Towards conflict resolution in collaborative clustering

Germain Forestier, Cédric Wemmert and Pierre Gancarski
LSIIT - CNRS - University of Strasbourg - UMR 7005
Pôle API, Bd Sébastien Brant - 67412 Illkirch, France
email: forestier@unistra.fr

***Abstract*—In recent years, a lot of work has focused on the use of multiple clusterings for partitioning data. These approaches are supported by the existence of a huge number of clustering algorithms. Thus, different methods have been proposed to create alternative clustering results from the same data. However, the different clustering results are usually generated without sharing information and the user is often asked to select the final result. To cope with these issues, a new paradigm named collaborative clustering has been proposed recently. In collaborative clustering, different clustering methods work together (i.e. collaborate) to reach an agreement on the clustering of a common dataset. At the end of the collaboration, the results are expected to be strongly similar. In this paper, we address the problem of the collaboration of different clustering methods and we compare four collaboration strategies. Our experiments compare the different strategies on synthetic and real-life datasets and provide insight into the advantages and the drawbacks of each strategy.**

I. Introduction

Over the last fifty years, many new clustering algorithms have been developed, and existing methods have been modified and improved [1], [2]. This abundance of methods can be explained by the ill-posed nature of clustering. Indeed, each clustering algorithm is biased by the objective function used to build the clusters. Consequently, different methods can, from the same data, produce very different clustering results. Furthermore, even the same algorithm can produce different results, when parameters or initialization vary. A recent approach to cope with this problem is based on the idea that the information provided by different clustering algorithms may be complementary. Thus, the combination of different clustering results may increase their efficiency and accuracy.

In this paper, we address the problem of the collaboration between different clustering methods. In collaborative clustering, different clustering methods work together (i.e. collaborate) to reach an agreement on the clustering of a common dataset. One of the main challenges in collaborative clustering is the comparison of the different clustering results, in order to identify precisely where they are in disagreement. However, the clusters of one result do not have a trivial link with the clusters of another one. This lack of straightforward correspondence increases the challenge of discovering disparities (called *conflicts*) among the results. Once these conflicts are detected, the goal of a collaborative clustering process is to solve them by modifying the pair of results involved in the conflict. Thus, another challenging problem is the order in which the conflicts have to be solved. Indeed, many conflicts exist at the same time between the different results and the system has to choose which one to solve first. The main heuristic developed so far is the resolution of the most important conflict at each step of the algorithm. However, when the number of methods involved increases and when the dataset is complex, the size of the search-space (i.e. the space of existing solutions) increases tremendously. Consequently, it can have many local optima and always solving the most important conflict can lead to non-optimal solutions.

The purpose of this paper is to present alternative strategies of conflict resolution. We introduce three resolution strategies to improve the way the collaboration is processed. Three strategies are based on an iterative algorithm whereas the last one uses a genetic algorithm to drive the collaboration.

The rest of the paper is organized as follows. In Section 2, a review of related work is presented. Then in Section 3, the main principles of collaborative clustering are described. Section 4 introduces the different strategies studied in the paper, and Section 5 presents their assessments through various experiments. Finally, Section 6 concludes the paper and discusses future work.

II. Related work

In recent years, a lot of work has focused on the use of multiple clusterings as it is often difficult to design a single algorithm whose results reflect what users need and expect.

One of the existing approach is to generate a set of alternative clusterings and let the user select the solution according to its need. The challenging issue is to generate a set of solutions offering a good quality and a good diversity. For example, Caruana et al. [3] presented a method for automatically generating a diverse set of alternate clusterings, as well as methods for grouping clusterings into meta-clusters. At the end of the process, the user can navigate among the meta-clusters to select the final solution. Bae et al. [4] addressed the problem of creating an alternative clustering result starting from one existing solution. The key challenge is to use this initial solution to create a new alternative result which is different from the previous one but also reflecting a good clustering of the data. Davidson et al. [5] also addressed the problem of alternative clustering through the use of constraints. The constraints generated between data

objects from previous clusterings are used to constraint the generation of the alternative solution.

Another existing approach consists in creating a single clustering result, which summarizes a set of existing results. In the method proposed by Law et al. [6], different algorithms are applied on the same dataset and the final solution is created by selecting different clusters among the different results. A similar idea is presented by Jiamthapthaksin et al. [7], where the authors proposed an architecture for multi-run clustering where different clustering solutions are explored in an iterative way. The final result is created from information selected from every step of the process. However, in other existing methods, the final solution is not always composed of parts of the initial clustering results. For example, Gionis and al. [8] presented clustering aggregation algorithms creating a final clustering that minimizes the total number of disagreements among all clusterings. These approaches, generally referred as ensemble clustering, have received a strong interest during the recent years [9], [10], [11]. However, the ensemble clustering approaches do not generally address the problem of the generation of the initial results, and the algorithms used to create the initial results are not used in the combination process. Consequently, ensemble clustering approaches introduce a new bias, relative to the objective function chosen when merging the different clusterings.

The majority of these work focus on the merging process of the different clusterings, whereas our approach addresses the task of the collaboration between different methods. Indeed, in these approaches, the amount of information shared by the methods during the clustering process is relatively low. In our approach, the different methods collaborate and share information throughout the clustering process.

III. Collaborative clustering

Given a set $\breve{\mathcal{R}} = \langle \mathcal{R}^i \rangle_{i=1\dots m}$ of m different clustering results of the same dataset, the goal of collaborative clustering is to find a consensus among these results by reducing their disagreements. $\breve{\mathcal{R}}$ is composed of clustering results created with different algorithms or the same algorithm with different parameters. Let $\langle \mathcal{C}_k^i \rangle_{k=1\dots n_i}$ be the n_i clusters of one result $\mathcal{R}^i$ from $\breve{\mathcal{R}}$. Note that we work here with results that do not necessary have the same number of clusters (e.g. $n_i \neq n_j$ is possible if $i \neq j$).

To identify and solve the disagreements between the clusters of the different results, a similarity S between clusters from each pair of results has to be evaluated. We define that there is a *conflict* between a cluster $\mathcal{C}_k^i$ and a clustering $\mathcal{R}^j$ if no cluster of $\mathcal{R}^j$ is identical to $\mathcal{C}_k^i$ according to S. Thus, the conflicts $\breve{\mathcal{K}}$ in the ensemble $\breve{\mathcal{R}}$ can be defined as:

$$\begin{array}{l} \breve{\mathcal{K}} = \left\{ (\mathcal{C}_k^i, \mathcal{R}^j) : i \neq j, S\left(\mathcal{C}_k^i, \psi\left(\mathcal{C}_k^i, \mathcal{R}^j\right)\right) < 1 \right\} \\ \text{with } \psi\left(\mathcal{C}_k^i, \mathcal{R}^j\right) = \arg\max_{\mathcal{C}_l^j \in \mathcal{R}^j} S\left(\mathcal{C}_k^i, \mathcal{C}_l^j\right) \end{array} \quad (1)$$

Consequently, we have to design a *local similarity* measure capable to compare two clusters from two different results. This measure will then be used to identify the pairs of clusters (of two different results) having a poor similarity (i.e. sharing a poor overlap of data objects). These pairs will define the conflicts to solve to increase the similarity between the results.

Moreover, we also have to estimate the *global similarity* of all the clusterings involved in the collaboration to be able to assess the global usefulness of a local modification. Indeed, a conflict is identified between a pair of results whereas more than two methods can be involved in the collaboration process. Therefore, modification at the local level (i.e. between a pair of results) has to be assessed at a global level (i.e. all the results involved in the collaboration). The goal of collaborative clustering is to maximize this global similarity which is an indicator of the agreement among the set of results.

A. Local clusterings comparison

A large number of criteria exist to evaluate the similarity between a pair of clustering results [12]. However, these criteria only give a global evaluation of the similarity between two partitions. As we want to identify exactly which clusters are involved in the conflict, we have to compare each cluster of one result with all the clusters of the other results. In order to compare a pair of results, we use the confusion matrix or matching matrix (2). The *matching matrix* $\mathcal{M}^{i,j}$ between two results $\mathcal{R}^i$ and $\mathcal{R}^j$ is a $n_i \times n_j$ matrix defined by:

$$\mathcal{M}^{i,j} = \begin{pmatrix} \alpha_{1,1}^{i,j} & \dots & \alpha_{1,n_j}^{i,j} \\ & \vdots & \\ \alpha_{n_i,1}^{i,j} & \dots & \alpha_{n_i,n_j}^{i,j} \end{pmatrix} \text{ where } \alpha_{k,l}^{i,j} = \frac{\left|\mathcal{C}_k^i \bigcap \mathcal{C}_l^j\right|}{\left|\mathcal{C}_k^i\right|} \quad (2)$$

The *adequacy* $\omega_k^{i,j}$ of a cluster $\mathcal{C}_k^i$ compared to a clustering $\mathcal{R}^j$ is evaluated by observing the intersection (2) between the cluster $\mathcal{C}_k^i$ and its corresponding cluster (i.e. the most overlapping cluster) in $\mathcal{R}^j$, and by taking into account the distribution $\rho_k^{i,j}$ (4) of the cluster $\mathcal{C}_k^i$ in all the clusters of $\mathcal{R}^j$:

$$\omega_k^{i,j} = \rho_k^{i,j} \alpha_{l,k}^{j,i} \quad (3)$$

where $\alpha_{l,k}^{j,i} = max\langle \alpha_{l,k}^{j,i} \rangle_{l=1\dots n_j}$ and

$$\rho_k^{i,j} = \sum_{r=1}^{n_j} (\alpha_{k,r}^{i,j})^2 \quad (4)$$

To compute the *local similarity* between a pair of results, the adequacies of each pair of clusters of the two results are averaged. The similarities have to be computed in each of the two ways, as the matching matrices are not usually symmetric ($\omega_k^{i,j} \neq \omega_k^{j,i}$).

However, if we try to only maximize the local similarity of the results, we could easily end up with a trivial (and not wanted) solution like a unique cluster with all the data objects inside it. To cope with this problem, we introduce an evaluation of the quality of the two clusterings in the local similarity measure as follows:

$$\gamma^{i,j} = \frac{1}{2} \left(\underbrace{\left(\frac{1}{n_i} \sum_{k=1}^{n_i} \omega_k^{i,j} + \frac{1}{n_j} \sum_{k=1}^{n_j} \omega_k^{j,i} \right)}_{\text{similarity}} + \underbrace{\left(\delta^i + \delta^j\right)}_{\text{quality}} \right) \quad (5)$$

where δ^i is a measure of the quality of a result $\mathcal{R}^i$. This measure depends on the algorithm used in the process. Any partition validity index can be used here, like Silhouette [13] or Davies Bouldin [14].

B. Comparing global clusterings

The local similarity (5) evaluates the similarity and the quality of a pair of results. Though, more than two methods can be involved in the collaboration process. The *global similarity* evaluates the similarity of each pair of results involved in the collaboration and gives a global assessment of the results similarities:

$$\Gamma = \frac{1}{m}\sum_{i=1}^{m}\Gamma^i \quad \text{where} \quad \Gamma^i = \frac{1}{m-1}\sum_{\substack{j=1\\ j\neq i}}^{m}\gamma^{i,j} \tag{6}$$

C. Conflict definition and assessment

As introduced previously, there is a *conflict* between two clustering results $\mathcal{R}^i$ and $\mathcal{R}^j$ about the cluster $\mathcal{C}_i^k$ of $\mathcal{R}^i$, if there is no cluster in $\mathcal{R}^j$ similar to $\mathcal{C}_k^i$. Each conflict $\mathcal{K}_k^{i,j}$ is therefore identified by one cluster $\mathcal{C}_k^i$ and one result $\mathcal{R}^j$. Its importance $CI\left(\mathcal{K}_k^{i,j}\right)$ is computed according to the local similarity between $\mathcal{C}_k^i$ and $\mathcal{R}^j$ (3) as:

$$CI\left(\mathcal{K}_k^{i,j}\right) = 1 - \omega_k^{i,j} \tag{7}$$

$\breve{\mathcal{K}}$ (1) is the list containing all the conflicts of $\breve{\mathcal{R}}$. Each conflict has an importance (7) and involves a pair of results.

IV. Conflict resolution strategies

As already stated, the goal of collaborative clustering is to maximize the global similarity of the results (6), by solving the conflicts between the different results. In this section, we present different strategies to solve these conflicts. The four different strategies are detailed in the following subsections. Firstly, we introduce three iterative conflict resolution strategies and then a conflict resolution strategy based on a genetic algorithm.

A. Iterative resolution strategies

As presented in Algorithm 1, the main approach to solve the conflicts in collaborative clustering works in an iterative way. At each iteration, one conflict is selected and its resolution is computed. Algorithm 2 describes in details the resolution of a conflict using split, merge and recluster operators. Note that a parameter p_{cr} is introduced, representing the minimal similarity above which the clusters are evaluated as identical. The process iterates, and one conflict is solved at each step if this conflict improves the local agreement (5). If the resolution of this conflict is not relevant, this conflict is removed from the conflict list. When the list of conflicts is empty, the process stops.

Selecting the right conflict to solve is of particular interest. We present three different strategies corresponding to three different definitions of the `conflictChoice` function in Algorithm 1: worst conflict choice (WCC), stochastic conflict choice (SCC), and roulette-wheel conflict choice (R-WCC). In the following definitions of each strategy, $\mathcal{K}_{(i)}$ is the i^{th} conflict of $\breve{\mathcal{K}}$, and $p(\mathcal{K}_{(i)})$ its probability to be selected for the next resolution attempt.

1) Worst conflict choice (WCC): This first approach consists in choosing the worst conflict, i.e. the one having the highest conflict importance (7). This approach assumes that the resolution of the most important conflict between a pair of results will increase the global agreement between all

Algorithm 1 Collaborative clustering process

Let $\breve{\mathcal{R}} = \langle \mathcal{R}^i \rangle_{1\leq i\leq m}$ be the initial set of clusterings
Let $\breve{\mathcal{K}} = \texttt{conflicts}(\breve{\mathcal{R}})$ be the set of conflicts in $\breve{\mathcal{R}}$ as defined in section III-C
Let $\breve{\mathcal{R}}^{\texttt{best}} := \breve{\mathcal{R}}$ be the best temporary solution
Let $\breve{\mathcal{K}}^{\texttt{best}} := \breve{\mathcal{K}}$ be the conflicts of the best temporary solution
while $|\breve{\mathcal{K}}| \geq 0$ **do**
  $\mathcal{K}_k^{i,j} := \texttt{conflictChoice}\left(\breve{\mathcal{K}}\right)$
  $\breve{\mathcal{R}} := \texttt{conflictResolution}\left(\breve{\mathcal{R}}, \mathcal{K}_k^{i,j}\right)$ (Alg.2)
  if $\mathbf{\Gamma}(\breve{\mathcal{R}}) > \mathbf{\Gamma}(\breve{\mathcal{R}}^{\texttt{best}})$ **then**
    $\breve{\mathcal{R}}^{\texttt{best}} := \breve{\mathcal{R}}$
    $\breve{\mathcal{K}}^{\texttt{best}} := \breve{\mathcal{K}} := \texttt{conflicts}(\breve{\mathcal{R}})$
    $bt := 0$
  else if $\breve{\mathcal{R}}^{t+1} = \breve{\mathcal{R}}^t$ **then**
    $\breve{\mathcal{K}} := \breve{\mathcal{K}} \setminus \mathcal{K}_k^{i,j}$
  else
    $bt := bt + 1$
    $\breve{\mathcal{K}} := \breve{\mathcal{K}} \setminus \mathcal{K}_k^{i,j}$
    if $bt > |\breve{K}|$ **then**
      $\breve{\mathcal{R}} := \breve{\mathcal{R}}^{\texttt{best}}$
      $\breve{\mathcal{K}} := \breve{\mathcal{K}}^{\texttt{best}} \setminus \mathcal{K}_k^{i,j}$
    end if
  end if
end while
Consensus computation

Algorithm 2 Conflict resolution

Require: $\breve{\mathcal{R}}$ the ensemble of clusterings
Require: $\mathcal{K}_k^{i,j}$ the conflict to solve
Ensure: $\breve{\mathcal{R}}^\star = \texttt{conflictResolution}\left(\breve{\mathcal{R}}, \mathcal{K}_k^{i,j}\right)$ the new ensemble after the resolution
let $\kappa = \{\mathcal{C}_l^j, \forall 1 \leq l \leq n_j : \omega_k^{i,j} > p_{cr}\}$
if $|\kappa| > 1$ **then**
  $\mathcal{R}^{i'} = \mathcal{R}^i \setminus \{\mathcal{C}_k^i\} \cup \texttt{split}(\mathcal{C}_k^i, |\kappa|)$
  $\mathcal{R}^{j'} = \mathcal{R}^j \setminus \kappa \cup \texttt{merge}(\kappa, \mathcal{R}^j)$
else
  $\mathcal{R}^{i'} = \texttt{recluster}(\mathcal{R}^i \setminus \{\mathcal{C}_k^i\})$
end if
$\{\mathcal{R}^{i\star}, \mathcal{R}^{j\star}\} = \arg\max \gamma^{I,J}$ for $I \in \{i, i'\}, J \in \{j, j'\}$
$\breve{\mathcal{R}}^\star = \breve{\mathcal{R}} \setminus \{\mathcal{R}^i, \mathcal{R}^j\} \cup \{\mathcal{R}^{i\star}, \mathcal{R}^{j\star}\}$

the methods. This assumption is supported by the intuition that the resolution of an important conflict should increase significantly the similarity of a pair of solutions. In this case, the `conflictChoice` is defined as:

$$\mathcal{K} := \arg \max_{\mathcal{K}_{(i)} \in \breve{\mathcal{K}}} CI\left(\mathcal{K}_{(i)}\right) \tag{8}$$

It means that the probability of the most important conflict is $p(\mathcal{K}_{(1)}) = 1$ whereas the other conflict have no chance to be selected $p(\mathcal{K}_{(i>1)}) = 0$. As illustrated in Figure 1(a), only the first conflict of the list can be selected. Note that the list is ordered by conflict importance.

2) Stochastic conflict choice (SCC): This approach consist in choosing randomly a conflict to solve in the list of conflicts. This naive strategy assumes that solving a conflict with a high importance is not always relevant. In this case, the `conflictChoice` is defined as:

$$\mathcal{K} := \texttt{random}\left(\breve{\mathcal{K}}\right) \tag{9}$$

where the random function selects a conflict randomly in the list of conflicts $\breve{\mathcal{K}}$. Consequently, the probability $p(\mathcal{K}_{(i)})$ for this strategy is defined by $p(\mathcal{K}_{(i)}) = \frac{1}{N_c}$ with N_c the number of conflicts. As illustrated in Figure 1(b), each conflict has the same probability to be selected.

3) Roulette-wheel conflict choice (R-WCC): This approach is based on roulette-wheel selection or fitness proportionate selection well known in the field of evolutionary optimization [15]. The conflict importance is used to associate a probability of selection with each conflict. In this case, the probability of each conflict is defined as $p(\mathcal{K}_{(i)}) = \frac{CI\left(\mathcal{K}_{(i)}\right)}{\sum_{j=1}^{N_c} CI\left(\mathcal{K}_{(j)}\right)}$ with N_c the number of conflicts. While conflicts with a high importance will be more likely to be selected, there is still a chance that they may be avoided. In this case, the `conflictChoice` is defined as:

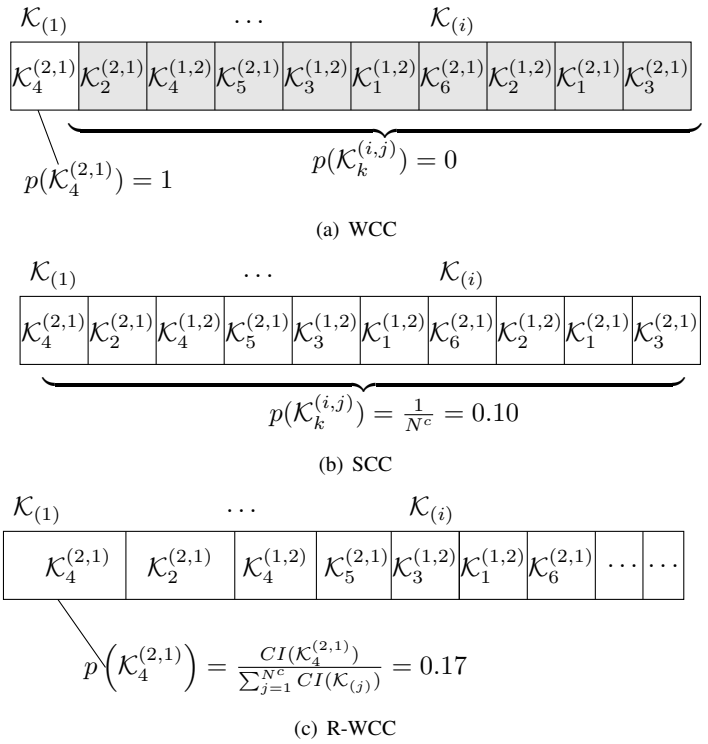
$$\mathcal{K} := \mathcal{K}_{(i)} | \left(\sum_{j=0}^{i-1} p(\mathcal{K}_{(j)}) \leq \nu \wedge \sum_{k=0}^{i+1} p(\mathcal{K}_{(k)}) > \nu \right) \tag{10}$$

where ν is a random value in $[0; 1]$.

As illustrated in Figure 1(c), each conflict has a probability to be selected which is proportionate to its importance. This probability is represented in the Figure 1(c) by the width of each cell.

B. Genetic resolution strategy (GR)

Genetic algorithms were first proposed as a way to solve problems where no other computational tractable algorithms exist. Genetic algorithms are heuristic searches and optimization techniques inspired by natural evolution [15] which find exact or approximate solutions of complex problems. They are used in various types of application, for example to find values of continuous variables. A genetic algorithm requires an initial population defined as a set of genotypes to perform the evolutionary process. In this process, the population evolves to obtain better genotypes at each generation, i.e. better solutions of the optimization problem under consideration. We used this



(a) WCC

(b) SCC

(c) R-WCC

Fig. 1. The different strategies for choosing the conflict to solve.

scheme to define a new strategy of conflict resolution. In this strategy, we consider g (the genotype in the genetic framework) as a set of clustering results $\langle \mathcal{R}^i \rangle_{i=1\ldots m}$. Each genotype represents a set of clustering results and consequently one solution to the collaborative clustering problem. To generate the initial population, we used the initial solution composed of different clustering results. We computed the conflicts list of this initial solution and we solved conflicts randomly. For each conflict resolution attempt, the generated alternative solution was stored as a member of the initial population. Another way to create the initial population could have been to make multiple runs of the algorithms used to create the initial solution. However, to be fair with the iterative methods we decided to only use the initial solution.

Once the initial population created, the algorithm relies on the following steps, which represent the transition between two generations:

1) *assessment of genotypes in the population*: each solution is evaluated (i.e fitness evaluation) according to its global similarity value (5). This fitness reflects the similarity and the quality of the set of results and is used to assess the quality of the solution. The higher the value, the better the solution.
2) *selection of genotypes for crossover operation* according to the fitness. A solution with a high global similarity value will have more chance to be selected than a solution with a lower one. This fitness proportionate selection uses the same principle we used in Section IV-A3 but based on the fitness of the solution and not

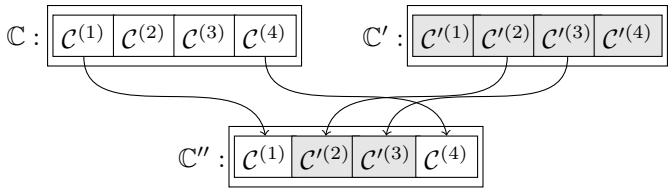


Fig. 2. Cross-over operation in GR algorithm.

the conflict importance.

3) *crossover*: the crossover operator is used to create new solutions from existing solutions in the population. Two solutions are selected and breed to create a new solution, by mixing the clustering results present in these two solutions (slicing cross-over). Figure 2 illustrates this process. Note that the order of the results in each solution is important and has to be kept. Indeed, each result is linked with the method used to generate this particular result, and used to solve the conflicts (see Algorithm 2). During the crossover operation, at each rank, it is equiprobable to select the result from the first parent or from the second one
4) *mutation*: at each generation a random conflict resolution is applied with a user-given probability. This attempts to avoid the genetic algorithm being trapped in a local minimum, and new alternative solutions are created to maintain a relative diversity within the population. Note that the best genotype of a generation is kept unchanged.

In our experiments, we considered the following parameters for the genetic algorithm: a population size of 20 genotypes, a mutation probability $\mathbb{P}_m$ of 10% and a number of generations equals to 100. The number of generations has been kept relatively low for computational reasons. At the end of the evolution, the best solution of the population is kept as the final solution. The relative high rate of mutation is motivated by the fact that the initial population is created from the same initial solution. We assume that this high mutation probability allows the algorithm to maintain a strong diversity in the population.

We used the Lamarckian evolutionary model [16], which consists in applying a local optimization on each individual of the population at each generation. This process is used to improve the convergence of the genetic algorithm. In our framework, the local optimization applied consists in solving the most important conflict of each solution at each generation. The local optimization is only applied if the resolution of the conflict is worthwhile (i.e. increase of the global similarity (5)). Experiments have been carried out with and without local optimization. We only report here the results of the version with local optimization, as it gave better or identical results.

V. Experiments

We evaluated the four proposed approaches on synthetic datasets included in the Cluster generators package [1], and real-life datasets from the UCI repository [17]. All the datasets are real valued and show a great heterogeneity in terms of number of features, instances and classes. The synthetic datasets contain four Gaussian shaped clusters described with two attributes in `2d-4c-no0` and `2d-4c-no1`, and with ten attributes in `10d-4c-no0` and `10d-4c-no1`. These datasets are freely available on the authors website. We used four different quality indexes to assess the quality of the results: the Rand index, the Jaccard index, the Folks & Mallows index and the F-Measure. This choice was made to reduce the bias involved by the selection of a single criteria. The true class information contained in the datasets was used to compute the value of these accuracy indexes.

In all of the experiments, we used the KMeans algorithm as the base clustering method. Five instances of KMeans were randomly initialized with a number of clusters randomly picked in $[2; 10]$ to create the initial set of results. Then, the different strategies of conflict resolution were applied, each one individually, but starting from the same initial set of results (i.e. the same initial solution). The experiments were carried out 100 times for each dataset, and the results were averaged. The results are summarized in Table I. The values in the table are mean accuracy among the five methods at the end of the process, and standard deviations are into brackets. The mean global similarity value (5), which reflects the similarity and the quality of the methods, is also presented in the table.

The results indicate that the genetic approach (GR) always gives the best results on the different tested datasets. The worst conflict choice strategy (WCC) gives good results when the datasets are simple (i.e. `2d-4c-no0` and `2d-4c-no1` with four clusters with two attributes). However, when the datasets are more complex (with ten attributes and from UCI) the roulette-wheel strategy (R-WCC) gives better results than worst conflict choice strategy (WCC). It can be explain by the fact that WCC is more likely to be trapped in a local minimum, as it always tries to solve the most important conflict. Conversely, roulette-wheel strategy (R-WCC) might avoid local optima by selecting less important conflict to solve. The last strategy, random selection (SCC), always gives the poorest results, except for one dataset (`10d-4c-no1`). These results reveal that the conflict importance (7) is an important indicator to chose the conflict to solve. However, it also highlights that blindly solving the most important conflict is not always relevant. These results are consistent as the genetic resolution strategy (GR) better explores the search space, and evaluates solutions which are not explored by the other iterative methods.

The good results of the genetic solution have to be balanced by its time and space complexity. As mentioned before, the genetic solution explores more solutions than the other methods, and consequently is time and space consuming. In all our experiments, the three iterative solutions (Section IV-A) took almost always the same time to compute (around 1 to 5 seconds depending of the dataset), and the genetic resolution took almost 10 times more (around 10 to 50 seconds). These results have been obtained on a Intel Core2Duo 6300 with 4GB RAM without any code optimization. It is worth noticing

[1] http://dbkgroup.org/handl/generators/

TABLE I
EVALUATION OF THE DIFFERENT STRATEGIES ON THE ARTIFICIAL AND UCI DATASETS.

	Strategy	Γ	Rand	Jaccard	Folks & Mallows	F-Measure
2d-4c-no0	WCC	0.924	0.772 (±0.074)	0.872 (±0.045)	0.869 (±0.049)	0.824 (±0.066)
-	RCC	0.872	0.745 (±0.095)	0.857 (±0.057)	0.850 (±0.065)	0.808 (±0.080)
-	R-WCC	0.896	0.767 (±0.064)	0.871 (±0.037)	0.866 (±0.041)	0.831 (±0.045)
-	GR	**0.957**	**0.850 (±0.037)**	**0.919 (±0.021)**	**0.919 (±0.021)**	**0.890 (±0.028)**
2d-4c-no1	WCC	0.897	0.643 (±0.163)	0.788 (±0.110)	0.769 (±0.137)	0.661 (±0.247)
-	RCC	0.813	0.616 (±0.097)	0.768 (±0.066)	0.756 (±0.081)	0.693 (±0.074)
-	R-WCC	0.800	0.612 (±0.110)	0.765 (±0.074)	0.750 (±0.088)	0.692 (±0.086)
-	GR	**0.941**	**0.769 (±0.056)**	**0.871 (±0.039)**	**0.868 (±0.038)**	**0.817 (±0.052)**
10d-4c-no0	WCC	0.852	0.900 (±0.100)	0.947 (±0.055)	0.944 (±0.060)	0.913 (±0.093)
-	RCC	0.781	0.828 (±0.177)	0.898 (±0.111)	0.888 (±0.123)	0.856 (±0.148)
-	R-WCC	0.875	0.937 (±0.080)	0.967 (±0.044)	0.965 (±0.048)	0.946 (±0.073)
-	GR	**0.887**	**0.958 (±0.011)**	**0.979 (±0.006)**	**0.979 (±0.006)**	**0.967 (±0.009)**
10d-4c-no1	WCC	0.836	0.814 (±0.226)	0.890 (±0.144)	0.876 (±0.171)	0.806 (±0.296)
-	RCC	0.835	0.925 (±0.104)	0.958 (±0.062)	0.956 (±0.066)	0.939 (±0.091)
-	R-WCC	0.849	0.937 (±0.079)	0.967 (±0.043)	0.965 (±0.045)	0.954 (±0.058)
-	GR	**0.865**	**0.972 (±0.049)**	**0.985 (±0.026)**	**0.985 (±0.027)**	**0.979 (±0.037)**
iris	WCC	0.870	0.586 (±0.006)	0.763 (±0.006)	0.739 (±0.005)	0.615 (±0.010)
-	RCC	0.811	0.589 (±0.043)	0.759 (±0.029)	0.740 (±0.035)	0.634 (±0.027)
-	R-WCC	0.828	0.594 (±0.036)	0.762 (±0.024)	0.744 (±0.029)	0.636 (±0.027)
-	GR	**0.898**	**0.608 (±0.020)**	**0.771 (±0.009)**	**0.756 (±0.015)**	**0.638 (±0.018)**
wine	WCC	0.740	0.712 (±0.099)	0.831 (±0.065)	0.826 (±0.074)	0.734 (±0.120)
-	RCC	0.733	0.670 (±0.149)	0.800 (±0.100)	0.789 (±0.120)	0.658 (±0.249)
-	R-WCC	0.743	0.747 (±0.064)	0.852 (±0.043)	0.850 (±0.045)	0.772 (±0.072)
-	GR	**0.803**	**0.752 (±0.139)**	**0.856 (±0.092)**	**0.849 (±0.115)**	**0.747 (±0.250)**
segment	WCC	0.829	0.328 (±0.024)	0.519 (±0.015)	0.491 (±0.028)	0.418 (±0.031)
-	RCC	0.767	0.294 (±0.073)	0.510 (±0.062)	0.448 (±0.091)	0.363 (±0.140)
-	R-WCC	0.787	0.301 (±0.064)	0.519 (±0.051)	0.457 (±0.076)	0.388 (±0.086)
-	GR	**0.849**	**0.332 (±0.078)**	**0.551 (±0.063)**	**0.493 (±0.091)**	**0.433 (±0.101)**

that these execution times could be greatly reduced as genetic algorithms are easily parallelizable.

VI. CONCLUSION

In this paper, we addressed the problem of the collaboration between different clustering methods. One of the main challenge in collaborative clustering is the comparison of the different clustering results in order to identify precisely where they are in strong disagreement. These are referred as conflicts, and we presented four different strategies to solve these conflicts. These four different conflicts resolution strategies were described and compared. The first three are iterative and solve one conflict at each step of the algorithm. The last one uses a tuned genetic algorithm which explores the complex search-space of the potential solutions. The results are in accordance with the intuition that the genetic algorithm is more likely to find the best solution. However, the genetic algorithm is also time and space consuming, which is a critical issue for some applications. One of the good trade off between time complexity and efficiency of the results is the roulette-wheel strategy (R-WCC), where the selection of the conflict to solve is weighted by its importance. This strategy gives good results when the datasets are complex and has acceptable execution times.

In further research, we plan to explore more deeply the different strategies and try to design new ones. Background knowledge integration is also in consideration to try to improve the collaboration between the methods.

REFERENCES

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[2] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[3] R. Caruana, M. Elhaway, N. Nguyen, and C. Smith, "Meta clustering," in *IEEE International Conference on Data Mining*, 2006, pp. 107–118.

[4] E. Bae and J. Bailey, "Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity," in *IEEE International Conference on Data Mining*, 2006, pp. 53–62.

[5] I. Davidson and Z. Qi, "Finding alternative clusterings using constraints," in *IEEE International Conference on Data Mining*, 2008, pp. 773–778.

[6] M. Law, A. Topchy, and A. Jain, "Multiobjective data clustering," in *IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 424–430.

[7] R. Jiamthapthaksin, C. F. Eick, and V. Rinsurongkawong, "An architecture and algorithms for multi-run clustering," in *Computational Intelligence Symposium on Data Mining*, 2009.

[8] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," in *International Conference on Data Engineering*, 2005, pp. 341–352.

[9] A. Fred and A. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 835–850, 2005.

[10] A. Topchy, A. Jain, and W. Punch, "Clustering ensembles: models of consensus and weak partitions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1866–1881, 2005.

[11] A. Strehl and J. Ghosh, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, no. 3, pp. 583–617, 2003.

[12] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[13] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, no. 1, pp. 53–65, 1987.

[14] D. Davies and D. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, no. 1, pp. 224–227, 2000.

[15] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[16] B. Ross, "A lamarckian evolution strategy for genetic algorithms," in *The Practical Handbook of Genetic Algorithms*. CRC Press, 1999, pp. 1–16.

[17] A. Asuncion and D. Newman, "UCI machine learning repository," 2007.