A Cluster-Oriented Genetic Algorithm for Alternative Clustering

Duy Tin Truong, Roberto Battiti
University of Trento, Italy

Abstract—**Supervised alternative clusterings is the problem of finding a set of clusterings which are of high quality *and* different from a given *negative* clustering. The task is therefore a clear multi-objective optimization problem. Optimizing two conflicting objectives requires dealing with trade-offs. Most approaches in the literature optimize these objectives sequentially or indirectly, resulting in solutions which are dominated. We develop a multi-objective algorithm, called COGNAC, able to optimize the objectives directly and simultaneously and producing solutions approximating the Pareto front. COGNAC performs the recombination operator at the *cluster level* instead of the object level as in traditional genetic algorithms. It can accept arbitrary clustering quality and dissimilarity objectives and provide solutions *dominating* those of other state-of-the-art algorithms. COGNAC can also be used to generate a sequence of alternative clusterings, each of which is guaranteed to be different from all previous ones.**

Keywords-**alternative clustering; multi-objective optimization; cluster-oriented; genetic algorithm.**

I. Introduction

Given a dataset, traditional clustering algorithms often provide a single set of clusters or a single view of that dataset. On complex tasks, different interesting ways of grouping items can exist, therefore a natural requirement is to ask for *alternative* clusterings to get complementary views. There have been many techniques developed for solving the *alternative* clustering problem. In *unsupervised alternative clustering*, the algorithm automatically generates a set of clusterings of high quality and different from each other. In *supervised alternative clustering*, the algorithm allows users to direct the search by explicitly labelling some clusterings as undesired or negative. This is useful when users already know some trivial or negative clusterings of the dataset, and they ask for different - and potentially more informative - clusterings.

This paper focuses on *supervised alternative clustering*: a multi-objective optimization problem (MOP) with two objectives of clustering quality and dissimilarity. The goal is to find a representative set of Pareto-optimal solutions. In MOP, a solution is termed Pareto-optimal if there is no solution which improves at least one objective without worsening the other objectives. The Pareto front is the set of all Pareto-optimal solutions in the objective space. Most approaches in the literature only optimize these two objectives *sequentially* (optimizing one objective first and then optimizing the other one) [1], [2] or *indirectly* by some heuristics [3]. Other methods combine these two objectives into a single one and then optimize this single objective [4]. Solving a multi-objective optimization problem in the above ways can result in solutions which are not Pareto-optimal, or in a single solution or in a very limited number of solutions on the Pareto front. The user flexibility is thus limited because the trade-off between the different objectives is decided *a priori*, before knowing the possible range of solutions. The trade-off can be decided in a better way *a posteriori*, by generating a large set of representative solutions along the Pareto front and then having the user pick the favorite one among them. To deal with the above issues, we propose an explicit multi-objective algorithm, called **C**luster-**O**riented **G**e**N**etic algorithm for **A**lternative **C**lusterings (**COGNAC**), capable of (i) optimizing directly and simultaneously the clustering quality and dissimilarity, (ii) generating a sequence of alternative clusterings, each of which is different from previous ones.

The rest of this paper is organized as follows. We describe our algorithm **COGNAC** in Section II and show how our algorithm can generate a sequence of different alternative clusterings and compare the performance of our algorithm with that of other state-of-the-art algorithms in Section III.

Related Work. Bae et al. [3] propose an alternative clustering algorithm, called **COALA**, which extends the traditional agglomerative hierarchical clustering algorithm by considering also *cannot-link* constraints (generated from negative clusterings) when merging two nearest clusters. As this algorithm only considers cannot-link constraints, useful information obtained through must-link constraints is lost. In addition, the application scope of the method is limited to agglomerative clustering algorithms. To overcome the scope limitation, Davidson et al. [1] propose a method, called **AFDT** which transforms the dataset into a different space, where the negative clustering is difficult to be detected, and then use an arbitrary clustering algorithm to partition the transformed dataset. However, such transformation can destroy the dataset's characteristics. Qi et al. [2] fix this problem by finding a transformation which minimizes the Kullback-Leibler divergence between the probability distribution of the dataset in the original space and in the transformation space, under the constraint that the negative clustering should not be detected. We refer this algorithm as **AFDT2**. Those algorithms can only accept one negative clustering. Nguyen et al. [4] propose **MinCEntropy**$^{++}$, which can accept a set of $N_{\overline{C}}$ negative clusterings. **MinCEntropy**$^{++}$ finds an alternative clustering $\mathbf{C}_*$ by maximizing the weighted sum: $N_{\overline{C}}\mathbb{I}(\mathbf{C};\mathbf{X}) - \lambda \sum_{i=1}^{N_{\overline{C}}} \mathbb{I}(\mathbf{C};\overline{\mathbf{C}}_i)$ where $\mathbb{I}(\mathbf{C};\mathbf{X})$ and $\mathbb{I}(\mathbf{C};\overline{\mathbf{C}}_i)$ are the mutual information of a clustering $\mathbf{C}$ with the dataset $\mathbf{X}$ and a negative clustering $\overline{\mathbf{C}}_i$, respectively.

II. A Cluster-Oriented Genetic Algorithm for Alternative Clustering

Evolutionary algorithms (EAs) are very successful in solving multi-objective problems. They can approximate the Pareto front in a single run by maintaining a solution set or population. In each iteration, this solution set $\mathbf{Q}$ is modified by a selection step and a variation step. The former chooses only well-adapted candidates from $\mathbf{Q}$ to form a set $\mathbf{P}$ of parent solutions. Then, the latter uses $\mathbf{P}$ to produce the next generation through recombination and mutation operators. The two steps are repeated until a number of generations is reached. In this paper, we adapt one of the most effective EAs, **NSGAII** [5], for the alternative clustering problem. The complexity of the selection step in **NSGAII** is $O(P^2\Lambda)$ where P is the population size, and Λ is the number of objectives. Besides, the complexity of the variation step is $O(P\Omega)$ where Ω is the complexity of computing Λ objectives. Therefore, when the number of objectives is 2, the total complexity of **NSGAII** is $O(T(P^2 + P\Omega))$ where T is the number of generations. The application of **NSGAII** to the alternative clustering problem requires the following choices: (i) two objective functions, (ii) a genetic encoding of clusterings, (iii) recombination and mutation operators, (iv) an effective initialization scheme. In the next sections, we present the above components for our algorithm.

A. Objective Functions

Clustering Quality: We consider the Vector Quantization Error (VQE), used in **K-Means** [6], for measuring the clustering quality, to simplify comparison with **AFDT** and **AFDT2** which use **K-Means** as the base clustering method. Besides, the quality objective used in **MinCEntropy**$^{++}$ is also proportional to VQE. The VQE of a clustering solution $\mathbf{C}_t$ is the sum of the square distance of each data object $\mathbf{x}_i$ to the centroid of the cluster $\mathbf{C}_t^k$ that $\mathbf{x}_i$ belongs to. Smaller VQE values imply better clustering quality. The cost of computing VQE for a clustering $\mathbf{C}_t$ is $O(ND)$ where N is the dataset size and D is the number of dimensions of data objects.

Clustering Dissimilarity: We deploy one of the most used clustering similarity metrics: the Adjusted Rand Index (ARI) measuring the number of common objects between clusters of two clusterings [7]. The maximum value of ARI is 1 when two clusterings are identical and is around 0 when they are very different. **COGNAC** aims at minimizing the maximum similarity between the alternative clustering and the negative clusterings. The complexity of computing ARI between two clusterings is $O(N)$ where N is the dataset size. Therefore, when optimizing VQE and ARI, the total complexity of **COGNAC** is $O(T(P^2 + PND))$. In other words, it increases *linearly* with the dataset size N and the number of dimensions D of data objects.

B. Genetic Encoding of Clusterings

We use the cluster-index based representation to encode clustering solutions. In detail, a clustering solution $\mathbf{C}_t$ of N data objects $\{\mathbf{x}_i\}_{i=1}^N$ is a N-dimensional vector where $\mathbf{C}_t(i)$ is the index of the cluster containing the data object $\mathbf{x}_i$ and $1 \le \mathbf{C}_t(i) \le K$ with K is the fixed number of clusters.

C. Cluster-Oriented Recombination Operator

As the traditional recombination operators mostly perform at the *object level* whereas the clustering meaning is considered at the *cluster level*, thus the offspring produced by these operators often do not inherit good properties from their parents. Inspired by the work of Falkenauer et al. [8] for the bin packing problem, we propose a *cluster-oriented* recombination operator performing on clusters rather than on separated objects to solve the above problem. The

Algorithm 1: Cluster-Oriented Recombination Operator

```
Initialize $\mathbf{C}_o$: $\forall i \in \{1,..,N\} : \mathbf{C}_o(i) = -1$
Find a perfect matching $\mathbf{M}$.
Copy clusters $\mathbf{C}_{p_1}^i$ where $i \in \mathbf{I}$ to the offspring:
$\forall \mathbf{x}_t \in \mathbf{C}_{p_1}^i : \mathbf{C}_o(t) = i$
for $\bar{i} \in \{1,..,K\} \setminus \mathbf{I}$ do
    $\mathbf{U} = \{\mathbf{x}_j : \mathbf{x}_j \in \mathbf{X} \wedge \mathbf{C}_o(j) = -1\}$
    if $\mathbf{C}_{p_2}^{\mathbf{M}(\bar{i})} \cap \mathbf{U} = \emptyset$ then
        $\forall \mathbf{x}_t \in \mathbf{C}_{p_2}^{\mathbf{M}(\bar{i})} : \mathbf{C}_o(t) = \bar{i}$
    else
        $\forall \mathbf{x}_t \in \mathbf{C}_{p_2}^{\mathbf{M}(\bar{i})} \cap \mathbf{U} : \mathbf{C}_o(t) = \bar{i}$
// Assign orphan objects.
if rand() % 2 = 0 then
    $\forall \bar{i} \in \{1,..,K\}\setminus\mathbf{I}.\ \forall \mathbf{x}_j \in \mathbf{C}_{p_1}^{\bar{i}} \wedge \mathbf{C}_o(j) = -1 : \mathbf{C}_o(j) = \bar{i}$
else
    $\forall i \in \mathbf{I}.\ \forall \mathbf{x}_j \in \mathbf{C}_{p_2}^{\mathbf{M}(i)} \wedge \mathbf{C}_o(j) = -1 : \mathbf{C}_o(j) = i$
return $\mathbf{C}_o$
```

pseudo-code of our cluster-oriented recombination operator is presented as in Algorithm 1. Let $\mathbf{C}_{p_j}^i$ be the i-th cluster of parent $\mathbf{C}_{p_j}$. We first use the Munkres's algorithm [9] to find a perfect matching $\mathbf{M}$ between clusters $\mathbf{C}_{p_1}^i$ and $\mathbf{C}_{p_2}^{\mathbf{M}(i)}$ of two parents such that the number of common objects between them is largest. Then, the uniform crossover on clusters is performed as follows. First, we select a set $\mathbf{I}$ of $K/2$ random positions in $\{1,..,K\}$ and copy clusters $\mathbf{C}_{p_1}^i$, with $i \in \mathbf{I}$, of the first parent $\mathbf{C}_{p_1}$ to the offspring $\mathbf{C}_o$. Let $\mathbf{U}$ be the set of all unassigned objects. Then, for each remaining position $\bar{i} \in \{1,..,K\} \setminus \mathbf{I}$, we compute the set $\mathbf{C}_{p_2}^{\mathbf{M}(\bar{i})} \cap \mathbf{U}$. If this set is empty, it means that $\mathbf{C}_{p_2}^{\mathbf{M}(\bar{i})}$ is strictly included in some cluster of the first parent. Therefore, we move all objects in $\mathbf{C}_{p_2}^{\mathbf{M}(\bar{i})}$ to cluster $\bar{i}$ to avoid empty clusters. Otherwise, we simply assign the unassigned objects in $\mathbf{C}_{p_2}^{\mathbf{M}(\bar{i})}$ to cluster $\bar{i}$. After merging clusters from two parents, the remaining unassigned (orphan) objects will be assigned to the clusters of one randomly chosen parent in order to preserve good characteristics from that parent.

An example of a dataset with 12 objects is in Fig.1a. The number of clusters is 3. The clusters of two parents are as in Fig.1b, 1c. The perfect matching $\mathbf{M}$ will match: $\mathbf{C}_{p_1}^1 \rightarrow \mathbf{C}_{p_2}^{\mathbf{M}(1)=3}$, $\mathbf{C}_{p_1}^2 \rightarrow \mathbf{C}_{p_2}^{\mathbf{M}(2)=1}$, $\mathbf{C}_{p_1}^3 \rightarrow \mathbf{C}_{p_2}^{\mathbf{M}(3)=2}$. Assume that $\mathbf{I} = \{1\}$. We copy cluster $\mathbf{C}_{p_1}^1$ from $\mathbf{C}_{p_1}$, and move the unassigned objects in two clusters $\mathbf{C}_{p_2}^{\mathbf{M}(2)=1}$, $\mathbf{C}_{p_2}^{\mathbf{M}(3)=2}$ from $\mathbf{C}_{p_2}$ to the offspring as in Fig.1d. Then, we assign the orphan object 5 to the cluster $\mathbf{C}_o^2$ as in the first parent to obtain the offspring as in Fig.1e. As can be seen, the offspring inherits all good properties from its parents and converges to a correct clustering solution.

D. Neighbor-Oriented Mutation Operator

In the traditional mutation operators, usually some objects are selected and moved randomly to different clusters. However, moving an object $\mathbf{x}_i$ to a *random* cluster $\mathbf{C}^k$ can radically decrease the clustering quality when $\mathbf{x}_i$ is too far from $\mathbf{C}^k$. Also, moving too few or too many objects can keep the algorithm in local minima or destroy the solution quality, respectively. To solve these problems, we replace the traditional mutation operators with a new operator called the *Neighbor-Oriented Mutation* operator. The pseudo-code

Algorithm 2: Neighbor-Oriented Mutation Operator

```
for each object $\mathbf{x}_i \in \mathbf{X}$ do
    Let $\mathbf{X}_{nn}$ be the set of the first $\gamma$ nearest neighbours of the object $\mathbf{x}_i$.
    Pick randomly a nearest neighbor $\mathbf{x}_j \in \mathbf{X}_{nn}$.
    if the size of cluster of $\mathbf{x}_i$ is greater than 1 then
        With probability of $\rho$, move $\mathbf{x}_i$ to $\mathbf{x}_j$'s cluster:
        $\mathbf{C}_o(i) = \mathbf{C}_o(j)$
return $\mathbf{C}_o$
```

of the new mutation operator is presented in Algorithm 2. In detail, with the probability of ρ, each data object $\mathbf{x}_i$ in a cluster with size greater than 1 is moved to the cluster of one of its γ nearest neighbors $\mathbf{x}_j$. In other words, a proportion ρ of the data objects will be selected randomly and moved to the clusters of one of their nearest neighbors. Besides, we do not move objects of singleton clusters, therefore no empty clusters are created. Moving an item in this manner avoids assigning it to a very far cluster, therefore the search space is reduced significantly. However, this operator can still produce arbitrarily-shaped clusters by linking near objects, e.g., a long cluster can be formed as a chain of near objects.

Setting the values of ρ and γ too small or too large can keep the algorithm in local minima, or damage the offspring quality, respectively. Therefore, we set these parameters similarly to the Simulated Annealing method [10]. At the beginning, both ρ and γ are assigned to large values ρ_{max} and γ_{max} to allow the algorithm to explore potential regions. Then, they will be decreased in each iteration by multiplying with decay factors ρ_{dec} and γ_{dec} until their values equal the minimum values ρ_{min} and γ_{min}. This helps the algorithm exploit the most promising regions carefully. ρ_{dec} and γ_{dec} are computed such that $\rho_{max}\rho_{dec}^T = \rho_{min}$, and $\gamma_{max}\gamma_{dec}^{T/2} = \gamma_{min}$ where T is the number of generations. Note that for parameter γ, our algorithm only decreases γ in the first $T/2$ generations and keep γ as γ_{min} in the remaining generations to guarantee minimal diversification.

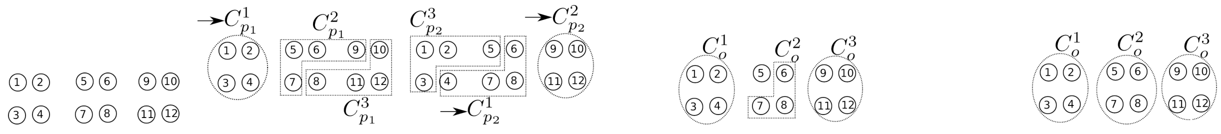
E. Initialization

We generate the initial population such that it contains both high-quality and dissimilar clusterings instead of random solutions to improve the convergence speed.

Generating dissimilar clusterings: Let P be the initial population size and K be the fixed number of clusters. We generate $P/2$ dissimilar solutions from pairs of negative clusterings and individual negative clusterings. For each pair of two negative clustering solutions $\mathbf{C}_1$ and $\mathbf{C}_2$, we first find a perfect matching $\mathbf{M}$ between their clusters. For each pair of matched clusters $\mathbf{C}_1^i \rightarrow \mathbf{C}_2^{\mathbf{M}(i)}$, we compute a common cluster $\mathbf{C}_{com}^i = \mathbf{C}_1^i \cap \mathbf{C}_2^{\mathbf{M}(i)}$, and a xor cluster $\mathbf{C}_{xor}^i = (\mathbf{C}_1^i \cup \mathbf{C}_2^{\mathbf{M}(i)}) \setminus \mathbf{C}_{com}^i$. Then, we randomly merge two nearest common clusters or xor clusters until their total number equals K to generate a dissimilar offspring $\mathbf{C}_o$. The distance between two clusters is the Euclidean distance between their centroids. The offspring is very dissimilar from its parents because in its parents, the objects in two common or xor clusters are in different clusters, but in the offspring they are placed in the same cluster.

For each negative clustering $\mathbf{C}_t$, we extract its K clusters $\{\mathbf{C}_t^i\}_{i=1}^K$ and for each cluster $\mathbf{C}_t^i$, we use **K-Means** [6] to partition this cluster into K sub-clusters $\{\mathbf{C}_t^{i_j}\}_{j=1}^K$. The remaining objects in $\mathbf{X} \setminus \mathbf{C}_t^i$ are assigned to the j-th nearest sub-cluster $\mathbf{C}_t^{i_j}$, with probability $\alpha^{-j} / \sum_{k=1}^{K} \alpha^{-k}$ to form a dissimilar offspring $\mathbf{C}_o$. The smaller α is, the more perturbed the offspring is, therefore we vary α from $\alpha_{min} = 2$ to $\alpha_{max} = 10$ to generate a diverse set of dissimilar solutions. In detail, from each cluster $\mathbf{C}_t^i$ and a value $\alpha \in \{\alpha_{min}, .., \alpha_{max}\}$, we generate $\frac{P/2 - N_{\overline{C}}(N_{\overline{C}}-1)/2}{N_{\overline{C}}K(\alpha_{max}-\alpha_{min}+1)}$ dissimilar offspring where $N_{\overline{C}}$ is the number of negative clusterings. The distance between an object and a sub-cluster is the Euclidean distance between that object and the sub-cluster centroid. The offspring generated by the above strategy is very dissimilar to its parents because the objects in each cluster $\mathbf{C}_t^i$ of their parents $\mathbf{C}_t$ are now assigned to different clusters. This strategy is inspired by Gondek et al. [11], but different in the perturbation parameter α to diversify the offspring set.

Generating high-quality clustering: We generate $\frac{P/2}{N_{\overline{C}}}$ high quality offspring from each negative clustering $\mathbf{C}_t$. First, we



(a) The dataset (b) Parent 1 (c) Parent 2 (d) Offspring after merging clusters (e) Offspring after adding orphan objects.

Figure 1: Cluster-Oriented Recombination Operator Example.

Table I: Parameter setting of **COGNAC** and datasets

(a) **COGNAC**'s parameters

# of generations	200
Population size	200
Mutation rate	0.2
$\rho_{min} - \rho_{max}$	0.1 - 0.3
$\gamma_{min} - \gamma_{max}$	10 - 40

(b) Dataset characteristics.

Dataset	CMU Face	WebKB	Birds	Flowers
Cardinality	128	1396	11766	14280
Dimensions	39	34	2	2
# of clusters	4	4	2	2

extract its K clusters $\{\mathbf{C}_t^i\}_{i=1}^K$ and compute their centroids. Then, we assign each object to its i-th nearest centroid with the probability $\alpha^{-i}/\sum_{k=1}^{K}\alpha^{-k}$ to obtain a new offspring. Similar to the procedure of generating dissimilar offspring, we also vary α from $\alpha_{min} = 2$ to $\alpha_{max} = 10$ for diversifying the high-quality offspring set.

III. Experiments

A. Datasets

In this paper, we use four datasets with a "ground truth" alternative clustering. The first two datasets are Escher images with multiple interpretations to the human eye as in Fig.2a, 2c. The *Flowers* and *Birds* images' size are 120×119 and 106×111, respectively. The RGB color space of each image is converted in the L*a*b* color space. The difference in the color of two pixels can be computed as the Euclidean distance between their a* and b* values. The images' negative clustering obtained by running **K-Means** to partition each into two clusters are shown in Fig.2b, 2d.

The third dataset is the *CMU Face* dataset on the UCI repository [12]. We extract 128 face images of 4 people (an2i, at33, boland, ch4f) taken with varying poses (straight, left, right, up). We apply PCA as in [4] to reduce the number of dimensions in each image to 39. Labelling each image by the person's name is used as the negative clustering of this dataset. The fourth dataset is the *WebKB* (www.cs.cmu.edu\~webkb) dataset, preprocessed by Cardoso-Cachopo (http://web.ist.utl.pt/acardoso/datasets/). It contains HTML documents collected mainly from four universities and classified under four groups. The labelling on groups is used the negative clustering. We apply the feature extraction algorithm in [13] to reduce the number of dimensions to 34. Table Ib summarizes these datasets.

B. Sequential Generation of Alternative Clusterings

In this section, we use a synthetic dataset with multiple clusterings and the *Flowers* dataset to show how **COGNAC** generates a set of different alternative clusterings. We also compare our algorithm with **MinCEntropy**$^{++}$.

Experimental Setup: The synthetic dataset consists of six Gaussian sub-clusters with the centroids of $\{(0,0),(6,0),(8,4),(6,8),(0,8),(-2,4)\}$ and the standard deviation of 0.5 on each coordinate. Each sub-clusters consists of 20 points. The negative clustering of this dataset obtained by **K-Means** is shown as in Fig.3a. The parameters of **COGNAC** are set as in Table Ia. For **MinCEntropy**$^{++}$, parameter m declaring that quality is m times important than dissimilarity is set to the default value of 2.5. We run both **COGNAC** and **MinCEntropy**$^{++}$ 10 times with different random seeds and record the best results. We then filter very similar solutions in the solution set returned by **COGNAC** and start from the solution with the highest quality on the filtered Pareto fronts. If the dissimilarity of the solution is satisfied, we add this solution to the negative clustering set, and rerun **COGNAC**. Otherwise, we move to the next solutions on the filtered fronts until we find a suitable one.

Experimental Results: Fig.3b-3d show the set of different alternative clusterings returned by **COGNAC** on dataset *Six-Gaussians*. It can be seen that these alternative clusterings are very different from each other and of high quality. Meanwhile, **MinCEntropy**$^{++}$ can only generate the first two alternative clusterings in Fig.3b, Fig.3c. The third solution generated by **MinCEntropy**$^{++}$ is very similar to the first solution in Fig.3b. On the *Flowers* dataset, **COGNAC** discovers the other two alternative clusterings (of the red and yellow colors) shown in Fig.4a, 4b while **minCEntropy**$^{++}$ produces alternative clusterings which are very similar to the negative clustering as shown in Fig.4c, 4d.

C. Comparison on the first alternative clustering

We compare the performance of our algorithm on the first alternative clustering with that of **COALA** [3], **AFDT** [1], **AFDT2** [2], **MinCEntropy**$^{++}$ [4] on four datasets.

Experimental Setup: The parameters of **COGNAC** are set as in Table Ia for the *Flowers* dataset. On the other smaller datasets, the number of generations is set to 100. For a specific configuration, **COALA**, and **AFDT2** can only produce one solution. In order to generate a set of different solutions from **COALA**, the parameter declaring user reference on clustering quality and dissimilarity w is changed from 0.1 to 1.0 with step of 0.1. Similarly, the trade-off parameter a of **AFDT2** is changed from 1.0 to 2.8 with step of 0.2. **AFDT** has no parameters and can only produce one solution. In **MinCEntropy**$^{++}$, we also vary its trade-off parameter m from 1 to 10 for generating a diverse set

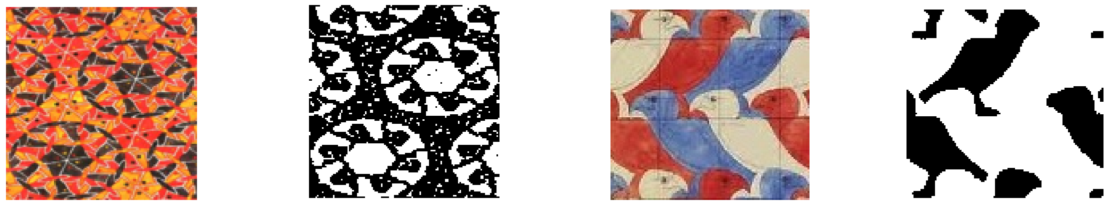

(a) The *Flowers* image (b) The *Flowers* image's negative clustering (c) The *Birds* image (d) The *Birds* image's negative clustering

Figure 2: Escher images.

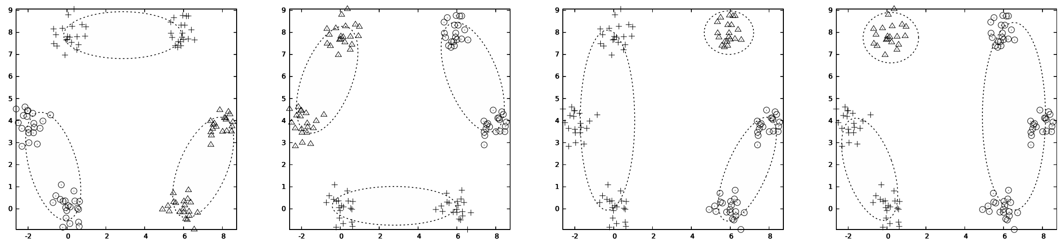

(a) The negative clustering (b) 1st alternative clustering (c) 2nd alternative clustering (d) 3rd alternative clustering

Figure 3: Alternative clusterings produced by **COGNAC** on the *Six-Gaussians* dataset.

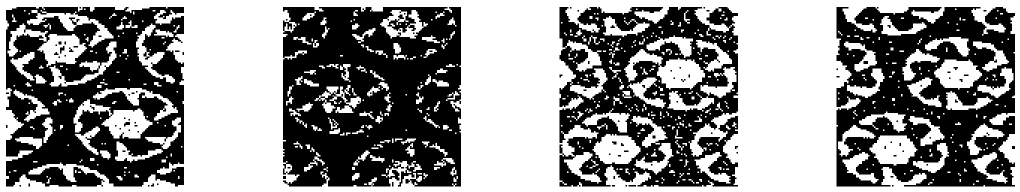

(a) 1st AC (**COGNAC**) (b) 2nd AC (**COGNAC**) (c) 1st AC (**minCEntropy**$^{++}$) (d) 2nd AC (**minCEntropy**$^{++}$)

Figure 4: Alternative clusterings (AC) on the *Flowers* dataset.

Table II: Run-time (in seconds) of five algorithms in a run.

Dataset	COALA	AFDT	AFDT2	MinCEntropy^{++}	COGNAC
CMU Face	0.16	0.60	0.01	0.01	2.49
WebKB	60.17	62.24	0.32	0.53	19.58
Birds	-	3050.70	8.45	13.25	63.43
Flowers	-	5418.13	12.54	20.50	132.87

of solutions. On each dataset, **COGNAC**, **MinCEntropy**$^{++}$ and the base algorithm **K-Means** of **AFDT** and **AFDT2** are run 10 times to eliminate the effect of local minima.

Experimental Results: Fig.5 and 6 show the performance of five algorithms on four datasets. We also plot the negative clusterings (denoted as **NC**) to present the trade-off between clustering quality and dissimilarity. For the sake of readability, we only plot some representative solutions on the Pareto front produced by **COGNAC**. On large datasets *Birds* and *Flowers*, **COALA** cannot finish after 24 hours of CPU time. As it can be observed, on most datasets our **COGNAC** provides diverse sets of high quality (in both clustering quality and dissimilarity) solutions. All solutions of **COALA**, **AFDT** and **AFDT2** are above the Pareto front of **COGNAC**. Thus, for each clustering solution produced by these three algorithms, there is always a solution produced by **COGNAC** of better quality in both objectives.

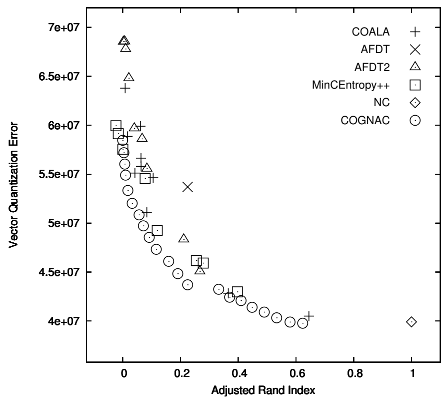
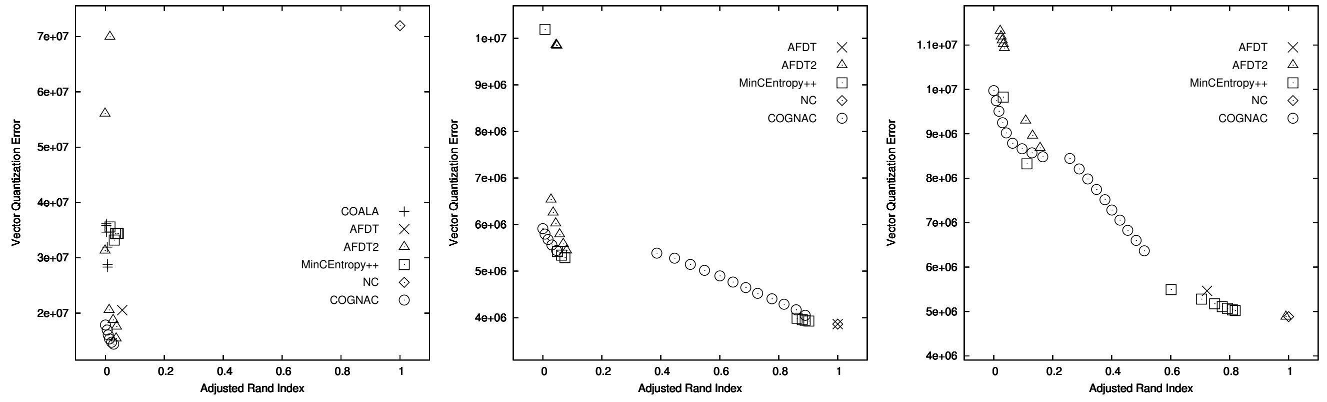


Figure 5: Performance comparison on dataset *CMU Face*.

Especially, on the *WebKB* and *Birds* datasets, our algorithm produces solutions of much higher quality. Besides, on the *WebKB* dataset, the negative clustering is not of high quality (in term of VQE), therefore **COGNAC** does not need to resolve the conflict between two objectives when optimizing them and converges to a small set of solutions. Moreover, **COGNAC** outperforms **MinCEntropy**$^{++}$ on datasets *CMU Face*, *WebKB*, and *Birds* in term of solution quality and dissimilarity. On the last dataset *Flow-*



(a) *WebKB* (b) *Birds* (c) *Flowers*

Figure 6: Performance comparison on three datasets.

ers, two algorithms explore different regions of the Pareto front. However, the solutions in the right-bottom corner discovered by **MinCEntropy**$^{++}$ are very similar to the negative clustering as presented in Section III-B. In addition, one solution of **MinCEntropy**$^{++}$ slightly dominates some solutions of **COGNAC** because **MinCEntropy**$^{++}$ optimizes the objectives at the object level instead of the cluster level as **COGNAC** does. Thus, **MinCEntropy**$^{++}$ can provide solutions which are finer than those of **COGNAC** when the number of objects is large and the number of clusters is very small (e.g., 2). Table II shows the average run-time of five algorithms in a run. Although approximating the whole Pareto front, **COGNAC** is much faster than **COALA** and **AFDT**. Compared to **AFDT2** and **MinCEntropy**$^{++}$, **COGNAC** is slower because in a single run, **COGNAC** approximates the whole Pareto front whereas the other two algorithms only compute one solution. However, the run-time of **COGNAC** is relatively small and scales up linearly with the dataset size and the number of generations.

IV. Conclusion

We have proposed an explicit multi-objective algorithm for alternative clustering, which provides better solutions than those produced by other state-of-the-art algorithms and can accept arbitrary objectives. The algorithm can work with a set of negative clusterings and can be used to generate a *sequence* of alternative clusterings. Although approximating the Pareto front in a single run, it is faster for large datasets as its run-time scales up linearly with the dataset size.

References

[1] I. Davidson and Z. Qi, “Finding alternative clusterings using constraints,” in *The eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 773–778.

[2] Z. Qi and I. Davidson, “A principled and flexible framework for finding alternative clusterings,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 717–726.

[3] E. Bae and J. Bailey, “Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity,” in *Data Mining, 2006. ICDM’06. Sixth International Conference on*. IEEE, 2006, pp. 53–62.

[4] N. Vinh and J. Epps, “mincentropy: A novel information theoretic approach for the generation of alternative clusterings,” in *The 10th International Conference on Data Mining (ICDM10)*. IEEE, 2010, pp. 521–530.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[6] S. Lloyd, “Least squares quantization in pcm,” *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[7] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[8] E. Falkenauer, “A new representation and operators for genetic algorithms applied to grouping problems,” *Evolutionary Computation*, vol. 2, no. 2, pp. 123–144, 1994.

[9] J. Munkres, “Algorithms for the assignment and transportation problems,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[10] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[11] D. Gondek and T. Hofmann, “Non-redundant clustering with conditional ensembles,” in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 70–77.

[12] A. Frank and A. Asuncion, “UCI machine learning repository.” [Online]. Available: http://archive.ics.uci.edu/ml

[13] J. Jiang, R. Liou, and S. Lee, “A fuzzy self-constructing feature clustering algorithm for text classification,” *IEEE Transactions on Knowledge and Data Engineering*, no. 99, 2009.