A Graph-based Approach for Dynamic Clustering

Haytham Elghazel, Hamamache Kheddouci, Véronique Deslandres and Alain Dussauchoy [1]
[1] LIESP (ex. PRISMa) Laboratory, Claude Bernard University of Lyon I, France
{elghazel,hkheddou,deslandres,dussauchoy}@bat710.univ-lyon1.fr

1 Introduction

Clustering algorithms consist in automatically discovering structure of large data set or providing coherent groups of objects independent of any user-defined classes. They are used in many domains: astronomy, information retrieval, image segmentation, biological applications and so on. Several clustering techniques have been proposed [1], there are either hierarchical techniques or partitioning techniques.

We have recently proposed a new partitioning clustering technique based on the *b-coloring* of graph [2]. This technique consists in coloring vertices with the maximum number of colors such that (i) no two adjacent vertices (vertices joined by an weighted edge representing the *dissimilarity* between objects) have the same color (*proper coloring*), and (ii) for each color c, there exist at least one vertex with this color which is adjacent (has a sufficient dissimilarity degree) to all other colors. This vertex is called *dominating vertex*, there can have many within the same class. This specific vertex reflects the properties of the class and also guarantees that the class has a distinct separation from all other classes of the partitioning.

The *b-coloring based clustering method* in [2] enables to build a fine partition of the data set (*numeric* or *symbolic*) in clusters when the number of clusters is not specified in advance.

Motivated by applications such as Web content management and information retrieval requiring a high rate of data set update, the current paper deals with the problem of *dynamic clustering*. It consists in updating clusters without having to frequently performing complete re-clustering. In the following, an *online* (or *incremental*) algorithm is proposed for the *b-coloring* technique presented in [2]. It relies solely on the knowledge of the *dissimilarity matrix* between instances and the *cluster dominating vertices*. The difference between this *learning b-coloring approach* and the traditional one [2] in particular is the ability to process new data as they are added to the data collection, eventually with an updating of existing clusters.

2 Dynamic *b-coloring* based Clustering Algorithm

Before introducing the *online* algorithm, we propose to briefly describe first the *b-coloring* based clustering method [2].

Consider the data set $\boldsymbol{X} = \{x_1,...,x_n\}$ to be clustered as an undirected complete edge-weighted graph $G = (\boldsymbol{V}, \boldsymbol{E})$ where $\boldsymbol{V} = \{v_1,...,v_n\}$ is the vertex set and $\boldsymbol{E} = \boldsymbol{V} \times \boldsymbol{V}$ is the edge set. Vertices in G correspond to instances (vertex v_i for instances x_i) and edges are weighted by the dissimilarity degree between pair of instances. It must be noticed that the possibility of a complete graph would not be interested for clustering problem because in such a case, the *b-coloring* algorithm would provide the trivial partition where each cluster is a singleton. Hence, our algorithm starts from a subgraph (non complete graph) from the original graph. The subgraph is a *superior threshold graph* which is commonly used in graph theory. Let $G_{>\theta}=(V,E_{>\theta})$ be the superior threshold graph associated with threshold value θ chosen among the dissimilarity matrix D. In other words, $G_{>\theta}$ is given by $\boldsymbol{V}=\{v_1,...,v_n\}$ as vertex set and $\{(v_i,v_j)|\ D(v_i,v_j) >\theta\}$ as edge set.

The data to be clustered are now depicted by a *non-complete edge-weighted graph* $G_{>\theta}=(\boldsymbol{V},\boldsymbol{E}_{>\theta})$. In order to divide the vertex set $\boldsymbol{V}$ into a partition $\boldsymbol{P}=\{C_1,C_2,..,C_k\}$ where for $\forall\ C_i,C_j \in \boldsymbol{P},\ C_i \cap C_j=\varnothing$ *for* $i\neq j$ (when the number of clusters k is not pre-defined), our *b-coloring based clustering algorithm* performed on the graph G consists of two steps: 1) generate an initial coloring of vertices using a maximum number of colors, and 2) removing colors without any dominating vertex using a *greedy procedure*.

The clustering algorithm is iterative and performs multiple runs, each of them increasing the value of the dissimilarity threshold θ. Once all threshold values passed, the algorithm provides the optimal partitioning (corresponding to one threshold value θ_o) which maximizes *Dunn's generalized index* ($Dunn_G$) [3]. $Dunn_G$ is designed to offer a compromise between the *intercluster separation* and the *intracluster cohesion*. So, it is the more appropriated to partition data set in *compact* and *well-separated* clusters.

The remaining of this section is devoted to the *dynamic clustering method*. The basic idea is, once the best partition (associated to the optimal threshold θ_o) returned from the *b-coloring based-clustering algorithm*, working to assign new instances to their respective clusters as they arrive. In the sequel, let suppose the data set $\boldsymbol{X}=\{x_1,...,x_n\}$ depicted by the *optimal threshold graph* $G=(\boldsymbol{V},\boldsymbol{E})$ and divided into $\boldsymbol{P}=\{C_1,C_2,..,C_k\}$ The adding of new instance x_{n+1} transforms the vertex set $\boldsymbol{V}$ on $\boldsymbol{V}\cup\{\ v_{n+1}\}$ and the edge set $\boldsymbol{E}$ on $\boldsymbol{E}\cup\{(v_i,v_{n+1})|\ v_i\in\boldsymbol{V}$ and $D(v_i,v_{n+1}) >\theta_o\}$. The main problem is to find the appropriate color to assign for v_{n+1} which is constrained to incrementally maintain the *b-coloring* of G and the clustering performances in terms of quality ($Dunn_G$ *value*) and runtime. Therefore, the incremental clustering problem is defined as follows: as each input instance is presented, either it is assigned to one of the current k clusters (or colors), or it starts off a new cluster.

Assuming that the vertices of G are colored, the following notations will be used:

- Δ: the maximum degree of G after the insertion of the new vertex v_{n+1}.
- $c(v_i)$: the color (integer value) of the vertex v_i in G.
- $N(v_i)$: the neighborhood of vertex v_i in G.
- $N_c(v_i)$: the neighborhood colors of vertex v_i.
- $Dom(v_i)$: the dominance of v_i. $Dom(v_i)$=1 if v_i is one dominant vertex of $c(v_i)$ and 0 otherwise.
- k: the current number of colors (clusters) in G.

As mentioned above, our *online algorithm* relies only on the knowledge of the dissimilarity matrix and the dominating vertices of each color. Under this hypothesis, the following scenarios are to be considered:

2.1 Scenario 1: v_{n+1} is adjacent to at least one dominating vertex of each color

When the neighborhood of v_{n+1} contains at least one dominating vertex from each k colors, v_{n+1} forms its own color $(k+1)^{th}$. Otherwise, the next *Scenario 2* is performed.

Proposition 1 After the creation of the new $(k+1)^{th}$ color, the coloring of G is a *b-coloring*.

Proof $\forall C_h \in \boldsymbol{P}=\{C_1,C_2,..,C_k\}\ \exists v \in (C_h \cap N(v_{n+1}))$ such that $Dom(v)$=1. Thus, $Dom(v_{n+1})$=1 and the vertex v remains dominating of its color $c(v)$ (*i.e.* $Dom(v)$=1). Consequently, $\forall\ C_h \in \boldsymbol{P}=\{C_1,C_2,..,C_k,C_{k+1}\}\ \exists\ v \in C_h$ such that $Dom(v)$=1: the coloring of G using $k+1$ colors is a *b-coloring*.

In order to improve the quality of the new partition $\boldsymbol{P}=\{C_1,C_2,..,C_k,C_{k+1}\}$ in terms of $Dunn_G$ *value*, the color of some vertices can be changed providing that the coloring of G remains a *b-coloring*. For that, the following definitions are introduced:

Definition 1 A vertex v_s is called "*supporting vertex*" if v_s is the only vertex colored with $c(v_s)$ in the neighborhood ($N(v_d)$) of one dominating vertex v_d. Thus, v_s cannot be re-colored.

Definition 2 A vertex v_c is called "*critical vertex*" if v_c is a *dominating* or a *supporting* vertex. Thus, v_c cannot be re-colored.

Definition 3 A vertex v is called "*free vertex regarding a color C*" if v is a non *critical vertex* and C is not in the neighborhood colors of v (*i.e.* $C \notin N_c(v)$). Thus, the color C can be assigned to v.

In order to evaluate the efficiency in the color change for *one free vertex v regarding one color C,* we compute the dissimilarity degree from the vertex v to the color C which is defined as the average dissimilarity from v to all vertices colored with C (eq.(1)). If this latter is lower to the dissimilarity degree from v to its current color, the color C is assigned to v.

$$d(v,C)=\frac{1}{|C|}\sum_{y\in C} D(v,y) \tag{1}$$

Due to this re-coloring, the *intraclass dissimilarity* can decrease which can maximally increase $Dunn_G$ by decreasing its numerator.

Suppose that the vertex v was initially assigned to $c(v)$ and re-colored with C. Since, the re-coloring of v causes to change the dissimilarity values $d(v_i, c(v))$ and $d(v_i,C)$ for each vertex v_i of G. Furthermore, although naive calculation of $d(v_i, c(v))$ and $d(v_i,C)$ takes $O(n^2)$, it can also be reduced to $O(n)$ using their old values as defined in the following equations:

$$d^{new}(v_i,c(v))=\frac{|c(v)|d^{old}(v_i,c(v))-D(v_i,v)}{|c(v)|-1} \tag{2}$$

$$d^{new}(v_i,C)=\frac{|C|d^{old}(v_i,C)+D(v_i,v)}{|C|+1} \tag{3}$$

2.2 Scenario 2: neighborhood of v_{n+1} has no dominating vertex of m colors

The neighborhood of v_{n+1} does not contain any dominating vertex from m colors among the k current colors. These colors are called "*available to receive* v_{n+1}". Two cases are then considered:

- ***Scenario 2.1 : m_1 colors ($m_1 \leq m$) are not present in v_{n+1} neighborhood colors***

The neighborhood colors of v_{n+1} does not contain m_1 among the m current colors. This means that there is no significant dissimilarity between vertex v_{n+1} and these m_1 colors. Among m_1 colors, the one having the smaller dissimilarity with v_{n+1} will color it. Otherwise, the *Scenario 2.2* is performed.

- ***Scenario 2.2: v_{n+1} is neighbor to at least one vertex in each m colors.***

Contrary to the previous scenario, v_{n+1} has at least one *non dominating vertex* per color in its neighborhood. We distinguish here the two following complementary sub-cases:

- *Scenario 2.2.1 : number of colors m=1*

If *m=1* that is only one color *C available to receive* v_{n+1}, we assign this color *C* to v_{n+1}. Since this assignment generates a *non proper coloring* of *G* due to the presence of some neighbors of v_{n+1} in *C*, the colors of these vertices must be changed. For each vertex v_i among the latter the transformation is feasible because it is *non dominating*. As our objective is to find a partition such that the sum of vertex dissimilarities within each class is minimized, the color whose dissimilarity with v_i is minimal (eq.(1)) will be selected if there is a choice between many colors for v_i.

Proposition 2 The new coloring given from *Scenario 2.2.1* is a *b-coloring*.

Proof $\forall v_i$ one vertex from *G* such that $c(v_i)=C$ and $v_i \in N(v_{n+1})$ we have $Dom(v_i)=0$. By the *dominance property*, $\exists\ h \in \{1,2,..,k\}$ such that $C_h \neq C$ and $C_h \notin N_c(v_i)$. Therefore, the color C_h will be assigned to v_i which guarantees *proper coloring*. In addition, $\forall\ h \in \{1,2,..,k\}$ such that $C_h \neq C$, $\exists\ v \in (C_h \cap N(v_{n+1}))$ having $Dom(v)=1$. Thus, *v* remains a dominating vertex of its color (*i.e.* $Dom(v)=1$) and likewise for v_{n+1} (*i.e.* $Dom(v_{n+1})=1$ in its color *C*). Consequently, there is at least one dominating vertex for each color ($\forall C_h \in \boldsymbol{P}=\{C_1,C_2,..,C_k\}\ \exists\ v$ such that $c(v)=C_h$ and $Dom(v)=1$): the *dominance property* is satisfied in ***P***. The coloring of *G* is a *b-coloring*.

- *Scenario 2.2.2 : number of colors m>1*

In this case, several colors are *available to receive* v_{n+1} ($m>1$). The following definition of *color transformation* is required:

Definition 4 A color *C* among the *m* candidate colors to receive v_{n+1} is called "*transformation subject*" if its transformation does not violates the *b-coloring* constraints for the (*m-1*) remaining colors. In other words, the color *C* is a *non transformation subject* if it exists at least one color *C'* (among *m*) such that all the neighbors in *C* for the dominating vertices of *C'* are in the neighborhood of v_{n+1}.

This shows that a color can undergo some transformations when a new vertex is presented (exclusion of vertices, change of dominating). Only the colors maintaining the *b-coloring* constraints are transformable. A relevant stage in the incremental approach will consist to identifying the number (m_2 among *m*) of *transformation subject* colors. The following sub-cases are then considered:

- *Scenario 2.2.2.1 : one color as a transformation subject*

In this case, only one color ($m_2=1$) is identified as a *transformation subject*. Therefore, the vertex v_{n+1} is assigned to this color and its transformation is allowed alike the previous *Scenario 2.2.1*.

- *Scenario 2.2.2.2 : m2>1 colors as a transformation subject*

The actual scenario considers the presence of a number m_2 ($1<m_2 \leq m$) *transformation subject* colors. The color whose dissimilarity with v_{n+1} is minimal (eq.(1)) will be selected to receive it. Since the neighbor vertices of v_{n+1} in these m_2 colors must change their colors behind the inclusion of v_{n+1}, these vertices do not contributes to compute the dissimilarity values. Once the color available to receive v_{n+1} being selected, we transform it alike the previous *Scenario 2.2.1*.

- *Scenario 2.2.2.3 : no color as a transformation subject*

If any color is selected as *transformation subject* among the *m* colors, v_{n+1} forms its own color $(k+1)^{th}$ it becomes its dominating vertex (*i.e.* $Dom(v_{n+1})=1$). Due to this transformation, the *m* colors becomes without dominating vertices. Regarding this problem, we define a procedure which tries to find a *b-coloring* of *G* where all colors are *dominating*. The idea is the following: each *non dominating* color *C* among the *m no subject transformation* colors can be changed. In fact, after removing *C* from the graph *G*, for each vertex v_i colored with *C* (i.e. $c(v_i)=C$), a new color is assigned to v_i which is different from those of its neighborhood. As our objective is to find a partition such that the sum of vertex dissimilarities within each class is minimized, the color whose distance with v_i is minimal will be selected if there is a choice between many colors for v_i. Before starting again with another *non dominating* color *C'* the procedure verifies if the remaining colors have now a dominating vertex (in such a case, these colors are identified as a dominating color).

Summary: The online algorithm has two steps: initialization and cluster update. It initially adopts the *b-coloring partition* associated to the optimal dissimilarity threshold θ_o and works to update it. In the initialization step it is better if we have a sample of the data set that is significant overall the feature space as that we can get a significant clustering, but we can work as well with a normal data set. If the data set used for initialization step does not reflect the true clusters structure, the online approach allows an eventual updating of existing clusters by re-coloring certain instances. Due to this re-assignment strategy, the *intraclass dissimilarity*, an increasing monotonous function of threshold θ, can decrease by improving the partition quality and monotonically decreasing the optimal dissimilarity threshold θ_o during the incremental process.

For more improving the partition quality, we propose an additional operation to optimize the groups of existing clusters called *color merging*. Typically, two colors are merged when the dissimilarity between them is below the optimal dissimilarity threshold θ_o. Consequently, the optimal threshold θ_o can increase although the *b-*

coloring constraints are violated. To solve this problem, the procedure used in Scenario 2.2.2.3 to find a *b-coloring* of *G* is applied for every color without dominating vertices.

3 Experimental Results

Experiments have been made using three relevant benchmark data sets chosen from UCI database [4]. The first data set (*Zoo data*) is a collection of 100 animals with 17 features (1 quantitative, 1 nominal and 15 boolean). The second data set (*Auto import data*) consists of 193 instances of cars with 24 features (14 quantitative and 10 nominal) and the third data set (*Tic-tac-toe data*) contains 958 instances, each described by 9 categorical attributes.

In order to examine the effectiveness of the *online b-coloring* algorithm, the experimental methodology in conducted as follows: for each data set, the *b-coloring partition* id firstly generated upon a data sample (which contains 50 instances for Zoo, 100 for Auto import and 700 for tic-tac-toe) from the original data sets; then the partition is updated by adding sequentially the remaining points and the value of $Dunn_g$ index is computed as more instances are included.

For an interesting assess of the results gained on these data set, our algorithm was compared against *original b-coloring*, *Single-Pass* and *k-NN*. The *original b-coloring* consists in performing complete re-clustering using *b-coloring clustering algorithm* [2]. The *Single-Pass clustering* algorithm basically processes instances sequentially, and compares each instance to all existing clusters (color). If the dissimilarity between the instance and any cluster is below the optimal threshold θ_o, then the instance is added to the closest cluster; otherwise it forms its own cluster. The *k-Nearest Neighbor clustering* algorithm computes for each new instance its dissimilarity to every other instance, and chooses the top *k* instances. The new instance is assigned to the cluster where the majority of the top *k* instances are assigned.

The figures 1, 2 and 3 show the evolution of $Dunn_g$ values according to the number of instances. The curves comparisons of the different clustering algorithms show the performance of the *online b-coloring algorithm*. It appears clearly that the online algorithm achieves better results than *k-NN* (*k*=5) and more significant results than *Single-Pass*. It appears that the incremental algorithm slightly improves the performance of the *original b-coloring algorithm* (except the runtime profit) especially due to the efficiency of the re-assignment strategy (re-coloring) which improves the partition quality in terms of $Dunn_g$ value.

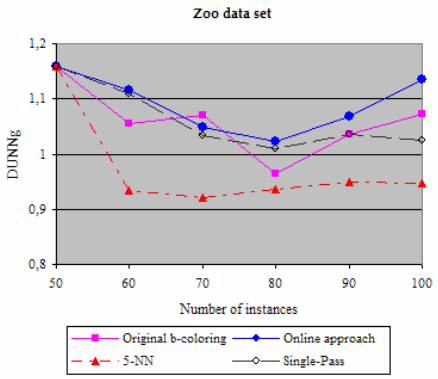

Fig. 1 – Performances on *Zoo*.

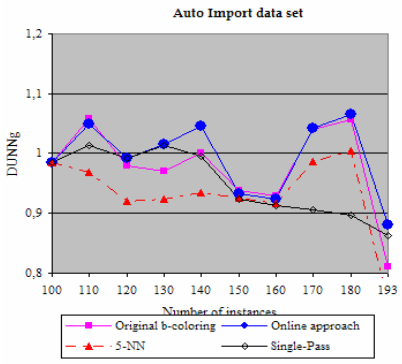

Fig. 2 – Performances on *Auto Import*.

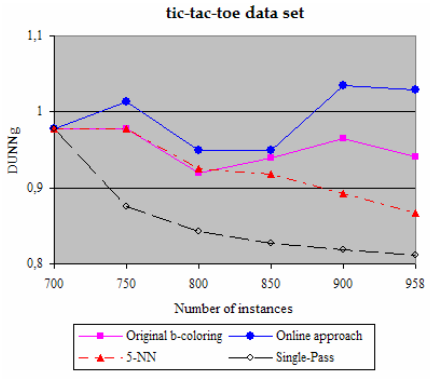

Fig. 3 – Performances on *tic-tac-toe*.

References

[1] Jain, A.K., M.N. Murty, and P.J. Flynn: Data Clustering: A Review. In: *ACM Computing Surveys*, Vol. 31, (1999), pp. 264-323.

[2] Elghazel, H. et *al.*: A new clustering approach for symbolic data and its validation: Application to the healthcare data. In F.Esposito et al.(Eds), editor, ISMIS2006 (Springer Verlag LNAI 4208) ,(2006), pp. 473–482.

[3] Kalyani, M. and M. Sushmita: Clustering and its validation in a symbolic framework. *Pattern Recognition Letters*, 24(14), (2003), pp. 2367-2376.

[4] Blake, C.L. and C.J. Merz: *UCI repository of machine learning databases*. University of California, Irvine, Dept. of Information and Computer Sciences. Available from http://www.ics.uci.edu/ ~mlearn/MLRepository.html, (1998).