is used first to quickly find 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 finding a 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 the most changes on V.

IV. Filtering algorithm for the WCSD criterion sum constraint

Using predefined constraints, Constraint (2) for the WCSD criterion can be implemented by the following:

- A set of reified constraints: for all $1 \leq i < j \leq n$,
$$S_{ij} = (G[i] == G[j])$$
S_{ij} is a boolean variable, which is equal to 1 iff $G[i] = G[j]$.
- A linear sum constraint:
$$V = \sum_{1 \leq i < j \leq n} S_{ij} d(i,j)^2$$

However, these constraints, while considered independently, do not offer enough propagation. For example, with $k = 2$, given 4 points from 1 to 4 and a partial assignment where $G[1] = 1$ and $G[2] = G[3] = 2$ as in Figure 2 (the number on each edge $\{i, j\}$ represents the value $d(i,j)^2$). We have three instantiated boolean variables: $S_{12} = S_{13} = 0, \;\; S_{23} = 1$ and three uninstanciated variables $S_{14}, S_{24}, S_{34} \in \{0, 1\}$. Let us assume that a solution with $V = 5$ was found. With the branch-and-bound search, this solution sets the upper bound of variable V to 5. A new constraint is added:

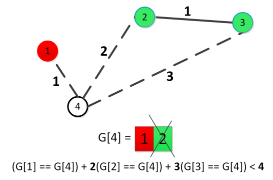

Figure 2. Example of filtering

$$\sum_{i<j} S_{ij} d(i,j)^2 < 5.$$

As $S_{12} = S_{13} = 0$, $S_{23} = 1$, this constraint becomes:

$$S_{14} + 2S_{24} + 3S_{34} < 4.$$

We can see that S_{24} and S_{34} must be equal since $G[2] = G[3]$ and then must not be equal to 1, otherwise the constraint is violated. We should then infer that point 4 cannot be in the same cluster as points 2 and 3, that means value 2 should be removed from the domain of $G[4]$. This filtering however is not done, since the constraints are considered independently.

Several recent works have proposed more efficient filtering for the sum constraint, when it is considered with other constraints. For a sum constraint $y = \sum x_i$ with inequality constraints $x_j - x_i \leq c$, a domain filtering algorithm reduces the domain of x_i when new bounds for y are known [18]. A bound-consistency algorithm is proposed for a sum constraint with increasing order constraints $x_i \leq x_{i+1}$[19] or with a constraint *alldifferent*$(x_1, \dots, x_n)$[20]. These cases however do not fit the WCSD criterion constraint (2) . A generic bound-consistency algorithm for a sum constraint with a set of constraints is proposed in [21]. In our case, the domain of $G[i]$ is a set of representative indices, which is not an interval in general, and where we wish to remove inconsistent values.

We have therefore developed a filtering algorithm for a new global constraint on a variable V, an array of variables G of size n and an array of constants a (where $a_{ij} = a_{ji}$ for $i, j \in [1, n]$), which is of the form:

$$V = \sum_{1 \leq i < j \leq n} (G[i] {==} G[j]) a_{ij}. \quad (3)$$

Taking into account the partitioning problem, the domain of each variable $G[i]$ is a set of the representative indices of all clusters, into which point i can be assigned. Let us assume that the domain of variable V is $[V.lb, V.ub)$ where $V.lb$ is the lower bound, which can initially be 0, and $V.ub$ is the upper bound, which can be the value of V in the last solution with a branch-and-bound search. Suppose that we have a partial assignment of variables in G, where there is at least one point assigned for each group (*e.g* the representative of the group, cf. sub-section III-B). Let K

be the set of points i which have been already assigned to a group ($G[i]$ is instanciated) and U the set of the unassigned points. The sum in (3) is split into three parts $V = V_1 + V_2 + V_3$, where:

- V_1 is the sum of dissimilarities between the assigned points:
$$V_1 = \sum_{i,j \in K, i<j} (G[i]{==}G[j]) a_{ij}$$
- V_2 is the sum of dissimilarities between the unassigned points and the assigned points:
$$V_2 = \sum_{i \in U, j \in K} (G[i]{==}G[j]) a_{ij}$$
- V_3 is the sum of dissimilarities between the unassigned points:
$$V_3 = \sum_{i<j, i,j \in U} (G[i]{==}G[j]) a_{ij}$$

The value of V_1 can be calculated exactly because the set K is already known. For the second part, the value of V_2 is unknown because of the unassigned points. However, a lower bound of V_2, denoted by $V_2.lb$, can be calculated by a sum of minimum contribution of all unassigned points. Since each unassigned point i will be assigned to a group, it will contribute to that group a sum of dissimilarities between point i and all the points of K that are in that group. The minimal contribution v_{2i} of the point i is the minimal added amount when considering all k groups, with respect to the assigned points:
$$v_{2i} = \min_{c \in [1,k]} \Big(\sum_{j \in K \cap C_c} a_{ij} \Big).$$
A lower bound of V_2 is then the sum of v_{2i}:
$$V_2.lb = \sum_{i \in U} v_{2i}.$$
For the third part, the value of V_3 is unknown too and we propose a heuristic to calculate a lower bound of V_3. We recall that V_3 is the sum of all a_{ij} with i and j in the same group. Let $p = |U|$, the minimal number of terms a_{ij} in the sum V_3 is the minimal number of within-group pairwise connections[2], while considering all partitions of p points into k groups. For example, with $p = 10$, $k = 3$ and with a partition into 3 groups of sizes 2, 3 and 5, the number of within-group pairwise connections is 14. The minimal value of this number is 12, corresponding to a partition into 3 groups of sizes 3, 3 and 4.

Let m be the quotient of the division of p by k and m' the remainder. Let the number of points in each group c be $m + \alpha_c$, with $\alpha_c < 0$ when the group c has less than m points, $\alpha_c \geq 0$ otherwise. We have then $\sum_{1 \leq c \leq k} (m + \alpha_c) = p = km + m'$, so $m' = \sum_{1 \leq c \leq k} \alpha_c$. The number of pairwise connections in a group c is $(m + \alpha_c)(m + \alpha_c - 1)/2$. The total number for all groups is:
$$\begin{aligned} & \textstyle\sum_{1 \leq c \leq k} (m + \alpha_c)(m + \alpha_c - 1)/2 \\ = \; & (\textstyle\sum_{1 \leq c \leq k} (m + \alpha_c)^2 - \sum_{1 \leq c \leq k} (m + \alpha_c))/2 \\ = \; & (km^2 + 2mm' + \textstyle\sum_{1 \leq c \leq k} \alpha_c^2 - km - m')/2 \end{aligned}$$
Since $m' = \sum_{1 \leq c \leq k} \alpha_c$, we have $m' \leq \sum_{1 \leq c \leq k} |\alpha_c| \leq \sum_{1 \leq c \leq k} \alpha_c^2$ (α_c are integers). Therefore the total number for all groups is greater or equal to $(km^2 + 2mm' - km)/2$, denoted by $f(p)$. The equality is reached when α_c is 1 for m' groups and is 0 for $k - m'$ groups. With a set U of unassigned points, with the constants a_{ij} ($i, j \in U$) ordered increasingly, a lower bound V_3, denoted by $V_3.lb$, is then calculated by the sum of the $f(|U|)$ first constants in this order.

An example is given in Figure 3 with 8 points to 2 groups (green and red). Suppose that there are 5 assigned points (3 red points and 2 green points) and 3 unassigned points. The value of V_1 is calculated exactly by the sum of solid black lines. The lower bound $V_2.lb$ is the sum of dash red lines and dash green lines. With 3 unassigned points, we have $p = 3, k = 2, m = 1$ and $m' = 1$, the minimum total number of connections is $f(3) = (km^2 + 2mm' - km)/2 = 1$. Therefore, the lower bound $V_3.lb$ is the dot blue line.

A lower bound of variable V is given by:
$$V.lb = V_1 + V_2.lb + V_3.lb$$
This lower bound is used for two purposes:

- Detecting the failure during the branch-and-bound search, it happens when $V.lb \geq V.ub$.
- Filtering inconsistent values of unassigned variables. For each value a of an unassigned variable $G[i]$, a new lower bound, denoted by $V'.lb$, will be calculated with the assumption $G[i] = a$. This value is inconsistent if $V'.lb \geq V.ub$. For example in Figure 2, value 2 can be removed from the domain of variable $G[4]$ because with the assumption $G[4] = 2$, the new lower bound $V'.lb = 6$, which is greater than the upper bound $V.ub = 5$.

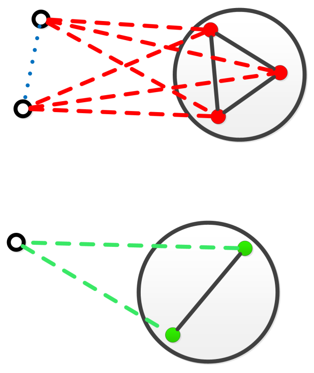
Figure 3. Example of $V.lb = V_1 + V_2.lb + V_3.lb$

[2]A group is like a clique and the number of pairwise connections is the number of edges in the clique.

The filtering algorithm is presented in Algorithm 1. This algorithm uses arrays add and min, where $add[i, c]$ is the added amount if i is assigned to group c ($add[i, c] = \sum_{j \in K \cap C_c} a_{ij}$) and $m[i]$ is the minimal added amount while considering all possible assignments for i ($m[i] = \min_c add[i, c]$). Since the constants a_{ij} must be ordered increasingly in the computation of $V_3.lb$, they are ordered once in the array ord, so $ord[pos]$ gives the constant a_{ij} in the order at position pos, and $px[pos]$ ($py[pos]$) gives the index i (j, resp.) of the constant. For the time being, the filtering algorithm is developed for the clustering task, where values in the domain of $G[i]$ are the representatives of all clusters for which point i can be assigned. Given an index a, the function $gr(a)$ gives the index of the cluster corresponding to a.

Algorithm 1: Filtering algorithm

```
1  $V_1 \leftarrow 0$; $V_2.lb \leftarrow 0$; $V_3.lb \leftarrow 0$; $V_4 \leftarrow 0$;
2  for $i \leftarrow 1$ to $n$ where $G[i]$ is instanciated do
3      for $j \leftarrow 1$ to $n$ do
4          if $G[j]$ is instanciated and $G[j] == G[i]$ and $i < j$ then $V_1 \leftarrow V_1 + a_{ij}$
5          if $G[j]$ is not instanciated then
           $add[j, gr(G[i])] \leftarrow add[j, gr(G[i])] + a_{ij}$
6  for $i \leftarrow 1$ to $n$ where $G[i]$ is not instanciated do
7      $m[i] \leftarrow \infty$;
8      foreach value $a \in Dom(G[i])$ do
9          if $m[i] > add[i, gr(a)]$ then
           $m[i] \leftarrow add[i, gr(a)]$
10     $V_2.lb \leftarrow V_2.lb + m[i]$;
11 $p \leftarrow$ number of uninstanciated variables in $G$;
12 $cpt \leftarrow 0$; $pos \leftarrow 1$;
13 while $cpt < f(p)$ do
14     $i \leftarrow px[pos]$; $j \leftarrow py[pos]$;
15     if $G[i]$ is not instanciated and $G[j]$ is not instanciated then
16         $cpt \leftarrow cpt + 1$;
17         $V_3.lb \leftarrow V_3.lb + ord[pos]$;
18         if $cpt \leq f(p-1)$ then $V_4 \leftarrow V_4 + ord[pos]$
19     $pos \leftarrow pos + 1$;
20 $V.lb \leftarrow \max(V.lb, V_1 + V_2.lb + V_3.lb)$;
21 for $i \leftarrow 1$ to $n$ where $G[i]$ is not instanciated do
22     foreach value $a \in Dom(G[i])$ do
23         if $V.lb + add[i, gr(a)] - m[i] - V_3.lb + V_4 \geq V.ub$ then
24             delete $a$ from $Dom(G[i])$;
```

The lower bound of V is revised in line 20. Lines 21 to 24 filter the domain of $G[i]$ ($i \in U$): for each value a in the domain, in case of assignment of i into group $gr(a)$, a new lower bound for $V.lb$ is $V'.lb = V'_1 + V'_2.lb + V'_3.lb$ with:

- $V'_1 = V_1 + add[i, gr(a)]$ because point i is supposed to be assigned to group $gr(a)$, the sum of dissimilarity between instantiated points increases a value of $add[i, gr(a)]$.
- $V'_2 = V_2.lb - m[i]$ because point i is no more unassigned, the contribution of point i in the calculation of $V_2.lb$ must be substracted.
- $V'_3.lb$ is the sum of the first $f(|U| - 1)$ elements of ord that are related to $U \setminus \{i\}$. In order to reduce the complexity of the filtering algorithm, we actually use V_4 instead of $V'_3.lb$. Here, V_4 is the sum of the first $f(|U| - 1)$ elements of ord (possibly some related to i). It is evidence that $V'_3.lb \geq V_4$.

The new lower bound is:

$$(V_1 + add[i, gr(a)]) + (V_2.lb - m[i]) + V'_3.lb$$

which is greater or equal to:

$$V.lb + add[i, gr(a)] - m[i] - V3.lb + V_4$$

So if this last value is greater than the actual upper bound of V, a is inconsistent.

The complexity of this algorithm is $O(n^2 + nk)$, since the domain of each $G[i]$ is of size at most k. Since $k \leq n$, the complexity is then $O(n^2)$.

V. Experiments

Experiments are performed on a PC Intel core i5 with 3.8 GHz and 8 GB of RAM. The model is applied to dataset Iris from UCI repository [22]. This dataset contains 150 samples of three species of iris flowers with 4 attributes. Our model is implemented with the Gecode 4.0.0 library[3]. In this newest version of Gecode released in 2013, float variable is supported. This property is important for our model to obtain exact optimal value.

Interest of filtering: Without the filtering algorithm, the propagation of the default linear sum constraint $V = \sum_{1 \leq i < j \leq n} (G[i] == G[j]) d(i, j)^2$ is weak because the propagation does not consider the relationship between variables $G[i]$ as mentioned before. In consequence, the model without filtering hardly solve dataset with more than 50 samples. Our filtering algorithm takes benefit from both assigned an unassigned points to have a better lower bound and a better filtering.

Table I shows the difference of performance of our model in two cases: with and without the filtering. In each case, the first column gives the number of nodes in the search tree, whereas the second column reports the total CPU time in seconds. The number of samples varies from $n = 20$ to $n = 45$ and the number of classes k is set to 3.

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

Table I
PERFORMANCE OF FILTERING ALGORITHM

n	without filtering		with filtering	
	#nodes	time	#nodes	time
20	667	0.004	375	0.002
25	2887	0.03	599	0.004
30	17183	0.2	867	0.01
35	47901	0.8	1207	0.02
40	1362113	29.7	1663	0.04
45	5687055	145.8	2071	0.06

Table II
SEPARATION CONSTRAINT WITH IRIS

Separation Constraint	WCSD	Total time
no constraint	573.552	4174s
δ = 2% $maxD$	573.552	1452s
δ = 4% $maxD$	573.552	84.4s
δ = 6% $maxD$	573.552	0.3s
δ = 8% $maxD$	2169.21	0.1s
δ = 10% $maxD$	2412.43	0.04s

Exact solution for WCSD criterion: There are few works dealing with finding the best optimum, and as far as we know, no work integrates user-constraints. Without user-constraints, our model is compared to the Repetitive Branch-and-Bound Algorithm (RBBA) [8][4], which, to the best of our knowledge, is the best exact algorithm for minimum within-cluster sum of dissimilarities partitioning. The distance between objects is the Euclidean distance and the dissimilarity is measured as the squared Euclidean distance. Without user-constraints, both our model and the RBBA approach can find the optimal solution with dataset Iris. Our model needs 4174s to complete the search whereas RBBA takes 3249s. However, there is no exact algorithm that handles user-constraints, while our model can handle different kinds of user-constraints.

WCSD and separation constraint: Let us add a separation constraint δ (the margin between two clusters must be at least δ), where δ ranges from 0% (no constraint) to 10% of the maximum distance between two objects ($maxD$). Table II reports the WCSD value of an optimal solution with the total time for computation. It shows that when the separation 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. With appropriate user-constraints, users can find meaningful partition while the optimal solution is always guaranteed.

WCSD and must-link constraint: Let us now add must-link (ML) constraints, where the number of ML constraints, generated from the real classes of objects, varies from 0.2 to 1% of the total number of pairs. Results are expressed in Table III, 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 real classes of all objects is used. The more ML constraints are added, the less computation time is needed. The reduction of 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 constraints, the total number of feasible solutions is always equal or less than the case with no constraint.

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

Table III
ML CONSTRAINTS WITH IRIS

# ML constraints	WCSD	Total time
no constraint	573.552	4174s
0.2%	602.551	1275s
0.4%	602.551	35.6s
0.6%	617.012	16.1s
0.8%	622.5	3.5s
1%	622.5	1.6s

Table IV
PROPERTIES OF DATASETS

Dataset	# Objects	# Attributes	# Clusters
Wine	178	13	3
Letter Recognition	600	16	3
Vehicle	846	18	4
GR666	666	2	not available

WCSD and combination of user-constraints: As mentioned above, finding an exact solution for minimizing the WCSD is difficult. However, with appropriate combination of user-constraints, the performance can be boosted. Table IV summarizes information about datasets for this section. The first three datasets are from the UCI repository [22]. 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 dataset GR666 is from the library TSPLIB [23], it contains coordinates of 666 European cities [24]. This dataset does not contain information about the number of clusters k and we choose $k = 3$ for the tests.

Table V presents some examples where our model can find exact solution with different user-constraints which reduce significantly the search space.

Table V
EXAMPLE OF COMBINATIONS OF USER-CONSTRAINTS

Dataset	User-constraints	Total time
Wine	separation: δ = 1.5% $maxD$ minimal capacity: α = 30	11.2s
GR666	separation: δ = 1.5% $maxD$ diameter: γ = 50% $maxD$	12.4s
Letter Recognition	# ML constraints = 0.1% total pairs # CL constraints = 0.1% total pairs separation: δ = 10% $maxD$	11.5s
Vehicle	separation: δ = 3% $maxD$ diameter: γ = 40% $maxD$	1.6s

VI. Conclusion

We have proposed in [1] a CP model for the constrained clustering task with a criterion to optimize. The model guarantees to find a global optimum if there exists one while all user-constraints are satisfied. In this paper, we propose a filtering algorithm for the WCSD criterion constraint. It improves significantly the performance of our model. Without user-constraints, our model is comparable to the best known exact approach for this criterion [8]. With appropriate user-constraints, the model is able to handle larger datasets. We plan to improve the efficiency of our model by working further on search strategies and on constraint propagators, thus being able to address larger datasets. We also work on extending the model so that it will be able to handle constrained clustering tasks where the user does not need to specify exactly the number k of clusters.

Moreover, the filtering algorithm was developed for a partitioning task. We aim to generalize it in order to extend the constraint to a more general form, so that it can be applied to other problems, as for instance the weighted max-cut problem.

References

[1] T. B. H. Dao, K. C. Duong, and C. Vrain, "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] K. Wagstaff and C. Cardie, "Clustering with instance-level constraints," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 1103–1110.

[3] S. Basu, I. Davidson, and K. L. Wagstaff, Eds., *Constrained Clustering: Advances in Algorithms, Theory and Applications*. Chapman & Hall/CRC Press, 2009.

[4] L. De Raedt, T. Guns, and S. Nijssen, "Constraint programming for itemset mining," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 204–212.

[5] L. D. Raedt, T. Guns, and S. Nijssen, "Constraint programming for data mining and machine learning," in *AAAI*, 2010.

[6] P. Boizumault, B. Crémilleux, M. Khiari, S. Loudni, and J.-P. Métivier, "Discovering Knowledge using a Constraint-based Language," *CoRR*, vol. abs/1107.3407, 2011.

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

[8] M. Brusco and S. Stahl, *Branch-and-Bound Applications in Combinatorial Data Analysis (Statistics and Computing)*, 1st ed. Springer, Jul. 2005. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0387250379

[9] D. Aloise, P. Hansen, and L. Liberti, "An improved column generation algorithm for minimum sum-of-squares clustering," *Math. Program.*, vol. 131, no. 1-2, pp. 195–220, 2012.

[10] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrdl, "Constrained k-means clustering with background knowledge," in *ICML*, 2001, pp. 577–584.

[11] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004, pp. 11–18.

[12] I. Davidson and S. S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," *PKDD*, pp. 59–70, 2005.

[13] Z. Lu and M. A. Carreira-Perpinan, "Constrained spectral clustering through affinity propagation," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2008, pp. 1–8.

[14] X. Wang and I. Davidson, "Flexible constrained spectral clustering," in *KDD '10: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 563–572.

[15] J.-P. Métivier, P. Boizumault, B. Crémilleux, M. Khiari, and S. Loudni, "Constrained Clustering Using SAT," in *IDA 2012, LNCS 7619*, 2012, pp. 207–218.

[16] M. Mueller and S. Kramer, "Integer linear programming models for constrained clustering," in *Discovery Science*, 2010, pp. 159–173.

[17] T. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[18] J.-C. Régin and M. Rueher, "Inequality-sum: a global constraint capturing the objective function," *RAIRO - Operations Research*, vol. 39, no. 2, pp. 123–139, 2005.

[19] T. Petit, J.-C. Régin, and N. Beldiceanu, "A $\theta(n)$ bound-consistency algorithm for the increasing sum constraint," in *Principles and Practice of Constraint Programming CP 2011*, 2011, pp. 721–728.

[20] N. Beldiceanu, M. Carlsson, T. Petit, and J.-C. Régin, "An $o(nlogn)$ bound consistency algorithm for the conjunction of an alldifferent and an inequality between a sum of variables and a constant, and its generalization," in *ECAI*, 2012, pp. 145–150.

[21] J.-C. Régin and T. Petit, "The objective sum constraint," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CPAIOR 2011*, 2011, pp. 190–195.

[22] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[23] G. Reinelt, "TSPLIB - A t.s.p. library," Universität Augsburg, Institut für Mathematik, Augsburg, Tech. Rep. 250, 1990.

[24] M. Grötschel and O. Holland, "Solution of large-scale symmetric travelling salesman problems," *Math. Program.*, vol. 51, pp. 141–202, 1991.