Improving Constrained Clustering Via Decomposition-based Multiobjective Optimization with Memetic Elitism

Germán González-Almagro[*]
DaSCI Andalusian Institute,
University of Granada
Granada, Spain
germangalmagro@ugr.es

Alejandro Rosales-Pérez
Centro de Investigación en
Matemáticas Apocada
Nuevo León, Mexico
arosalesp85@gmail.com

Julián Luengo
DaSCI Andalusian Institute,
University of Granada
Granada, Spain
julianlm@decsai.ugr.es

José-Ramón Cano
Dept. of Computer Science, EPS of
Linares, University of Jaén
Jaén, Spain
jrcano@ujaen.es

Salvador García
DaSCI Andalusian Institute,
University of Granada
Granada, Spain
salvagl@decsai.ugr.es

ABSTRACT

Clustering has always been a topic of interest in knowledge discovery, it is able to provide us with valuable information within the unsupervised machine learning framework. It received renewed attention when it was shown to produce better results in environments where partial information about how to solve the problem is available, thus leading to a new machine learning paradigm: semi-supervised machine learning. This new type of information can be given in the form of constraints, which guide the clustering process towards quality solutions. In particular, this study considers the pairwise instance-level must-link and cannot-link constraints. Given the ill-posed nature of the constrained clustering problem, we approach it from the multiobjective optimization point of view. Our proposal consists in a memetic elitist evolutionary strategy that favors exploitation by applying a local search procedure to the elite of the population and transferring its results only to the external population, which will also be used to generate new individuals. We show the capability of this method to produce quality results for the constrained clustering problem when considering incremental levels of constraint-based information. For the comparison with state-of-the-art methods, we include previous multiobjective approaches, single-objective genetic algorithms and classic constrained clustering methods.

CCS CONCEPTS

• **Computing methodologies → Genetic algorithms**; **Semi-supervised learning settings**;

[*]Correspondign author

KEYWORDS

Semi-supervised learning, constrained clustering, pairwise instance-level constraints, multiobjective optimization, memetic elitis MOEA.

ACM Reference Format:
Germán González-Almagro, Alejandro Rosales-Pérez, Julián Luengo, José-Ramón Cano, and Salvador García. 2020. Improving Constrained Clustering Via Decomposition-based Multiobjective Optimization with Memetic Elitism. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Canún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3390187

1 INTRODUCTION

Clustering constitutes a key research area in data science. It is one of the most successful techniques within the unsupervised learning paradigms, where no information other than the raw dataset is given to perform an analysis, so no information on how it should be handled is available. Traditionally unsupervised, clustering received renewed attention when new types of information were included into the task, leading to the machine learning paradigm known as Semi-supervised Learning [6]. Within this framework we are able to incorporate background information into the clustering process in order to augment the capabilities of the algorithms. This newly-added type of information can be given in the form of constraints, resulting in constrained clustering (CC). The goal of this technique is to find a partition of the dataset that satisfies a constraint set and that meets the characteristics of a classic clustering method result. It has been successfully applied in many fields of knowledge, among which it is worth mentioning: advanced robotics applications [33], applied marketing [34], obstructive sleep apnea analysis [23], terrorist sub-communities detection [31], electoral district design [4], and lane finding in GPS data [38] among others.

Many distinctions can be done within the general CC framework. Three main ways to include constraints into the clustering problem are known: cluster-level [3], instance-level [10] and feature-level CC [32]. Regarding the inclusion of constraints into the clustering process, two main approaches are discussed in the literature: (1) in *distance-based* methods the goal is to learn a new metric that reflects the information contained in the constraint set [39–41], (2) in *clustering-engine adapting* methods a clustering method is modified to be able to handle the constraints by using them as hints to guide the clustering process [11, 30, 38]. Finally, we can also distinguish between the concepts of hard [38] and soft [21] constraints. Hard constraints must necessarily be satisfied, while soft constraints are taken as a strong guide for the algorithm that uses them but can be partially satisfied in the output partition [34]. This study focuses on the Must-link (ML) and Cannot-link (CL) soft pairwise instance-level constraints, which tell us if two specific instances of a dataset must be placed in the same or in different clusters respectively.

The use of ML and CL constraints makes the constrained clustering problem **NP**-complete [12]. Multiobjective Evolutionary Algorithms (MOEAs) are presented as a promising option to solve the CC problem, not only because of their excellent capability to handle **NP**-complete problems [7], but also because of their ability to combine multiple clustering-oriented objective functions that leads to consistent and high quality partitions. Many measures can be used to guide the clustering process towards a quality solution [35], although it is often difficult to integrate them in a single function that could be optimized by a standard optimizer. Similarly, when dealing with constraints in a single-objective optimization scheme, it could be difficult to find an appropriate weighting for the constraints [16]. Multiobjective optimization schemes provide us with a powerful tool to overcome all these drawbacks.

Many MOEAs have been developed to solve problems with different characteristics and needs on exploration and exploitation; among the most successful methods are: Strength Pareto Evolutionary Algorithm 2 (SPEA2) [43], Pareto Archived Evolution Strategy (PAES) [20], Pareto Envelope-based Selection Algorithm (PESA) [9], MultiObjective Messy Genetic Algorithm (MOMGA) [37]. Although MOEAs have been successfully applied to clustering before [25, 26], very little work has been done on investigating their suitability for the CC problem; two relevant studies in this topics are the one presented in [16], where the MOCK technique (an adaptation of PESA-II [8]) is extended to include constraints, and the one in [22], where a MOEA is used to perform spectral clustering taking into account a set of constraints. However, the exploitation-exploration tradeoff featured in these proposal does not seem to be adequate for the CC problem. In [16] a k-means based initialization step and a highly biased mutation operator cause the population to converge in early stages of the exploration of the solutions space, while in [22] a classic (non evolutionary) clustering method is finally applied to get a partition from the results delivered from the used MOEA.

In this study we focus on Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) [42]. We propose Memetic Elitist-MOEA/D (ME-MOEA/D) to enhance exploitation in the exploration of the solutions space, which we have found to favor CC. We have chosen MOEA/D as our basis because of its capability to decompose the optimization of several objective functions into subproblems that can be locally optimized via a Local Search (LS) procedure. The proposed method features a CC-oriented exploitation-exploration tradeoff that is able to properly explore the solutions space and exploit certain regions of it without compromising the mentioned exploration capability. It produces partitions of the dataset as result without the need for any further clustering method, either in initialization or in post-processing. Another memetic variant of MOEA/D (conceptually different to the one proposed in this study) can be found at [1].

The rest of this paper is organized as follows: in Section 2 we introduce background related to CC and multiobjective optimization, Section 3 describes our proposal and its application to CC, the experimental setup is explained in Section 4, results and its analysis are presented in Sections 5 and 6 respectively. Finally, conclusions are discussed in Section 7.

2 BACKGROUND

2.1 Constrained Clustering

Partitional clustering consists in grouping instances of a dataset into a fixed number of clusters, which we will call k. More formally, a dataset $X = \{x_1, \cdots, x_n\}$ is composed of n instances, each one of them described by u features, and with the ith instance noted as $x_i = (x_{[i,1]}, \cdots, x_{[i,u]})$. A typical clustering algorithm assigns a class label l_i to each instance $x_i \in X$. As a result, we obtain the set of labels $L = \{l_1, \cdots, l_n\}$, with $l_i \in \{1, \cdots, k\} \forall i \in \{1, \cdots, n\}$, that effectively splits X into k non-overlapping clusters c_i to form a partition called C. The criterion used to assign an instance to a given cluster is the similarity to the rest of elements in that cluster, and the dissimilarity to the rest of instances of the dataset, which can be obtained with some kind of distance measurement [18].

In most clustering applications it is common to have some kind of information about the dataset to be analyzed. In pairwise instance-level CC this information is given in the form of pairs of instances. A constraint states whether the instances which it refers to must, or must not, be assigned to the same cluster. It is possible to obtain a better result by using this type of information than by using completely unsupervised clustering algorithms. We can now formalize the two types of constraints mentioned:

- Must-link constraints $C_{=}(x_i, x_j)$: instances x_i and x_j from X must be placed in the same cluster.
- Cannot-link constraints $C_{\neq}(x_i, x_j)$: instances x_i and x_j from X cannot be assigned to the same cluster.

The goal of CC is to find a partition (or clustering) of k clusters $C = \{c_1, \cdots, c_k\}$ of the dataset X that ideally satisfies all constraints in the constraint set. As in the original clustering problem, the sum of instances in each cluster c_i must be equal to the number of instances in X, which we have defined as $n = |X| = \sum_{i=1}^{k} |c_i|$.

2.2 Multiobjective Optimization

The multiobjective optimization problem (MOP) can be formalized as in Equation 1:

$$\begin{aligned} &\text{minimize} \;\; F(y) = (f_1(y), \cdots, f_m(y)) \\ &\textbf{s.t.} \;\; y \in \Omega \end{aligned} , \tag{1}$$

where Ω is the variable space and $F : \Omega \longrightarrow R^m$ consists of m real-valued functions (objective functions). R^m is known as the objective space and the attainable object set is defined as $\{F(y) | y \in \Omega\}$. The MOP presented in 1 is said to be continuous if $y \in R^n$ and Ω is defined as in Equation 2, where h_j are continuous functions.

$$\Omega = \{y \in R^n | h_j(y) \leq 0, j = 1, \cdots, m\}. \tag{2}$$

The goal of MOP techniques is to balance all objective functions in Equation 1, which in the general case is not trivial due to conflicts between them. The tradeoff balance found by a MOP technique can be defined in terms of Pareto optimality. Let $v, w \in R^m$, then v dominates w if and only if $f_i(v) \leq f_i(w) \forall i \in \{1, \cdots, m\}$ and if $\exists j | f_j(v) < f_j(w), j \in \{1, \cdots, m\}$. This is: v dominates w if and only if v is better than w in at least one objective function and as good as w in the rest of them.

A point $y^* \in \Omega$ is said to be Pareto optimal if there is no other point $y \in \Omega$ such that y dominates y^*; when this is fulfilled then y^* is a Pareto optimal objective vector. We refer to the set of Pareto optimal points as Pareto Set (PS), and their associated objective vectors are called the Pareto Front (PF). The aim of a MOP technique is to find the best possible PF for any given optimization problem. Please note that the above definition of MOP applies to minimization problems, although the maximization version of it can be obtained by simply reversing all inequalities.

3 CONSTRAINED CLUSTERING THROUGH MEMETIC ELITIST MOEA/D

In this section we describe in detail the Memetic Elitist MOEA/D (ME-MOEA/D) optimization scheme, which is based on the MOEA/D method [42], and its application to the CC problem.

Representation Scheme. We start by defining the representation scheme for the CC problem. There are three main representation schemes for the clustering problem when approached from the point of view of evolutionary computing: prototype-based representation, label-based representation and graph-based representation. An extensive review of the advantages and disadvantages of each one of them can be found in [14]. For the CC problem we choose the label-based representation, where each individual p_i of a population P with size $|P|$ defines a partition of the dataset X by explicitly assigning a label to each one of its instances. With this we have that $p_i = \{p_{[i,1]}, \cdots, p_{[i,n]}\}$ where $p_{[i,j]} = l | l \in \{1, \cdots, k\} \forall j \in \{i, \cdots, n\}$. This means that every position $p_{[i,j]}$ of p_i contains the label of the jth instance of the dataset X. This representation lets us keep a better control of the number of clusters and allows for a straightforward evaluation of the partition encoded in each individual of the population.

Multiobjective Problem Decomposition. The classic MOEA/D approaches the multiobjective optimization problem from the point of view of decomposition. It keeps a population P of $|P|$ individuals and a weight vector λ for each one of them. Each individual in P is referred to as p_i, and its associated weight vector as $\lambda_i = \{\lambda_{[i,1]}, \cdots, \lambda_{[i,m]}\}$. The set of weight vectors $\Lambda = \{\lambda_i, \cdots, \lambda_{|P|}\}$ is used to introduce the decomposition factor into the optimization process, where every λ_i is composed of m values such that $\lambda_{[i,j]} \geq 0 | j \in 1, \cdots, m$ and $\sum_{j=1}^{m} \lambda_{[i,j]} = 1$. This way, the m-dimensional λ-space is defined. MOEA/D decomposes the problem by approximating the PF as separated scalar problems. For the CC problem the Tchebycheff [24] decomposition scheme, shown in Equation 3, is utilized.

$$\begin{array}{ll} \text{minimize} & g^{te}(p_i|\lambda_i, z^*) = max\{\lambda_{[i,j]}|f_j(p_i) - z_j|\} \\ \textbf{s.t.} & p_i \in \Omega \end{array} , \quad (3)$$

where $z^* = (z_1^*, \cdots, z_m^*)$ is the reference point, which for a minimization problem would be defined as $z_j^* = \min\{f_j(p_i) | j \in 1, \cdots, m\}$ with $p_i \in \Omega$. With this we have that, for each Pareto Optimal point y^*, there is a weight vector λ such that y^* is optimal for Equation 3 and each optimal solution of Equation 3 is optimal for Equation 1. This way, by modifying the weight vectors $\{\lambda_1, \cdots, \lambda_{|P|}\}$, one is able to obtain different PFs.

One of the major concepts behind MOEA/D is the neighborhood in the λ-space. Let us assume that the optimal solution of $g^{te}(p_i|\lambda_i, z^*)$ should be close to the one for $g^{te}(p_i|\lambda_j, z^*)$ if λ_i and λ_j are close in the λ-space. Then, we can use any information about g^{te}s with weight vectors close to λ_i to improve $g^{te}(p_i|\lambda_i, z^*)$. In order to do so, the concept of neighborhood must be defined: the neighborhood of λ_i is composed by its γ closest weight vectors in Λ. Distances between weight vectors are calculated using the Euclidean distance.

Genetic Operators. As most evolutionary algorithms, the proposed ME-MOEA/D uses the crossover and mutation operators to introduce explore the solutions space. For every individual p_i in the population, a new individual is generated by means of the crossover operator, which combines characteristics from two already explored individuals. The mutation operator is applied to this new individual to randomly modify some of its inherited characteristics. For the CC problem, the uniform crossover operator and uniform mutation operator are used. When it comes to selecting two individuals for the crossover operator, ME-MOEA/D uses a biased selection operator which always randomly chooses a first individual p_a from the λ-neighborhood of p_i. A second individual p_b is (also randomly) chosen from the same λ-neighborhood or from the external population (EP), which is the set of non-dominated solutions, with a certain probability given by a parameter $\gamma \in (0, 1)$. This parameter γ is meant to control the exploration-exploitation tradeoff of ME-MOEA/D. When $\gamma = 0$, the unbiased selection operator is used, so p_b is always chosen from P, whereas if $\gamma = 1$, p_b is always chosen from EP.

Memetic Elitism. In the ME-MOEA/D optimization scheme, a bias towards high-quality individuals is introduced by means of an LS procedure and a biased selection operator. In ME-MOEA/D the elite of the population must be obtained in each generation. To do so the *dominance index* of each individual p_i has to be computed. The dominance index refers to the number of individuals in population P dominating p_i, so that the ν lower dominance index individuals are selected as the elite of the population. Then, an LS procedure is applied to the elite of the population, transferring its result not to the population P but to EP instead. Note that the LS procedure is a single-objective optimization method, so it cannot be applied to optimize all m objective functions at the same time. However, once again we can use the weight vectors in our favor: since each individual p_i has a weight vector λ_i associated, we can determine what would be the less significant objective function when computing $g^{te}(p_i|\lambda_i, z^*)$, which would be $f_\alpha | \alpha = \text{argmin}_{i=1}^{m}\{\lambda_{[i,1]}, \cdots, \lambda_{[i,m]}\}$. The proposed LS optimizes $f_\alpha(p_i)$, under the assumption that the classic MOEA/D optimization scheme is in charge of optimizing the rest of the functions for each individual. We have found that this helps the population to converge to quality solutions for the CC problem.

The goal of the LS procedure is to locally improve solutions (individuals p_i from P) in a non-exhaustive way. To do so, it randomly chooses an instance from the dataset (an index from an individual p_i) and iteratively assigns it to different clusters. When improvement in the fitness function is detected, the change is transferred to the solution; when there is no possible improvement, the LS is said to and the fails counter is increased. When the maximum number of fails is reached the LS procedure stops. This maximum number of fails is given in the form of a proportion $\xi \in (0,1)$ of the number of instances in the dataset X (the length of each individual p_i). Parameter ξ allows for an effective control of the exploration-exploitation tradeoff. Algorithm 1 summarizes the LS procedure describe above.

Algorithm 1: Local Search

```
Input: Dataset $X$, constraint sets $C_=$ and $C_{\neq}$, individual to
       be locally improved $p_i$, weights vector $\lambda_i$, fail
       percent $\xi$, number of clusters $k$.
// Find the least significant cost function index
$\alpha \leftarrow \text{argmin}_{i=1}^{m}\{\lambda_{[i,1]},\cdots,\lambda_{[i,m]}\}$
$fails \leftarrow 0$
while $improvement$ or $fails < n \times \xi$ do
    $improvement \leftarrow$ false
    $j \leftarrow$ RandInt($\{1,\cdots,n\}$)
    // Random shuffle labels set
    $RSL \leftarrow$ RandomShuffle($\{1,\cdots,K\}$)
    for $l \in RSL$ and while not $improvement$ do
        $p_i' \leftarrow p_i$
        // Move instance $i$ to the cluster
           associated with label $l$
        $p_{[i,j]}' \leftarrow l$
        if $f_\alpha(p_i') < f_\alpha(p_i)$ then
            $p_i \leftarrow p_i'$
            $improvement \leftarrow$ true
        end
    end
    if not $improvement$ then
        $fails \leftarrow fails + 1$
    end
end
return $p_i$
```

Target Functions for the CC Problem. We have found that the three target functions ($m = 3$) described below work best for the CC problem. The *Davies-Bouldin* function [13] is the ratio of the within-cluster mean distance to the between-cluster separation. We use it to keep clusters as compact and separated from each other as possible. The within-cluster mean distance for a cluster c_i is defined in Equation 4.

$$\overline{c_i} = \frac{1}{|c_i|}\sum_{x_j \in c_i} \|x_j - \mu_i\|^2, \tag{4}$$

where μ_i is the centroid of cluster c_i. The distance between two clusters c_i and c_j is computed as $d_{i,j} = \|\mu_i - \mu_j\|^2$, which is the Euclidean distance between their centroids. Then, we define $R_i = \max_{j,j\neq i}\{(\overline{c_i} + \overline{c_j})/d_{i,j}\}$, so that the Davies-Bouldin function for a partition C and a dataset X can be defined as in Equation 5. A partition that minimizes DB(C,X) will result in a high quality clustering.

$$\text{DB}(C,X) = \frac{1}{k}\sum_{i=1}^{k} R_i. \tag{5}$$

Secondly, the *Connectedness* function is utilized to keep the number of clusters of the generated solutions under control [15]. Given an instance of the dataset, it is related to the number of neighboring instances that are in different clusters. It is computed as in Equation 6 for a given partition C and a dataset X.

$$\text{Conn}(C,X) = \sum_{i=1}^{n}\left(\sum_{j=1}^{\epsilon} x_{i,nn_i(j)}\right), \tag{6}$$

where $nn_i(j)$ is the jth neighbor of instance x_i, ϵ is a parameter that defines the size of the neighborhood for every instance and $x_{i,nn_i(j)}$ is computed as in Equation 7. A good partition would minimize Conn(C,X).

$$x_{i,nn_i(j)} = \begin{cases} \frac{1}{j} & \nexists c_k | x_i, nn_i(j) \in c_k \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

Lastly, the *Infeasibility* function is used to integrate constraints into the clustering process. It simply measures the number of violated constraints in a given partition and can be computed for a given partition C, a set of ML constraints $C_=$, and a set of CL constraints $C_{\neq}$ as in Equation 8.

$$\text{Infs}(C, C_=, C_{\neq}) = \sum_{i=0}^{n}\sum_{j=0}^{n} V(C_=(x_i,x_j)) + V(C_{\neq}(x_i,x_j)), \tag{8}$$

where $V(\cdot)$ is a function that returns 1 if the constraint given as argument exists and is violated and 0 otherwise. It is clear that Infs$(C, C_=, C_{\neq})$ is a function to minimize.

The Algorithm. In each generation the current population is composed of the best solution found so far for each individual p_i (subproblem), and an external population (EP) is maintained to store all non-dominated solutions found. It is also necessary to maintain the target function values for each individual in the population; to this purpose, ME-MOEA/D uses the set $\{FV_1,\cdots,FV_{|P|}\}$, where FV_i is the F-value of individual p_i. Algorithm 2 summarizes the overall ME-MOEA/D optimization process.

For the CC problem, we want to minimize all three objective functions described above in this section, so we initialize z such that it is always updated in the first generation of the optimization process, before it is used in g^{te} (line [14] in Algorithm 2). We set it to $z = [\infty,\infty,\infty]$ during the initialization process (line [4] in Algorithm 2). Regarding the set of weight vectors Λ, we generate it according to a normal distribution to ensure a good spread of values. This will cause average the distances in the λ-neighborhoods to be similar for every λ_i, and hence for every p_i. Note that these are problem-specific initialization methods.

Algorithm 2: Memetic Elitist MOEA/D

```
Input: Dataset $X$, constraint sets $C_=$ and $C_{\neq}$, size of the
       population $|P|$, size of the $\lambda$-neighborhoods $\delta$,
       number of clusters $k$, selection operator bias
       probability $\gamma$, size of the elite of the population $\nu$, set
       of weight vectors $\Lambda$.
// Initialization Step
$EP = \emptyset$
Obtain the $\lambda$-neighborhood for every $\lambda_i \in \Lambda$ as
  $\{\lambda_i^1, \cdots, \lambda_i^\delta\}$ by computing the Euclidean distances in $\Lambda$.
  Then, for every $i \in \{i, \cdots, |P|\}$ set $\Delta(i) = \{i_1, \cdots i_\delta\}$
Generate the initial population $P = \{p_1, \cdots, p_{|P|}\}$ and get
  their fitness values as $FV_i = F(p_i) \forall i \in \{i, \cdots, |P|\}$
Initialize $z = [z_i, \cdots, z_m]$
while stopping criteria are not met do
    for $i \in \{i, \cdots, |P|\}$ do
        // Selection Operator
        if RandInt(0, 1) $< \gamma$ then
            Select $p_a$ randomly from the $p_i$ $\lambda$-neighborhood
              and select $p_b$ randomly from EP
        else
            Select $p_a$ and $p_b$ randomly from the $p_i$
              $\lambda$-neighborhood.
        end
        // Crossover Operator
        Obtain a new individual $p_{new}$ by applying the
          uniform crossover operator to $p_a$ and $p_b$.
        // Mutation Operator
        Mutate $p_{new}$ by applying the uniform mutation
          operator
        // Update reference point $z$
        For each $j \in \{1, \cdots, m\}$ set $z_j = f_j(p_{new})$ if
          $f_j(p_{new}) < z_j$
        // Update the neighborhood of $p_i$
        For each $j \in \Delta(i)$ set $p_j = p_{new}$ and $FV_j = F(p_{new})$
          if $g^{te}(p_{new}|\lambda_j, z) \leq g^{te}(p_j|\lambda_j, z)$
        // Update the external population EP
        Remove from EP all vectors dominated by $F(p_{new})$
        Add $F(p_{new})$ to EP if it is not dominated by any
          vector in EP
        // Memetic Elitism
        Obtain the indices of the best $\nu$ individuals in $P$
          according to the dominance index as EliteIndices
        for $j \in$ EliteIndices do
            Apply LS to $p_j$ having into account $\lambda_j$ to get an
              improved individual $p_{LS}$
            Remove from EP all vectors dominated by
              $F(p_{LS})$
            Add $F(p_{LS})$ to EP if it is not dominated by any
              vector in EP
        end
    end
end
return EP
```

4 EXPERIMENTAL SETUP AND CALIBRATION

For our experiments, we will compare the results obtained by the proposed method and five other well-known approaches to CC over 15 datasets and 3 constraint sets for each one of them. Most of these datasets can be found at the Keel-dataset repository[1] [36], though some of them have been obtained via `scikit-learn` python package[2] [27]. Table 1 displays a summary of every dataset.

Table 1: Summary of datasets used for the experiments.

Name	No. Instances	No. Classes	No. Features
Appendicitis	106	2	7
Breast Cancer	569	2	30
Bupa	345	2	6
Haberman	306	2	3
Heart	270	2	13
Ionosphere	351	2	33
Iris	150	3	4
Monk2	432	2	6
Newthyroid	215	3	5
Pima	768	2	8
Saheart	462	2	9
Sonar	208	2	60
Soybean	47	4	35
Spectfheart	267	2	44
Wdbc	569	2	30

Classification datasets are commonly used in the literature to test CC algorithms; the reason behind this is that they enable us to generate constraints with respect to the true labels. We will use the method proposed in [38] to generate artificial constraint sets. This method consists of randomly selecting two instances of a dataset, then comparing its labels, and finally setting an ML or CL constraint depending on whether the labels are the same or different.

We will generate, for each dataset, three different sets of constraints (CS_{10}, CS_{15} and CS_{20}) that will be associated with three small percentages of the size of the dataset: 10%, 15% and 20%. With n_f being the fraction of the size of the dataset associated with each of these percentages, the formula $(n_f(n_f - 1))/2$ tells us how many artificial constraints will be created for each constraint set; this number is equivalent to how many edges a complete graph with n_f vertices would have. Table 2 shows the number of constraints of each type obtained for each dataset.

4.1 Evaluation Method

Since we have the true labels associated with each of the datasets, we can use them to evaluate the results provided by each method. We will use the Adjusted Rand Index (ARI) to measure the accuracy of the predictions resulting from each method we test [17]. The basic Rand Index computes the degree of agreement between two partitions C_1 and C_2 of a given dataset X. C_1 and C_2 are viewed as collections of $n(n-1)/2$ pairwise decisions [29].

For each pair of instances x_i and x_j in X, a partition assigns them to the same cluster or to different clusters. We take a as the number of pairings where x_i is in the same cluster as x_j in both C_1 and C_2, and b as the opposite event (x_i and x_j are in different clusters in C_1 and C_2). Then, the degree of similarity between C_1 and C_2 is calculated as $\text{Rand}(C_1, C_2) = (a + b)/(n(n-1)/2)$.

[1]https://sci2s.ugr.es/keel/category.php?cat=clas
[2]https://scikit-learn.org/stable/datasets/index.html

Table 2: Number of constraints used in experiments.

Dataset	CS_{10}		CS_{15}		CS_{20}	
	ML	CL	ML	CL	ML	CL
Appendicitis	39	16	71	49	164	67
Breast Cancer	876	720	1965	1690	3487	2954
Bupa	323	272	699	627	1201	1145
Haberman	304	161	634	401	1135	756
Heart	178	173	396	424	744	687
Ionosphere	330	300	732	646	1299	1186
Iris	26	79	82	171	136	299
Monk2	473	473	979	1101	1917	1824
Newthyroid	108	123	270	258	449	454
Pima	1604	1322	3595	3075	6452	5329
Saheart	595	486	1292	1123	2330	1948
Sonar	100	110	245	251	436	425
Soybean	4	6	6	22	12	33
Spectfheart	233	118	543	277	965	466
Wdbc	840	756	1925	1730	3472	2969

The ARI is a corrected-for-chance version of the Rand Index. This correction uses the expected similarity of all comparisons between clusterings specified by a random model to set up a baseline. The ARI is computed as in Equation (9).

$$\mathrm{ARI}(C_1, C_2) = \frac{\mathrm{Rand}(C_1, C_2) - \text{Expected Index}}{\text{Maximum Index} - \text{Expected Index}}, \tag{9}$$

where Maximum Index is expected to be 1 and Expected Index is the already mentioned expected degree of similarity with a random model. It is easy to see that $\mathrm{ARI}(C_1, C_2) \in [-1, 1]$, such that an ARI value close to 1 means a high degree of agreement between C_1 and C_2, a positive value close to 0 means no agreement and a value smaller that 0 means that the $\mathrm{Rand}(C_1, C_2)$ is less than expected when comparing with random partitions. To summarize, the higher the ARI, the greater the degree of similarity between C_1 and C_2. For more details on ARI see [17].

Our goal is to quantify the quality of the solutions obtained as a result of the methods presented in this study. To accomplish this task we just set one of the two partitions given as parameters to the ARI function as the ground truth labels. We also include the *Unsat* measure, that refers to the percentage on unsatisfied constraints.

4.2 Validation of results

In order to validate the results that will be presented in Section 5, we will use Bayesian statistical tests, instead of the classic Null Hypothesis Statistical Tests (NHST). In [2] we find an in-depth analysis of the disadvantages of NHST, and a new model is proposed for carrying out the comparisons researchers are interested in. *"In a nutshell: NHST do not answer the question we ask"*. To put it clear, the disadvantages of the NHST that the authors highlight in [2] are based on the trap of black-and-white thinking, this is: to reject, or not to reject?

Most of the problems of NHST can be avoided by using Bayesian tests instead of NHST. In particular, we will use the Bayesian sign test, which is the Bayesian version of the frequentist non-parametric sign test. To make use of it, we will employ the R package `rNPBST`, whose documentation and guide can be found in [5].

The Bayesian sign test is based on obtaining the statistical distribution of a certain parameter ρ according to the difference between the results, under the assumption that said distribution is a Dirichlet distribution. To get the distribution of ρ we count the number of times that $A - B < 0$, the number of times where there are no significant differences, and the number of times that $A - B > 0$. In order to identify cases where there are no significant differences, we define the region of practical equivalence (rope) $[r_{\min}, r_{\max}]$, so that $P(A \approx B) = P(\rho \in \text{rope})$. Using these results we calculate the weights of the Dirichlet distribution and sample it to get a set of triplets with the form described in Equation 10.

$$\begin{array}{l}[P(\rho < r_{\min}) = P(A - B < 0),\quad P(\rho \in \text{rope}) \\ P(\rho > r_{\max}) = P(A - B > 0)]\end{array}. \tag{10}$$

4.3 Calibration

We describe the parameters used in them in this section. Parameters used for the proposed ME-MOEA/D method are described in Table 3. For the two evolutionary algorithms considered in this study (ME-MOEA/D and MOCK), the stopping criterion for the optimization process is the maximum number of target function evaluations, which will be set to 300,000.

Table 3: Parameters setup for ME-MOEA/D.

Parameter	Meaning	Value
$\lvert P \rvert$	Internal population size	100
δ	Size of the λ-neighborhoods	10
γ	Selection operator bias probability	0.25
ν	Elite population size	25
ξ	Fail percent for the LS procedure	0.1
ϵ	Size of the Connectivity measure neighborhood	10
k	Output partition number of clusters	No. Classes (Table 1)

We compare our proposed method with five state of the art methods described below. Table 4 summarizes parameter setting.

MOCK: This method is based on the PESA-II multiobjective optimization strategy, whose selection operator is based on crowding [8]. MOCK optimizes compactness and connectedness, and it is extended to the CC problem by adding the unsatisfied number of constraints to the objective functions. It initializes the population with K-means and minimum spanning trees based methods [16].

COP-kmeans: This method introduces a modification to the assignment rule of instances to clusters of the K-means algorithm so that an instance can be assigned to a cluster only if the assignment does not violate any constraint [38].

LCVQE: The Linear Constrained Vector Quantization Error algorithm introduces a modification of the cost function of CVQE to make it less computationally complex [28].

TVClust: Two Views Clustering incorporate the constraints into the clustering problem by making a soft interpretation of them. The authors model dataset and constraints in different ways, perform clustering methods and try to find a consensus between them [19].

RDPM: Relation Dirichlet Process - Means is a deterministic derivation of the TVClust model. This method can be viewed as an extension of K-means that includes side information (constraints) and has the property that the number of clusters (k) does not need to be specified.[19].

Table 4: Parameters setup used for previous proposals.

Method name	Parameters name and values
MOCK	`internal_population_size` = 10 `external_population_size` = 100 `grid_size` = 16 `neighborhood_size` = 10
COPKM	`max_iter` = 300 `tolerance` = $1 * 10^{-4}$ `init_mode` = ``rand''
LCVQE	`max_iter` = 300; `initial_centroids` = $\emptyset$
RDPM	`max_iter` = 300; $\xi_0 = 0.1$; $\xi_{\text{rate}} = 1$ λ is calculated on the basis of the mean distances in the dataset.
TVClust	`max_iter` = 300; $\alpha_0 = 1.2$ `stop_threshold` = $5 * 10^{-4}$

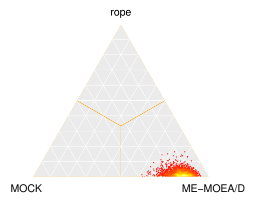
Parameter values have been assigned following the guidelines of the original creators of the different proposals. Given that the purpose of this work is to draw a fair comparison between the algorithms and assess their robustness in a common environment with multiple datasets, we have not included a tuning step to maximize any particular performance metric.

5 EXPERIMENTAL RESULTS

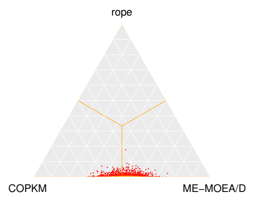
In this section we present experimental results for all datasets and constraint sets presented in Section 4. Note that some of the previous proposals are not deterministic algorithms, so the results may vary between runs. To lessen the effect this may cause we present average results of 25 runs in our tables. Please note that when COPKM is not able to produce an output partition we assign to that particular run the worst possible benchmark values. Table 5 shows the results obtained with the methodology described before.

Since multiobjective optimization methods return a PF as the solution and not a single individual, we need to choose the one that best meets our requirements. This is typically done by a problem-specific method. As our goal is to prove that ME-MOEA/D is able to provide quality results for the CC problem, we will make use of the known labels for each dataset to select the best partition from the PF in each case; this is represented in result tables as *ARI** and *Unsat**. However, we cannot consider this rigorous enough in terms of real-world applications, so we also use the simplest PF selection method: choose the individual whose Euclidean distance to the vector of ideal values is the smallest.

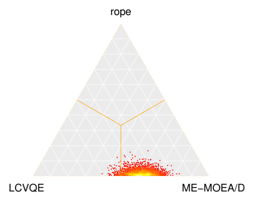
In Table 5, results for the CS_{10} constraint sets are firstly presented. We can see how ME-MOEA/D is able to provide better average results than the other methods, even with the lowest level of constraint-based information. It is worth noting how MOCK is able to provide better results in terms of the solution closest to the ideal objective values. Results obtained for COPKM should also be highlighted, given that it is able to produce partitions close to the ground-truth, at the expense of not being able to produce results for the majority of the cases. It is for the CS_{15} constraint sets that we start to see major differences between ME-MOEA/D and MOCK in terms of ability to find partitions closer to the ground-truth. ME-MOEA/D produces peak performance of almost twice the quality of MOCK, while in real-world performance both methods are similar. We also see how ME-MOEA/D is able to scale the quality of the results with respect to the amount of constraint-based information provided, which is indicative of a proper constraint-integration scheme. COPKM continues to produce very high-quality results in most cases in which it is able to deliver an output partition. Finally, for the CS_{20} constraint sets, we continue to observe the trend already present in the previous table: ME-MOEA/D outscales MOCK in terms of *Best ARI*. Nonetheless, in this case ME-MOEA/D is also capable of producing better real-world results. Again, this is indicative of a good constraint-integration scheme.



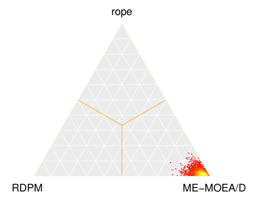
(a) MOCK vs ME-MOEA/D



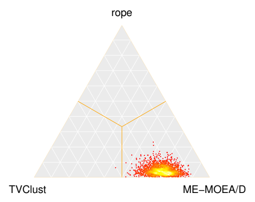
(b) COPKM vs ME-MOEA/D



(c) LCVQE vs ME-MOEA/D



(d) RDPM vs ME-MOEA/D



(e) TVClust vs ME-MOEA/D

Figure 1: Heatmaps for the comparison of ME-MOEA/D with previous proposals.

6 STATISTICAL ANALYSIS OF RESULTS

One of the major advantages of the Bayesian sign test is that we can obtain a very illustrative visual representation of its results. We can produce a representation of the triplet set in the form of a heatmap, where each triplet constitutes one point whose location is given by barycentric coordinates. Figure 1 shows heatmaps comparing ARI results obtained by our proposal and the rest of the methods considered in this study. We take results produced by the proposed method as the *A* set of results in Equation 10, and the set of results obtained the other methods as *B*. Particularly, we compare the ARI (or ARI* when available) measure, which is a measure to maximize.

Table 5: Experimental results obtained for all constraint sets and all datasets

Constraint Level	Dataset	ME-MOEA/D				MOCK				COPKM		LCVQE		RDPM		TVClust	
		ARI*	Unsat*(%)	ARI	Unsat(%)	ARI*	Unsat*(%)	ARI	Unsat(%)	ARI	Unsat(%)	ARI	Unsat(%)	ARI	Unsat(%)	ARI	Unsat(%)
CS_{10}	Appendicitis	**0.672**	2.909	0.465	8.000	0.104	32.727	-0.024	36.364	-	-	0.450	7.273	0.267	29.273	0.211	31.818
	Breast Cancer	0.602	17.494	0.106	39.436	0.586	19.987	0.581	20.163	-0.604	80.000	**0.917**	3.446	0.502	24.373	0.016	44.524
	Bupa	0.054	34.353	0.004	40.235	0.023	40.840	0.003	40.605	-	-	**0.149**	32.134	-0.006	47.361	-0.009	48.874
	Haberman	**0.373**	18.581	0.199	24.989	0.121	32.946	0.071	33.548	-0.807	90.000	0.019	34.108	0.088	41.806	0.332	22.430
	Heart	0.058	38.974	0.019	38.348	0.025	41.652	0.001	40.228	-	-	0.006	35.442	0.032	51.083	**0.223**	34.758
	Ionosphere	**0.535**	18.254	0.196	34.317	0.272	30.127	0.224	31.143	-	-	0.060	34.286	0.202	38.873	0.004	47.460
	Iris	0.758	6.667	0.683	10.476	0.765	5.905	0.588	15.619	-0.105	50.000	**0.769**	2.857	0.567	19.333	0.511	15.524
	Monk2	0.101	36.469	0.008	41.670	0.303	32.326	0.301	32.410	0.982	0.000	**0.575**	14.059	0.092	42.125	0.096	46.628
	Newthyroid	0.432	21.472	0.160	37.749	0.562	20.346	0.515	22.597	-	-	**0.791**	3.766	0.370	29.394	0.695	12.165
	Pima	0.211	34.190	0.040	42.618	0.119	38.489	0.117	38.510	-	-	0.034	46.036	0.075	44.874	**0.803**	9.686
	Saheart	0.260	28.640	0.105	37.761	0.106	38.187	0.077	38.742	0.974	0.000	0.020	42.969	0.028	45.680	**0.367**	28.529
	Sonar	0.037	33.810	0.000	38.667	0.069	37.238	0.005	38.476	-	-	**0.109**	25.714	0.007	42.667	0.000	52.381
	Soybean	0.279	8.000	0.023	42.000	0.604	2.000	0.440	14.000	0.613	0.000	0.551	9.000	**0.635**	13.000	0.000	60.000
	Spectfheart	**0.471**	14.017	0.206	23.647	-0.035	37.892	-0.065	38.405	-	-	0.014	31.624	-0.118	49.202	0.000	33.618
	Wdbc	0.643	15.326	0.130	39.662	0.601	19.373	0.596	19.612	-	-	**0.917**	3.759	0.502	24.875	0.011	46.842
	Average	**0.366**	21.943	0.156	33.305	0.281	28.669	0.228	30.694	-0.529	74.666	0.358	21.764	0.216	36.261	0.217	35.682
CS_{15}	Appendicitis	**0.991**	0.333	0.462	23.000	0.338	27.333	0.101	33.500	-	-	0.379	14.167	0.335	29.333	0.261	35.500
	Breast Cancer	0.808	8.788	0.364	29.472	0.567	20.706	0.567	20.706	**1.000**	0.000	0.979	1.231	0.502	23.803	0.096	41.869
	Bupa	0.308	28.507	0.145	36.501	0.034	42.112	0.028	41.614	**1.000**	0.000	-0.002	46.591	-0.006	47.474	0.092	43.220
	Haberman	0.599	16.097	0.333	25.758	0.110	36.309	0.103	36.058	**1.000**	0.000	0.001	43.652	0.079	48.329	0.973	0.966
	Heart	0.446	22.561	0.253	31.829	0.094	39.951	0.075	40.122	**1.000**	0.000	0.053	45.244	0.030	48.512	0.435	28.024
	Ionosphere	0.781	9.463	0.485	22.830	0.357	28.476	0.355	28.273	**1.000**	0.000	0.004	47.968	0.261	36.023	0.004	46.814
	Iris	0.531	18.103	0.441	18.893	0.912	3.399	0.912	3.399	-	-	**0.941**	0.791	0.543	22.372	0.524	17.470
	Monk2	0.616	17.288	0.195	38.625	0.358	31.856	0.353	31.510	**1.000**	0.000	0.671	15.240	0.144	37.240	0.098	44.827
	Newthyroid	0.600	15.114	0.288	33.712	0.634	16.818	0.634	16.818	-	-	0.835	4.299	0.455	24.034	**0.848**	6.345
	Pima	0.384	29.187	0.140	39.481	0.187	38.099	0.187	38.099	-	-	0.022	48.036	0.075	44.423	**0.803**	9.676
	Saheart	0.457	24.116	0.246	33.565	0.131	38.377	0.116	38.708	**1.000**	0.000	0.017	48.406	0.030	47.081	0.808	9.321
	Sonar	**0.502**	18.468	0.222	31.573	0.181	34.758	0.162	34.879	-	-	0.037	39.375	0.014	43.306	0.000	50.605
	Soybean	**0.675**	5.714	0.271	28.571	0.654	10.714	0.652	10.714	0.656	0.000	0.550	5.000	0.593	11.786	0.000	78.571
	Spectfheart	0.771	8.073	0.388	20.634	-0.002	40.171	-0.019	40.073	**0.983**	0.000	0.167	34.390	-0.116	50.049	0.000	33.780
	Wdbc	0.831	8.202	0.503	24.306	0.593	19.962	0.592	19.934	**1.000**	0.000	0.931	3.557	0.502	24.897	0.019	46.435
	Average	**0.62**	15.334	0.315	29.25	0.343	28.602	0.321	28.960	0.309	33.333	0.372	26.529	0.229	35.910	0.330	32.894
CS_{20}	Appendicitis	**0.983**	0.519	0.801	7.013	0.321	29.004	0.255	29.004	-	-	0.050	33.766	0.380	22.424	0.419	21.472
	Breast Cancer	0.872	6.350	0.531	22.295	0.560	21.428	0.559	21.376	**1.000**	0.000	0.944	2.748	0.502	24.018	0.098	41.413
	Bupa	0.596	18.193	0.219	36.837	0.058	42.600	0.058	42.600	**1.000**	0.000	-0.002	49.403	-0.006	49.642	0.093	44.625
	Haberman	0.786	8.874	0.633	15.463	0.178	34.976	0.176	34.860	**1.000**	0.000	0.001	47.065	0.102	43.934	**1.000**	0.000
	Heart	0.733	11.502	0.147	37.540	0.080	41.174	0.075	41.104	**1.000**	0.000	0.042	45.716	0.034	45.856	0.488	24.626
	Ionosphere	0.858	6.406	0.653	16.193	0.380	28.700	0.379	28.676	**1.000**	0.000	-0.002	49.336	0.324	31.899	0.004	47.630
	Iris	0.824	6.207	0.659	12.046	0.887	3.678	0.881	3.724	-	-	**0.941**	2.299	0.597	19.954	0.573	19.517
	Monk2	0.797	9.366	0.283	34.408	0.336	31.660	0.336	31.660	**1.000**	0.000	0.883	5.667	0.196	37.616	0.199	39.695
	Newthyroid	0.677	12.890	0.536	20.111	0.650	15.083	0.650	15.083	-	-	0.801	10.443	0.445	25.759	**0.924**	2.813
	Pima	0.469	25.343	0.303	32.744	0.168	38.555	0.167	38.481	**1.000**	0.000	0.010	49.079	0.076	44.487	0.901	4.820
	Saheart	0.587	18.784	0.350	30.061	0.107	40.482	0.104	40.468	**1.000**	0.000	0.009	49.673	0.030	46.697	0.808	9.187
	Sonar	0.715	11.359	0.437	24.065	0.171	38.653	0.154	38.560	**1.000**	0.000	-0.004	49.593	0.043	41.777	0.000	49.361
	Soybean	**0.941**	3.556	0.459	28.889	0.793	6.667	0.570	17.778	-0.609	80.000	0.560	8.889	0.641	18.444	0.000	73.333
	Spectfheart	0.831	5.856	0.430	20.154	0.016	42.027	-0.021	42.236	**1.000**	0.000	-0.004	50.035	-0.123	52.075	0.000	32.565
	Wdbc	0.873	5.965	0.442	26.415	0.588	20.466	0.588	20.466	**1.000**	0.000	0.972	1.553	0.522	24.468	0.097	41.630
	Average	**0.769**	10.078	0.458	24.282	0.352	29.010	0.328	29.738	0.492	25.333	0.346	30.351	0.250	35.27	0.373	30.179

Figures 1a, 1d and 1e feature heatmaps comparing the results obtained by ME-MOEA/D with those obtained by MOCK, RDPM and TVClust respectively and with $rope = [-0.01, 0.01]$. We can draw the same conclusion for all three of them: there exists clear statistical evidence in favor of ME-MOEA/D, given that the majority of the triplets are represented in the right third of the triangle. In Figure 1c we see the heatmap comparing ME-MOEA/D with LCVQE; similarly to the three diagrams mentioned before, this heatmap presents statistical evidence in favor of ME-MOEA/D, although a small number of triplets are represented in the left third of the triangle, assigning a low (but existent) probability to LCVQE performing better than ME-MOEA/D. Lastly, Figure 1b shows the comparison of ME-MOEA/D with COPKM. In this case, the Bayesian sign test does not give a clear advantage to any method, although it certainly suggests that the two methods are never equivalent as no triplets are represented in the top third. This can also be said of the other four diagrams.

7 CONCLUSIONS

In this study we approach the CC problem from the point of view of multiobjective optimization. We develop a memetic elitist evolutionary algorithm based on decomposition to bias the exploration of the solutions space towards quality solutions for the CC problem. To achieve this, we take MOEA/D as the basis for our proposal. We introduce memetic elitism into it by means of an LS procedure applied to the elite of the population that only transfers its results to the non-dominated solution archive. We also propose a selection operator that takes into consideration the aforementioned archive, so that the target function evaluations spent in the LS procedure help the population to converge to a good spread of quality solutions.

We use Bayesian statistical tests to prove that the proposed ME-MOEA/D method is capable of finding high-quality results that in most cases outperform the current state-of-the-art and that rival them in particular cases. We also prove the suitability of the constraint-integration scheme by showing how the quality of the results increases with the amount of constraint-based information.

ACKNOWLEDGMENTS

Our work has been supported by the research project TIN2017-89517-P and PP2016.PRI.I.02.

REFERENCES

[1] Ahmad Alhindi, Abrar Alhindi, Atif Alhejali, Abdullah Alsheddy, Nasser Tairan, and Hosam Alhakami. 2019. MOEA/D-GLS: a multiobjective memetic algorithm using decomposition and guided local search. *Soft Computing* 23, 19 (2019), 9605–9615.

[2] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. 2017. Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. *The Journal of Machine Learning Research* 18, 1 (2017), 2653–2688.

[3] P. S. Bradley, K. P. Bennett, and A. Demiriz. 2000. *Constrained K-Means Clustering.* Technical Report. MSR-TR-2000-65, Microsoft Research.

[4] Andreas Brieden, Peter Gritzmann, and Fabian Klemm. 2017. Constrained clustering via diagrams: A unified theory and its application to electoral district design. *European Journal of Operational Research* 263, 1 (2017), 18–34.

[5] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. 2017. rNPBST: An R Package Covering Non-parametric and Bayesian Statistical Tests. In *International Conference on Hybrid Artificial Intelligence Systems.* Springer, 281–292.

[6] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. 2010. *Semi-Supervised Learning* (1st ed.). The MIT Press.

[7] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. 2007. *Evolutionary algorithms for solving multi-objective problems.* Vol. 5. Springer.

[8] David W Corne, Nick R Jerram, Joshua D Knowles, and Martin J Oates. 2001. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation.* 283–290.

[9] David W Corne, Joshua D Knowles, and Martin J Oates. 2000. The Pareto envelope-based selection algorithm for multiobjective optimization. In *International conference on parallel problem solving from nature.* Springer, 839–848.

[10] Ian Davidson and Sugato Basu. 2007. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from data* 1 (2007), 1–41.

[11] Ian Davidson and SS Ravi. 2005. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *European Conference on Principles of Data Mining and Knowledge Discovery.* Springer, 59–70.

[12] Ian Davidson and SS Ravi. 2005. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining.* SIAM, 138–149.

[13] David L Davies and Donald W Bouldin. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), 224–227.

[14] Alvaro Garcia-Piquer, Albert Fornells, Jaume Bacardit, Albert Orriols-Puig, and Elisabet Golobardes. 2013. Large-scale experimental evaluation of cluster representations for multiobjective evolutionary clustering. *IEEE transactions on evolutionary computation* 18, 1 (2013), 36–53.

[15] Julia Handl and Joshua Knowles. 2005. Exploiting the trade-off: the benefits of multiple objectives in data clustering. In *International Conference on Evolutionary Multi-Criterion Optimization.* Springer, 547–560.

[16] Julia Handl and Joshua Knowles. 2006. On semi-supervised clustering via multiobjective optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation.* 1465–1472.

[17] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2, 1 (1985), 193–218.

[18] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. 1999. Data clustering: a review. *ACM computing surveys (CSUR)* 31, 3 (1999), 264–323.

[19] Daniel Khashabi, John Wieting, Jeffrey Yufei Liu, and Feng Liang. 2015. Clustering With Side Information: From a Probabilistic Model to a Deterministic Algorithm. *arXiv preprint arXiv:1508.06235* (2015).

[20] Joshua Knowles and David Corne. 1999. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Vol. 1. IEEE, 98–105.

[21] Martin HC Law, Alexander Topchy, and Anil K Jain. 2004. Clustering with soft and group constraints. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR).* Springer, 662–670.

[22] Juanjuan Luo, Licheng Jiao, and Jose A Lozano. 2015. A sparse spectral clustering framework via multiobjective evolutionary algorithm. *IEEE Transactions on Evolutionary Computation* 20, 3 (2015), 418–433.

[23] Son T Mai, Sihem Amer-Yahia, Sébastien Bailly, Jean-Louis Pépin, Ahlame Douzal Chouakria, Ky T Nguyen, and Anh-Duong Nguyen. 2018. Evolutionary Active Constrained Clustering for Obstructive Sleep Apnea Analysis. *Data Science and Engineering* 3, 4 (2018), 359–378.

[24] Kaisa Miettinen. 2012. *Nonlinear multiobjective optimization.* Vol. 12. Springer Science & Business Media.

[25] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos Artemio Coello Coello. 2013. A survey of multiobjective evolutionary algorithms for data mining: Part I. *IEEE Transactions on Evolutionary Computation* 18, 1 (2013), 4–19.

[26] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos A Coello Coello. 2013. Survey of multiobjective evolutionary algorithms for data mining: Part II. *IEEE Transactions on Evolutionary Computation* 18, 1 (2013), 20–35.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[28] Dan Pelleg and Dorit Baras. 2007. K-means with large and noisy constraint sets. In *European Conference on Machine Learning.* Springer, 674–682.

[29] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.

[30] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. 2007. C-dbscan: Density-based clustering with constraints. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing.* Springer, 216–223.

[31] Firas Saidi, Zouheir Trabelsi, and Henda Ben Ghazela. 2018. A novel approach for terrorist sub-communities detection based on constrained evidential clustering. In *2018 12th International Conference on Research Challenges in Information Science (RCIS).* IEEE, 1–8.

[32] Jana Schmidt, Elisabeth Maria Brandle, and Stefan Kramer. 2011. Clustering with attribute-level constraints. In *2011 IEEE 11th International Conference on Data Mining.* IEEE, 1206–1211.

[33] Samaneh Hosseini Semnani, Otman A Basir, and Peter Van Beek. 2016. Constrained clustering for flocking-based tracking in maneuvering target environment. *Robotics and Autonomous Systems* 83 (2016), 243–250.

[34] Alex Seret, Thomas Verbraken, and Bart Baesens. 2014. A new knowledge-based constrained clustering approach: Theory and application in direct marketing. *Applied Soft Computing* 24 (2014), 316–327.

[35] Weiguo Sheng, Stephen Swift, Leishi Zhang, and Xiaohui Liu. 2005. A weighted sum validity function for clustering with a hybrid niching genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35, 6 (2005), 1156–1167.

[36] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. 2017. KEEL 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems* 10, 1 (2017), 1238–1249.

[37] David A Van Veldhuizen and Gary B Lamont. 2000. Multiobjective optimization with messy genetic algorithms. In *Proceedings of the 2000 ACM symposium on Applied computing-Volume 1.* 470–476.

[38] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. 2001. Constrained K-means Clustering with Background Knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning.* Morgan Kaufmann Publishers Inc., 577–584.

[39] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. 2003. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems.* 521–528.

[40] Dit-Yan Yeung and Hong Chang. 2007. A kernel approach for semisupervised metric learning. *IEEE Transactions on Neural Networks* 18, 1 (2007), 141–149.

[41] Xuesong Yin, Songcan Chen, Enliang Hu, and Daoqiang Zhang. 2010. Semi-supervised clustering with metric learning: An adaptive kernel method. *Pattern Recognition* 43, 4 (2010), 1320–1333.

[42] Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation* 11, 6 (2007), 712–731.

[43] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report* 103 (2001).