Clustering Formulation Using Constraint Optimization

Valerio Grossi[1(✉)], Anna Monreale[1,2], Mirco Nanni[2], Dino Pedreschi[1], and Franco Turini[1]

[1] KDDLab, University of Pisa, Largo B. Pontecorvo, 3, Pisa, Italy
{valerio.grossi,anna.monreale,dino.pedreschi,franco.turini}@di.unipi.it
[2] KDDLab, ISTI-CNR, Via G. Moruzzi, 1, Pisa, Italy
{anna.monreale,mirco.nanni}@isti.cnr.it

Abstract. The problem of clustering a set of data is a textbook machine learning problem, but at the same time, at heart, a typical optimization problem. Given an objective function, such as minimizing the intra-cluster distances or maximizing the inter-cluster distances, the task is to find an assignment of data points to clusters that achieves this objective. In this paper, we present a constraint programming model for a centroid based clustering and one for a density based clustering. In particular, as a key contribution, we show how the expressivity introduced by the formulation of the problem by constraint programming makes the standard problem easy to be extended with other constraints that permit to generate interesting variants of the problem. We show this important aspect in two different ways: first, we show how the formulation of the density-based clustering by constraint programming makes it very similar to the label propagation problem and then, we propose a variant of the standard label propagation approach.

1 Introduction

One of the most important and more diffuse task in data mining and machine learning is clustering. This data mining method partitions a set of data objects into subsets without any supervisory information such as data labels. Each subset is called cluster and contains objects very similar to each other, and dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a clustering. In the literature different clustering algorithms have been proposed suitable to operate on various kind of data objects such as text, multimedia, databases, spatio-temporal databases, networks and so on [8, 15,18,23]. Most of the clustering algorithms can be seen as typical optimization problems since they try to optimize a criterion specifying the clustering quality.

In this paper, we propose a formalization of some clustering problems using the constraint programming (CP) paradigm. In particular, we introduce a CP model for *K-medoids*, a prototype-based clustering, *DBSCAN*, a density-based clustering and *Label Propagation* a community discovery algorithm (i.e., a cluster algorithm working on network data). The advantage that derives from the use

D. Bianculli et al. (Eds.): SEFM 2015 Workshops, LNCS 9509, pp. 93–107, 2015.
DOI: 10.1007/978-3-662-49224-6_9

of this kind models is twofold. On one side, relying on CP paradigm we are able to find clustering representing the optimal solution. This is an important aspect because often imperative algorithms find local optima due to the high complexity of the clustering problems. On the other side, by using these models we can exploit the expressive power of the constraint programming formulation that makes it easy to extend standard clustering problems with new user-specified constraints, that can express some background knowledge about the application domain or the data set, improving the clustering results.

The key contribution of this paper is focused on the fact that by our CP models we show how by slightly changing some constraints in the formulation of the standard problems we can easily obtain new and interesting variants that capture new properties and characteristics of the clusters. In particular, we introduce a new variant of the Label Propagation algorithm and we show that by formulating the DBSCAN as a community discovery problem it become very similar to Label Propagation.

The remaining of the paper is organized as follows. Section 2 discusses the clustering problem and describes the imperative algorithms of *K-medoids*, *DBSCAN* and *Label Propagation*. In Sect. 3, we introduce the constraint programming model for the three clustering algorithms. Section 4 shows the expressive power of CP formulation of the clustering models by introducing some variants of the standard clustering methods. Section 5 discusses the state-of-the-art and, lastly Sect. 6 concludes the paper.

2 Clustering Problem

Clustering is one of the most important and well-known data mining methods. The goal of a cluster algorithm is to group together data objects according to some notion of similarity. Therefore, it assigns data objects to different groups (clusters) so that objects that were assigned to the same groups are more similar to each other than to objects assigned to another clusters.

Clustering algorithms can work on different types of data such as text, multimedia, databases, spatio-temporal databases, networks and so on. The main requirement is to have objects described by a set of features or relationships that the algorithm uses to measure the similarity between objects and determine the object's cluster. In the literature, different algorithms have been proposed and each one is designed to find clusters with specific properties [6,14,15,18]. Other clustering algorithms have been designed to work in particular data having a specific structure such as network data. In this field, the clustering algorithms are commonly called *community discovery algorithms*. The concept of a "community" in a (web, social, or informational) network is intuitively understood as a set of individuals that are very similar, or close, to each other, more than to anybody else outside the community. An exhaustive survey about this kind of algorithms are discussed in [8].

Most of the clustering algorithms can be seen as typical optimization problems because in their search of finding a grouping of data objects, they try to

optimize a criterion, such as minimizing the intra-cluster distances or maximizing the inter-cluster distances.

We formally define a clustering $\mathcal{C}$ as a set of clusters $C_1, \ldots, C_k$ that represent a partition of a set of data points $P = \{p_1, \ldots, p_n\}$ obtained by optimizing a specific criterion; so we have $C \subseteq P$, $\forall C \in \mathcal{C} : C \subseteq D, \bigcup \mathcal{C} = D, \bigcap \mathcal{C} = \emptyset$. Note that we consider non-overlapping clusters. Each point p_i is represented by an m-dimensional vector. In order to determine the similarity between data objects we consider a distance function $d(p_i, p_j)$ between two data objects. The data objects are described by attributes, which may be qualitative or quantitative. In case of quantitative attributes, we consider the euclidean distance between two m-dimensional vectors.

The optimization criterion determines the quality of the clustering. There are many different ways to measure the goodness of a clustering:

Sum of Squared Inter-cluster Distances. Given some distance function $d(\cdot, \cdot)$ over points, e.g., the Euclidean distance, we can measure the sum of squared distances over each cluster as follows: $\sum_{C \in \mathcal{C}} \sum_{i,j \in |C|, i<j} d^2(p_i, p_j)$.

Sum of Squared Error to Centroid. A more common measure is the "error" of each cluster, that is, the distance of each point in the cluster to the mean (centroid) of that cluster.

Considering the *centroid* of a cluster as the mean of the data points that belong to it, i.e., $m_C = \dfrac{\sum_{p \in C} p}{|C|}$ then, the sum of squared error is measured as: $\sum_{C \in \mathcal{C}} \sum_{p \in C} d^2(p, m_C)$.

Sum of Squared Error to Medoids. Instead of using the mean (centroid) of the cluster, we can also use the medoid of the cluster, that is, the point that is most representative of the cluster. Let the medoid of a cluster be the point with smallest average distance to the other points: $m_C = argmin_{y \in C} \frac{\sum_{p \in C} d^2(p,y)}{|C|}$. The sum of squared error to the medoids is then measured as follows: $\sum_{C \in \mathcal{C}} \sum_{p \in C} d^2(p, m_C)$.

Cluster Diameter. Another measure of coherence is to measure the diameter of the largest cluster, where the diameter is the largest distance between any two points of a cluster. This leads to the following measure of maximum cluster diameter: $\max_{C \in \mathcal{C}} \max_{i,j \in |C|, i<j} d(p_i, p_j)$.

Inter-cluster Margin. The margin between two clusters is the minimal distance between any two points that belong to the different clusters. The margin gives an indication of how different the clusters are from each other (e.g. how far apart they are). This can be optimized using the following measure of minimum inter-cluster margin: $\min_{i,j \in |\mathcal{C}|, i<j} \min_{p_1 \in C_i, p_2 \in C_j} d(p_1, p_2)$.

2.1 Prototype-Based Clustering

Among the most studied and applied clustering methods there are the prototype-based ones that require the number of clusters to be known in advance. These

techniques create a partitioning of the data where each cluster (partition) is represented by a prototype called the *centroid* of the cluster. Two popular algorithms employing this approach are *K-means* and *K-medoids*. K-means represents each centroid as the average of all points in the cluster, while K-medoids considers the most representative actual point in the cluster. Both methods are heuristic algorithms, that operate as follows: given a user-specified value k, the algorithm selects k initial centroids. Successively, each point is assigned to the closest centroid based on a distance measure. Finally, the centroids are updated iteratively based on the points assigned to the clusters. This process stops when centroids do not change. The quality of the clustering is computed as the sum of squared distances of each cluster compared to its centroid. The goal is to minimize this value. Therefore, given a set of n points, and a k value, the aim is to find an assignment of points that minimizes the Sum of Squared Error to centroid (SSE):

$$SSE = \sum_{C \in \mathcal{C}} \sum_{p \in C} d^2(p, m_C)$$

where m_C is the centroid of the cluster C (i.e., the mean of the data points that belong to C in K-means while the medoid in K-medoids).

2.2 Density-Based Clustering

The approaches presented in Sect. 2.1 require to know in advance the number of clusters to be found. Moreover, they tend to provide clusters with globular shapes. Unfortunately, in many real applications, we are in the presence of non-globular regions or regions that are quite dense surrounded by areas with low density, typically formed by noise. In this perspective, clusters can also be defined implicitly by the regions of higher data density, separated from each other by regions of lower density. The price for this flexibility is a difficult interpretation of the obtained clusters. One of the most famous algorithm based on the notion of density of regions is DBSCAN [15]. This approach does not rely on an optimization algorithm, it is based on measuring the data density at a certain region of the data space and then, defining clusters as regions that exceed a certain density threshold. The final clusters are obtained by connecting neighboring dense regions. Figure 1 shows an example for the two-dimensional space. Four groups are recognized as clusters and they are separated by an area where the data density is too low. DBSCAN identifies three different classes of points:

Core points. These points are in the interior of a density-based cluster. A point is a core point if the number of points within a given neighborhood around the point as determined by the distance function and a user- specified distance parameter, ϵ, exceeds a certain threshold, $MinPts$, which is also a user-specified parameter.

Border points. These points are not core points, but fall within the neighborhood of a core point. A border point can fall within the neighborhoods of several core points.

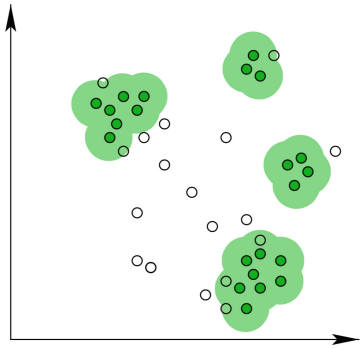

Fig. 1. Example of density-based clusters [5].

Noise points. A noise point is any point that is neither a core point nor a border point.

The DBSCAN algorithm is the following:

1. Label all points as core, border, or noise points.
2. Eliminate noise points.
3. Put an edge between all core points that are within ϵ of each other.
4. Make each group of connected core points into a separate cluster.
5. Assign each border point to one of the clusters of its associated core points.

2.3 Label Propagation

In the network field a task very similar to clustering is *community discovery*, which can be seen as a network variant of standard data clustering. The concept of a "community" in a (web, social, or informational) network is intuitively understood as a set of individuals that are very similar, or close, to each other, more than to anybody else outside the community [8]. This has often been translated in network terms into finding sets of nodes densely connected to each other and sparsely connected with the rest of the network. An interesting community discovery algorithm is the Label Propagation algorithm [23] that detects communities by spreading labels through the edges of the graph and then labeling nodes according to the majority of the labels attached to their neighbors, iterating until a general consensus is reached.

Iterative Label Propagation (LP). Suppose that a node v has neighbors $v_1, v_2, \ldots, v_k$ and that each neighbor carries a label denoting the community that it belongs to. Then, v determines its community based on the labels of its neighbors. [23] assumes that each node in the network chooses to join the community to which the maximum number of its neighbors belong to. As the labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label. At the end of the propagation process, nodes with the same

labels are grouped together as one community. Clearly, a node with an equal maximum number of neighbors in two or more communities will take one of the two labels by a random choice. For clarity, we report here the procedure of the LP algorithm. Note that, in the following $C_v(t)$ denotes the label assigned to the node v at time (or iteration) t.

1. Initialize the labels at all nodes in the network. For any node v, $C_v(0) = v$.
2. Set $t = 1$.
3. Arrange the nodes in the network in a random order and set it to V.
4. For each $v_i \in V$, in the specific order, let $C_{v_i}(t) = f(C_{v_{i1}}(t-1), \ldots, C_{v_{ik}}(t-1)$. f here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.
5. If every node has a label that the maximum number of their neighbors have, or t hits a maximum number of iterations t_{max} then stop the algorithm. Else, set $t = t + 1$ and go to (3).

The drawback of this algorithm is the fact that *ties are broken uniformly randomly*. This random behavior can lead to different results for different executions and some of these results cannot be optimal. In Fig. 2, we show how given the same network as input of LP we obtain four different results.

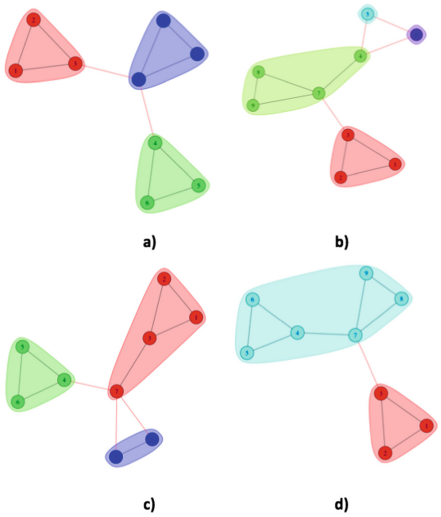

Fig. 2. The result of four executions of LP Algorithm.

2.4 Constraint-Based Clustering

Clustering algorithms are generally used in an unsupervised way, i.e., they do not require any external knowledge to be run. However, sometimes in real application

domains, it is often the case that the data analyst possesses some background knowledge (about the application domain or the data set) that could be useful in clustering the data. Therefore, incorporating user knowledge into algorithms could make them better both in terms of efficiency and quality of results. These constraints help to reduce the task complexity and to find clusters that satisfy user-specified constraints.

The introduction of user-specified constraints in the standard algorithms leads to the study of a new branch of clustering algorithms [3,22,24,27]. The possible constraints that can be specified are *cluster-level constraints*, that define some specific requirements on clusters such as the minimum or the maximum number of elements, and *instance-level constraints*, that define a property of two data object such as the fact that they must be or cannot be in the same cluster [25]. In the following we recall the definition of the most important user constraints:

- *Must-link constraint* requires that two points belong to the same cluster, so given two points p_i and p_j it is expressed by: $\exists C \in \mathcal{C} : \ p_i \in C \wedge p_j \in C$.
- *Cannot-link constraint* requires that two points belong to a different clusters, so given two points p_i and p_j it is expressed by: $\forall C \in \mathcal{C} : \ \neg(p_i \in C \wedge p_j \in C)$.
- *Cluster size constraints* require that the found clusters have a minimum or a maximum numbers of elements. These constraints are respectively expressed by: $\forall C \in \mathcal{C} : |C| \geq \alpha$ and $\forall C \in \mathcal{C} : |C| \leq \alpha$.
- *Cluster diameter constraint* requires that each cluster must have a diameter at most β. In this case, we express this constraint by: $\forall C \in \mathcal{C}, \forall p_i, p_j \in C : d(p_i, p_j) \leq \beta$.
- *Margin constraint* requires that the margin between any two different clusters to be above a certain threshold δ. This constraint is also called δ-constraint and is expressed as follows: $\forall C, C' \in \mathcal{C}, \forall p_i \in C, \ p_j \in C' : \ d(p_i, p_j) \geq \delta$.

3 Modeling Clustering by Constraint Programming

This paper proposes a constraint programming formulation of some of the most famous clustering methods: *K-medoids*, *DBSCAN* and *Label Propagation*. Constraint Programming (CP) is a declarative programming paradigm where the relations between variables are defined in terms of constraints. Constraints do not specify a step or sequence of steps to be executed (as in imperative programming), but rather the properties of a solution to be found. CP is a powerful paradigm in solving combinatorial search problems. A Constraint Satisfaction Problem (CSP) is a triple (V, D, Y) composed of a set of variables V, a set of domains D and a set of constraints Y over the variables V. Sometimes a CSP can be associated to an objective function to be optimized to find a solution. In this case we have a Constraint Optimization Problem that is a quadruple (V, D, Y, f), where we have also the objective function f that must be minimized or maximized.

The clustering problems, described as an optimization problem, must optimize one of the criterion specified in Sect. 2 representing the quality of the clustering.

In this paper, we introduce the formalization of some clustering methods as optimization problems by using constraint programming formulation. We denote by A the matrix representing the possible clusters where $a_{i,j} = 1$ if data point p_i belongs to the cluster j and $a_{i,j} = 0$ otherwise.

Clearly, we re-define all the user-specified constraints described above by using the constraint programming formulation. Therefore, we have:

- *Must-link constraint* between two points p_i and p_j can be expressed by: $\exists t \in \mathcal{C} : \ a_{i,t} + a_{j,t} = 2$.
- *Cannot-link constraint* between two points p_i and p_j can be expressed by: $\forall t \in \mathcal{C} : a_{i,t} + a_{j,t} \leq 1$.
- *Cluster size constraints* can be expressed as follows: $\forall t \in \mathcal{C} : \sum_{\forall i} a_{i,t} \geq \alpha$ and $\forall t \in \mathcal{C} : \sum_{\forall i} a_{i,t} \leq \alpha$.
- *Cluster diameter constraint*, given a threshold β, can be formulate as: $\forall p_i, p_j.\ i < j$ and $d(p_i, p_j) \geq \beta \rightarrow \forall t \in \mathcal{C}\ a_{i,t} + a_{j,t} = 2$.
- *Margin constraint*, given a threshold δ, can be expressed as: $\forall p_i, p_j.\ i < j$ and $d(p_i, p_j) \geq \delta \rightarrow \forall t \in \mathcal{C}\ a_{i,t} + a_{j,t} \leq 1$.

3.1 CP Model for K-medoid Clustering

In this section, we provide a constraint formalization of the K-medoid clustering starting from the definition of SSE to medoids introduced in Sect. 2. In this context, given a set of points and a parameter K specifying the number of clusters to find, the aim is to find an assignment of points as well as medoids such that the sum of squared error to medoids is minimized. We recall that the problem of finding clusters providing an optimal solution considering Euclidean metric measure for a general value of K, e.g. minimizing SSE, is a NP Hard problem [17].

The proposed model assumes that both the *assignment of points to clusters* and *the actual medoids* are discovered during solution search. More precisely we do not explicitly constrain the medoid according to its cluster and inversely, we do not impose any constraints among the points. As shown in Table 1, we introduce all the variables representing point membership (4) and medoids (5). The matrix A stores the clustering partition. In particular, the final model is represented by an assignment of the points to the clusters. Moreover, the model provides the selection of most representative points as medoids in $m_{i,j}$. The K-medoid algorithm is based on euclidean distance computed by the function $d(i, j)$.

Our formalization defines the optimization function, as proposed in Sect. 2, i.e. it sums for all points $i \in P$ and cluster $C_j \in \mathcal{C}$, the squared distance between i and the medoid of j. Furthermore, two additional constraints are required to enforce that a point can only belong to one cluster (7) implying that $C_i \cap C_j = \emptyset$, $\forall C_i, C_j \in \mathcal{C}$, and each cluster can have only one medoid (8).

Table 1. A K-medoid based clustering formulation.

$$\underset{\mathcal{C}}{\text{minimize}} \quad SSE(\mathcal{C}), \tag{1}$$

s.t.

$$P \text{ is the set of points} \tag{2}$$

$$K \text{ number of required clusters} \tag{3}$$

$$a_{i,C_j} \in \{0,1\} : a_{i,C_j} = 1 \text{ iff point } i \in P \text{ is assigned to cluster } C_j \in \mathcal{C} \tag{4}$$

$$m_{i,C_j} \in \{0,1\} : m_{i,C_j} = 1 \text{ iff point } i \in P \text{ is the medoid of cluster } C_j \in \mathcal{C} \tag{5}$$

$$SSE(C_j \in \mathcal{C}) = \sum_{i \in P} a_{i,C_j} \sum_{h \in P \& h \neq i} m_{h,C_j}\, d^2(i,h) \tag{6}$$

$$\sum_{C_j \in \mathcal{C}} a_{i,C_j} = 1 \quad \forall i \in P \tag{7}$$

$$\sum_{i \in P} m_{i,C_j} = 1 \quad \forall C_j \in \mathcal{C} \tag{8}$$

$$|\bigcup_{C \in \mathcal{C}} C| = |P| \tag{9}$$

$$|\mathcal{C}| = K \tag{10}$$

Table 2. DBSCAN formulation.

$$\text{maximize}(\sum_{l_j \in L} min(1, \sum_{i \in P} k_{i,l_j})), \tag{11}$$

s.t.

$$P \text{ is the set of points} \tag{12}$$

$$L = \{l_1, \ldots, l_n, l_{n+1}\} \text{ is an ordered set of colors (or labels)} \tag{13}$$

$$a_{i,j} \in \{0,1\} : a_{i,j} = 1 \text{ iff distance between points } i,j \in P : \ d(i,j) \leq \epsilon \tag{14}$$

$$k_{i,l_j} \in \{0,1\} : k_{i,l_j} = 1 \text{ iff point } i \in P \text{ has color } l_j \in L \tag{15}$$

$$r_i \in \{0,1\} : r_i = 1 \text{ iff point } i \in P \text{ is a core point, i.e., } \sum_{j \in P} a_{i,j} \geq minp \tag{16}$$

$$\sum_{l_j \in L} k_{i,l_j} = 1 \quad \forall i \in P \tag{17}$$

$$r_h = 1 \wedge r_p = 1 \wedge a_{h,p} = 1 \wedge k_{h,l_j} = 1 \Rightarrow k_{p,l_j} = 1 \tag{18}$$

$$r_i = 0 \Rightarrow k_{i,l_j} = 1 \tag{19}$$

$$\text{where } l_j = min\{\{l_j \in L \setminus \{l_{n+1}\} | a_{h,i} = 1 \wedge r_h = 1 \wedge k_{h,l_j} = 1\} \cup \{l_{n+1}\}\}$$

3.2 CP Model for DBSCAN Clustering

Let us now introduce a constraint programming model for the density-based clustering DBSCAN. In particular, we reformulate the problem in the context of networks by considering the set of points P as nodes and setting an edge between two nodes i and j, if the distance between i and j is less than a given ϵ. Clearly, in this way we have that the neighbors of a node (point) i are the set of points within a distance ϵ. Our intended objective is to capture the basic idea of that "each node has the *same* label of all its neighbors".

DBSCAN algorithm does not rely on an optimization algorithm however, its constraint programming formulation shows how, re-defining this task as a community discovery problem in a network, this approach becomes very similar to the Label Propagation approach that finds clusters of nodes in networks [23]. Moreover, we show how the expressivity introduced by the formulation of the problem by constraint programming makes the standard problem easy to be extended with other constraints that permit to generate interesting variants of the same problem.

More in detail, the model in Table 2 is described as follows. Variables k_{i,l_j} denotes the color (label l_j) of a point (node i). Variable $a_{i,j}$ indicates the presence or absence of an edge between two nodes i and j. Variables r_i denote whether node i is a core point or not.

We also consider two domains: (a) the set of points P, and (b) the ordered set of colors L. Note that in the set of colors we have the color l_{n+1} that is an additional color used for coloring the noise points. The model imposes that a point has one and only one color, and that all the connected core points must have the same color (18). Another requirement is that each point that is not a core point takes the same color of the core points that are connected to it. If it does not have any core point around, then this point takes the additional color l_{n+1} because it is a noise (19). Such a constraint also captures the special case in which the point i can be connected to more than one core with different colors. In this case, the model assigns to i the color of the core point that in the ordered set $C \setminus \{l_{n+1}\}$ has a lower rank. Finally, the model is intended to maximize the number of different colors. Notice that a solution where all points have distinct colors does not satisfy Constraint (18) because connected points do not have the same color.

3.3 CP Model for Label Propagation

Let us now propose a constraint programming model for the community discovering problem based on label propagation. Our aim is to capture the basic idea that "each node takes the label of the majority of its neighborhood". The model is reported in Table 3 and described as follows. Variables $a_{i,j}$ indicate the presence of an edge between nodes i and j. Variables k_{i,l_j} denote the color (label) l_j of a node i in the network. There are three domains: (a) the set of nodes N, (b) the set of edges E, and (c) an ordered set of colors L. A node can be assigned one and only one color. Variables n_{i,l_h} denote the number of neighbors of node

Table 3. Label Propagation formulation.

$$\text{maximize}(\sum_{l_j \in L} min(1, \sum_{i \in N} k_{i,l_j})), \tag{20}$$

s.t.

$$E \text{ is the set of edges} \tag{21}$$

$$N \text{ is the set of nodes} \tag{22}$$

$$L = \{l_1, \ldots, l_n\} \text{ is an ordered set of colors (or labels)} \tag{23}$$

$$a_{i,j} \in \{0,1\} : a_{i,j} = 1 \text{ iff the edge between nodes } i \text{ and } j, (i,j) \in E \tag{24}$$

$$k_{i,l_j} \in \{0,1\} : k_{i,l_j} = 1 \text{ iff node } i \in N \text{ has color } l_j \in L \tag{25}$$

$$\sum_{l_j \in L} k_{i,l_j} = 1 \quad \forall i \in N \tag{26}$$

$$\sum_{\forall j . a_{i,j}=1} k_{j,l_h} = n_{i,l_h} \quad l_h \in L \tag{27}$$

$$\forall i \in N : k_{i,l_j} = 1 \text{ where } l_j = min\{l_j \in L | max\{n_{i,l_1}, \ldots, n_{i,l_n}\} = n_{i,l_j}\} \tag{28}$$

i with assigned color l_h. The model assigns to the node i the color l_h if it is the most popular among its neighbors, as shown in (28). Such a constraint also captures the case of ties. In such a case, node i is assigned the color that has the lowest rank in the ordered set L. Finally, the model maximizes the number of different colors in the network, as shown in (20).

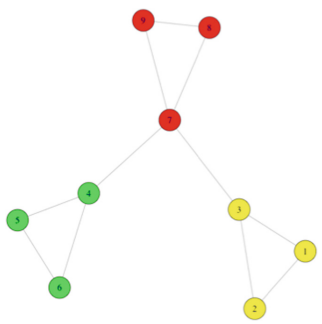

Fig. 3. The result of the execution of CP-LP model.

This model highlights the similarity between Label Propagation and the Density-based clustering problem, and thanks to the constraint programming formulation we can notice that the model for density-based clustering is a variant of the standard label propagation. Indeed, the only difference is due to the fact that the Density-based model requires that "each node has the *same* label

Table 4. Formulation of a variant of standard Label Propagation.

$$\text{maximize}\Big(\sum_{l_j \in L} min(1, \sum_{i \in N} k_{i,c_j})\Big), \tag{29}$$

s.t.

$$E \text{ is the set of edges} \tag{30}$$

$$N \text{ is the set of nodes} \tag{31}$$

$$L = \{l_1, \ldots, l_n\} \text{ is an ordered set of colors (or labels)} \tag{32}$$

$$a_{i,j} \in \{0,1\} : a_{i,j} = 1 \text{ iff the edge between nodes } i \text{ and } j,\ (i,j) \in E \tag{33}$$

$$k_{i,l_j} \in \{0,1\} : k_{i,l_j} = 1 \text{ iff node } i \in N \text{ has color } l_j \in L \tag{34}$$

$$g_i \in \{0,1\} : g_i = 1 \text{ iff the node } i \text{ has degree} \geq \gamma \tag{35}$$

$$\sum_{l_j \in L} k_{i,l_j} = 1 \qquad \forall i \in N \tag{36}$$

$$\sum_{\forall j . a_{i,j}=1} k_{j,l_h} g_j = n_{i,l_h} \qquad l_h \in L \tag{37}$$

$$\forall i \in N : k_{i,l_j} = 1 \text{ where } l_j = min\{l_j \in L | max\{n_{i,l_1}, \ldots, n_{i,l_n}\} = n_{i,l_j}\} \tag{38}$$

of all its neighbors", and not the *most frequent* label. Constraints (18) and (19) in Table 2 and Constraint (28) in Table 3 express this difference. By executing our model we obtain the optimal solution depicted in Fig. 3, where we consider as input the same network in Fig. 2.

4 Variants of the Standard Clustering Algorithms

The expressivity introduced by the formulation of the problems by constraint programming makes the standard methods easy to be extended with other constraints that permit to generate interesting variants.

For each formulation of the clustering problems introduced above, we could easily specify some constraints on the cluster size, or on the maximum diameter of the clusters and so on, by adding one of the user-specified constraints defined in Sect. 3. However, we can also extend the problems by changing one of the constraints of the standard formulation. We extended the standard Label propagation problem as follows.

Given the neighborhood of a node, we give to each node an importance on the basis of its degree; more specifically, in order to compute the most frequent label of its neighborhood the nodes with a degree less than a specific threshold γ are not considered. This means that the propagation of labels is guaranteed by *important* nodes (i.e., nodes with a minimum degree equal to γ), while the nodes with a low degree takes a label in a passive way. We recall that the degree of a node i is the number of nodes that are directly connected to i.

As reported in Table 4, the model presented in Sect. 3.3 can be easily transformed in order to consider this new constraint. We can simply add a variable g_i that is equal to 0 if the node i has a degree less then γ (Constraint (35)); then, the computation of the neighborhood takes into consideration also this variable to filter out the nodes with low degree (37).

5 Related Work

Initial approaches in constrained clustering focused on the introduction of instance-level constraints, especially *must-* and *cannot*-link [26,27]. Several properties are related to instance-level constraints [11]. Furthermore, [10,12,13] define δ and ϵ-constraint forcing clustering with specific spatial properties among items. COP-KMeans [27] represents a popular approach for the integration of instance level constraints in the classical K-means approach. The algorithm takes as input the data, the number of cluster required, a set of must-link, and a set of cannot-link. COP-KMeans ensures that when the clustering is updated none of the specified constraints is violated, performing a hard constraint satisfaction. [22] proposes a modification of COP-KMeans introducing an ensemble approach for managing constraint priorities in order to provide a model even though all the constraints cannot be satisfied. Similarly, PC-KMeans [3] permits that some constraints can be violated enabling a soft constraint satisfaction. In this case every must- and cannot-link has a weight w associated to it. By setting the value w, the algorithm looks for a trade-off between minimizing the total distance between points and cluster centroids, and the cost of violating the constraints. Other approaches do not require that any constraint must be satisfied but they perform metric learning from constraints or adopt an hybrid solution including both constraints satisfaction and metric learning [4]. The set of constraints is mapped into distance measures [2]. The basic idea of these approaches is to weight the features differently, based on their importance in the computation of the distance measure based on the available constraints [7,28]. E.g. [28] parametrizes the Euclidean distance using a symmetric positive-definite matrix, simultaneously minimizing the distance between must-linked instances and maximizing the distance between cannot-linked instances. A probabilistic model based on Hidden Markov Random Fields (HMRF) can be used for incorporating constraints into prototype-based clustering. In [4] the objective function is derived from the posterior energy of the HMRF framework, and the authors propose a EM-based clustering algorithm (HMRF-KMeans) for finding (local) minimum of this objective function. The distance measure is estimated during clustering to validate the user-specified constraints as well as to incorporate data variance. MPCK-Means [7] learns an individual metric for each cluster allowing violations by imposing penalties. Recently, [16,21] proposed the use of constraint programming for modeling data mining tasks. In particular, [16] addressed the problem of searching frequent k-patterns, that cover the whole dataset, while [21] proposed a CP formulation of the constraint-based sequence mining task, that has the goal to find sequences of symbols in a large number of input sequences and that

satisfies some constraints specified by the user. In [20], instead authors proposed an approach based on Integer Linear Programming where the model, knowing a set of candidate clusters, searches for the best clustering among the subset of clusters. Integer Linear programming is also used in [1], where a constraint-based approach for K-means is investigated by using column generation; this work is based on [19]. In [9] authors proposed a declarative and generic framework, based on CP, which enables to design clustering tasks by specifying an optimization criterion and some constraints either on the clusters or on pairs of objects.

6 Conclusion

In this paper, we have introduced a CP model for a medoid based clustering. Moreover, we have proposed a model for a density based clustering and one for a community discovery problem corresponding to a clustering algorithm in network data. We showed the powerful of the expressivity introduced by the constraint programming formulation that enables an easy integration of the standard clustering problems with other constraints. We presented this aspect in two different ways: first, we showed how the formulation of the density-based clustering by constraint programming makes it very similar to the label propagation problem and then, we proposes a variant of the standard label propagation approach.

Acknowledgements. This work was supported by the European Commission under the project Inductive Constraint Programming (ICON) contract number FP7-284715.

References

1. Babaki, B., Guns, T., Nijssen, S.: Constrained clustering using column generation. In: Simonis, H. (ed.) CPAIOR 2014. LNCS, vol. 8451, pp. 438–454. Springer, Heidelberg (2014)
2. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance functions using equivalence relations. In: ICML, pp. 11–18 (2003)
3. Basu, S., Banerjee, A., Mooney, R.J.: Active semi-supervision for pairwise constrained clustering. In: SDM (2004)
4. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In: KDD, pp. 59–68 (2004)
5. Berthold, M.R., Borgelt, C., Hppner, F., Klawonn, F.: Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data, 1st edn. Springer, London (2010)
6. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell (1981)
7. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML, ACM (2004)
8. Coscia, M., Giannotti, F., Pedreschi, D.: A classification for community discovery methods in complex networks. Stat. Anal. Data Min. **4**(5), 512–546 (2011)
9. Dao, T.-B.-H., Duong, K.-C., Vrain, C.: A declarative framework for constrained clustering. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part III. LNCS, vol. 8190, pp. 419–434. Springer, Heidelberg (2013)

10. Davidson, I., Ravi, S.S.: Clustering with constraints: feasibility issues and the k-means algorithm. In: SDM (2005)
11. Davidson, I., Ravi, S.S.: Identifying and generating easy sets of constraints for clustering. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI), pp. 336–341 (2006)
12. Davidson, I., Ravi, S.S.: The complexity of non-hierarchical clustering with instance and cluster level constraints. DMKD **14**(1), 25–61 (2007)
13. Davidson, I., Ravi, S.S.: Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. Data Min. Knowl. Discov. **18**(2), 257–282 (2009)
14. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J. Cybern. **3**(3), 32–57 (1974)
15. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) KDD, pp. 226–231. AAAI Press (1996)
16. Guns, T., Nijssen, S., Raedt, L.D.: k-pattern set mining under constraints. IEEE Trans. Knowl. Data Eng. **25**(2), 402–418 (2013)
17. Hansen, P., Aloise, D.: A survey on exact methods for minimum sum-of-squares clustering. http://www.math.iit.edu/Buck65files/msscStLouis.pdf, pp. 1–2, January 2009
18. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: a k-means clustering algorithm. J. Roy. Stat. Soc. Ser. C (Appl. Stat.) **28**(1), 100–108 (1979)
19. Merle, O.D., Hansen, P., Jaumard, B., Mladenović, N.: An interior point algorithm for minimum sum of squares clustering. SIAM J. Sci. Comput. **21**, 1485–1505 (1997)
20. Mueller, M., Kramer, S.: Integer linear programming models for constrained clustering. In: Pfahringer, B., Holmes, G., Hoffmann, A. (eds.) DS 2010. LNCS, vol. 6332, pp. 159–173. Springer, Heidelberg (2010)
21. Negrevergne, B., Guns, T.: Constraint-based sequence mining using constraint programming. In: Michel, L. (ed.) CPAIOR 2015. LNCS, vol. 9075, pp. 288–305. Springer, Heidelberg (2015)
22. Okabe, M., Yamada, S.: Clustering by learning constraints priorities. In: ICDM, pp. 1050–1055 (2012)
23. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E **76**(2), 036106+ (2007)
24. Ruiz, C., Spiliopoulou, M., Menasalvas, E.: C-DBSCAN: density-based clustering with constraints. In: An, A., Stefanowski, J., Ramanna, S., Butz, C.J., Pedrycz, W., Wang, G. (eds.) RSFDGrC 2007. LNCS (LNAI), vol. 4482, pp. 216–223. Springer, Heidelberg (2007)
25. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: ICML, pp. 1103–1110 (2000)
26. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: AAAI/IAAI, p. 1097 (2000)
27. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained k-means clustering with background knowledge. In: ICML, pp. 577–584 (2001)
28. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems, vol. 15, pp. 505–512. MIT Press (2002)