A Partially Dynamic Clustering Algorithm for Data Insertion and Removal

Haytham Elghazel, Hamamache Kheddouci, Véronique Deslandres, and Alain Dussauchoy

LIESP Laboratory, Lyon 1 University, 43 Bd du 11 novembre 1918, 69622 Villeurbanne cedex, France
{elghazel,hkheddou,deslandres,dussauchoy}@bat710.univ-lyon1.fr

Abstract. We consider the problem of *dynamic* clustering which has been addressed in many contexts and applications including dynamic information retrieval, Web documents classification, etc. The goal is to efficiently maintain homogenous and well-separated clusters as new data are inserted or existing data are removed. We propose a framework called *dynamic b-coloring clustering* based solely on pairwise dissimilarities among all pairs of data and on cluster dominance. In experiments on benchmark data sets, we show improvements in the performance of clustering solution in terms of quality and computational complexity.

Keywords: Dynamic clustering, graph *b-coloring*, dissimilarity, dominance.

1 Introduction

Cluster analysis is one of the most important aspects in the data mining process for discovering groups and identifying interesting distributions or patterns over the considered data sets [1]. Clustering algorithms are widely used in many areas including information retrieval, image segmentation and so on.

In [2] a new partitioning clustering scheme is introduced. It is based on the *b-coloring of graph* [3]. This technique consists in coloring the vertices of a graph G 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.

In dynamic information environments, such as the World Wide Web, it is usually desirable to apply adaptive methods for document organization such as clustering. Incremental clustering methods are of great interest in particular when we examine their ability to cope with a high rate of dataset update. In this paper, we consider the problem of *online clustering* in the form of data insertion and removal. The difference

V. Corruble, M. Takeda, and E. Suzuki (Eds.): DS 2007, LNAI 4755, pp. 78–90, 2007.

between these *learning approaches* and the traditional ones in particular is the ability to process instances as they are added (new data) or deleted (outmoded or inefficient data) from the data collection, eventually with an updating of existing clusters without having to frequently performing complete re-clustering.

In the dynamic setting, instances arrive or leave one by one, and we need to deal with an arriving or removed data before seeing any future instances. Problems faced by such algorithms include how to find the appropriate cluster to assign for a new object, how to deal with deletion of an existing object, and how to reassign objects to other clusters

Many algorithms are proposed to investigate the dynamic clustering problem. The *Single-Pass clustering* algorithm basically processes instances sequentially, and compares each instance to all existing clusters. If the dissimilarity between the instance and any cluster[1] is above a certain threshold, then the instance is added to the closest cluster; otherwise it forms its own cluster. The *k-Nearest Neighbor clustering* [4] 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 most frequent class label among the k nearest training.

In this paper, a dynamic algorithm is proposed for the *b-coloring based clustering approach* presented in [2]. It depends only on pairwise dissimilarities among all pairs of data and on *dominance property* of vertices.

The paper is structured as follows: in Section 2, the *b-coloring* technique is introduced in broad outline. Section 3 is devoted to the dynamic algorithm. Some experiments using relevant benchmarks data set are shown in Section 4. Further works and applications linked with dynamic clustering will be proposed in conclusion.

2 Clustering with Graph b-Coloring

In this section, we briefly introduce the *b-coloring based clustering approach* and we refer the reader to [2] for more details.

When the dissimilarities among all pairs of data to be clustered $\{x_1,...,x_n\}$ are specified, these can be summarized as a weighed dissimilarity matrix D in which each element $D(x_i; x_j)$ stores the corresponding dissimilarity. Based on D, the data can also be conceived as a weighted linkage graph $G = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = \{v_1,v_2,...,v_n\}$ is the vertex set which correspond to the data (vertex v_i for data x_i), and $\boldsymbol{E} = \boldsymbol{V} \times \boldsymbol{V}$ is the edge set which correspond to a pair of vertices $(v_i; v_j)$ weighted by their dissimilarities $D(v_i; v_j)$. 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

[1] The dissimilarity between an instance x and a cluster C is the average of dissimilarities between x and instances of C.

$\boldsymbol{P}=\{C_1,C_2,..,C_k\}$ 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 $G_{>\theta}$ consists of two steps: 1) generate an initial coloring of vertices using a maximum number of colors, and 2) removing each color that has no dominating vertices yet using a greedy algorithm. Step 2 is performed until the coloring is stable, i.e. each color of $G_{>\theta}$ has at least one dominating vertex.

Let illustrate the *b-coloring* algorithm on one example. $\{A,B,C,D,E,F,G,H,I\}$ is the data set to analyse for which dissimilarity matrix D is given in table 1. Figure 1 shows the *superior threshold graph* for θ =0.15. Therefore here, the *b-coloring* of $G_{>0.15}$ (*cf.* Fig.2) gives four classes, namely: C_1={**B**}, C_2={**A**,**D**}, C_3={**C**,**E**,G,H,I} and C_4={**F**}. Bold characters show dominating vertices.

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* ($Dunn_G$) [5]. $Dunn_G$ 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. As an illustration, successive threshold graphs are constructed for each threshold θ selected from the dissimilarity Table 1, and our approach is used to give the *b-coloring* partition of each graph. The value of the *Dunn's generalized index* is computed for the obtained partitions. We conclude that the partition θ=0.15 has the maximal $Dunn_G$ among other ones with different θ.

Table 1. Dissimilarity matrix

v_i	A	B	C	D	E	F	G	H	I
A	0								
B	0.20	0							
C	0.10	0.30	0						
D	0.10	0.20	0.25	0					
E	0.20	0.20	0.10	0.40	0				
F	0.20	0.20	0.20	0.25	0.65	0			
G	0.15	0.10	0.15	0.10	0.10	0.75	0		
H	0.10	0.20	0.10	0.10	0.05	0.05	0.05	0	
I	0.40	0.075	0.15	0.15	0.15	0.15	0.15	0.15	0

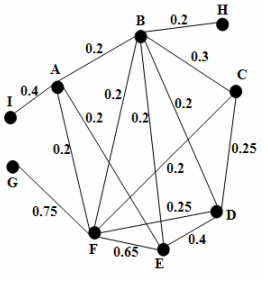

Fig. 1. *Superior threshold graph* $G_{>0.15}$ *(*θ *=0.15)*

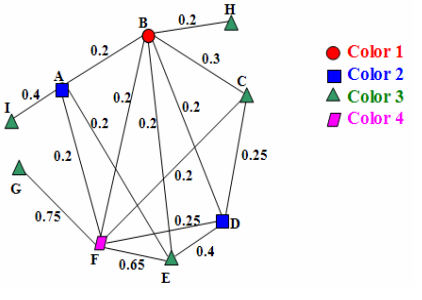

Fig. 2. *b-coloring* of graph $G_{>0.15}$: four classes are identified

3 Online *b-Coloring* Based Clustering Algorithm

We now present the *online* clustering algorithm based on the above scheme (the *b-coloring* based clustering). The algorithm works incrementally by receiving a new document or removing existing data. The principle 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 or to rearrange the partition when existing instances leave the system. 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 deletion of one instance $x_m\in\boldsymbol{X}$ transforms the vertex set $\boldsymbol{V}$ on $\boldsymbol{V}-\{v_m\}$ and the edge set $\boldsymbol{E}$ on $\boldsymbol{E}-\{(v_i,v_m)|\ v_i\in\boldsymbol{V}$ and $D(v_i,v_m)>\theta_o\}$. The main problem is to find the appropriate color to assign for v_{n+1} (*i.e.* in the case of insertion) or to rearrange the coloring of G (*i.e.* in the case of deletion) which is constrained to incrementally maintain the *b-coloring* of G and the clustering performances in terms of quality ($Dunn_G$ *value*) and runtime.

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

- Δ: the maximum degree of G.
- $c(v_i)$: 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.

3.1 Adding a New Instance x_{n+1}

When a new instance x_{n+1} is introduced which corresponds to the vertex and edges adding in G, the following update operations on G are allowed:

- v_{n+1} is assigned to one of the existing k colors of G.
- v_{n+1} forms its own color.
- The insertion of v_{n+1} in G launches the merge of some colors in G.

As mentioned above, our *dynamic 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:

3.1.1 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:

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

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

Procedure Scenario 1()

```
BEGIN
c(v_{n+1}):=k+1 ;
For each free vertex v_i regarding the color k+1 do
 If (d(v_i,k+1)<d(v_i,c(v_i)) then
    for each vertex v_j from G do
     Update(d(v_j,k+1);// using eq.(2)
     Update (d(v_j,c(v_i));// using eq.(3)
    Enddo
    c(v_i):=k+1;
 EndIf
Enddo
find_dominating();
END.
```

Therefore, after the re-coloring of every *free vertex* v_i *regarding the color* k, the method *find_dominating()* of order $O(n)$ tries to identify the new dominating vertices in G.

Proposition 2. The procedure *Scenario 1()* runs in $O(n^2)$.
Proof. After the coloring of v_{n+1} using the $(k+1)^{th}$ color, the algorithm for *Scenario 1* verifies if the color of each free vertex v_i regarding the color $k+1$ (at most n) can be changed by $(k+1)$. In this case, for each vertex v_j from G we update the dissimilarities $d(v_j,k+1)$ and $d(v_j,c(v_i))$ using the formulas eq.(2,3) in $O(n)$. Therefore, *Scenario 1* uses at most $(n*n)$ instructions, and the complexity is $O(n^2)$.

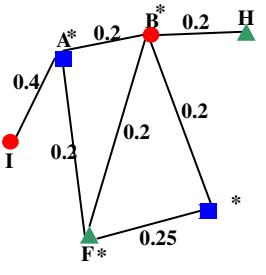

Fig. 3. Optimal Partition of $\{A,B,D,F,H,I\}$ on 3 clusters for $\theta=0.15$. "*" is used to denote the dominating vertices.

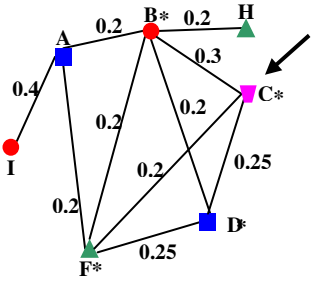

Fig. 4. Insertion of vertex C using *Scenario 1*: the neighborhood of C contains at least one dominating vertex of each color

3.1.2 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 (*cf.* Fig.5). 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.

```
Procedure Scenario 2.1()
BEGIN
H := {h | h∉Nc(vn+1)};
c(vn+1):= {C| d(vn+1,C)=minh∈H(d(vn+1,h))};
For each vertex vi from G do
   Update(d(vi, c(vn+1)) ;// using eq.(2)
Enddo
For each vertex vi ∈ N(vn+1) do
   test_dominance(vi) ;
Enddo
END.
```

After the insertion of v_{n+1} in the selected color (among m_1), some vertices from the neighborhood of v_{n+1} became dominating vertices. These vertices needed only one neighbor within the selected color to become dominating. In order to verify this situation, we need to recall the method *test_dominance(vertex)* which is in order $O(1)$.

Proposition 3. The procedure *Scenario 2.1()* performs in $O(n)$.
Proof. After the coloring of v_{n+1} using the selected color, the procedure *Scenario 2.1()* tries to update the dissimilarity $d(v_i,c(v_{n+1}))$ for each vertex v_i from G using the formula eq.(2) ($O(n)$). Afterward, it verifies the dominance property of the neighbors of v_{n+1} (at most Δ) using *test_dominance method* ($O(1)$). Therefore, the procedure *Scenario 2.1()* uses at most ($\Delta+n*1$) instructions, and the complexity is $O(n)$.

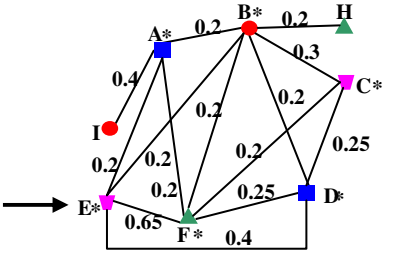

Fig. 5. Insertion of vertex E using *Scenario 2.1*: the color of C does not belong to the neighborhood colors of E. Consequently, the color of C is assigned to E.

❖ ***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.

Procedure Scenario 2.2.1()
BEGIN
c(v_{n+1}):= C;// C the color available to receive v_{n+1}
For each vertex v_i from G **do**
Update(d(v_i, c(v_{n+1})) ;// using eq.(2)
Enddo
For each vertex $v_i \in N(v_{n+1})$ **such that** c(v_i)=C **fo**
H := {h | h$\notin N_c(v_i)$};
k := {color| d(v_i,color)=min$_{h\in H}$(d(v_i,h))};

```
   For each vertex v_j from G do
      Update (d(v_j,k);// using eq.(2)
      Update (d(v_j,c(v_i));// using eq.(3)
   Enddo
   c(v_i):=k ;
Enddo
For each vertex v_i ∈ N(v_{n+1}) do
   test_dominance(v_i) ;
Enddo
END.
```

Proposition 4. 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 $\boldsymbol{P}$. The coloring of G is a *b-coloring*.

Proposition 5. The procedure *Scenario 2.2.1()* performs in $O(n\Delta)$.

Proof. When the color C is assigned to v_{n+1}, the neighbor vertices of v_{n+1} colored with C (at most Δ) change their colors which require the updates of the dissimilarities values in $O(n)$. Afterward, the dominance property of the neighbors of v_{n+1} (at most Δ) is verified using the *test_dominance method* ($O(1)$). Therefore, the procedure *Scenario 2.2.1()* uses at most ($\Delta * n + \Delta * 1$) instructions, and the complexity is $O(n\Delta)$.

- *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}.

Example: As an illustration, the figure 6 shows two colors C_1 and C_2 *available to receive* the vertex **F** ($m=2$). The unique neighbor in C_1 to the dominating vertex of C_2 (the vertex **B**) is the vertex **A** (called a *supporting vertex*) which belong to the neighborhood of **F**. Thus, the color C_1 is a *non transformation subject*. In fact, if the color C_1 is affected to the vertex **F**, the vertex **A** (dissimilar to **F**) must be re-colored. Due to this transformation, the color C_2 is removed from the neighborhood colors of **B** which becomes a non dominating vertex and the color C_2 without dominating vertices. Consequently, the transformation of C_1 is forbidden. Contrary to C_1, C_2 is a *transformation subject* and it is hence available to receive **F**.

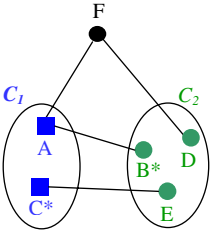

Fig. 6. A *transformation subject* colors identification

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).

Discussion

In order to process new data instances as they are arrived, the learning 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.

3.2 Removal of an Existing Instance v_m

When an instance x_m is introduced which corresponds to the vertex and edges deleting in G, we must rearrange the coloring of G in order to maintain the *b-coloring* properties and a high quality clustering. Likewise to the previous scenarios, our idea is based 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:

3.2.1 Scenario 3: v_m Is the Sole Dominating Vertex of Its Color $c(v_i)$

In this case, v_m is the only one dominating vertex of its color. Therefore, by removing v_m, the color $c(v_m)$ becomes without dominating vertices and the coloring of G is not a *b-coloring*. Consequently, the colors of the remaining vertices of $c(v_m)$ 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. In the opposite case of this scenario, the following *Scenario 4* is performed.

3.2.2 Scenario 4: v_m Is a Supporting Vertex of All Dominating Vertices from at Least One Color

In this situation, v_m is the sole vertex colored with $c(v_m)$ in the neighborhood of all dominating vertices from at least one color C. As a result, the deletion of v_m, pushes these vertices to become *non dominating* and C without dominating vertices. 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 C without dominating vertices.

If v_m does not verify any of the previous *Scenarios* 3 and 4, the dynamic algorithm process the deletion of v_m without any rearrangement and the new coloring of G is a *b-coloring*.

4 Experimental Results

Experiments have been made using three relevant benchmark data sets chosen from UCI database [6]. 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*[2] and *k-NN* (*k*=5). The *original b-coloring* consists in performing complete re-clustering using *b-coloring clustering algorithm* [2].

We note that the Euclidian distance is applied to define the dissimilarity level between two instances characterized with *m* features a_f ($f \in \{1...m\}$) as given by the following formula:

$$d_{i,j} = D(x_i, x_j) = \left(\sum_{f=1}^{m} \left(\frac{g_f(a_{i,f}, a_{j,f})}{m_f} \right)^2 \right)^{1/2} \tag{4}$$

where m_f is the *normalized coefficient* for the attribute a_f and g_f is the comparative dissimilarity function between the two attribute values $a_{i,f}$ and $a_{j,f}$ corresponding respectively to the instances x_i and x_j.

For *numeric* attributes, g_f is: For *categorical* attributes, g_f is:

$$g_f(a_{i,f}, a_{j,f}) = |a_{i,f} - a_{j,f}| \qquad g_f(a_{i,f}, a_{j,f}) = \begin{cases} 1 & iff \quad a_{i,f} \neq a_{j,f} \\ 0 & iff \quad a_{i,f} = a_{j,f} \end{cases} \tag{5}$$

The figures 7, 8 and 9 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. Finally, one can see that similar experiments may be done for removal instances.

[2] The dissimilarity threshold used in the *Single-Pass* algorithm is the optimal threshold θ_o.

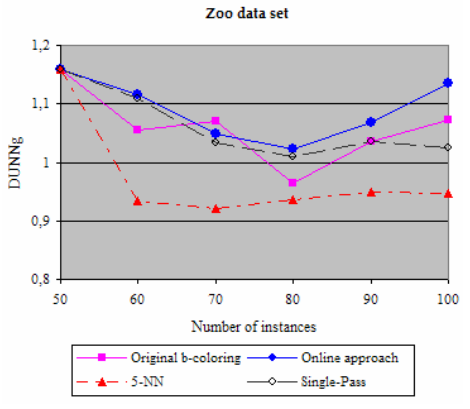

Fig. 7. Performances on *Zoo*

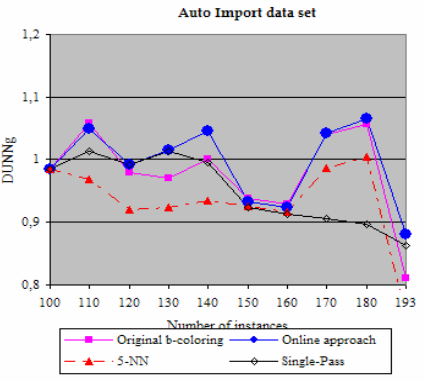

Fig. 8. Performances on *Auto Import*

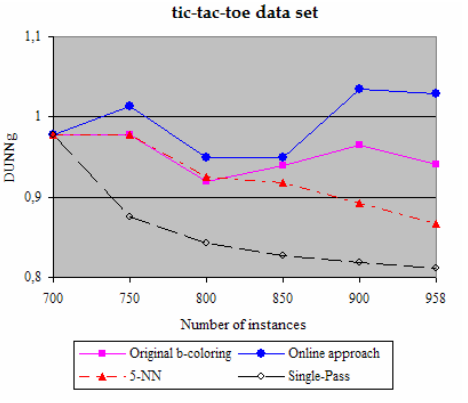

Fig. 9. Performances on *tic-tac-toe*

5 Conclusion

We proposed a dynamic *version* for the *b-coloring based clustering approach* which relies only on *dissimilarity matrix* and c*luster dominating vertices* in order to cluster new data as they are added to the data collection or to rearrange a partition when an existing data is removed. A real advantage of this method is that it performs a dynamic classification that correctly satisfies the *b-coloring* properties and the clustering performances in terms of quality ($Dunn_G$ *value*) and runtime, when the number of clusters is not pre-defined and without any exception on the type of data. The results obtained over three *UCI data sets* have illustrated the efficiency of our algorithm to generate good results than *Single-Pass* and *k-NN* algorithms.

There are many interesting issues to pursue: (1) leading additional experiments on a larger medical data set where a patient stay typology is required and an inlet patient

stay is regular and has to be incorporate to the typology, (2) extending the incremental concept to add or remove simultaneously sets of instances, and (3) to define some operators which permit to combine easily different clusterings constructing on different data.

References

[1] Jain, A.K., Murty, M.N., Flynn, P.J.: Data Clustering: A Review. ACM Computing Surveys 31, 264–323 (1999)
[2] Elghazel, H., et al.: A new clustering approach for symbolic data and its validation: Application to the healthcare data. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) ISMIS 2006. LNCS (LNAI), vol. 4203, pp. 473–482. Springer, Heidelberg (2006)
[3] Irving, W., Manlove, D.F.: The b-chromatic number of a graph. Discrete Applied Mathematics 91, 127–141 (1999)
[4] Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
[5] Kalyani, M., Sushmita, M.: Clustering and its validation in a symbolic framework. Pattern Recognition Letters 24(14), 2367–2376 (2003)
[6] Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences (1998), http://www.ics.uci.edu/ mlearn/MLRepository.html