Instituto Tecnologico y de Estudios Superiores de Monterrey

Estado de Mexico Campus

School of Engineering and Sciences

Evolutionary Clustering using Classifiers: Definition, Implementation, Scalability, and Applications

A thesis presented by

Benjamin Mario Sainz Tinajero

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Monterrey, Nuevo León, April, 2022

Dedication

This project is the consequence of the opportunities and courage given to me by my hard-working parents Araceli and Benjamin, and my brave little sister Karol. Mom, thank you for being by my side in times of glory and despair. Dad, thank you for being an example of a responsible person willing to do anything for his family. Karol, the jingle bell of this family, thank you for always believing in me. To my grandmother, Cristina, thank you for always giving us a piece of your strength, wisdom, and hospitable heart. Marisol, thank you for sharing this ride with me and enriching it with your kindness, intelligence, and empathy. To my friends, thank you for every moment of laughter and trustfulness when I needed it. To John, Jose, and Stephen, thank you for the continuous inspiration.

What is science if not a reflection of our endless pursuit of answers?
This goes for every human being looking for them.

And for me.

Acknowledgements

I want to thank Dr. Gutierrez-Rodriguez for being such a supportive advisor, it was an honor working with you. Thank you for guiding my first years conducting research and believing in my potential and work ethic. Thank you for fostering my sense of creativity, which is, after all, what makes us human. To my colleagues, Dachely Otero-Argote, Carmen E. Orozco-Mora, and Gabriel I. Perez-Landa, thank you for accepting to work in this project. The application of our latest method in computer vision would have been impossible without your hard work.

I also want to express my gratitude to professors who left a memorable mark on my academic development. Dr. Cantu-Ortiz and Dr. Ceballos-Cancino, thank you for being an example of knowledgeable researchers and encouraging me to scale my work into scientific publications. Dr. Ortiz-Bayliss and Dr. Amaya-Contreras, thank you for always pushing me to deliver research work of the highest quality. Dr. Gonzalez-Mendoza, thank you for supporting my first research tangent on data mining, letting me assimilate the impact that applied research can have on relevant topics to society. Dr. Molina-Espinosa, thank you for letting me be instrumental in developing undergraduate students during my time as a Teaching Assistant. Dr. Monroy-Borja, thank you for making graduate school a place for Computer Science researchers to pursue their ideas and drive science and technology forward through novel contributions on Artificial Intelligence. Last but not least, Dr. Medina-Perez, thank you for rather than being a professor, being a mentor. To every professor who has shared with me even a small piece of knowledge, thank you for making me a complete scientist.

This thesis was supported by Tecnologico de Monterrey, and CONACYT.

Evolutionary Clustering using Classifiers: Definition, Implementation, Scalability, and Applications

by

Benjamin Mario Sainz Tinajero

Abstract

Clustering is a Machine Learning tool for partitioning multi-dimensional data automatically into mutually exclusive groups, aiming to reflect the patterns of the phenomena it represents. Clustering algorithms perform this task conditioned by the clustering criterion modeled in its objective function. However, selecting the optimal criterion is a domain-dependent task that requires information on the cluster structure that a user often does not count on due to the unsupervised nature of the technique. Available approaches accentuate this problem as they perform clustering according to a similarity notion often limited to the concepts of compactness and connectedness, inducing bias and favoring clusters with certain shape, size, or density properties from using conventional distance functions. However, we cannot consider this a complete notion of a cluster because not every dataset will comply with both notions in the same proportion. Hence, research on this topic has not converged to a standard definition of a cluster, which raises the need for algorithms that produce adaptive solutions that mirror the underlying structures and relations within the data.

This thesis is focused on the design of single-objective Evolutionary Clustering Algorithms that generate solutions that are not biased towards any cluster structure by optimizing a novel generalization clustering criterion. To achieve that, we designed objective functions modeled as a supervised learning problem, considering that a good partition should induce a well-trained classifier. That is how we decided to assess the quality of a clustering solution, according to its capability to train an ensemble of classifiers. The main contribution of this thesis is our series of Evolutionary Clustering Algorithms using Classifiers (the ECAC series), which introduces the aforementioned clustering criterion along with evolutionary computation. This meta-heuristic allows us to model distinct criteria to optimize while creating and evaluating multiple solutions along the process. The experimental results in the design of our family of methods ECAC, F1-ECAC, and ECAC-S, show an increase in similarity between the partitions created by our algorithms and the ground truth labels (obtained from the publicly available repositories where we retrieved the data) with a maximum Adjusted RAND Index of 0.96. Our second algorithm, F1-ECAC, proved the competitiveness of our contributions against traditional, single, and multi-objective Evolutionary Clustering algorithms showing no statistically significant difference against k-means, HG-means, and MOCLE. Our latest contribution, ECAC-S, was tested on a satellite image segmentation task, and it produced segmentations with higher average Adjusted RAND Index than k-means, Spectral-clustering, Birch, and DBSCAN in 4 out of 10 images.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The adoption of technology within industrial processes has enabled data generation at unprecedented amounts and rates [71]. Machines used for handling packaging processes, plastic injection, and piece evaluation exemplify how automation can propel designs that collect and transmit data. Virtually every sector in a productive system produces data that can be analyzed to retrieve insights for further process enhancement (e.g., a manufacturing line that has seen a rise in average stationary time, a supply chain that has seen unexplained efficiency drops, or market sectors that behave unnaturally) [67]. This revolution has even reached our approach to personal computing and the devices we use every day. Smartphones (and the applications within them) are instrumented for tracking usage data of almost every possible transaction, and social media are the most representative case in recent times. Gathering and storing data implies a cost; therefore, data shall be considered an investment and an asset [67].

Still, how do we process all these data? This question implies the need for methods capable of modeling phenomena that would be impossible to be processed without numerical methods. Data Science methods have the objective of assisting the knowledge extraction task to transform it into decisions. For instance, in the manufacturing example mentioned before, the collected data can help find root causes and propose solutions to improve productivity. Regarding social networks, data mining can be useful for market segmentation, testing a new feature, or modifying an existing product [67]. Standardized procedures such as the Cross-Industry Standard Process for Data Mining, or CRISP-DM (which we have used in previous studies to analyze the impact of the COVID-19 vaccination progress [79]), propose an organized approach towards understanding and preparing data to evaluate and deploy a model [93].

Machine Learning is the collection of methods for extracting models from data and is a sub-field of Artificial Intelligence that closely aligns with scientific disciplines such as Applied Statistics and Pattern Recognition. In the end, its main purpose is concerned with the analysis of data for identifying informative patterns. Supervised and Unsupervised Learning are two sub-fields from Machine Learning that can be useful for answering questions depending on the intrinsic character of the task to be solved. For instance, looking for natural groups within a set of customers would be an unsupervised approach, whereas predicting a customer's likelihood of buying a product based on previous examples would require a supervised technique (complete historical data on a specific target label is provided for prediction) [93, 79].

Clustering is an Unsupervised Learning technique widely used in exploratory data analysis for finding patterns at the beginning of a Data Science pipeline to get an outlook on the existing relations within a dataset [48, 39, 46, 67]. One of the results of clustering a dataset is a set of disjoint subsets of the data, which we refer to as *clusters* [48, 23]. The primary purpose of clustering is to create groups that present high similarity levels within but not between them [48]. This group of methods could be helpful when dealing with questions of open character. For instance, an image could be clustered to evaluate if its pixels might fall into informative segments regarding color similarity [24, 34].

In general, clustering aims to find compact and isolated groups [15, 71, 32, 38, 14]. However, this ideal condition does not hold for most datasets in real applications, causing an absence of a cluster definition that is accepted across the literature [42, 15]. Hence this relative concept has given rise to multiple clustering methods following divergent similarity notions to group unlabeled objects into clusters [71, 38, 15, 40, 94]. Clustering criterion is a model capable of capturing and representing the internal structures in a dataset associated with an algorithm [14]. Taxonomies across the literature often divide clustering algorithms into three categories: partitional, hierarchical, and density-based methods. An algorithm in these varieties aims to handle possible cluster shape, size, density, and noise [71, 38, 32]. In recent decades, we have witnessed the design of dozens of publicly-available algorithms [94]. Some examples of well-maintained libraries offering straightforward setup of clustering algorithms are TensorFlow by Google [1], PyTorch by Facebook [64], and Scikit-Learn [65].

The interpretation of a phenomenon is susceptible to the bias encountered in the Machine Learning method that was used to model it. In clustering, a partition contains groups formed according to a specific similarity notion modeled in an algorithm conditioned by its intrinsic clustering bias [67, 71]. Each algorithm offers parameters that inevitably favor groups complying with specific cluster structures [94]. Hence performing clustering on a dataset with multiple algorithms will raise insights that might vary significantly upon each method's tendency towards certain cluster shapes. Even using the same algorithm could return different partitions with the same data due to their sensitivity regarding hyper-parameter setting [32, 48]. Clustering bias refers to a performance enhancement when the same structural conditions are met by a clustering algorithm's objective function and the data being analyzed. This problem is accentuated by the natural characteristic of data from real sources, which is not likely to perfectly match the clustering bias induced by algorithms optimizing conventional distance functions [15, 71, 32, 38, 14]. Therefore, there is no clustering algorithm that can capture every underlying structure from data [15].

The clustering approach has stayed the same while more algorithms are released each year. Clustering criterion selection is usually a manual task that consists of choosing the algorithm and hyper-parameters that seem most likely to return revealing partitions [82]. The main disadvantage of this procedure is the dependency on experience-based knowledge [44, 15]. Consequently, a user might face failure in displaying informative clusters, which implies an iterative adjustment of the clustering criterion. This evaluation takes place after running each algorithm and requires enough knowledge on the domain (or an external expert on the discipline) to determine whether the partitions offer an informative representation of the data or not [15]. However, most data mining applications do not count on information about the cluster structures within the data beforehand. This factor, and the vast amount of available methods, makes it unfeasible to know the proper clustering criterion a-priori [71, 32].

The relation between clustering bias and clustering criterion selection is causal, as the former complicates the latter. For instance, k-means, could be useful when dealing with clusters forming hyper-spheres. The main drawback of using it is that most real datasets present irregular cluster shapes, thus not complying with the clustering bias induced by this algorithm. It is limiting and insufficient to try fitting our anthropocentric concepts of geometry when modeling multi-dimensional data [52, 91, 45, 51, 37, 33].

Clustering can be modeled as an optimization problem, in which an objective function should be designed according to a desired internal quality criterion to maximize or minimize [32, 18]. Therefore, a partition's quality is delimited by the capability of the cluster property favored by the algorithm to capture the event reflected in a dataset in a natural manner. To tackle the problems mentioned before, clustering algorithms based on nature-inspired meta-heuristics aim to optimize one or more objectives for creating clusters of arbitrary shapes and avoid falling into local minima [37, 58, 57, 60]. Among them, Evolutionary Algorithms are the most popular for clustering [32]. Contrary to traditional methods that optimize only one conventional criterion, single and multi-objective Evolutionary Clustering algorithms compute fitness combining distance functions fundamentally different but equally desirable [23, 32]. Another advantage is that they explore the search space efficiently, and create and combine partitions during the clustering process. However, existing approaches still optimize criteria related to compactness or connectedness, and these two properties are not enough to formulate a complete definition of a cluster because it does not hold from real data sources [37, 58, 57, 60, 33].

1.1 Hypothesis

The clustering algorithms developed for this thesis return partitions that reflect the intrinsic relations within a dataset, achieving competitive performance against state-of-the-art methods by optimizing the novel cluster quality criterion of generalization. The algorithms do not present any clustering bias towards a geometrical shape or structure and create and assess solutions along, and not at the end of the clustering process.

1.2 Research Questions

The following questions describe the project:

- How can we design an objective function that favors adaptive partitions?
- What would be the best search strategy to evaluate solutions during the process?
- What would be our algorithm's position and role within the literature?
- How extensive can this type of algorithm be for dataset variety?

1.3 Objectives

The main objective of the present thesis project is to design clustering algorithms that return high-quality solutions. The sub-objectives on which this project focuses are the following:

- Design an efficient objective function that models the generalization cluster quality criterion, leading our methods to adapt to the dataset and not the other way around.
- Implement a search strategy to generate solutions that iteratively increase in quality.
- Analyze improvement areas of the algorithms to enhance efficiency.
- Perform a benchmark on a real application using our ultimate algorithm.

1.3.1 Scope and Limitations

The boundaries that limit the project are established up to the design of clustering algorithms that require as input a dataset and a number of clusters for recommending a partition. The related work under consideration ranges from clustering and cluster validity indices to novel techniques of evolutionary computation.

1.4 Contributions

This project contributed to the literature with the Evolutionary Clustering Algorithm using Classifiers (ECAC) series of publications. The cluster quality criterion in every version of ECAC is based upon the philosophy behind the cluster Validity Index using Classifiers by Rodriguez *et al.* [71], considering that a high-quality partition induces a well-trained classifier. In this way, we take advantage of the benefits of modeling the objective function of a clustering algorithm as a supervised problem to optimize the generalization capabilities of a partition. The main point of differentiation of our methods is the inclusion of an ensemble of classifiers to compute fitness in single-objective Evolutionary Clustering algorithms. Therefore, we compute the quality of a partition according to the quality of the classification model it can induce, giving us an informative approximation of its capability of capturing the relations within a dataset while avoiding the disadvantages of using conventional distance functions such as clustering bias.

During the development of this thesis project, we designed three versions of ECAC. In every iteration of the algorithm, we aimed to find the best encoding, genetic operators, objective function, and initialization schema for the Evolutionary Clustering process. The first version of the algorithm, ECAC, constitutes the establishment of our algorithm's position within the single-objective Evolutionary Clustering literature using publicly available datasets for experimentation. The article related to ECAC was published and presented at the IEEE Congress on Evolutionary Computation (July 2021), and is available at Ref. [77] (implementation available at Ref. [73]). The Evolutionary Clustering Algorithm using Classifiers and the F_1 score (F1-ECAC) represents the second version of ECAC and was developed involving major changes to the objective function. This is an enhanced version of our first algorithm in terms of performance and efficiency and offered competitive performance in a benchmark

against state-of-the-art multi-objective clustering algorithms. The article related to F1-ECAC was published at IEEE Access Journal in September 2021 and is available at Ref. [78] (implementation available at Ref. [74]). The Simultaneous Evolutionary Clustering Algorithm using Classifiers (ECAC-S) represents the third and final contribution of this thesis, and offers efficiency improvements in its implementation and design, introducing parallel computing and an application on satellite image segmentation. The article related to ECAC-S is under preparation for submission (implementation available at Ref. [76]).

The remainder of this thesis is organized as follows. Chapter 2 describes clustering methods according to the most prevalent taxonomies within the literature. In Chapter 3, we describe our ECAC series of contributions. Chapter 4 goes into detail the experimental framework used to test each of our developments, and their results are depicted and discussed in Chapter 5. Finally, our conclusions are stated in Chapter 6.

Chapter 2

Related Work

This chapter introduces the most representative clustering algorithms belonging to multiple families, their role in the literature, and their influence on this project. It is essential to understand the main drawbacks and pinpoint design decisions that could be of great use to answer the research questions to be tackled. Taxonomies found in the literature often vary according to the criteria or purpose [3, 42, 94]. Handl and Knowles [32] categorize clustering algorithms based on the criterion that each algorithm optimizes. The three cluster structures considered in this taxonomy are:

1. *Compactness* within a cluster: algorithms favoring this property return groups with low intra-variation.
2. *Connectedness* within a cluster: algorithms favoring this property tend to group neighbors together.
3. *Spatial Separation* within and between clusters: algorithms favoring this property tend to consider density-related criteria to form groups.

2.1 Traditional Clustering Algorithms

We refer to algorithms in this category as those performing optimization without using metaheuristics. The methods presented in this, and the following sections, were used to assess the performance of our proposals in further Chapters.

2.1.1 k-means

This partitional algorithm (it searches for a solution iteratively) is the most widely used since its introduction in the 1960s due to its straightforward configuration and interpretability [35, 42, 45, 52]. k-means optimizes compactness within clusters by minimizing the sum of the squared distances from each point to its cluster centroid [42, 45, 52]. Its dependency on conventional distance functions as the Euclidean metric favors clusters forming hyperspheres and an increased number of clusters, making k-means unsuitable when dealing with complex structures (which are likely to appear in data from real sources). Another disadvantage is its sensitivity towards initial centroids, which might lead to local minimums and trivial

solutions [42]. The cluster quality criterion used in k-means has inspired the design of Evolutionary Clustering Algorithms that do not fail to converge to a global optimum by introducing varying heuristics and implementations [2, 29, 45, 51].

2.1.2 Agglomerative Clustering

Hierarchical methods form sequential partitions started either by locating every point 1) in a singleton cluster or 2) in a clustering containing the whole dataset. These methods were published around 1985, and build a dendrogram structure according to the selected distance function and the affinity metric [91, 94]. A dendrogram can be split at the required height to produce a desired number of clusters [39, 42]. Using a single-linkage affinity (SL) considers the distance between the closest points from two groups to create the nested clusters. In this case, the connectedness cluster quality criterion is optimized, inducing clustering bias towards continuous groups (e.g., spiral-shaped structures), making it an inappropriate solution for data from real sources [39].

2.1.3 Density-based Spatial Clustering of Applications with Noise

Spatial Separation is a cluster quality criterion susceptible to converging to trivial solutions; hence it is commonly combined with other heuristics. This is an objective more commonly found as cluster quality indexes and is not widely found as a heuristic in clustering algorithms [30, 32]. The Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was introduced in 1996, and it aims to take care of the spatial separation between clusters, a property not considered by the compactness and connectedness quality criteria [13, 81]. Contrary to the previously mentioned methods, DBSCAN does not require the number of clusters as an initial hyper-parameter. Its operation consists of computing neighborhood density within clusters and placing objects in a group if and only if its aggregation complies with a minimum threshold (this is the only required hyper-parameter by the algorithm) and lets the user set the minimum objects required to form a cluster [13].

2.1.4 Balanced Iterative Reducing and Clustering using Hierarchies

Zhang *et al.* [96] developed the Balanced Iterative Reducing and Clustering using Hierarchies (Birch) algorithm in 1996, a method that focuses on tracking densely occupied portions of the data and creating a compact summary. This robust method computes connectedness and compactness as it builds a height-balanced tree. This method groups objects incrementally and can handle data points outside the main underlying pattern (considered noise). Birch only requires one scan of the data and increases performance if allowed further iterations, but is highly dependent on a proper hyper-parameter setting.

2.1.5 Spectral Clustering

In 1973, Donath and Hoffman proposed constructing graph partitions based on eigenvectors of an adjacency matrix [9, 90]. This finding led to the establishment of the fundamental theory behind Spectral Clustering, made popular by the work of Shi and Malik [9, 82]. Spectral

Clustering is a series of discriminative methods that do not make assumptions of the structures within the data (they are even capable of solving problems like cluster structures formed as intertwined spirals) and compute local evidence on the likelihood of two points being clustered together to proceed to make global decisions to create the disjoints sets of data according to some criterion [90]. This criterion can even be decoded with an embedding framework, where the clustering relationships could be preserved in a lower-dimensional interpretable representation [95]. Once the similarity graph is built, a linear problem is solved in the relaxed continuous domain by eigendecomposition (using heuristics such as k-means to get a discrete solution, or partition, from eigenvectors in a lower-dimensional space) with no risk of falling into local minima and no further need of restarting the clustering process with different initializations [90, 95].

2.2 Single-objective Evolutionary Clustering

Modeling an internal cluster quality criterion as an optimization problem can diminish the impact of the absence of a domain expert. Thus, a cluster property embodied in an objective function to be maximized or minimized, will define the nature of the solutions returned in an optimization problem (Ω, P) to find the optimal clustering C^* for which

$$P(C^*) = \min_{C \in \Omega} P(C), \tag{2.1}$$

where Ω is the set of reachable solutions, C is a clustering, or partition, of dataset E, and P is an internal criterion modeled after a particular cluster assignment notion [18, 32]. The cluster properties mentioned at the beginning of this chapter have been modeled using innovative objective functions, operators, and encoding schemes. In the following section, we will go into detail about one of the most representative single-objective Evolutionary Clustering methods among the literature [29, 37, 58, 57, 60].

2.2.1 HG-means

Gribel and Vidal's algorithm [29] (published in 2018), performs clustering in a single-objective schema, introducing a genetic approach. Recombination and mutation are the main components of HG-means' (HGM) evolutionary search pipeline that turn a random initial population into solutions that minimize compactness. As performed by k-means and other methods following meta-heuristics [33, 37], its objective function is computed as

$$Comp(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k)^2, \tag{2.2}$$

where $\delta(.,.)$ represents the Euclidean metric as distance function, and i is an object from dataset E, with centroids μ_k for each cluster C_k contained in C.

In each iteration, a binary tournament is in charge of selecting two parents required for the crossover procedure. This operator solves a bipartite matching problem to align the closest cluster centers of both parents into pairs and then selects one of them randomly with equal

probability, continuing the process with k centroids. Subsequently, the mutation operator removes one of them and reassigns it randomly to any remaining point in the dataset. This centroid set is used to execute a local search based on k-means at the end of each generation. This powerful algorithm incorporates diversity management strategies to foster distinct solutions and avoid premature convergence. Nonetheless, HG-means also has the disadvantage of optimizing a quality criterion that assumes that the cluster structures within the data are spherical.

2.3 Multi-objective Evolutionary Clustering

Despite the proven benefits of evolutionary search to avoid local minima, the inability of existing methods to produce partitions enabling two or more properties suggests the inclusion of multi-objective methods for clustering. This could lead to high-quality solutions, reducing the clustering criterion selection dilemma mentioned before, which could still be propelled by Evolutionary Clustering Algorithms that optimize one objective function that induces bias in its operation, resulting in unacceptable results. Therefore, an inadequate selection of the optimal Evolutionary Clustering algorithm is more prone when dealing with single-objective algorithms. Multi-objective Evolutionary Clustering Algorithms overcome this problem by modeling the compromise of multiple complementary cluster properties, implying a more natural data processing. A problem $(\Omega, P_1, ..., P_m)$ aims to search C^* for which

$$\begin{aligned} P_t(C^*) &= \min_{C \in \Omega} P_t(C) \\ t &\in 1, ..., j, \end{aligned} \tag{2.3}$$

where Ω is a set considering all possible clusterings, C is a partition of the dataset E, and P_t are the j multiple clustering criteria to be optimized [18, 32]. Pareto optimality is introduced in these problems to handle the fitness balance between objective functions. Given solutions $C_1, C_2 \in \Omega$, solution C_1 is considered to dominate solution C_2 ($C_1 \prec C_2$) if and only if

$$\forall t : P_t(C_1) \leq P_t(C_2) \wedge \exists t : P_t(C_1) < P_t(C_2), \tag{2.4}$$

and the Pareto-optimal solutions are consequently stated as

$$\Pi = \{C \in \Omega : \nexists C^{'} \in \Omega : C^{'} \prec C\}, \tag{2.5}$$

which generates an objective space denominated as the Pareto front (there is no solution outside of the Pareto front dominating any inside of it). Simultaneous search using multiple objectives is expected to find solutions at least in equal capacity compared to its separate objective functions in a single-objective framework whereas acquiring an increased computational cost [18, 32]. Multi-objective Evolutionary Clustering Algorithms have been commonly designed and implemented to optimize no more than two objectives. This novel discipline has witnessed the release of relevant methods in recent years, and now, we will go into detail about the two most dominant among the literature [31, 63, 70, 97].

2.3.1 Multi-objective Clustering Ensemble

The Multi-objective Clustering Ensemble (MOCLE) method is an approach to Evolutionary Clustering using Pareto-optimality and was introduced by Faceli *et al.* [15, 16] in 2006. MOCLE starts with an initial population constituted of partitions of the dataset created by conceptually diverse clustering algorithms (with hyper-parameter variations). Such partitions range from k to $2 * k$ clusters. The encoding of each individual representing a partition is performed as an array of sets containing the members' labels per cluster. The Pareto front returned by MOCLE guarantees to offer a group of individuals representing each of its regions, as the result of optimizing two cluster quality criteria modeled in two separate objective functions that should be minimized:

1. *Deviation* is a similarity metric that resembles compactness (see Equation 2.2) but does not use squared distance values [32], and is computed as

$$Dev(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k), \tag{2.6}$$

2. *Connectivity* quantifies the inverted frequency in which neighboring points are clustered together and is computed as

$$Conn(C) = \sum_{i \in E} (\sum_{l=1}^{L} x_{i,nn_{i,l}})$$
$$x_{r,s} = \begin{cases} \frac{1}{l} \text{ if } \nexists C_k : r \in C_k \wedge s \in C_k \\ 0, \text{otherwise} \end{cases}, \tag{2.7}$$

   where nn_{il} is the l_{th} nearest neighbor of object i, and L is the hyper-parameter for delimiting the number of considered neighbors, and E is the dataset under analysis [32].

As MOCLE optimizes objective functions found in other algorithms of its family, the gradual elimination of low-performing solutions and the generation of new ones is mainly in charge of the recombination operation and the fitness computation. The crossover operator used in this algorithm follows an innovative approach. First, two parents are selected through a binary tournament. Afterward, a new chromosome is created with the participation of both parents in a clustering ensemble problem for finding a consensus partition instead of performing slicing operations. The results returned by MOCLE will vary significantly according to the ensemble method used in this operator. If the cluster ensemble technique does not automatically select the number of clusters for the child generated from both parents, the k value must be randomly chosen from the cluster range of both parents. MOCLE does not use a mutation operator, which sets it apart from other Evolutionary Clustering methods. The authors claim that this design decision parts from assuming that the relevant structures from the data are contained and captured within the partitions created by the set of algorithms at the beginning of the procedure. Therefore, the evolutionary process of MOCLE works in a restricted search space.

2.3.2 Multi-objective Clustering with Automatic k-determination

In 2017, Garza-Fabre *et al.* [23] proposed an update to the Multi-objective Clustering with Automatic k-determination method (MOCK, from 2007) [32], Δ-MOCK. The enhanced version of this Evolutionary Clustering Algorithm introduces modifications to the genotype representation framework and its initialization, resulting in efficiency benefits. Δ-MOCK acts in accordance with an evolutionary strategy, and during each iteration, a mating selection binary tournament and two genetic operators create new offspring that compete between them and with their parents to survive. The authors state that the optimization of two fundamentally different but equally desirable quality criteria models the trade-off of two objective functions to be minimized:

1. *Variance* within clusters, which is computed as

$$Var(C) = \frac{1}{N} \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k)^2, \tag{2.8}$$

   where N represents the length of dataset E. This criterion computes the mean compactness as seen in Equation 2.2 and improves as k increases.

2. *Connectivity* within clusters, as presented in Equation 2.7. The gradually decreasing penalty $1/j$ emphasizes to nearest neighbors, and this objective improves as the k value decreases.

Both objectives balance each other's tendency towards the value of k, avoiding trivial solutions that could be retrieved if using any of these separate objectives [15, 32].

Δ-MOCK is natively designed to work with the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) [6] to search for a Pareto front with a balanced representation per region. The initial population is built upon partitions with varying cluster numbers to the maximum set by the user. A Minimum-spanning Tree (MST) sets up the conformation of the genetic material to be enhanced during the evolutionary search. Partitions are recombined through a uniform crossover operator that joins the genotypes of two parents [86]. In this procedure, a brand new solution is started as an exact copy of a parent (baseline genotype), and its information is interchanged with the remaining one (modifier genotype) conditioned by a hyper-parameter p_c that delimits the individual discrete probability to decide if a gene is preserved or modified. Each parent is used once as the baseline genotype and once as the modifier genotype. Using this operator instead of slicing the whole chromosome to give rise to new solutions can produce any possible combination between two parents. Δ-MOCK uses a neighborhood-biased operator that also computes the individual mutation probability p_m per gene (or link, because of the Δ-locus encoding used to represent a solution) to increase the chances of discarding unfavorable links. Such probability for mutating a gene encoding a link associating two solutions $i \to h$ is computed as

$$p_m(i \to h) = \frac{1}{|\Gamma|} + \left(\frac{nn_i(h)}{|\Gamma|} \right)^2, \tag{2.9}$$

where $|\Gamma|$ is the number of positions in the representation. The mutation of a link $i \to h$ replaces the link with a new link $i \to z$, where $z \in \{i, nn_{i1}, .., nn_{iL}\} \setminus \{h\}$ [23].

2.4 Supervised and Unsupervised Learning

Diverse approaches have used supervised learning to solve unsupervised problems and the other way around. Some of them will be described in this section to analyze their design principles and their relevance to the current project.

2.4.1 Optimized Ensemble Classifier with Cluster Size Reduction

The method introduced by Jan *et al.* [43] in 2020 is relevant to this proposal due to the inclusion of clustering in a classification problem. Clustering, an ensemble of classifiers, and an evolutionary search strategy constitute the Optimized Ensemble Classifier with Cluster Size Reduction. This method performs incremental clustering to generate varied class-pure groups to pre-process the data in unbalanced problems. The obtained partitions train and test a set of classifiers using the accuracy metric. An evolutionary process aims to retain an optimized set of classifiers. Even though our project belongs to clustering, it is essential to consider methods from other families that have proven the benefits of using supervised and unsupervised learning together to develop innovative proposals.

2.4.2 Cluster Validity Index using Classifiers

Rodriguez *et al.* [71] proposed a cluster validity index in 2018 that uses the performance of the classifiers induced by a partition as a quality criterion, the Cluster Validity Index using Classifiers (VIC). This internal metric to evaluate a partition implies that the performance of a supervised classifier has a positive relationship with the capacity of a partition to reflect the intrinsic relations within a dataset. We refer to this indicator as generalization, a novel criterion that differs from the traditional ones mentioned before. Consequently, this index shall be maximized when modeled in an optimization problem. An objective function reflecting this criterion will assess a solution based on its capacity of inducing a good set of classifiers (using a partition as class labels and the dataset's attributes to train them). The numerical assessment is the resulting maximum average Area Under the Curve (AUC) obtained by a classifier in a five-fold cross-validation schema. The authors suggest avoiding the accuracy metric to test the classifiers as it is misleading in unbalanced problems. For this thesis project, VIC is the foundation of the criterion optimized in the ECAC series of algorithms.

2.5 Satellite Image Segmentation

Image Segmentation is a processing technique (that composes one of the main areas of computer vision) to divide an image into multiple regions of interest [72, 24]. This tool has been applied in various applications, for example, medical analysis, autonomous vehicles, security cameras, and augmented reality [19, 55]. Methods allowing image segmentation could be divided as general-purpose or use-case specific. Some of the most prevalent methods among the image segmentation literature are Single-linkage, Hybrid-linkage, Spatial Clustering, and Split-and-Merge techniques [34]. In this thesis, we present an evaluation of the performance of our finest contribution on a satellite image segmentation task.

2.6 Remarks

Single and multi-objective Evolutionary Clustering algorithms offer innovative search strategies that might emulate the role of an expert to evaluate a partition while avoiding local minima iteratively. However, they still operate by balancing the predefined cluster properties inherent in the criteria they optimize. Cluster structure assumptions when modeling the objective function of a clustering algorithm are a limiting factor to be addressed in this proposal by using the generalization cluster quality criterion defined in Section 2.4.2. The benefits, disadvantages, induced bias affecting data interpretation, and difference in performance between the clustering algorithms in validation and real application tasks will be revisited in further chapters.

Chapter 3

The ECAC Series

This chapter will elaborate on the definition, implementation, and scalability of our ECAC series of contributions. Every algorithm belonging to our series of Evolutionary Clustering Algorithms using Classifiers was designed upon the search strategy of the genetic algorithm, and they optimize the novel generalization cluster quality criterion. Their main pipeline is depicted in Fig. 3.1. A set of random partitions represent the initial genetic material that the algorithm seeks to improve according to the selected clustering criterion through an evolutionary process. In each generation, the genetic operators are in charge of generating new chromosomes that form upcoming populations. Our algorithms run for a number of iterations delimited by the user or until the overall best individual achieves a fitness value of 1 (except for ECAC-S, which also incorporates a convergence exit parameter).

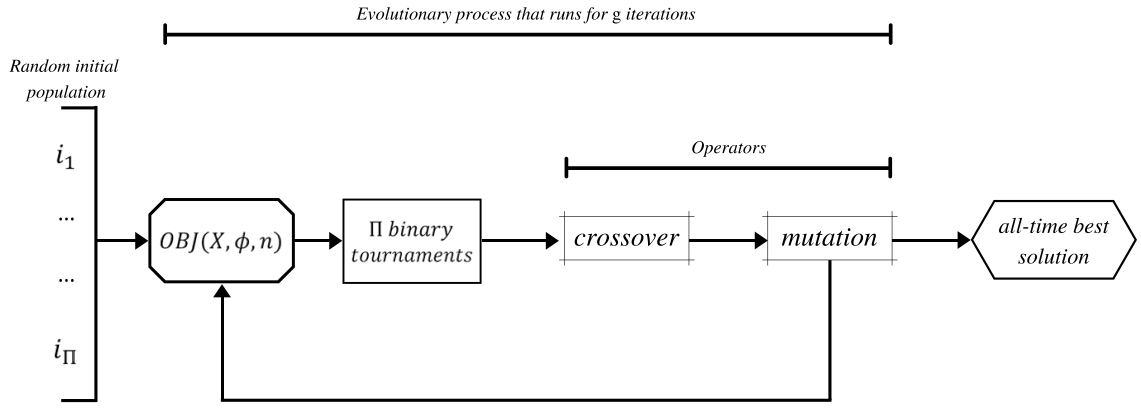

Figure 3.1: Pipeline followed by our series of Evolutionary Clustering Algorithms using Classifiers.

ECAC [77, 73], F1-ECAC [78, 74], and ECAC-S [76] optimize modified versions of the cluster quality proposed by Rodriguez *et al.* [71] presented in Chapter 2.4.2 and escalate it into clustering algorithms. VIC is the first internal index that introduces an ensemble of classifiers as cluster quality criterion, and as a consequence, the ECAC series is the first group of algorithms that optimizes this index within their objective function. In this thesis, we present three single-objective clustering algorithms that reduce the need for prior knowledge on the most suitable clustering criterion by entrusting classifiers to compute the separation between

classes of partitions created and combined along an evolutionary process. Our approaches avoid the clustering bias induced by traditional criteria based on distance functions and do not assume cluster structures within the data, leading to adaptive clusters with arbitrary shapes. To summarize, the ECAC series is composed by algorithms that optimize the capability of a partition to train an ensemble of classifiers by using each chromosome's genes as class labels following the search strategy of the genetic algorithm.

3.1 ECAC

The original edition of the Evolutionary Clustering Algorithm using Classifiers (ECAC) [77, 76] was our initial approach to evaluating the possible scope of using the innovative generalization quality criterion in a clustering algorithm designed as a single-objective problem. We wanted to model an objective function capable of detecting the underlying structures within a dataset; hence, we decided that supervised classifiers were the best option for computing an individual's fitness. The computation of ECAC can be found in Algorithm 1. To begin with, a population Φ of Π individuals (the population size is maintained in the evolutionary process) is randomly generated, and its fitness is computed. Afterward, an evolutionary process follows these steps in each generation.

1. Π binary tournaments are held. Each time, two random individuals are selected from Φ using the uniform distribution, and the one that presents better fitness (higher value, as we are tackling a maximization problem) is maintained.

2. Π parent pairs are formed in the following order: an individual i_1 is taken as $parent_1$, another individual i_2 is taken as $parent_2$, and the pairing is repeated in the opposite order (i_2 is taken as $parent_1$ and i_1 is taken as $parent_2$) and so on. It is necessary to include both orderings for the parent selection to generate Π offspring, considering that each pair produces one child (the recombination process of $parent_1$ and $parent_2$ is totally independent to the recombination process of $parent_2$ and $parent_1$).

3. The Π pairs of parents are used for generating Π children $C'_1 ... C'_\Pi$ by recombining and mutating them. These new offspring contain exchanged information from their parents and constitute a new population Φ'.

4. The fitness of each chromosome in Φ' is calculated, and the all-time best solution is stored in b.

3.1.1 Solution Representation

Our ECAC series of methods uses an integer label-based encoding to represent a clustering solution [37]. An individual is a partition that goes through an evolutionary process. An individual is related to one genotype, or chromosome, which is a vector of length N, and each position within it is known as a gene, and it stores the cluster membership information of the object from dataset E that it represents. Therefore, the qth position in a genotype represents the cluster assigned to the qth object in E. Fig. 3.2 presents a chromosome for

Algorithm 1 ECAC algorithm.

function ECAC(E, Π, g, k)
▷ E: data; Π: population size; g: maximum generations; k: number of clusters.
  $\Phi \leftarrow i_1, ..., i_\Pi$ ▷ Generation of initial population Φ with k clusters.
  $X \leftarrow$ data E features.
  Compute the fitness of each individual in Φ using OBJ(X, i, k). ▷ See Algorithm 2
  for $u = 1.g$ **do**
    $selected \leftarrow \Pi$ individuals are selected from Φ through a binary tournament.
    $parents \leftarrow selected$ elements form Π pairs.
    $C_1 ... C_\Pi \leftarrow crossover(parents)$.
    $C'_1, ..., C'_\Pi \leftarrow mutation(C_1, ..., C_\Pi)$.
    $\Phi' \leftarrow C'_1, ..., C'_\Pi$.
    Compute $C'_1, ..., C'_\Pi$'s fitness with OBJ(X, i, k).
    $\Phi \leftarrow \Phi'$. ▷ New population.
    $b \leftarrow$ all-time best solution.
    if b's fitness=1 **then**
      break
    end if
  end for
  return b
end function

a dataset with 12 objects clustered into three groups. Label-based representations have the disadvantages of proportionally increasing length as a function of N and causing repetition in the search space by the $k!$ possible genotypes to represent the same solution. However, this is an encoding widely found in the literature that offers an interpretable approach that lets the solutions be analyzed at any point of the clustering process [29, 37, 58, 57]. This representation was selected hand in hand to fit the classifiers' requirements of getting as input a data structure with one target label per object to be trained, making it the optimal alternative for this application.

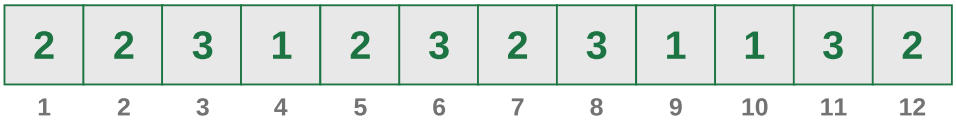

Figure 3.2: Genotype representation exemplified with one partition with three groups.

3.1.2 Initial Population

The initialization routine of ECAC consists of a chromosome generator that is in charge of creating Π individuals that constitute the initial generation of solutions. Every time a genotype is formulated, one cluster is assigned to gene q_i (where $i \in \{1, .., N\}$), forming mutually exclusive groups while avoiding cloned solutions. The cluster allocation is performed randomly

(uniform distribution) in the range $q_i \in \{1, .., k\}$, where k is the fixed number of required groups. Other methods explore the search space within the boundaries delimited by the solutions within the initial population and even omit a mutation operator, as seen in Chapter 2.3.1. Contrastingly, we attribute the capacity of our algorithms to discern low from high-quality partitions to their objective function and consider it the main driver of their natural selection process.

3.1.3 Objective Function

Algorithm 2 shows the steps followed by the objective function of ECAC to compute fitness. This function receives as input the dataset features X and an individual's chromosome i. Each classifier in the objective function is trained and tested using X and i as class labels with a fixed rate of training samples set to 25% of the data (the train-test split is held before every fitness computation, and the same data is used for every classifier). This low training rate favors short and quick trainings whereas still approximating a partition's capability to detect the underlying structures in the data, which is why we did not implement VIC purely in ECAC, F1 ECAC, or ECAC-S' objective function. The metric to evaluate a classifier $\Psi_v, v \in \{1, .., w\}$ is the average Area Under the Curve (AUC), which is computed according to a Receiver Operating Characteristic Curve (ROC) that depicts relative predictive trade-offs made by a classifier. The True Positive Rate (TPR) and False Positive Rate (FPR) are defined as

$$\begin{aligned} TPR &= \frac{TP}{TP + FN} \\ FPR &= \frac{FP}{FP + TN}, \end{aligned} \tag{3.1}$$

where TP represents True Positives in the resultant confusion matrix (objects correctly classified as positive), FN represents False Negatives (objects that should have been classified as positive but were predicted as negative), FP represents False Positives (objects that should have been classified as negative but were predicted as positive), and TN represents True Negative observations (objects correctly classified as negative) [67]. Plotting the TPR versus the FPR using the prediction probability returned by the classifier with multiple thresholds allows us to compute a classifier Ψ_v AUC, which is expressed as a fraction of the unit square in a range from 0 to 1. In problems with $k > 2$ the process is repeated k times per classifier using a binarized one-vs-rest training schema, and the average AUC of every class training is set as AUC_v. If a classifier cannot converge, AUC_v is assigned as 0. The returned fitness value of an individual is the average of the mean AUC obtained by all the classifiers (or $AUC_v, v \in \{1, .., w\}$). This unweighted metric is to be maximized, and values close to 1 imply a well-trained set of classifiers.

The classifier selection plays a huge role and will directly influence the objective function's performance (categorical and numerical data are supported as long as the classifiers are capable of processing them) [71]. The classifiers in ECAC's objective function are:

1. *Support Vector Machine* (SVMs): this versatile classifier maximizes the margin of a hyper-plane boundary delimited by support vectors for distinguishing between classes.

Algorithm 2 ECAC Objective Function Algorithm.

function OBJ(X, i)
▷ X: data features; i: an individual's chromosome.
  for $v = 1.w$ **do** ▷ w is the number of classifiers.
    Ψ_v classifier training with X, using i as class labels.
    for $z = 1.k$ **do**
      $fpr_z \leftarrow$ FPR of Ψ_v's predictions with variable thresholds.
      $tpr_z \leftarrow$ TPR of Ψ_v's predictions with variable thresholds.
      $AUC_{v,z} \leftarrow$ area under the curve of (fpr_z, tpr_z)
    end for
    $AUC_v \leftarrow$ mean($AUC_{v,1}...AUC_{v,k}$)
    if Ψ_v could not converge **then**
      $AUC_v \leftarrow 0$
    end if
  end for
  return mean($AUC_1...AUC_w$)
end function

SVMs optimize the model's structure and parameters simultaneously, returning globally optimal solutions. This classifier maps the features into a higher-dimensional space according to a specified kernel [66, 88].

2. *k-Nearest Neighbors*: this classification method relies on a similarity notion based on a distance metric and a number of neighbors k used for assigning a class to unseen data points. The performance of this classifier is enhanced on scenarios as data increases (along with its computational cost). k-Nearest Neighbors executes queries without building a model using non-linear decision boundaries and offers a straightforward parameter setup [26].

3. *Logistic Regression*: even though this classifier might impose a misnomer, the target values in the data are categorical. The class probability estimate of a Logistic Regression is computed as a function of $f(x)$ (i.e., the distance to the decision boundary). This curve is called sigmoid due to its 'S' shape and is in charge of compressing the probabilities into a range between zero and one [50, 56, 67].

3.1.4 Genetic Operators

Each of our ECAC methods uses two operators to conduct the genetic process that creates new offspring each iteration, which runs a reproductive strategy Π times.

Standard One-point Crossover

This operator proceeds if a random crossover probability (uniform distribution) p_c complies with a crossover rate of 0.95, implying a 95% probability of two parents being recombined. If the operator does not proceed, $parent_1$ is returned unmodified immediately. When the

crossover is performed, the genetic material of two parents is interchanged delimited by a random crossover point in the range $cp \in \{1, .., N\}$ (uniform distribution to be able to cut the chromosomes of the parents at any point). Afterward, a slicing operation will induce the formation of a child created from slicing $parent_1$ from gene q_1 to gene q_{cp} and $parent_2$ from gene q_{cp+1} to gene q_N, similar to the operator used by Handl and Knowles [32]. Fig. 3.3 depicts the procedure followed by the Standard One-point Crossover. This example uses the genotype introduced in Fig. 3.2 as $parent_1$ and an additional one as $parent_2$. The result is an offspring with the information of both parents delimited by the crossover point $cp = 8$. As seen in Chapter 2.3.2, this operator has been applied with variations in state-of-the-art Evolutionary Clustering Algorithms. We ensured that this operator keeps at least one member per group after its execution to avoid invalid solutions and the context insensitivity issues caused by the label-based solution representation detailed in Chapter 3.1.1 as mentioned by Hruschka [37] and Falkenauer [17].

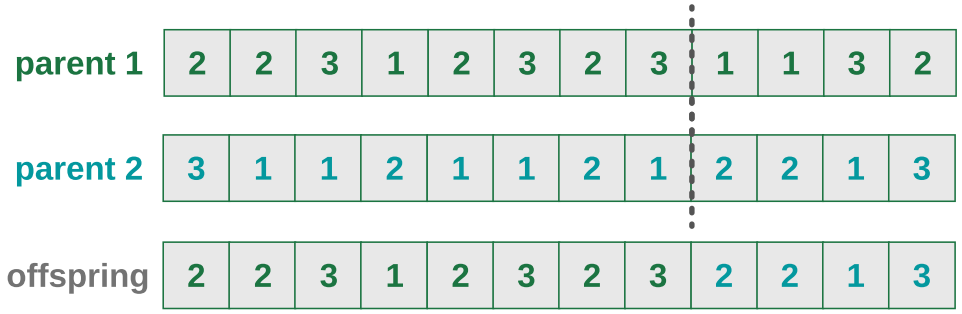

Figure 3.3: Crossover operator creating new offspring from the genetic material of two parents.

One-point Mutation

In this operator, a random number p_m (uniform distribution) is computed to determine if the mutation of a child produced by the crossover operator will proceed or not. The mutation rate is established at 0.98, meaning that there is a 98% probability of mutating an individual's genotype during the reproductive process, similar to the Δ-MOCK algorithm described in Chapter 2.3.2. A mutation point $mp \in \{1, .., N\}$ is produced randomly to indicate the gene from a chromosome that will be modified. The value of gene g_{mp} is then updated to the cluster assigned to gene g_{mp+1}. Fig. 3.4 depicts the mutation procedure in the resultant offspring from Fig. 3.3. The mutation point mp was randomly set as 4; thus, q_4 changed its cluster allocation from 1 to 2 (gene q_4 gets the cluster assignment of q_5).

3.2 F1-ECAC

The Evolutionary Clustering Algorithm using Classifiers and the F_1 score (F1-ECAC) [78, 74] is the successor to ECAC and offers improved areas to simplify the operating pipeline while increasing its performance. This section will outline every modification done to ECAC leading to its second version, F1-ECAC.

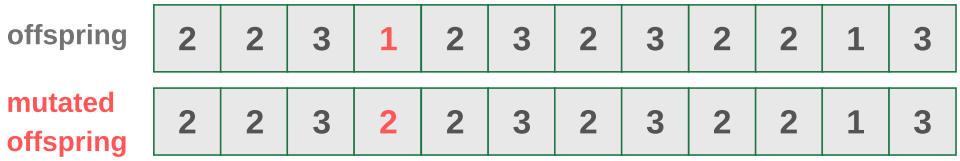

Figure 3.4: Mutation operation performed to the fourth gene of a chromosome, where it is updated using the information of the fifth gene.

3.2.1 Objective Function

The fitness computation of F1-ECAC also uses an ensemble of classifiers to optimize the generalization cluster quality criterion instead of using a typical distance metric. The unweighted average of the predictive performance induced to three classifiers by a clustering solution is the fitness value returned by the objective function. It accounts for our multi-expert approach to evaluate a partition in a single-objective schema (for efficiency purposes). Each classifier uses a set of features X and an individual i's chromosome as labels for training and testing, assessing quality according to a partition's capability to induce multiple well-trained classifiers by using its representative genotype as target labels; as done by ECAC. The design changes made to ECAC that give rise to F1-ECAC are summarized in Table 3.1 and will be described in this section.

Table 3.1: Improvements performed to ECAC in the design of F1-ECAC.

Modification	ECAC	F1-ECAC
Classifiers	Support Vector Machine k-Nearest Neighbors Logistic Regression	Support Vector Machine k-Nearest Neighbors Decision Tree
Training schema	one-vs-rest	multi-class
Evaluation metric	AUC	F_1 score
Label binarization	✓	-
Mutation operator	1-point	Proportional: 5%

The first modification relies upon selecting the classifiers within the objective function. These classifiers are:

1. *Support Vector Machine* as described in Chapter 3.1.3.

2. *k-Nearest Neighbors* as described in Chapter 3.1.3.

3. *Decision Tree*: this intelligible predictive model determines the class of an unseen object starting at the root node of a tree modeled after the data and descending through the

interior nodes based on the feature vector until reaching a terminal node. At this point, a decision is made according to some quality criterion [56, 67].

Decision Trees were selected as the third classification algorithm in F1-ECAC's objective function instead of Logistic Regression models due to the latter's computational cost, which is amplified by the one-vs-rest approach for training the models (requiring every classifier to be trained k times). This way, we end up with a set of classifiers from different families (reducing classifier bias induced by predictive models as suggested by Rodriguez *et al.* [71]) that require only one training per fitness computation by taking advantage of multi-class training and avoiding label binarization prior to the training phase.

Another differentiating point of F1-ECAC is the metric used to evaluate each classifier. The performance metric to test them is their macro F_1 score (computing the unweighted mean of the F_1 score per class obtained by the classifier for multi-class problems with $k > 2$), which is the harmonic mean of precision and recall and returns a numerical value between 0 and 1. The F_1 score is computed as

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \\ F_1 &= 2 * \frac{precision * recall}{precision + recall}, \end{aligned} \tag{3.2}$$

where $precision$ and $recall$ (or $sensitivity$) are used to consider the influence of TP, FN, and FP examples [83]. The F_1 score has the advantage of a more efficient computation because it does not require the generation of prediction probabilities as done by the AUC metric. This design decision has complexity benefits and still exploits the confusion matrices resultant from the classifiers' predictions. Its inclusion also inspired the name of our second contribution in the ECAC series of publications, F1-ECAC. Considering these modifications, the objective function of F1-ECAC computes the fitness of an individual as demonstrated in Algorithm 3, which is exemplified in Fig. 3.5 using the offspring created in the genetic procedure from Fig 3.4.

3.2.2 5% Proportional Mutation

As performed by ECAC and as mentioned in Chapter 3.1.4, this operator is executed if a mutation probability threshold p_m of 98% is met by a randomly generated number (uniform distribution). If the mutation proceeds, a genotype is taken as input, and 5% of its genes are modified in contrast to the One-point Mutation in ECAC, which only modified one of the genes regardless of the dataset size. The extended execution range of this operator has the purpose of keeping important effects in the search space regardless of the value of N. The mutation points are selected randomly (uniform distribution) in the range $mp_a \in \{1, .., N\}$, where $a \in \{1, .., N * 0.05\}$. The cluster assignment of a gene q_{mp} is updated according to gene

Algorithm 3 F1-ECAC Objective Function Algorithm.

function OBJ(X, i)
▷ X: data E features; i: an individual's genotype.
  $X_{train}, X_{test}, y_{train}, y_{test} \leftarrow$ train-test split of X and i.
  for $v = 1.w$ **do** ▷ w is the followed in the fitness computation number of classifiers.
    Ψ_v classifier training with X_{train}, using i_{train} as target labels.
    $F_v \leftarrow F_1$ score (macro-average) of Ψ_v's predictions using X_{test} against i_{test}.
    if Ψ_v could not converge **then**
      $F_v \leftarrow 0$
    end if
  end for
  $fitness \leftarrow$ mean($F_1, ..., F_w$)
  return $fitness$
end function

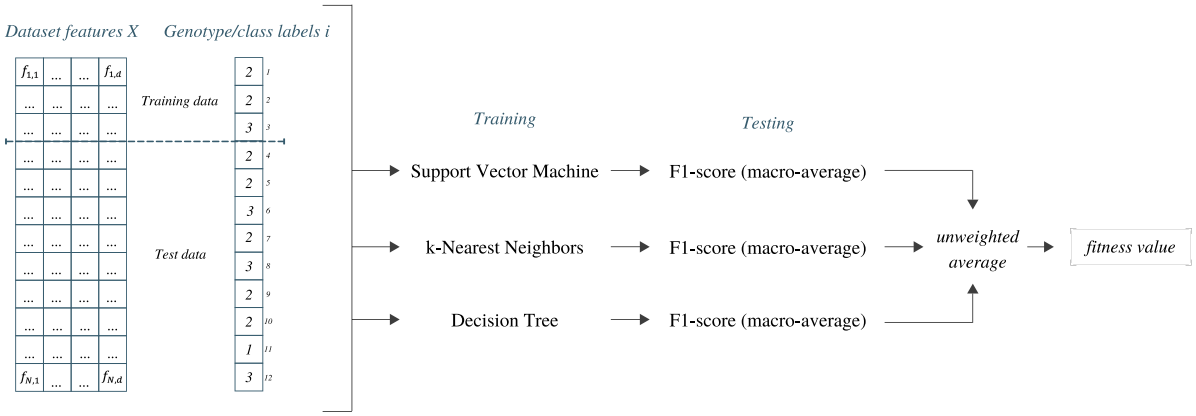

Figure 3.5: Pipeline followed by the objective function of F1-ECAC.

following the mutation point q_{mp+1}, in a similar procedure to the operator used by Murthy and Chowdhury [59]. For instance, if the first mutation point $mp_1 = 3$, gene q_3 is allocated in the cluster where gene q_4 is grouped. As done by ECAC's version, this operator guarantees to keep at least one member per cluster after the mutation procedure. This is the last modification to ECAC that converts it into F1-ECAC, and all of the phases and details of the evolutionary process that were not mentioned in this section were kept unmodified.

3.3 Improvements Leading from F1-ECAC to ECAC-S

The third and final version of our ECAC family of contributions is the Simultaneous Evolutionary Clustering Algorithm using Classifiers or ECAC-S, for short (following the naming convention of our previous algorithms and highlighting the benefits of its parallel implementation) [76]. ECAC-S preserves the design advantages of F1-ECAC presented in Chapter 3.2, but we tackled some aspects that represented an opportunity for an efficiency boost in its implementation. In contrast to F1-ECAC, ECAC-S was designed by testing multiple sets of

modifications instead of only one.

In forthcoming sections, we will go into detail about each improvement done to F1-ECAC focused on retaining solution quality while reducing runtime that brought us to the definitive version of the ECAC series, ECAC-S. The modifications were tested sequentially in varying aspects of the algorithm, as outlined in Fig. 3.6.

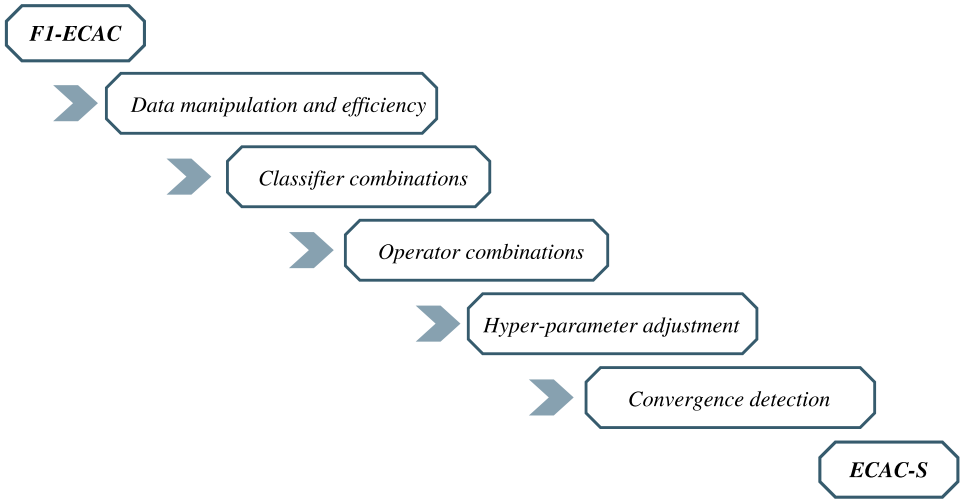

Figure 3.6: Improvements to F1-ECAC tested sequentially in the design process of ECAC-S.

3.3.1 Data Manipulation

The first set of modifications to F1-ECAC involves mandatory changes in data manipulation from the insights raised after its development, as seen in Table 3.2. The central aspect of this series of proposals is the parallel computation of three processes to take advantage of the available hardware:

1. Initial population generation.
2. Fitness computing of a population.
3. Reproductive process including the crossover and mutation operators.

We introduced a heuristic to sort the dataset in the initialization phase of F1-ECAC according to the groups suggested by the Single-linkage clustering algorithm described in Chapter 2.1.2 to avoid the original dataset sorting affecting the solutions through the mutation operator (which updates cluster assignment from a gene q_{mp} according to gene q_{mp+1} as mentioned in Chapter 3.2.2). This way, we keep a consistent exploration of the search space while

assisting the mutation operator and avoiding bias towards the source sorting of the data. Another implementation modification is related to changing the main pipeline of individuals and fitness computation to lists instead of dictionaries. Moreover, we ensured to keep at least one object per class in the training fold resulting from the train test split in the objective function to guarantee that the classifiers are always trained with k classes. All of the components of F1-ECAC that were not mentioned in this section were left intact (i.e., the rest of the aspects of the algorithm work as detailed in Chapter 3.2).

Table 3.2: Data manipulation improvements proposed to F1-ECAC.

Process	F1-ECAC	Proposed Modification
Computing of fitness, reproduction, and initial population	Serial	Parallel
Dataset sorting	Source order	Single-linkage
Main data structure of the implementation	Dictionaries	Lists

3.3.2 Classifier Combinations

Previous studies show that one classifier is insufficient to give a confident approximation of a partition's generalization capabilities [78]. For that reason, we aim to find the best pair of classifiers for the objective function (instead of using three, as F1-ECAC) to reduce computational runtime but still retain our algorithm's capability to return meaningful solutions. The six classifier combinations to be tested are presented in Table 3.3. The previous algorithms from the ECAC series compute the average predictive performance of the classifiers as fitness value to equally weigh their contribution as would be done by a group of experts. Nevertheless, we included the maximum F_1 score returned by the classifiers in our analysis to determine if this metric positively impacted fitness computation as done by VIC (described in Chapter 2.4.2). This approach implies that a positive opinion of one expert is enough to consider a solution as good.

Table 3.3: Classifier combinations tested for the objective function of ECAC-S.

Classifiers	Metric 1	Metric 2
Support Vector Machine/ Decision Tree	Mean F_1 score	Max F_1 score
k-Nearest Neighbors/ Decision Tree	Mean F_1 score	Max F_1 score
k-Nearest Neighbors/ Support Vector Machine	Mean F_1 score	Max F_1 score

3.3.3 Genetic Operator Combinations

We propose three genetic operators and seek to observe the most beneficial combination for performing the recombination and mutation procedures. The operator combinations to be evaluated are reported in Table 3.4. The three previously unseen genetic operators are:

1. **Two-point Crossover**: this is an operator that requires the computation of a probability to determine if two parents will be recombined, as done by the One-point Crossover presented in Chapter 3.1.4. There is a 95% probability of proceeding with the recombination in this case. If the operator does not proceed, $parent_1$ is also returned. In case the crossover is held, this operator slices and merges two parents delimited by two random crossover points $cp_1, cp_2 \in \{1, .., N\}$. An offspring is a result of slicing genes 1 to q_{cp_1-1} from $parent_1$ with genes q_{cp_1} to q_{cp_2-1} from $parent_2$ with genes q_{cp_2} to q_N from $parent_1$.

2. **Individual Gene Probability Crossover**: this operator is initialized by using $parent_1$ as the base genotype. A random (uniform distribution) discrete probability p_c between 0 and 1 is computed for each gene q_{cp} to determine individually if it will acquire the cluster membership of the gene in the same position $q_{cp}, cp \in \{1, .., N\}$ but from the opposite parent. If $p_c = 0$, the gene remains unmodified, otherwise, the cluster assignment of gene g_{cp} is interchanged with gene g_{cp} from $parent_2$.

3. **Individual Gene Probability Mutation**: following the same methodology as the Individual Gene Crossover Probability, instead of mutating 5% of genes by selecting them randomly, a mutation probability p_m is computed for each gene $g_{mp}, mp \in \{1, .., N\}$, at 5% a mutation rate. This operator and the Individual Gene Probability Crossover are similar to those implemented by Garza-Fabre *et al.* [23] discussed in Chapter 2.3.2.

Table 3.4: Operator combinations tested in the design process of ECAC-S.

Crossover Operator	Mutation Operator
2-point	5% Proportional Mutation
Individual Gene Probability	5% Proportional Mutation
1-point	Individual Gene Probability
2-point	Individual Gene Probability
Individual Gene Probability	Individual Gene Probability

3.3.4 Hyper-parameter Adjustment

Two previously-unseen values of 50% and 25% are proposed for the test size argument to determine the percentage of random objects assigned to the testing fold in each fitness computation (the 75% test size value used by F1-ECAC is also considered to find the optimal one). The amount of data that is located for testing the classifiers in the train-test split will have a positive correlation with runtime; increasing it will induce a good set of classifiers in

exchange for higher computation time, whereas decreasing it might result in less CPU time but cause classifiers to have lower performance (which in this application is not considered an issue because the classifier might still be able to offer a confident fitness evaluation and return meaningful partitions without compromising the quality of the solutions).

3.3.5 Convergence Detection

An early-stopping parameter was implemented to stop the execution of ECAC-S if the best individual is not updated after an established number of iterations. This modification breaks the evolutionary process once convergence has been reached, which is reflected in the absence of improvement of the highest-performing partition. A positive outcome of including an early-stopping parameter is the direct reduction of 10% or 20% of the generations (to be tested along with 100%, or no early-stopping, as done in F1-ECAC). However, limiting the search space so abruptly could reduce the quality of the solutions, despite diminishing runtime significantly.

To summarize, the set of proposed changes to be tested starts with F1-ECAC and ends with ECAC-S, our ultimate algorithm in the ECAC series. We seek to find an adequate balance between performance and runtime with the present modifications. The sequential evaluation implies retaining the properties of an outperforming component/parameter combination with a particular focus on reducing computational runtime. The definitive version of ECAC-S will be detailed now, and the statistical analysis leading to each design decision on the properties mentioned before can be found in Chapter 5.3.

3.4 ECAC-S

In this section, we present the latest contribution of the current thesis project, the Simultaneous Evolutionary Clustering Algorithm using Classifiers, ECAC-S[1]. This algorithm is the result of multiple versions and iterations as discussed in Chapter 3.3 on its components and parameters, leading to the highest solution quality with a more efficient implementation and design, compared to F1-ECAC. Algorithm 4 outlines the computation of ECAC-S, the definitive version of the ECAC series, which requires as input a dataset E with either numerical or categorical features, the required number of clusters k, the population size Π (maintained along the clustering process), and the number of maximum iterations g.

3.4.1 Evolutionary Process

The clustering pipeline of ECAC-S follows a genetic approach as search strategy. It begins by sorting E according to a partition suggested by the Single-linkage algorithm. This way, we provide the mutation operator with a heuristic to continue its procedure of assigning cluster membership conditioned by the position of an object represented by a gene $q_1, .., q_N$ within E with a more informed and guided search path, avoiding the source sorting of the data to affect the solutions. The initial population of ECAC-S is constituted by Π random individuals, each representing a partition, in a parallel computing schema (creating one individual per core) using the label-based representation (described in Chapter 3.1.1) to assign each object into a

[1]The implementation of ECAC-S is available in Ref. [76].

Algorithm 4 ECAC-S algorithm.

function ECAC-S(E, Π, g, k)
▷ E: data; Π: population size; g: max. gens.; k: number of clusters.
  $E^{'} \leftarrow$ sorted E according to Single-linkage clustering.
  $X \leftarrow$ data $E^{'}$ features.
  $\Phi \leftarrow i_1, ..., i_\Pi$ ▷ Initial population with k clusters.
  $f_\Phi \leftarrow$ fitness computing with OBJ(X, Φ). ▷ See Algorithm 5.
  for $u = 1.g$ **do**
    $selected \leftarrow \Pi$ individuals are selected from Φ.
    $parentPairs \leftarrow selected$ elements form Π pairs.
    $C_1...C_\Pi \leftarrow$ 2-point Crossover operator applied to $parentPairs$.
    $C_1^{'}, ..., C_\Pi^{'} \leftarrow 5\%$ Mutation applied to $C_1, ..., C_\Pi$.
    $\Phi^{'} \leftarrow C_1^{'}, ..., C_\Pi^{'}$.
    $f_{\Phi'} \leftarrow$ fitness computing with OBJ($X, \Phi^{'}$).
    $\Phi \leftarrow \Phi^{'}$. ▷ New population handover.
    $b \leftarrow$ all-time best solution.
    if b's fitness=1 or no change for $g * 0.20$ iterations **then**
      break
    end if
  end for
  $b^{'} \leftarrow$ reordered b according to the original E's sorting.
  return $b^{'}$
end function

cluster $C_1, .., C_k$ and ensuring presence of all clusters. An evolutionary process creates new solutions that increase in quality over time. The 2-point Crossover and 5% Mutation Operators (see Chapters 3.3.3 and 3.2.2 for more details) are in charge of recombining and modifying individuals to create enhanced offspring in each generation using parallel computing allocating one parent pair per core. The clustering process is terminated if the early-stopping condition detects convergence (i.e., the best solution is not updated after 20% of the maximum generations g), a global maximum of 1 is found, or the number of iterations has reached g. The best solution b is defined as the partition that maximizes the generalization criterion being optimized (b is sorted back to the original order from E before it is returned at the end of the process).

3.4.2 Objective Function

The objective function of ECAC-S is computed as indicated in Algorithm 5 for computing the fitness of a population using a parallel framework (allocating one individual per core). This is done by performing the following process per individual. A partition's representative genotype is used as class labels to train and test two classifiers along with data E's features X after splitting the data into training and testing folds (random training fold proportion of 25%). The ensemble in the objective function is constituted by the Support Vector Machine and Decision Tree classifiers with the parameters mentioned in Chapter 4.2.2. The fitness

Algorithm 5 ECAC-S Objective Function algorithm.

function OBJ(X, Φ)
▷ X: data features; Φ: a population of individuals.
  for $j = 1.\Pi$ **do**
    $X_{train}, X_{test}, y_{train}, y_{test} \leftarrow$ train-test split of X and individual i_j.
    SVM training with X_{train} and y_{train} as target labels.
    DTC training with X_{train} and y_{train} as target labels.
    $pred_{SVM}, pred_{DTC} \leftarrow$ predictions with X_{test}.
    $f_{SVM} \leftarrow F_1$ score of y_{test} and $pred_{SVM}$.
    $f_{DTC} \leftarrow F_1$ score of y_{test} and $pred_{DTC}$.
    $fitness_j \leftarrow mean(f_{j,SVM}, f_{j,DTC})$.
  end for
  $f_\Phi \leftarrow fitness_1 ... fitness_\Pi$.
  return f_Φ
end function

value returned to evaluate a partition is defined as both classifiers' mean macro-average F_1 score.

3.4.3 Remarks

The hyper-parameters for delimiting the number of members per population and maximum generations Π and g set as 200 return high-quality solutions in problems of similar size to the multi-dimensional general-purpose and cross-domain data used for validation of our design (as seen in Chapter 5.3). Nevertheless, if ECAC-S is being used in a data mining task requiring a more extensive search space, we suggest quadrupling its size by changing both hyper-parameters to 400. To summarize, ECAC-S is an Evolutionary Clustering algorithm that uses the benefits of parallel computing to propose partitions that optimize an ensemble of classifiers as an approximation of its generalization capabilities, which is the cluster quality criterion to be maximized. At this point, we conclude to elaborate on the definition, design process, implementation, and scalability enhancements of our ECAC series of contributions.

Chapter 4

Experimental Frameworks

We worked with three separate experimental frameworks, each one developed for the article related to ECAC [77], F1-ECAC [78], and ECAC-S; they aim to validate the performance of an ECAC method in multiple scenarios, are self-contained, and should not be analyzed in an interspersed manner, as important variables change between them.

4.1 ECAC Experimental Setup

This section outlines the inputs of ECAC used for experimentation, such as the datasets and hyper-parameters. The firstborn in the ECAC family was tested against the authors' implementation of the HG-means algorithm [28] described in Chapter 2.2.1. We chose to test ECAC against a remarkable algorithm belonging to our algorithm's family (single-objective Evolutionary Clustering Algorithms) as a proof of concept to realize how competitive our idea was against one of the most representative methods of its kind. The implementation of ECAC was programmed using the Scikit-learn [65] and Pandas [54, 68] libraries in the Python programming language [87] for coding the supervised classification models and data manipulation, respectively. The experiments were held on a dual-core Intel Core i5 2.7 GHz processor with 8 GB of RAM. The source of ECAC is available at Ref. [73].

4.1.1 Datasets

The ten datasets selected from the UCI Machine Learning Repository [10] and their structural details can be found in Table 4.1. Datasets of reduced dimensions are included in this framework, but as shown in further sections, the experimental phases introduce larger datasets and more complex experimentation as the efficiency of our algorithms is enhanced. The features of the datasets were standardized to enhance convergence and avoid context insensitivity issues.

4.1.2 Hyper-parameter Calibration

The number of clusters k that we required the algorithms to include in their solutions was set as specified in the source from where the datasets were retrieved [10]. The population size Π

Table 4.1: Dataset information regarding the experimental framework of ECAC.

Dataset	Classes	Features	Objects
breast-tissue	6	9	106
dermatology	6	34	366
ecoli	8	7	336
forest	4	27	523
glass	6	9	214
iris	3	4	150
leaf	36	14	340
liver	16	5	345
transfusion	2	4	748
wine	3	13	178

was set to 20 individuals, and the g argument, which limits the number of generations held in the evolutionary process, was established at 2,000 for both algorithms.

The classifiers within the objective function of ECAC have the following configuration:

- *Support Vector Machine*: linear kernel, L_2 regularization parameter equal to 1 for 5,000 maximum iterations.
- k-*Nearest Neighbors*: two nearest neighbors with uniform weights and Euclidean distance metric.
- *Logistic Regression*: L_2 regularization parameter equal to 1 for 100 maximum iterations.

An equal input setting in data and hyper-parameters suggests a fair benchmark with equal conditions. Now that the hyper-parameters used to obtain solutions have been outlined, we will proceed to explain the pipeline followed to evaluate the performance of ECAC against HG-means.

4.1.3 Computational Experiments

We tested the quality of solutions created by ECAC and HG-means, considering their quality and the time required to produce them. Every time an algorithm is run to obtain a partition, we store the fitness of the returned partition and runtime. Subsequently, the Adjusted RAND Index (ARI) [4, 84] of an algorithm's partition is computed against the ground truth labels retrieved from the same source as the datasets [10]. The ARI is considered an objective external measure of cluster quality to compare a solution returned by an algorithm against a ground truth partition and is mainly used in controlled experimental environments [32]. We are aware that most real applications do not count on reference partitions, but for these sets of tests, they are only used to validate the performance of our methods and are not involved in the clustering process. ECAC and HG-means were run ten independent times per dataset to avoid their stochastic nature affecting the analysis.

4.2 F1-ECAC Experimental Setup

In this section, we describe the experimental framework followed in the development of F1-ECAC. In this case, we decided to test our proposal against k-means, Single-linkage Agglomerative Clustering, DBSCAN, HG-means, MOCLE, Δ-MOCK (previously described in Chapter 2) and ECAC, to identify our algorithm's position within the literature regarding solution quality considering algorithms with complex designs and from different families. The classifiers within the objective function of F1-ECAC and the k-means and Single-linkage Agglomerative Clustering methods were coded with the Scikit-learn and Pandas libraries in the Python programming language [65, 54, 68, 87]. The experiments were performed in a dual-core Intel Core i5 2.7 GHz processor with 8 GB of RAM.

Table 4.2: Datasets used in the experimental framework of F1-ECAC.

Dataset	Classes	Features	Objects
aggregation	7	2	788
breast-cancer-wisconsin	2	30	569
breast-tissue	6	9	106
dermatology	6	34	366
ecoli	8	7	336
forest	4	27	523
glass	6	9	214
iris	3	4	150
jain	2	2	373
leaf	36	14	340
liver	16	5	345
parkinsons	2	22	195
pathbased	3	2	300
r15	15	2	600
seeds	3	7	210
segment	7	19	210
spiral	3	2	312
transfusion	2	4	748
wine	3	13	178
zoo	7	16	101

4.2.1 Datasets

We used 20 numerical datasets from the UCI Machine Learning Repository [10] and the Clustering Benchmark repository of Fränti and Sieranoja [20, 25, 5, 89, 41]. Table 4.2 displays the information of every dataset used for testing F1-ECAC. Fig. 4.1 depicts the ground truth groups of the 2-D synthetic data used for testing our second algorithm[1]. Using artificially-generated data helps understand the biases and structures favored by clustering algorithms,

[1]Fig. 4.1 and the following figures were retrieved using the Seaborn library [92] in the Python programming language [87].

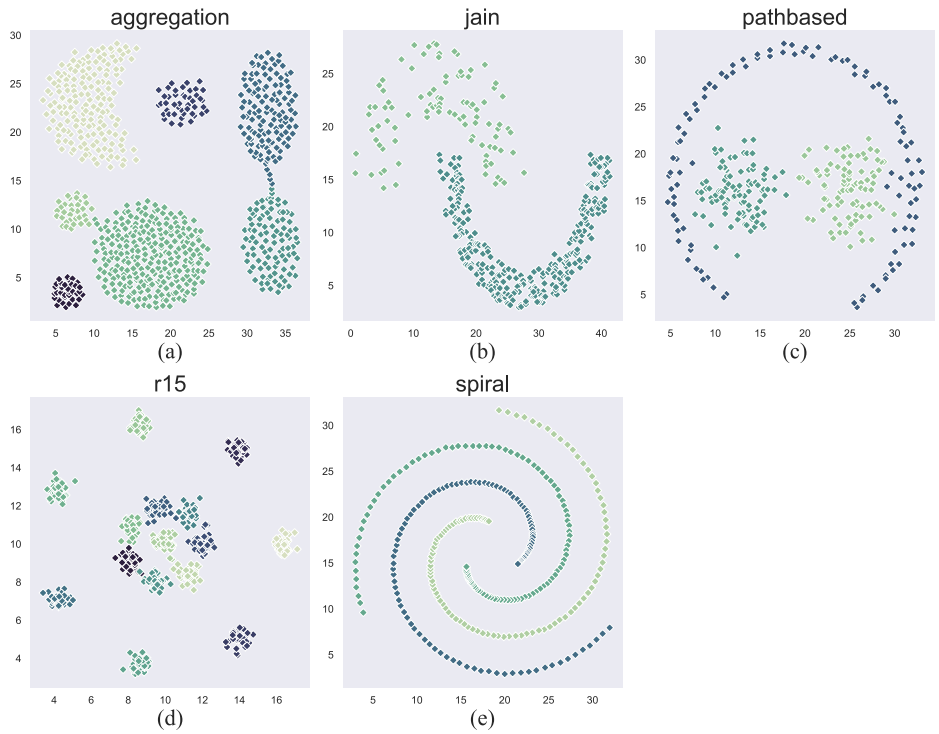

Figure 4.1: Visualizations of the synthetic data included in the experimental setup of F1-ECAC.

raising meaningful behavioral insights about them. However, we understand that its processing will be inherently different from clustering data from real studies; thus, we decided to limit the proportion of synthetic datasets to 25%. Every dataset was standardized before performing clustering with the algorithms.

4.2.2 Hyper-parameter Calibration

In these subsections, we will specify the hyper-parameters used to set up each algorithm in the validation benchmark of F1-ECAC.

- k-*means*: 10 random initializations for 250 maximum iterations.
- *Single-linkage Agglomerative Clustering*: Euclidean distance metric to compute dissimilarity.
- *DBSCAN*: seven minimum samples per group and ϵ set to 0.30.
- *HG-means*: population size Π set to 20 for 2,000 generations g. We used the author's implementation available in Ref. [28].
- *MOCLE*: as the implementation of this algorithm was not publicly available up to the development of this thesis project, we implemented it from scratch. We used NSGA-II as the search engine (based on the code from Ref. [47]). Our implementation of

MOCLE creates variable population sizes each time, and was set to 50 maximum generations. We introduced a label-based representation for interpretability, and the initial population is constituted by clusterings created using k-means, Agglomerative Clustering with Single and Average-linkage affinities (all of them using the Scikit-learn library [65]), and the Shared Nearest Neighbors method [11] with varied hyper-parameters using the authors' implementation from Ref. [12]. For the connectedness objective function aforementioned in Chapter 2.3.1, we used 5% of N as L. The crossover operation was performed by the Meta-clustering Algorithm by Strehl and Ghosh [85] using the implementation by Kultzak from Ref. [49], avoiding cloned solutions. Our implementation of MOCLE is available at Ref. [75].

- Δ-*MOCK*: we used the C++ implementation of the authors from Ref. [22] with 100 individuals for 100 maximum generations. We used the Δ-locus representation and set a maximum k of 50.

- *F1-ECAC*: Π and g arguments set to 200; thus, 120,000 classifier trainings are held along the evolutionary process. The parameters for the classifier ensemble within the algorithm's objective function are:

  - *Support Vector Machine*: linear kernel with L_2 regularization value of 1 for unlimited iterations.
  - k-*Nearest Neighbors*: five nearest neighbors with an inverse weight according to their Euclidean distance.
  - *Decision Tree*: unlimited maximum depth using the entropy criterion for building the tree.

  The Python implementation of F1-ECAC is available in Ref. [74].

- *ECAC*: we include F1-ECAC's predecessor as it is crucial to evaluate the runtime and performance differences between them. Π of 20 and g set to 2,000. The parameters of the classifiers inside ECAC's objective function were set up as recommended in its article [77] (as detailed in Chapter 4.1.2). We used ECAC's published implementation available at Ref. [73].

4.2.3 Computational Experiments

Each algorithm was run ten independent times per dataset, and the k values were set as specified in the source from which we retrieved each algorithm (for the algorithms that require this argument as input). Again, the ground truth class labels from the datasets are used as a reference for evaluating the solutions returned by the algorithms against them using the ARI metric [4, 84] and are not influential in the clustering process. An equal-condition testing environment across all datasets aims to propel a fair comparison between the methods. F1-ECAC was tested against the seven mentioned contestant methods with varying indicators to be validated using hypothesis testing between observations. For testing comparisons between two treatments, we used the non-parametric Wilcoxon test [8, 69], and for benchmarks over two

treatments, we used the non-parametric $1xN$ Friedman test [21] with the Holm post-hoc [8], both with a significance level of $\alpha = 0.05$[2].

We selected the solution with the highest ARI when dealing with multi-objective algorithms that return a Pareto front with multiple non-dominated solutions; thus we keep one solution per run across every contestant method. Another remark is that we wanted to test our decision to include three classifiers to compute fitness in an ensemble schema as an additional experiment. Hence we ran F1-ECAC using each classifier separately to evaluate if one classifier could be enough to create meaningful partitions. Therefore, the objective function evaluates a partition by training and testing only one classifier in this experiment, which reduces computational runtime but might compromise solution quality.

4.3 ECAC-S Experimental Setup

As mentioned in Chapter 3.3, there are multiple areas of F1-ECAC that have to be tested to discover the most suitable combination of components and parameters that will constitute the definitive version of ECAC-S. Hence, we will compare the effects of each modification done to F1-ECAC[3] as detailed in Chapters 3.3.1 through 3.3.5. The implementation of each version was programmed with Pandas [54, 68] in the Python programming language [87] (the parallel operations were implemented using its native Multi-processing library to make use of every core in the computer), and the classifiers in the objective function of each one of them were coded using the Scikit-learn [65] library. These experiments were performed in an eight-core Apple M1 3.2 GHz processor with 8 GB of unified memory.

4.3.1 Datasets

The 30 datasets selected for evaluating each intermediate version between F1-ECAC and ECAC-S were obtained from the UCI Machine Learning Repository [10] and are listed in Table 4.3; Fig. 4.2 depicts a histogram with the number of classes per dataset. The features across all datasets were standardized, and those containing categorical features were encoded into numerical ones.

4.3.2 Hyper-parameter Calibration

The arguments Π and g were set at 200 across every version leading from F1-ECAC to ECAC-S to evaluate their differences under equality conditions. The parameters for the classifiers were set as the published version of F1-ECAC (see Chapter 4.2.2).

[2]The number of datasets is at least two times the number of treatments; thus, this assumption for the Friedman test holds for the experiments.

[3]For comparing F1-ECAC against its modified versions leading to ECAC-S, we used the F1-ECAC implementation used for publication available at Ref. [74].

Table 4.3: Datasets used in the experimental framework of ECAC-S.

Dataset	Classes	Features	Objects
absenteeism-at-work	19	19	740
arrhythmia	13	279	452
breast-cancer-wisconsin	2	30	569
breast-tissue	6	9	106
car-evaluation	4	6	1728
dermatology	6	34	366
echocardiogram	2	10	75
ecoli	8	7	336
forest	4	27	523
forest-fires	12	11	517
german-credit	2	20	1000
glass	6	9	214
hepatitis	2	19	155
image-segmentation	7	19	2310
ionosphere	2	34	351
iris	3	4	150
leaf	30	14	340
liver	16	5	345
parkinsons	2	22	195
seeds	3	7	210
segment	7	19	210
sonar	2	60	208
soybean-large	19	35	307
student-performance	17	32	649
tic-tac-toe	2	9	958
transfusion	2	4	748
user-knowledge-modeling	4	5	403
wine	3	13	178
yeast	10	8	1484
zoo	7	16	101

4.3.3 Computational Experiments

Every dataset was run ten independent times in each modified version of the algorithm. The k values were set as specified in the original dataset information (in cases where the theoretical value differed from the number of clusters found in the reference labels, the real k was used for a fair comparison). The solutions by each algorithm version were contrasted against the reference labels of each dataset (retrieved from the UCI Machine Learning Repository [10]). Such partitions are only used for evaluating algorithm performance in a comparison held after the clustering process of an algorithm has ended. Just as held in previous frameworks, the metric to juxtapose a clustering solution and its ground truth is the ARI [4, 84] between them to assess their similarity degree [4, 32]. This way, we estimate the capability of a clustering algorithm to partition data as close as a human expert would.

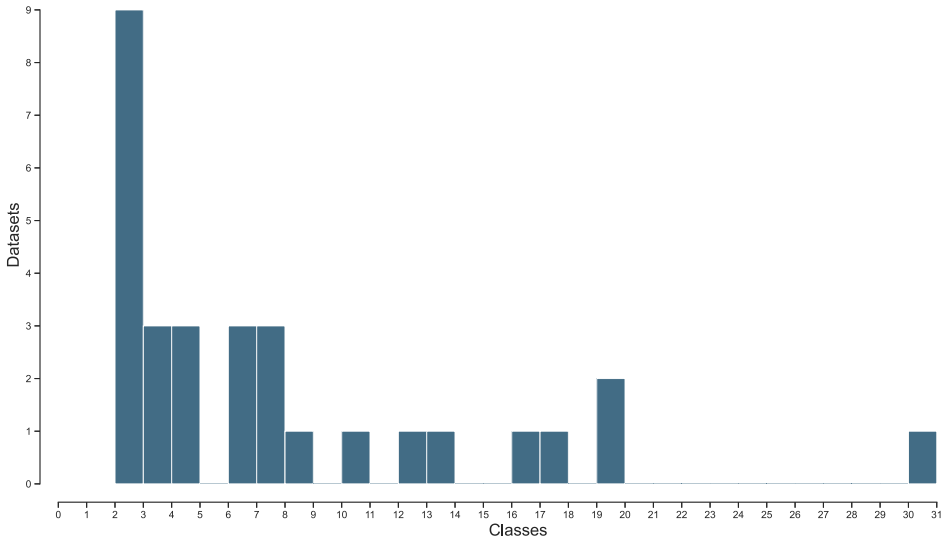

Figure 4.2: Histogram of the number of classes per dataset used in the design process of ECAC-S.

When comparing the mean ARI or time per dataset returned by two algorithms, we used the non-parametric Wilcoxon test [8, 69]. In cases over two treatments, we used the non-parametric NxN Friedman test and the Nemenyi post-hoc [21, 61] with an α of 0.05 to identify statistically significant differences. Hypothesis testing was implemented using the Autorank [36] and Orange [7] Python libraries. In the statistical analysis, an average rank close to one implies better solutions (partitions from an algorithm presenting higher average ARI values or lower computational runtime than the rest).

4.3.4 Image Segmentation Application

As an additional analysis, we present a real application of ECAC-S in an image segmentation task. The images in this benchmark were obtained from Google Maps [27]. We selected ten locations and captured source images of 180x180 pixels as presented in Fig. 4.3. We created a ground truth partition for each image by arbitrarily segmenting them into different classes. We propose a number of clusters according to the natural distribution of each capture. This way, we generated segmentation masks that reflect the color relations within the pixels in a fashion we would like a clustering algorithm to be capable of doing. We resized each image and reference masks to dimensions of 64x64 pixels and used this resolution for clustering. The features X, in this case, were the standardized Red, Green, and Blue pixel values; thus, we are working with ten datasets with 4,096 observations and three attributes each. The ground truth labels were generated by parsing each pixel in the reference masks and assigning it a numerical cluster label.

We compared ECAC-S against leading algorithms that have been successfully applied to image segmentation [82, 53] using the following hyper-parameters:

Figure 4.3: Source satellite captures used in the image segmentation benchmark.

- **Birch**: threshold of 0.50 and branching factor set to 50 [96].
- **DBSCAN**: 7 minimum samples per cluster and ϵ set to 0.30 [13, 81].
- k**-means**: 10 random initializations for 250 maximum iterations [52].
- **Spectral-clustering**: eigenvector quantity set equal to k, with discretizing technique for assigning labels in the embedding space [62].
- **ECAC-S**: values of 400 set to Π and g (population size and maximum generations, respectively) with the implementation used for publishing the article (Ref. [76]).

All competing clustering algorithms were coded with the Scikit-learn library [65] in the Python programming language [87]. Every algorithm processed each image ten independent times. Once again, the metric used to assess the similarity between algorithm-generated solutions and its ground truth mas was the ARI due to its applicability in comparing partitions with group labels that have no semantic relation [4, 80]. The NxN Friedman test [21] and the Nemenyi post-hoc [61] were used for the statistical analysis using the Autorank [36] and Orange[7] Python libraries, considering an $\alpha = 0.05$ (an algorithm with an average rank close to 1 implies solutions of overall higher-quality). The image segmentation benchmark was performed in an eight-core Apple M1 3.2 GHz processor with 8 GB of unified memory.

These have been the experimental frameworks related to the development and application of ECAC, F1-ECAC, and ECAC-S; further chapters will be focused on the insightful discussion of the results obtained from them.

Chapter 5

Results and Discussion

In the following sections we will address the experimental results, according to the previously-defined experimental frameworks. Similar to the design of the benchmarks, the analysis of the results gets more profound as the evolution of the ECAC series requires it. The following sections evidence the capabilities and limitations of the clustering criterion and search schema in ECAC, F1-ECAC, and ECAC-S.

5.1 ECAC Experimental Results

The outputs of the tests detailed in Chapter 4.1 are presented in the following sections for evaluating ECAC against a representative method of its direct family, HG-means.

Table 5.1: Mean runtime per dataset of HG-means and ECAC, and average fitness of the solutions returned by our first algorithm. Values in bold represent the lowest average computational runtime and the highest average ARI per dataset.

Dataset	HG-means (s)	ECAC (s)	HG-means (ARI)	ECAC (ARI)
breast-tissue	**0.0232**	2,250.97	0.0999	**0.3765**
dermatology	**0.8802**	4,265.44	-0.0072	**0.0186**
ecoli	**0.1618**	3,100.36	**0.4262**	0.2165
forest	**0.2028**	7,141.61	**0.4987**	0.0044
glass	**0.0762**	2,451.73	**0.2598**	0.2375
iris	**0.0161**	1,201.93	0.7302	**0.7793**
leaf	**1.3717**	14,254.7	0.2950	**0.3477**
liver	**0.3784**	6,018.51	0.0305	**0.06**
transfusion	**0.0280**	1,422.60	**0.0795**	0.0339
wine	**0.0168**	1,376.43	0.3711	**0.7785**

5.1.1 Clustering Performance Analysis

The mean runtime and ARI per dataset obtained by HG-means and ECAC are presented in Table 5.1. HG-means was more efficient than ECAC due to its exceptionally fast implementation, and we noted this aspect for improvement in the posterior releases of the ECAC series of publications. Still, we believe that outperforming a method in solution quality (ARI) is prioritized and justifies an increase in runtime. Furthermore, the average fitness of the solutions created by ECAC with values close to 1 might imply an overfitting issue in the classifiers. Nevertheless, the fitness value is only an approximate measurement of the generalization capabilities of a partition computed by the classifiers for finding the separation between classes. The boundary of this project is set up to the recommendation of a partition and neither the solutions nor the trained classifiers from the last generation are intended to be used in a further predictive classification problem.

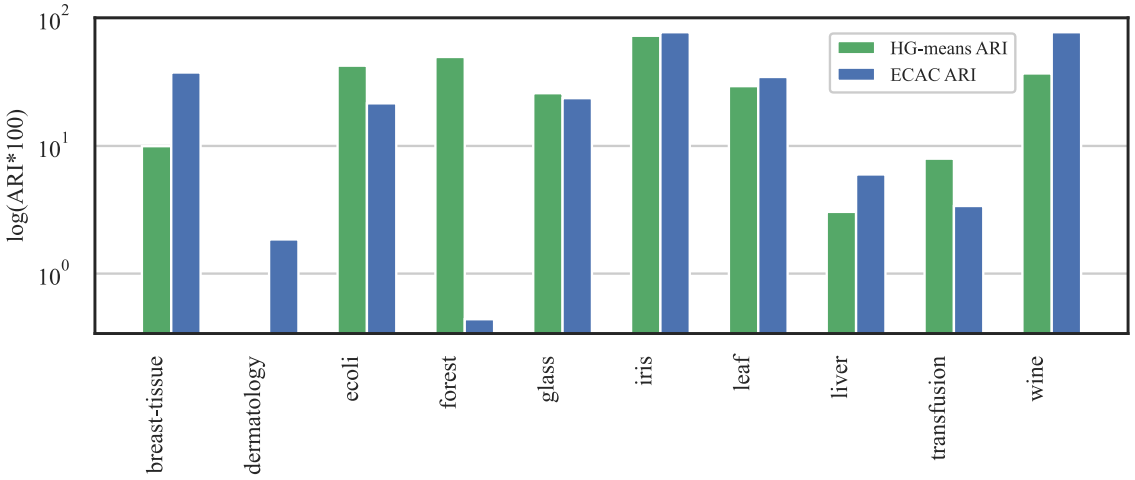

Figure 5.1: Mean ARI per dataset obtained by ECAC and HG-means.

Fig. 5.1 depicts the performance comparison between ECAC and HG-means, contrasting their mean ARI obtained per dataset. The x-axis contains the ten datasets detailed previously in Table 4.1, and the average ARI is located in the y-axis using the logarithmic scale for a clearer plot of the results. ECAC outperformed HG-means in six out of ten datasets in this metric, returning solutions that are closer to the reference partition in most datasets. HG-means struggled especially to cluster the dermatology dataset, and it even failed to converge to an acceptable solution with the hyper-parameters used (see Chapter 4.1.2), getting an average ARI of -0.0072. Fig. 5.2 contains the correlation plot to compare the solutions of ECAC and HG-means. The highly-positive correlation between the fitness values returned by the objective function of ECAC and their resultant ARI suggests that our algorithm favors partitions that are similar to their ground truth and is able to adapt to the structures within a dataset without imposing a predefined hyper-shape into it. Moreover, our algorithm's fitness values are not correlated with the k value, opposite to the behavior of k-means (see Chapter 2.1.1). However, one drawback is the susceptibility of ECAC towards large datasets, as they tend to affect its performance.

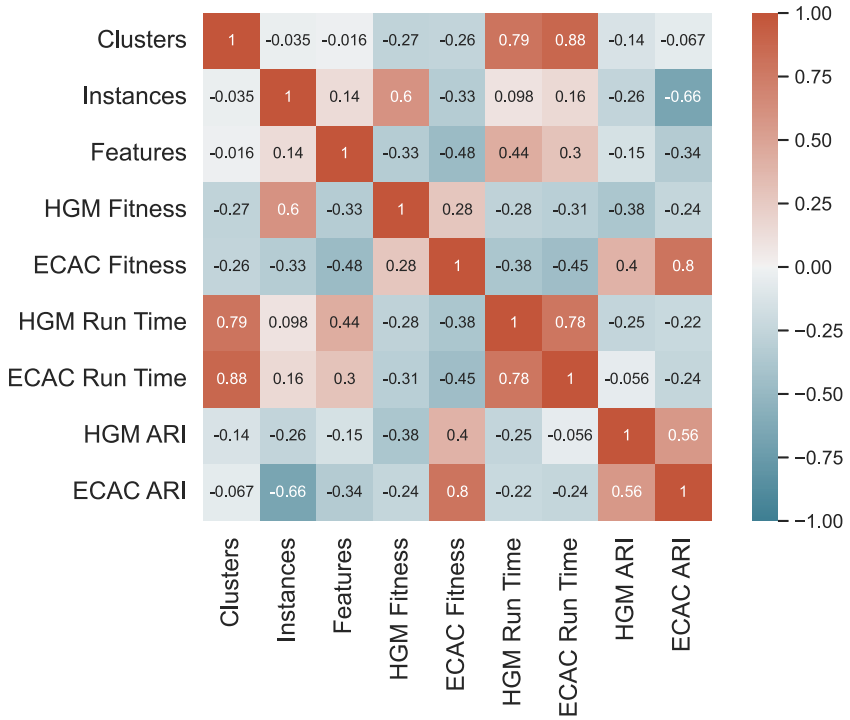

Figure 5.2: Correlation plot with multiple performance metrics for comparing the solutions of ECAC and HG-means.

The three scatter plots in Fig. 5.3 illustrate the 178 objects from the wine dataset, which contains wine samples from three Italian cultivars according to their chemical and physical attributes (a clustering algorithm should be able to detect these groups without either the need for an external expert or previous knowledge on the subject). The three subplots demarcate the relations between the proanthocyanins and total phenols of every wine in the dataset, and the color of each point is related to the cluster assignment from the ground truth labels and the partitions returned by ECAC and HG-means. This was the best-performing dataset of our algorithm, and the increased similarity to the reference partition is perceptible visually in the bottom subplot.

5.1.2 Remarks

The cluster quality criterion and search schema used in ECAC led to an increase in similarity in the solutions against reference partitions created by a human or an experimental study. The inclusion of classifiers for evaluating partitions in an evolutionary process implies a model that is flexible towards the structures of the clusters within a dataset. ECAC is an algorithm suitable for generating meaningful partitions that can be used and analyzed for further decision-making when working with datasets carrying the natural condition of having overlapping clusters that do not follow any geometrical shape.

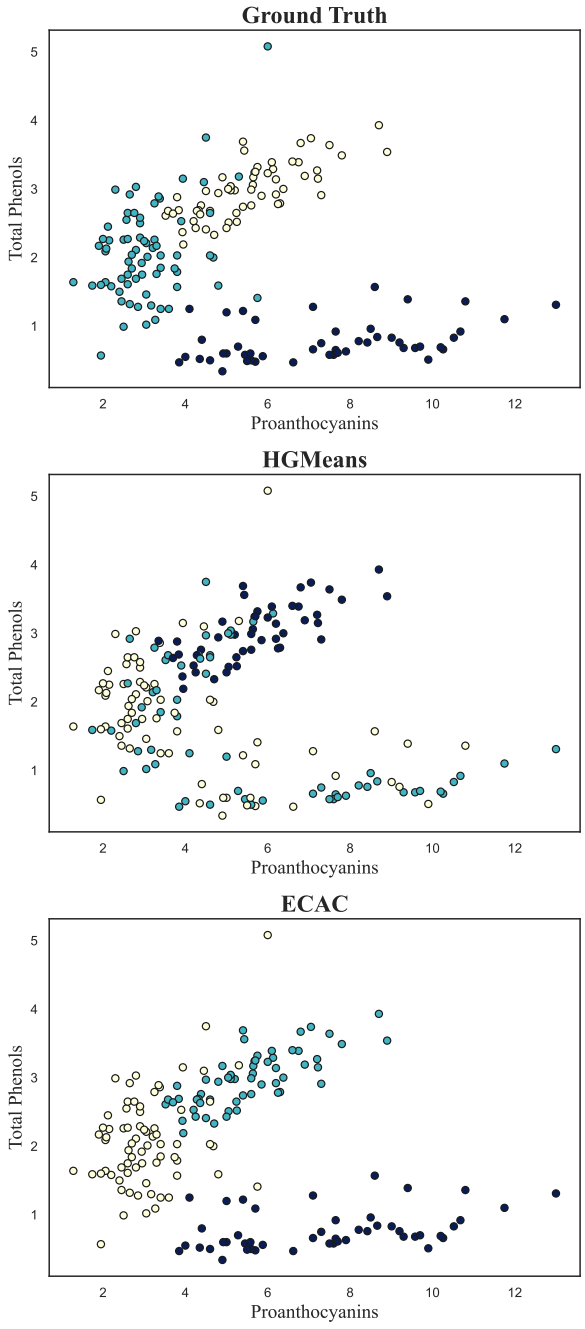

Figure 5.3: Partitions of the wine dataset as suggested by its ground truth, HG-means, and ECAC.

5.2 F1-ECAC Experimental Results

This section points out the experiments' results as mentioned in Chapter 4.2. First, we will contrast the performance differences as a consequence of using one or three classifiers in the objective function of F1-ECAC, to then proceed to present the differentiation in solution quality and runtime of F1-ECAC and its predecessor ECAC (see chapter 3.1). Afterward, a general benchmark is held between F1-ECAC and other evolutionary and traditional methods.

5.2.1 Ensemble vs. One Classifier

Table 5.2 contains the average ARI per dataset of F1-ECAC (in its full classifier ensemble objective function) versus the resulting versions of the algorithm obtained by replacing the ensemble with only one classifier (we used the same classifiers detailed in Chapter 3.2, just including each of them separately). F1-ECAC returned higher scores than versions with only one classifier in 14 out of 20 datasets. Remarkably, the versions using one classifier got a higher mean ARI than F1-ECAC in 3 out of the 5 synthetic datasets. In such cases, there was a coincidence between the evident structure and class separation of artificial data (which does not present overlapping clusters) and the strong bias of the function optimized by the classifiers. The advantages of using a classifier ensemble instead of a single classifier schema suggest that the multi-expert approach leads to partitions of higher quality. The multi-expert schema emulated by F1-ECAC ensemble objective function acts as a voting system for assessing generalization where the purpose is to find if a partition is capable of inducing a set of well-trained classifiers.

Fig. 5.4 displays the fitness of the best individual and entire population per generation (mean value) for visualizing the convergence of F1-ECAC's evolutionary process when creating a partition of the Iris dataset[1]. The enhancement of the population as they go through the evolutionary process starts very quick (as seen in the upwards trend of both plots) and reaches a convergence point of marginal increase around generation 100; however, we keep the process going for more iterations to increase the search space within reasonable computational time. For the last generations, the quality of the solutions in the populations does not differ strongly from the overall best solution.

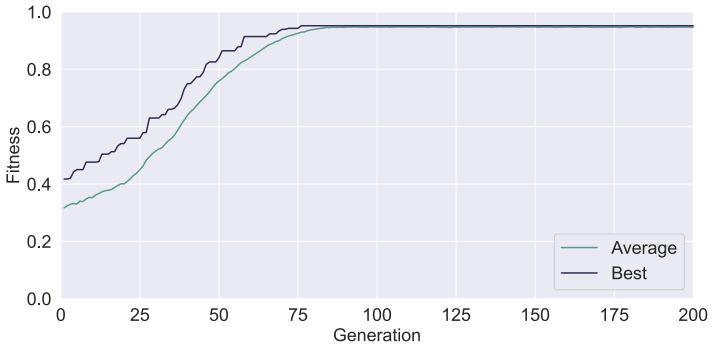

Figure 5.4: Convergence plot of a solution created by F1-ECAC for the Iris dataset.

[1]The partition we used to create this plot got a maximum fitness of 0.95 and ARI of 0.96.

Table 5.2: Mean ARI per dataset obtained by F1-ECAC with its full ensemble objective function versus using one classifier to compute fitness. Values in bold represent the highest mean ARI per dataset (solutions of higher quality).

Dataset	SVM	k-NN	Decision Tree	F1-ECAC
aggregation	**0.4484**	0.1007	0.1034	0.2685
breast-cancer-wisconsin	0.0226	0.0360	0.0191	**0.0570**
breast-tissue	0.4248	0.4094	0.4857	**0.5046**
dermatology	0.0195	0.0235	0.0172	**0.0283**
ecoli	0.3704	0.2993	0.2714	**0.4506**
forest	0.0037	0.0065	0.0020	**0.0100**
glass	0.2699	0.2091	0.2392	**0.2880**
iris	0.8021	0.7399	0.8294	**0.9169**
jain	0.2936	0.0829	0.0196	**0.3513**
leaf	0.4515	0.4050	0.4197	**0.4993**
liver	**0.0939**	0.0851	0.0857	0.0880
parkinsons	0.0368	0.0447	**0.0597**	0.0419
pathbased	**0.3657**	0.0940	0.1577	0.2900
r15	0.4389	0.4951	0.3770	**0.5478**
seeds	**0.7908**	0.6135	0.5937	0.7763
segment	0.6311	0.5386	0.5521	**0.6777**
spiral	-0.0013	0.0447	**0.0575**	0.0134
transfusion	0.0505	0.0252	0.0122	**0.0542**
wine	0.6457	0.6208	0.5935	**0.7577**
zoo	0.0856	0.0770	0.0846	**0.1173**

5.2.2 F1-ECAC vs. ECAC

The mean ARI per dataset returned by F1-ECAC and ECAC can be found in Fig. 5.5, plotted in the logarithmic scale (x-axis) for ease of visualization. F1-ECAC got an improved solution quality in all datasets except for the Parkinson's one, with a negligible decrease of 0.023. The Wilcoxon test determined that the observations generated with both algorithms had a statistically significant difference. Along with this increase in performance, the runtime decrease of F1-ECAC is evident in Fig. 5.6 (also using the logarithmic scale). The CPU runtime contrast between both algorithms is positive in favor of F1-ECAC across every dataset and had a statistically significant difference. Considering mean metrics per dataset, F1-ECAC had an ARI improvement of 83% and was seven times faster than its previous version, ECAC.

5.2.3 Clustering Performance Analysis

The benchmark including algorithms from multiple families, is presented in two sub-analyses depending on the dataset type according to their source and origin.

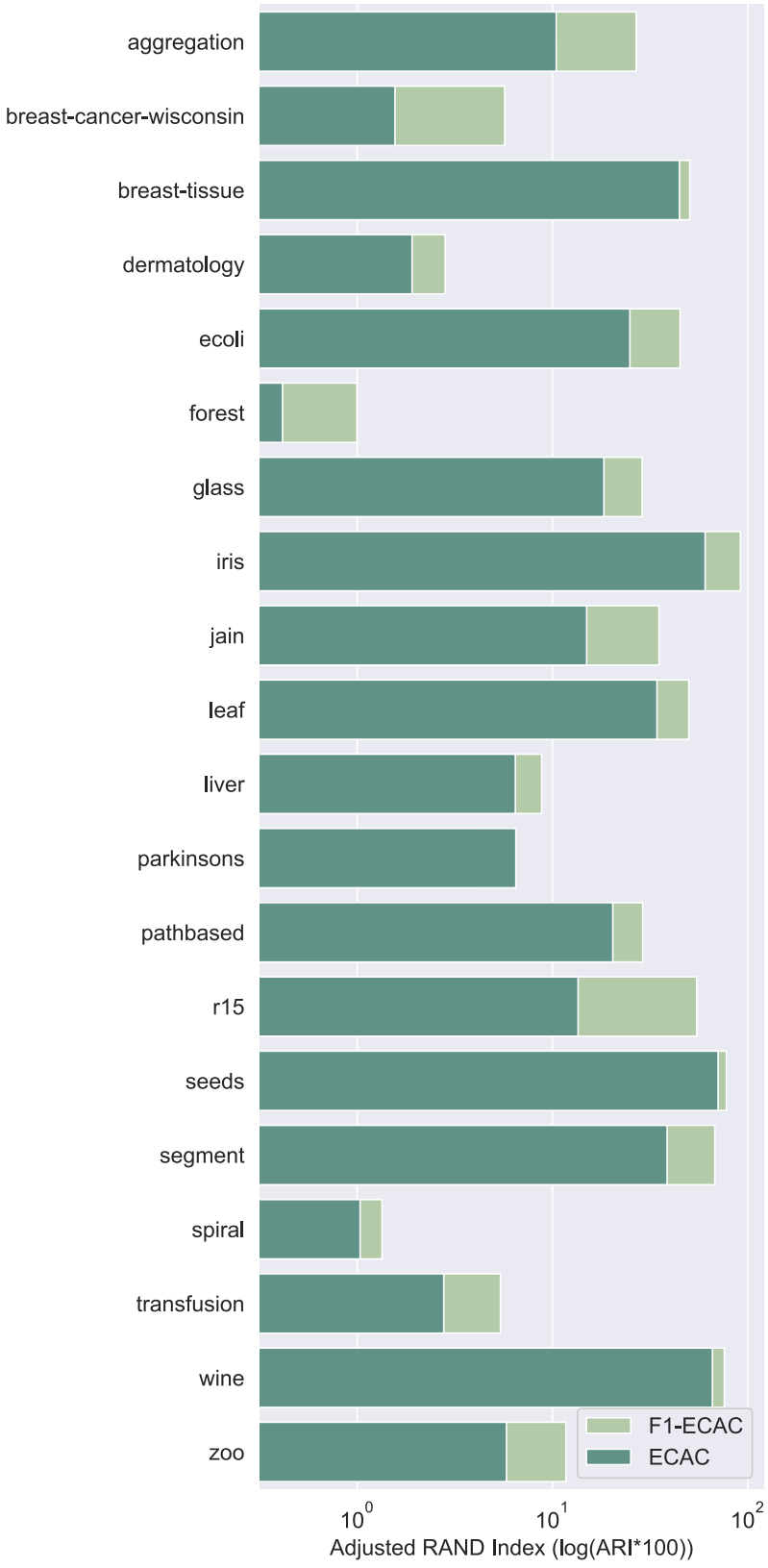

Figure 5.5: Difference in solution quality between F1-ECAC and ECAC.

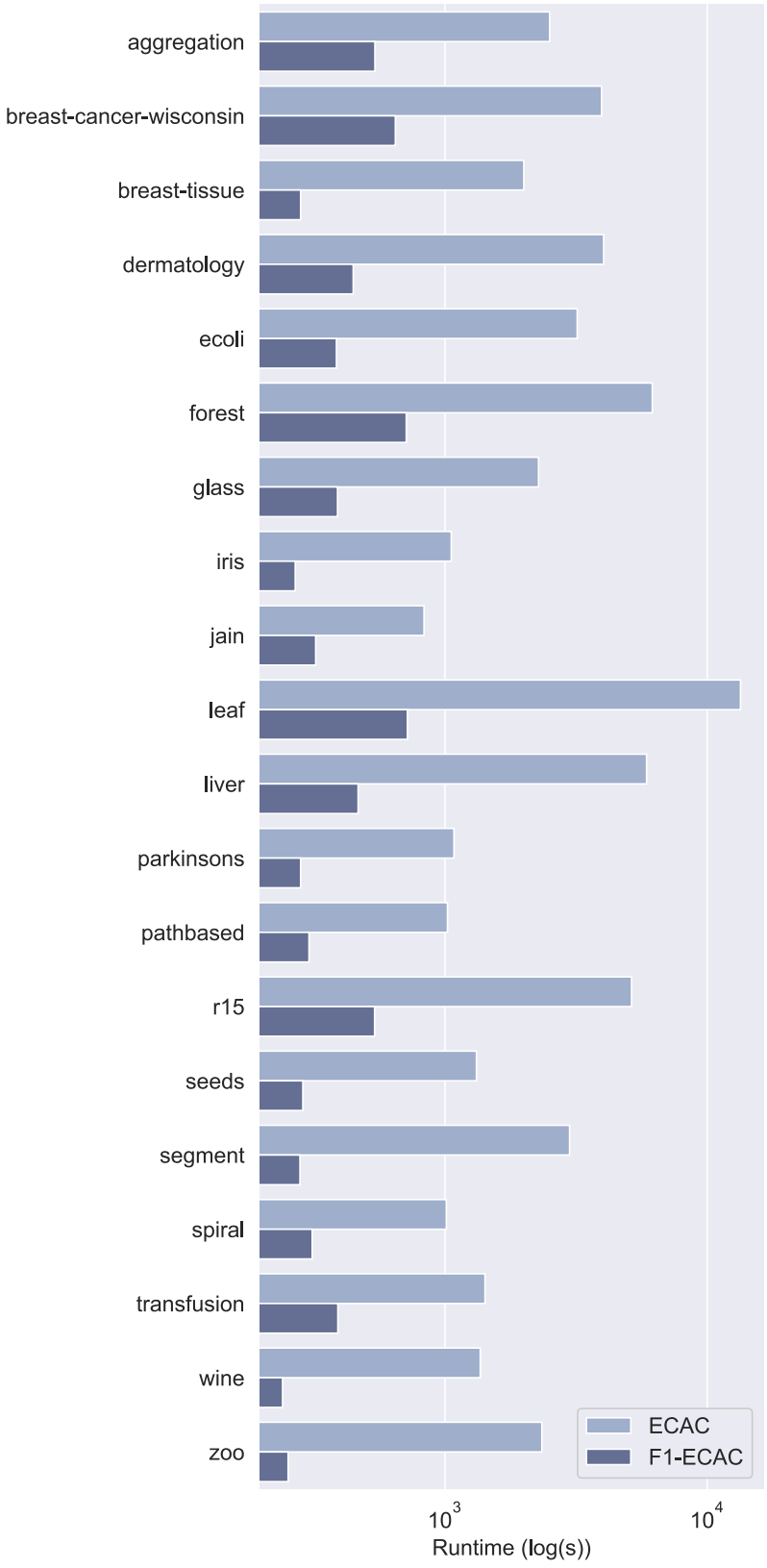

Figure 5.6: Comparison of runtime of F1-ECAC against ECAC.

Synthetic Data

Data generated artificially can be a valuable tool for evaluating the application range of a clustering algorithm, making evident their bias towards specific structures. The average ARI of the partitions created by each clustering algorithm for each synthetic dataset is presented in Table 5.3. The experiments where no solution could be retrieved were left as a blank space. Δ-MOCK was capable of partitioning synthetic data with impressive results, getting the highest score in four out of the five datasets. Moreover, k-means, MOCLE, and Δ-MOCK were tied at the top-performing set of methods for the R15 dataset. We consider synthetic data clustering a limitation of F1-ECAC. However, we target our proposals to applications using real-source datasets, and they are not designed to favor either the cluster structures in the artificial datasets used or any geometrical shape due to their unlikely presence in real-world data mining tasks. Clustering algorithms designed to favor specific 2-D structures struggle with cross-domain applicability caused by their inherent solid clustering bias despite their visualization advantages.

Table 5.3: Mean ARI of the solutions generated by the algorithms using synthetic data during the experimental phase of F1-ECAC. Values in bold represent the highest mean ARI per dataset (solutions of higher quality).

Dataset	k-means	Single-linkage	DBSCAN	HG-means	MOCLE	Δ-MOCK	F1-ECAC
aggregation	0.7272	0.8058	0.7338		0.9991	**1**	0.2686
jain	0.5528	0.0099	0.9316	0	0.4893	**1**	0.3513
pathbased	0.4797	0.0009	0.2255		0.4851	**0.7271**	0.2900
r15	**0.9928**	0.5425	0.2637	0.0011	**0.9928**	**0.9928**	0.5478
spiral	-0.0058	**1**	0.4573		0.0438	0.6203	0.0134

Real-source Data

The main focus of the analysis of the clustering performance relies on the results raised from processing data from authentic sources (giving us a confident quantification of their solutions' quality when using data behaving naturally), which will be displayed in this section. Table 5.4 summarizes the mean ARI of the partitions created by each algorithm, and Fig. 5.7 displays the number of datasets in which each algorithm got the highest score at least once. F1-ECAC obtained a higher mean ARI than the rest in 7 out of 15 datasets, followed by MOCLE with 4, k-means with 3, and Δ-MOCK with 1. We attribute the absence of Single-linkage Agglomerative Clustering in this list to its strong bias towards connected clusters. DBSCAN failed to return acceptable clusterings in most cases to its sensitivity towards its hyper-parameter setting, which is firmly attached to domain knowledge. HG-means was the only evolutionary algorithm that did not outperform the other algorithms in any dataset. F1-ECAC shows stable solutions across the datasets, which is a positive outcome considering their variation in dimensions and cluster structure shape, leading us to the insight of our algorithm's adaptive capabilities towards the intrinsic nature of the data.

The p-value from the Friedman test equalled 0 for all experiments, and Table 5.5 presents the unadjusted and adjusted p-values of the Friedman test with the Holm post-hoc. The $1xN$

Table 5.4: Solution quality (mean ARI) of each algorithm using the UCI ML Repository datasets in the experimental phase of F1-ECAC. Values in bold represent the highest average ARI per dataset.

Dataset	k-means	SL	DBSCAN	HGM	MOCLE	Δ-MOCK	F1-ECAC
breast-cancer-wisconsin	**0.6707**	0.0048	0	0.4914	0.5415	0.0048	0.0570
breast-tissue	0.2781	0.0007	0	0.0053	0.1824	0.0419	**0.5046**
dermatology	**0.7124**	0.2759	0	0.0034	0.1717	0.0060	0.0283
ecoli	0.5023	0.0399	0	0.4262	**0.7446**	0.0283	0.4506
forest	0.4325	0.0017	0	0.4987	**0.5392**	0.0076	0.0100
glass	0.1715	0.0143	-0.0076	0.2598	0.2762	0.2709	**0.2880**
iris	0.6201	0.5584	0.0641	0.7302	0.8008	0.0043	**0.9169**
leaf	0.3565	0.0221	0	0.2950	0.2880	0.3182	**0.4993**
liver	0.0148	0.0083	0	0.0305	0.0640	-0.0039	**0.0880**
parkinsons	-0.0978	-0.0134	0	0	**0.1674**	0.0813	0.0419
seeds	0.7733	0	0	0.7166	0.7416	0.7132	**0.7763**
segment	0.4687	0.0010	0	0.4063	0.4524	0.0012	**0.6777**
transfusion	0.0527	-0.0036	0.0002	0.0795	**0.0836**	0.0086	0.0542
wine	**0.8975**	-0.0068	0	0.3711	0.3652	0.4025	0.7577
zoo	0.7581	0.4425	-0.0578	0.7088	0.7561	**0.7670**	0.1173

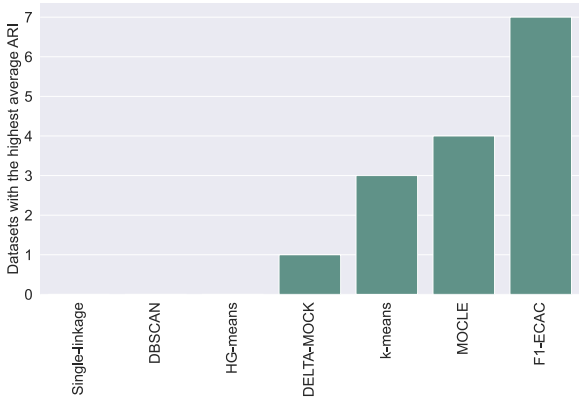

Figure 5.7: Number of datasets in which each algorithm outperformed the rest using the real data in the experimental framework of F1-ECAC.

comparisons are held by comparing each method against F1-ECAC, which was the highest-ranked according to Table 5.6. Getting this version of our algorithm in the top rank proves F1-ECAC's competitive performance even in this challenging benchmark. The difference between the results of F1-ECAC versus DBSCAN, Single-linkage, and Δ-MOCK is statistically significant in favor of F1-ECAC. Therefore, k-means, HG-means, and MOCLE do not have a significant difference with F1-ECAC.

We performed a correlation analysis to visualize the statistical relationships between the

Table 5.5: Unadjusted and adjusted p-values returned by the Friedman test using the Holm post-hoc in the experimental evaluation of F1-ECAC.

Algorithm	Unadjusted P	P_{Holm}
DBSCAN	0	0
Single-linkage	0.0001	0.0004
Δ-MOCK	0.0088	0.0352
HG-means	0.0759	0.2278
k-means	0.6121	1.2242
MOCLE	0.9327	1.2242

Table 5.6: Average ranks of each treatment of the experimental evaluation of F1-ECAC (values close to 1 suggest better performance). Value in bold represents the best average ranking (overall higher ARI values).

Algorithm	Ranking
F1-ECAC	**2.4**
MOCLE	2.4667
k-means	2.8
HG-means	3.8
Δ-MOCK	4.4667
Single-linkage	5.5333
DBSCAN	6.5333

structural relations of the data, k values, and the solution quality of each algorithm using natural data. The Spearman correlation between 10 variables is shown in Fig. 5.8; we consider a threshold of an absolute value of 0.50 to determine high correlations. The absence of correlation between the quality of the partitions created with F1-ECAC and other methods implies that the objective function of our algorithm discerns cluster quality in a contrasting manner to conventional approaches. Moreover, F1-ECAC also does not correlate with k, retaining the stability properties of its previous version.

Fig. 5.9 displays the ground truth and a partition created by the algorithms that scored the highest mean ARI at least once (see Fig. 5.7). For this plot, we considered synthetic and real data and included the best-performing solution of each selected algorithm (see Tables 5.3 and 5.4). We decided to include the Ecoli instead of the Zoo dataset in Subfigure *(e)* to visualize better the partition clustered by MOCLE. F1-ECAC clustered the Iris dataset almost perfectly, only locating two of the 150 flowers in the dataset in the wrong group, achieving a maximum ARI of 0.96 (which more complex methods were not capable of doing). This tough clustering problem is often tackled to see if an algorithm can identify the difficult differentiation between the overlapping groups (see Subfigure *(g)* from Fig. 5.9).

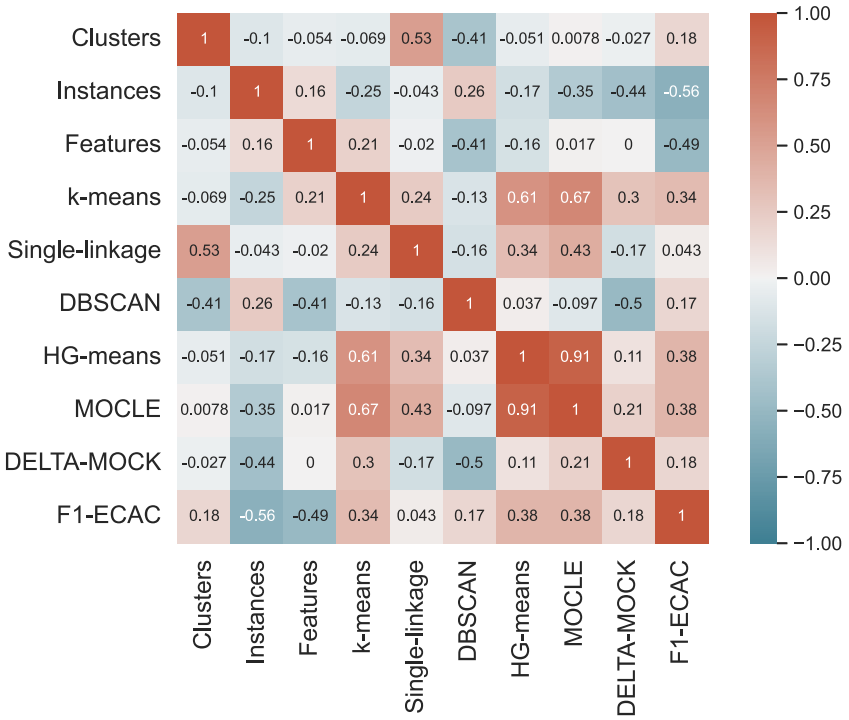

Figure 5.8: Correlation plot of the structural details of the datasets and the ARI of the partitions returned by the algorithms using the UCI Repository datasets in the development of F1-ECAC.

5.2.4 Remarks

Achieving similar performance to leading clustering algorithms is a relevant achievement of this project. Besides, we avoided some of the theoretical issues found in other algorithms. For instance, the disadvantages of distance-based dissimilarity metrics optimized by traditional and single-objective evolutionary clustering algorithms (which imposes a cluster structure into the data, thus inducing clustering bias) are overcome by the ensemble of classifiers in F1-ECAC's objective function; computing class separation according to the kernel in each classifier. This is evident in Subfigures *(b)* and *(d)* from Fig. 5.9, where the bias towards compactness and connectedness is noticeable in the partitions created by k-means and Single-linkage for the R15 and Spiral datasets. Hence the application range of F1-ECAC is not limited to data complying with predefined structures and offers diverse solutions that are not tied to them. The multi-objective evolutionary approaches generated clusters of arbitrary hyper-shapes by optimizing complementary objectives as shown in Subfigures *(f), (h),* and *(j)* from Fig. 5.9. However, the solutions returned by k-means, HG-means, and MOCLE got a high correlation between them as seen in Fig. 5.8 due to the inclusion of the former in the latter two, suggesting that the solutions of evolutionary clustering algorithms might still be biased towards certain structures if they optimize clustering criteria based on distance functions. Another advantage of F1-ECAC is the high degree of interpretability offered by 1) the label-based representation used in F1-ECAC, contrary to the representation schemes used by evolutionary algorithms such as Δ-MOCK (see Chapter 2.3.2) and 2) the intelligible

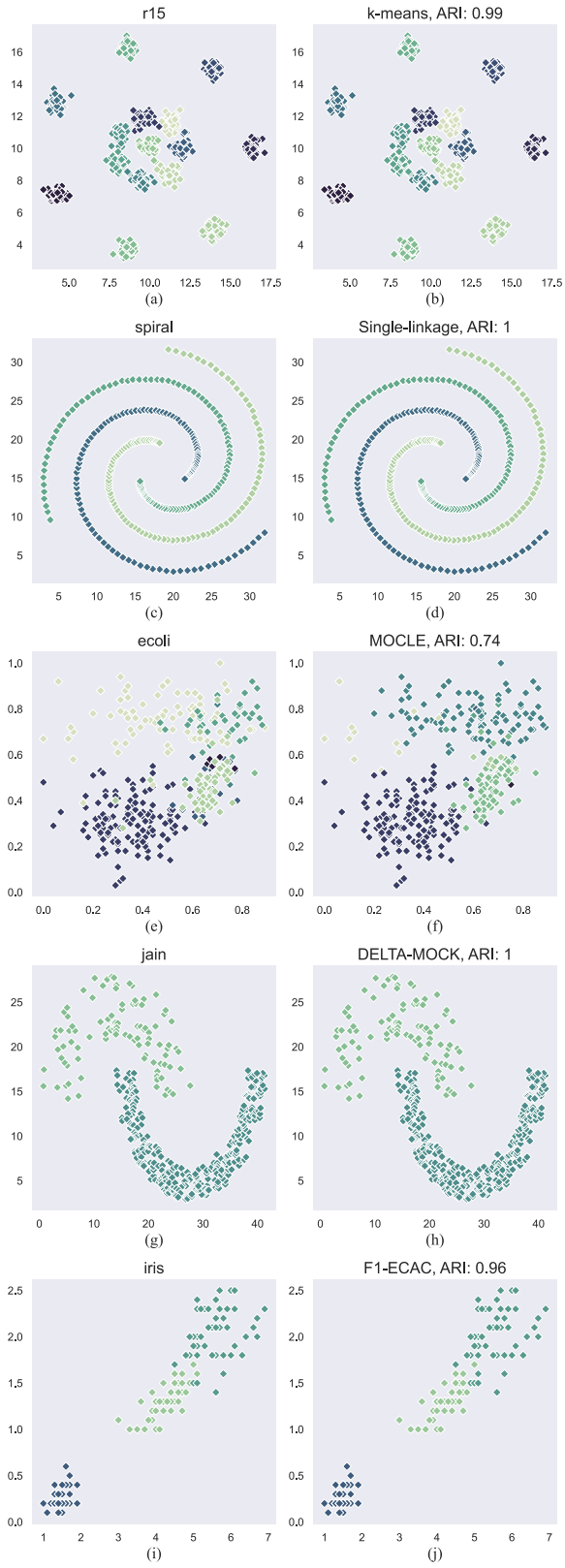

Figure 5.9: Reference labels and best partition created by the algorithms that scored the highest ARI at least once while developing F1-ECAC.

single-objective evolutionary process used to generate the search space, which can be stopped and analyzed at any point and avoids selecting solutions from a Pareto front.

F1-ECAC (as every version in the ECAC family) has boundaries limited to the generation of a partition from an unlabeled dataset; hence fitness values close to one are not relevant to this application even though it might suggest an overfitting issue and induced classifier variance. It is not within the scope of this project to train classifier models ready to be deployed as we are tackling a clustering problem. Regarding the negative correlation between F1-ECAC's performance and the number of instances in Fig. 5.8, it is important to remark that F1-ECAC is not intended to be used in data mining tasks involving big data, because just as the other methods mentioned in Chapter 2, the dissimilarity computation to be optimized becomes unfeasible when dealing when data of that volume. We conclude that the design decisions leading to F1-ECAC brought us to the fulfillment of our goal of improving our first algorithm's (ECAC) competitiveness in a complex cross-domain clustering benchmark by modifying its optimization pipeline and objective function.

5.3 ECAC-S Experimental Results

This sections present the results from each modification done to F1-ECAC mentioned in Chapter 3.3 according to the experimental framework from Chapter 4.3. The components from F1-ECAC are maintained the same unless explicitly specified that any part of the algorithm is modified to test its possible impact on reducing runtime.

5.3.1 Data Manipulation

We held a comparison against the published version of F1-ECAC (detailed in Chapter 3.2) and its *parallelized* version, which includes the changes proposed in Chapter 3.3.1 to the algorithm's data manipulation pipeline. In this 1-vs.-1 benchmark, both versions had a statistically significant difference in the mean ARI and runtime per dataset metrics (ARI difference in favor of F1-ECAC, runtime difference in favor of the *parallelized* version). The *parallelized* version got a lower average runtime in 19 of the 30 datasets, with a median of 69.33 seconds per dataset, whereas it was 159.59 seconds for F1-ECAC. Despite the significant difference in solution quality, the mean ARI per dataset only decreased by a negligible drop of 0.0178.

5.3.2 Classifier Combinations

The following iterations of the algorithm contain the properties of the *parallelized* version, which proved its benefits in Chapter 5.3.1. In this section, we present the results of comparing the six classifier and metric combinations mentioned in Chapter 3.3.2 with the *parallelized* version (which uses the mean F_1 score of the three classifiers from the original F1-ECAC, as seen in Chapter 3.2.1). We can visualize the results of this benchmark in Fig. 5.10 and Fig. 5.11, where we show the average ranks and significant differences for the ARI and runtime metrics, respectively. The *parallelized* version had a mean runtime per dataset of 402.88 seconds and was the worst-ranked, proving that using two classifiers instead of three will lead

to an inherent efficiency boost. In contrast, the *Support Vector Machine/Decision Tree (Mean)* version took 70.94 seconds on average to return a solution across all datasets and was the top-ranked. Despite such improvement in runtime, there is no statistically significant difference in ARI between the top-6 ranking algorithms. Thus we decide to continue with further improvements using a classifier ensemble in the objective function of our algorithm constituted by the Support Vector Machine and Decision Tree classifiers, using the mean F_1 score value to compute fitness due to its arguable superiority in runtime without compromising solution quality.

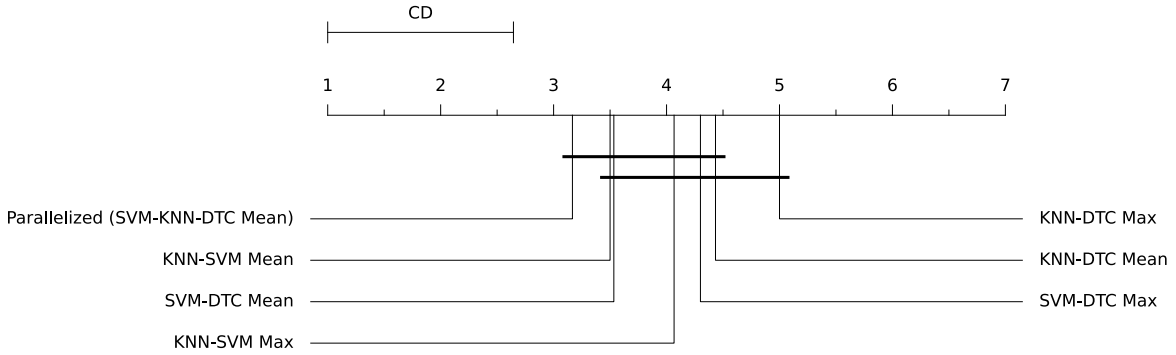

Figure 5.10: Adjusted Rand Index Critical Difference (CD) diagram for the classifier-metric combinations benchmark in the design process of ECAC-S.

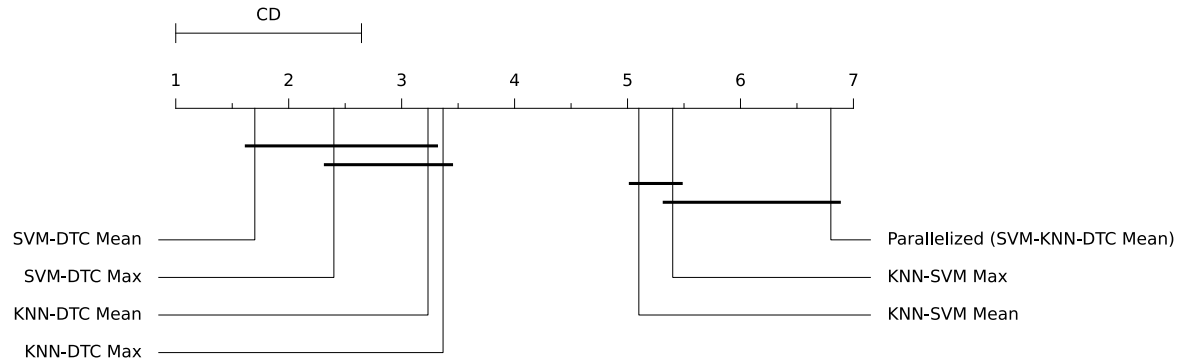

Figure 5.11: Runtime in seconds Critical Difference (CD) diagram for the classifier-metric combinations benchmark in the design process of ECAC-S.

5.3.3 Genetic Operator Combinations

We tested every proposed change to the genetic procedures of F1-ECAC (see Chapter 3.3.3) along with the *Support Vector Machine/Decision Tree (Mean)* version due to its efficiency advantages against the rest as proven in Chapter 5.3.2. The *Support Vector Machine/Decision*

Tree (Mean) version includes the genetic operators from the original F1-ECAC (i.e. 1-point Crossover and 5% Mutation; see Chapter 3.2) and is included as a reference for comparison. Fig. 5.12 and Fig. 5.13 present the resulting Critical Difference (CD) diagrams for the ARI and runtime metrics. The *Individual Gene Probability* Crossover was the worst-performing operator in both metrics. It was ranked last in every combination, and had a significant difference from the four best-performing operator combinations. The *2-point Crossover/5% Mutation* version was the top-ranked in mean ARI and runtime. We decide to proceed using this version as it returned an average runtime decrease of 5.56 seconds per dataset without presenting a significant difference in solution quality to its previous version (i.e. *Support Vector Machine/Decision Tree (Mean)*).

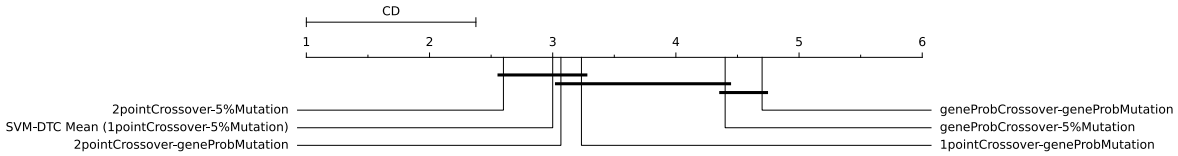

Figure 5.12: Adjusted Rand Index CD diagram for the genetic operator combinations benchmark in the design process of ECAC-S.

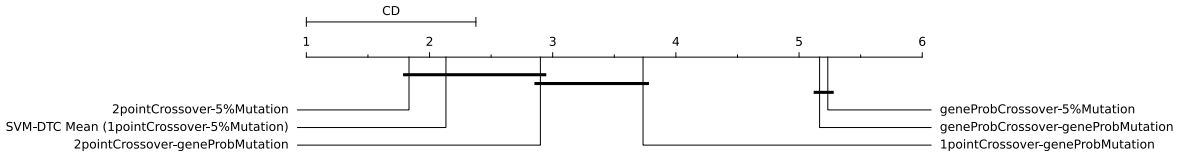

Figure 5.13: Runtime in seconds CD diagram for the genetic operator combinations benchmark in the design process of ECAC-S.

5.3.4 Hyper-parameter Adjustment

We developed a hyper-parameter adjustment benchmark using the *2-point Crossover/5% Mutation* version resultant from the analysis in Chapter 5.3.3. The updated hyper-parameters proposed in Chapter 3.3.4 were implemented and tested against the *2-point Crossover/5% Mutation* version, which uses a test size of 75% in proportion as done in the original F1-ECAC as seen in Chapter 3.2. The CD diagrams in Fig. 5.14 and Fig. 5.15 show the CD diagrams corresponding to the ARI and runtime metrics. We decide to select the *2-point Crossover/5% Mutation* version to proceed with future iterations because the 75% test size hyper-parameter returned significant runtime improvements and obtained the top-rank in such metric. Additionally, this runtime reduction does not present a significant difference in solution quality with the *50% test size* version, which was the best-ranked in this metric.

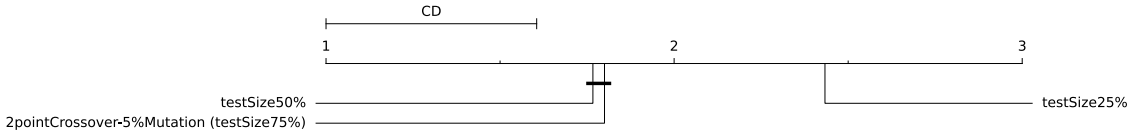

Figure 5.14: Adjusted Rand Index CD diagram for the hyper-parameter adjustment benchmark in the design process of ECAC-S.

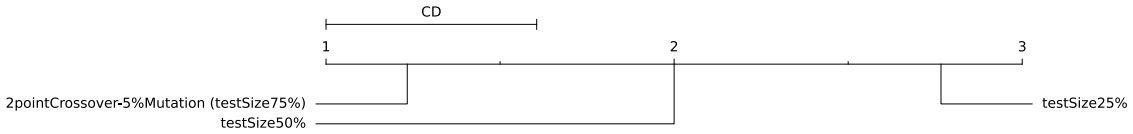

Figure 5.15: Runtime in seconds CD diagram for the hyper-parameter adjustment benchmark in the design process of ECAC-S.

5.3.5 Convergence Detection

We implemented the condition for detecting convergence proposed in Chapter 3.3.5 with the test size hyper-parameter of 75% (*2-point Crossover/5% Mutation* version), according to the results from the experiments in Chapter 5.3.4. We can find the mean ARI and runtime per dataset ranks and critical differences from testing multiple early-stopping hyper-parameters in Fig. 5.16 and Fig. 5.17. We include the *75% test size* version for reference as it does not include an early-stopping condition (as done by F1-ECAC). An early-stopping hyper-parameter that terminates the optimization process after 10% of the generations with no change in the best solution resulted in an evident runtime advantage. However, the significant difference in solution quality of the *Early-stopping 10%* and the *75% test size* versions does not let us choose this version as the most suitable. Therefore, we selected the *Early-stopping 20%* version as the best option because it offers an acceptable balance between quality and runtime, being the top-ranked in ARI while still having a significant difference in CPU time compared to the version with no early-stopping (i.e. *75% test size* version).

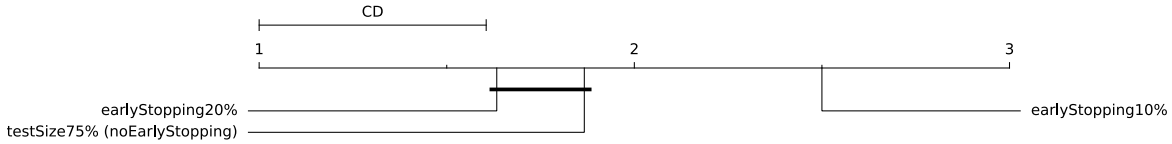

Figure 5.16: Adjusted Rand Index CD diagram for the convergence detection benchmark in the design process of ECAC-S.

Now that every modification to F1-ECAC has been sequentially implemented and tested, we claim the *Early-stopping 20%* version of our algorithm to be the youngest member in the ECAC family, and we refer to it as ECAC-S.

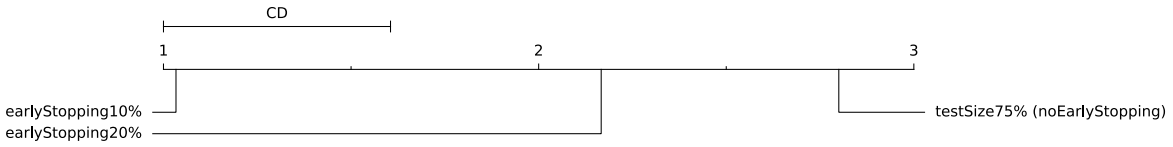

Figure 5.17: Runtime in seconds CD diagram for the convergence detection benchmark in the design process of ECAC-S.

5.3.6 ECAC-S vs. F1-ECAC

To summarize the consequences in solution quality and runtime of the incremental updates leading to establishing the final version of ECAC-S, we present the mean ARI and time in seconds per dataset of F1-ECAC, ECAC-S, and their intermediate versions in Tables 5.7 and 5.8. We can visualize the results in the relational plot found in Fig. 5.18. This figure plots the mean ARI in the x-axis and the mean runtime in seconds in the y-axis, using the logarithmic scale. Each dot in the plot depicts the relation between solution quality and runtime for a specific dataset using some version of the algorithm starting with F1-ECAC and ending up with ECAC-S, which defines the color code of the plot (i.e., F1-ECAC, *parallelized* version, *SVM-DTC (Mean)* version, *2-point Crossover/5% Mutation* version, and ECAC-S). The size of each point represents the standard deviation in ARI of the solutions it represents. The x-axis is reversed; therefore, a good result for an algorithm would be a set of small dots located close to the lower-left corner of the figure. Following this logic, ECAC-S offers noticeable advantages against the rest of the version as it generated partitions with lower runtime and higher quality across most datasets.

The CD diagrams in Fig. 5.19 and Fig. 5.20 show the average rank and significant differences between F1-ECAC, ECAC-S, and in-between versions. The minimum mean time per dataset obtained by F1-ECAC was 53,64 seconds, whereas it was 14.15 seconds for ECAC-S. On the other hand, the maximum average runtime per dataset was 2,507.07 seconds for F1-ECAC and 370.61 seconds for ECAC-S. The average runtime considering the mean of all datasets of ECAC-S was 252.74 lower than the results obtained by F1-ECAC. The top-rank of ECAC-S in Fig. 5.20 supports this version's efficiency benefits compared to the rest. Additional to the faster pipeline followed by ECAC-S, there is no significant difference in performance, as proven by Fig. 5.19. Moreover, the maximum average ARI even increased from 0.84 (F1-ECAC) to 0.88 (ECAC-S).

To summarize, we performed substantial enhancements to F1-ECAC in developing ECAC-S in its data manipulation pipeline, objective function, genetic operators, convergence detection, and hyper-parameters. ECAC-S is 4.22 faster on average than F1-ECAC and has no significant difference in performance. ECAC-S offers an important reduction in runtime due to its modified implementation and design compared to F1-ECAC, and is capable of processing datasets or greater size without compromising solution quality, which we will assess in the forthcoming section.

Table 5.7: Solution quality (mean ARI) per dataset of every intermediate version between F1-ECAC and ECAC-S. Values in bold represent the highest average ARI per dataset.

Dataset	F1-ECAC	Parallelized	SVM-DTC Mean	2pointCr/5%Mut	ECAC-S
absenteeism-at-work	0.0099	0.0150	0.0165	**0.0172**	0.0151
arrhythmia	0.0039	0.0047	0.0034	**0.0055**	0.0015
breast-cancer-wisconsin	**0.0524**	0.0337	0.0363	0.0319	0.0413
breast-tissue	**0.4742**	0.4094	0.3595	0.3533	0.3576
car-evaluation	**0.0068**	**0.0068**	0.0061	0.0057	0.0066
dermatology	0.0255	**0.2733**	0.2439	0.2661	0.2297
echocardiogram	**0.2894**	0.2650	0.1203	0.0663	0.1262
ecoli	**0.4196**	0.3644	0.3760	0.3754	0.3623
forest	**0.0098**	0.0081	0.0066	0.0082	0.0088
forest-fires	0.0407	0.0361	0.0371	**0.0451**	0.0383
german-credit	**-0.0002**	-0.0015	-0.0010	-0.0020	-0.0003
glass	**0.2894**	0.2644	0.2726	0.2319	0.2285
hepatitis	0.0690	**0.0991**	0.0924	0.0902	0.0857
image-segmentation	**0.1541**	0.1141	0.1030	0.1214	0.1300
ionosphere	-0.0007	0.0095	**0.0175**	0.0114	0.0092
iris	0.8293	0.8378	0.7997	0.8474	**0.8808**
leaf	**0.4848**	0.2511	0.2503	0.2209	0.2528
liver	**0.0962**	0.0619	0.0708	0.0621	0.0484
parkinsons	**0.0654**	0.0348	0.0315	0.0202	0.0307
seeds	0.7727	0.6977	0.7512	**0.7769**	0.7209
segment	**0.6715**	0.4489	0.4060	0.4479	0.4894
sonar	0.0240	0.1112	0.0850	0.1731	**0.1955**
soybean-large	**0.3956**	0.2214	0.2416	0.2438	0.2541
student-performance	**0.0080**	0.0078	0.0066	0.0072	0.0078
tic-tac-toe	0.0039	0.0068	**0.0099**	0.0069	0.0060
transfusion	**0.0683**	0.0325	0.0358	0.0458	0.0404
user-knowledge-modeling	-0.0017	-0.0008	**-0.0001**	-0.0027	-0.0023
wine	**0.8402**	0.7596	0.5667	0.7036	0.7207
yeast	0.0102	0.0089	0.0086	0.0098	**0.0109**
zoo	0.0766	0.2745	0.2786	**0.2967**	0.2829

Table 5.8: Runtime (mean time in seconds) per dataset of every intermediate version between F1-ECAC and ECAC-S. Values in bold represent the lowest average runtime per dataset.

Dataset	F1-ECAC	Parallelized	SVM-DTC Mean	2pointCr/5%Mut	ECAC-S
absenteeism-at-work	516.4233	843.0994	**84.7060**	87.2755	85.9511
arrhythmia	857.7323	2378.5323	176.5866	175.0729	**84.7259**
breast-cancer-wisconsin	309.4416	758.0020	56.0044	56.4008	**45.6905**
breast-tissue	62.1918	33.2178	21.4870	21.5318	**17.8256**
car-evaluation	605.8827	151.7303	112.9758	111.1874	**111.0698**
dermatology	194.2949	701.5605	37.6859	37.6814	**37.6062**
echocardiogram	53.6448	30.1839	19.2559	19.3214	**14.1472**
ecoli	112.3737	43.4201	29.7016	28.8953	**28.7623**
forest	330.3182	769.5698	60.7830	**60.2331**	60.6081
forest-fires	232.9348	69.7273	49.9447	49.8162	**49.8075**
german-credit	434.9185	850.0128	63.0129	**61.2365**	61.8491
glass	83.5681	38.3212	23.9405	24.0246	**21.5541**
hepatitis	61.0789	32.9577	21.5929	21.7754	**18.2448**
image-segmentation	2507.0743	1528.9290	374.5877	**368.8092**	370.6107
ionosphere	174.9465	701.1325	36.4350	37.3823	**26.7860**
iris	59.4812	30.2069	19.4150	18.5467	**15.0757**
leaf	**216.3903**	292.2674	394.4758	238.0254	232.1517
liver	144.2059	50.2775	34.7003	35.3387	**28.0793**
parkinsons	72.2985	35.0240	24.1872	24.2276	**17.2885**
seeds	68.9605	33.3604	21.5747	21.5152	**19.1141**
segment	89.6397	38.9343	27.0316	26.9451	**25.9677**
sonar	125.6720	678.8756	31.0691	31.3806	**28.6788**
soybean-large	194.2130	721.2563	41.6588	42.6496	**39.8902**
student-performance	452.0130	863.5029	**75.9638**	76.2097	76.3470
tic-tac-toe	235.6782	68.9275	42.7874	41.6843	**40.8273**
transfusion	139.9635	49.4009	32.3401	31.7268	**31.5593**
user-knowledge-modeling	112.3340	42.0112	28.1148	27.9042	**27.2173**
wine	69.2086	33.8220	22.0842	21.8553	**19.7849**
yeast	802.5682	184.8799	140.6381	**140.5524**	140.8310
zoo	59.2900	33.1688	23.5196	22.2031	**18.5349**

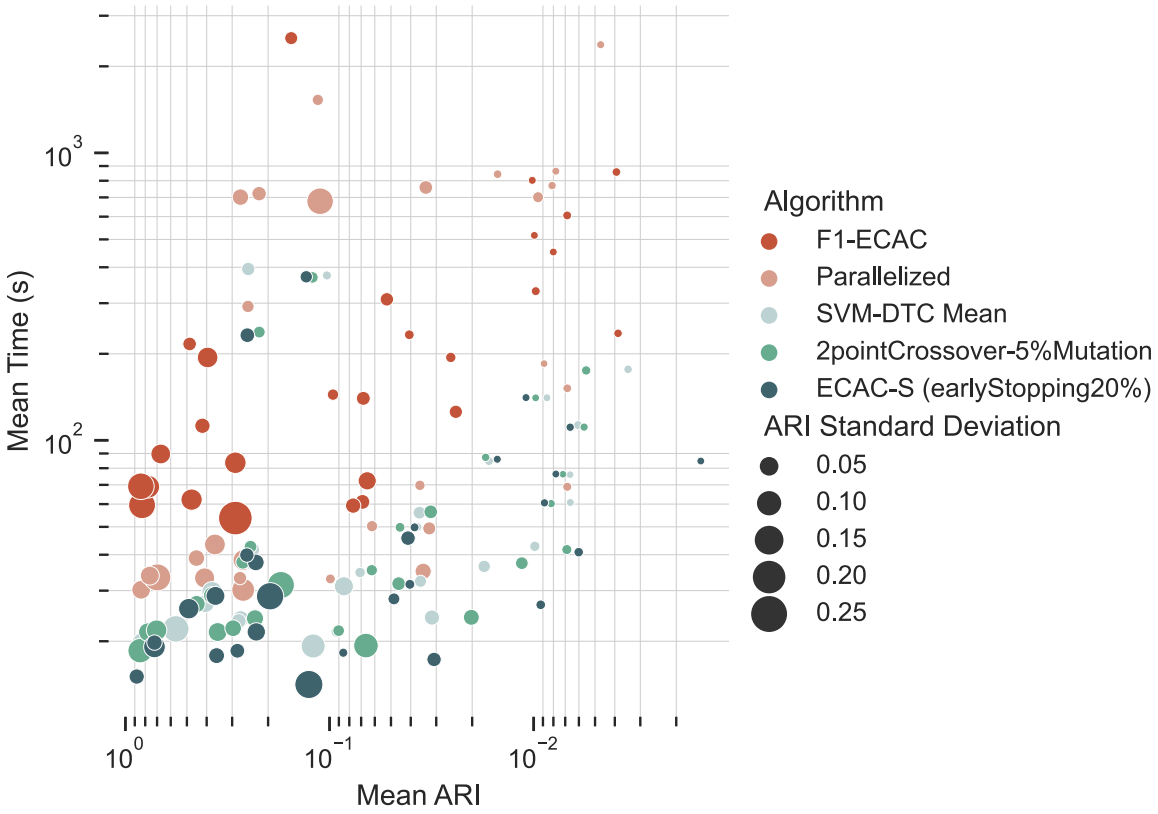

Figure 5.18: Mean ARI, time in seconds, and ARI standard deviation per dataset of the solutions returned by F1-ECAC, ECAC-S, and their intermediate versions.

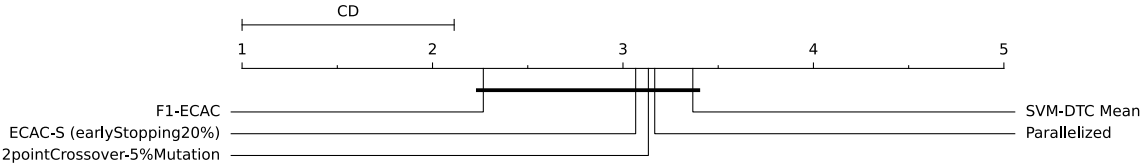

Figure 5.19: Adjusted Rand Index CD diagram for the version evolution benchmark including F1-ECAC, ECAC-S, and in-between versions.

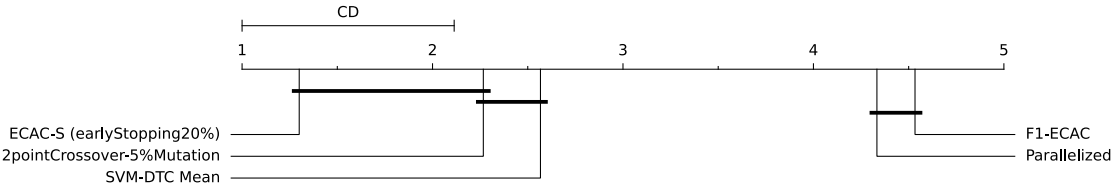

Figure 5.20: Runtime in seconds CD diagram for the version evolution benchmark including F1-ECAC, ECAC-S, and in-between versions.

5.3.7 Application on Satellite Image Segmentation

Table 5.9 presents the average ARI resulting from processing each image according to the experimental framework presented in Chapter 4.3.4. Fig. 5.21 displays the number of images in which each algorithm achieved the best performance according to Table 5.9 (we only include those methods which surpassed the rest at least once). ECAC-S and Spectral-clustering obtained solutions of average higher quality than the other algorithms in four out of the ten images, followed by k-means and Birch, with one each.

Table 5.9: Mean ARI of the solutions returned by k-means, DBSCAN, Spectral-clustering, Birch, and ECAC-S in the image segmentation benchmark. Values in bold represent the highest average ARI per image.

Image	k-means	DBSCAN	Spectral	Birch	ECAC-S
canada	0.6904	0.0005	**0.6997**	0.6675	0.4379
coast	**0.2132**	-0.0009	0.1350	0.1091	0.2043
highway-in-the-desert	0.0739	-0.0025	0.0441	0.0356	**0.3975**
london	0.5813	0.0386	**0.6752**	0.1509	0.3207
parking-lot	0.6097	0.0362	**0.7350**	-0.0181	0.1428
port	0.4226	0.0073	0.4170	0.4040	**0.5298**
port-city	0.6253	0.0000	0.6313	**0.6406**	0.1896
road-with-trees	0.4511	-0.0004	0.4505	0.5328	**0.6042**
varadero	0.4790	0.0002	0.5722	0.3034	**0.5889**
white-containers	0.6731	0.0004	**0.6753**	0.6540	0.2292

We can also remark positive results in the robustness comparison. The results considering every solution generated with the five algorithms are illustrated in the boxplots in Fig. 5.22. We observe that the median of the solutions of ECAC-S and Birch are comparable. In contrast, Spectral-clustering and k-means outperformed the former two in this metric. The interquartile range of the solutions by ECAC-S falls in an acceptable margin and is even smaller than the one generated by the partitions of the Birch algorithm. The mean standard deviation ARI per image of ECAC-S was 0.1723, whereas it was 0.2684 for Spectral-clustering despite being the highest-ranked treatment. Such standard deviation decrease suggests that our algorithm offers minor variation in results when using data of varying nature. Moreover, k-means returned a minimum mean ARI per image of 0.0739, whereas ECAC-S had a value of 0.1428. Hence we can state that ECAC-S could adapt to the natural phenomena represented in the data in greater capacity because even its lowest-performing solutions start from a low-boundary mean ARI that is noticeably better than the values of the rest of the algorithms. Achieving high performance in a real segmentation task stands for the vast application range of our ECAC series of contributions, thanks to its optimization process and novel cluster quality criterion. In Fig. 5.23, we introduce the CD diagram with the rankings and significant differences resulting from the average ARI per image obtained by the algorithms. Here, we present one of the most outstanding insights from this project. Our ultimate proposal, ECAC-S, does not present a significant difference in quality with the solutions created by k-means, Spectral-clustering, and Birch.

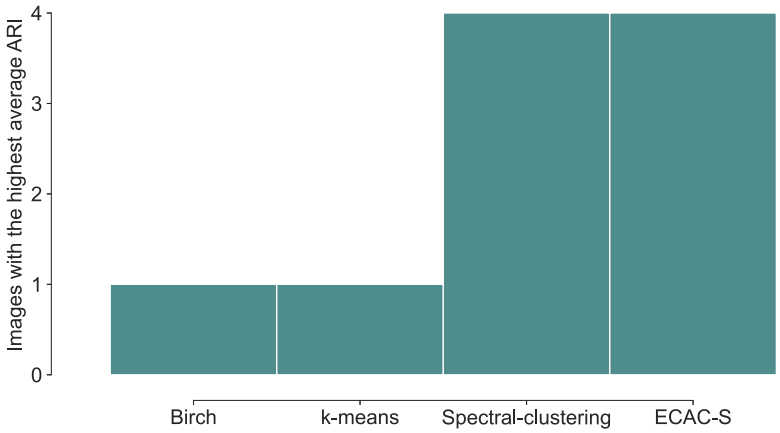

Figure 5.21: Number of images where each algorithm outperformed the rest in the mean ARI metric, including only those who achieved it at least once.

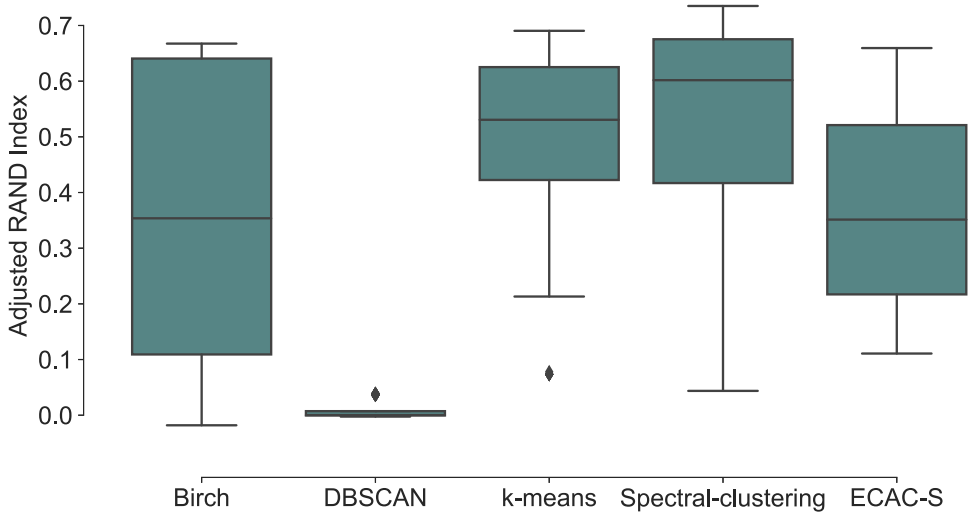

Figure 5.22: ARI of the partitions created by the algorithms from the satellite image segmentation benchmark visualized as a boxplot.

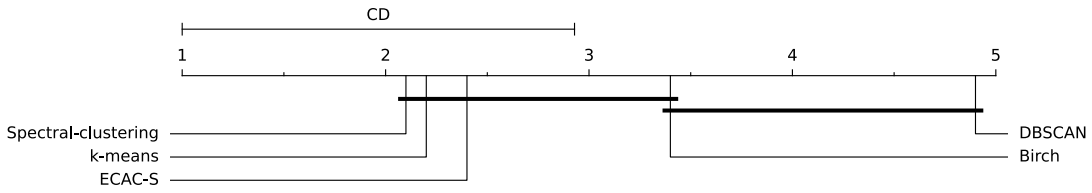

Figure 5.23: CD diagram of the mean ARI obtained by each algorithm in the image segmentation task.

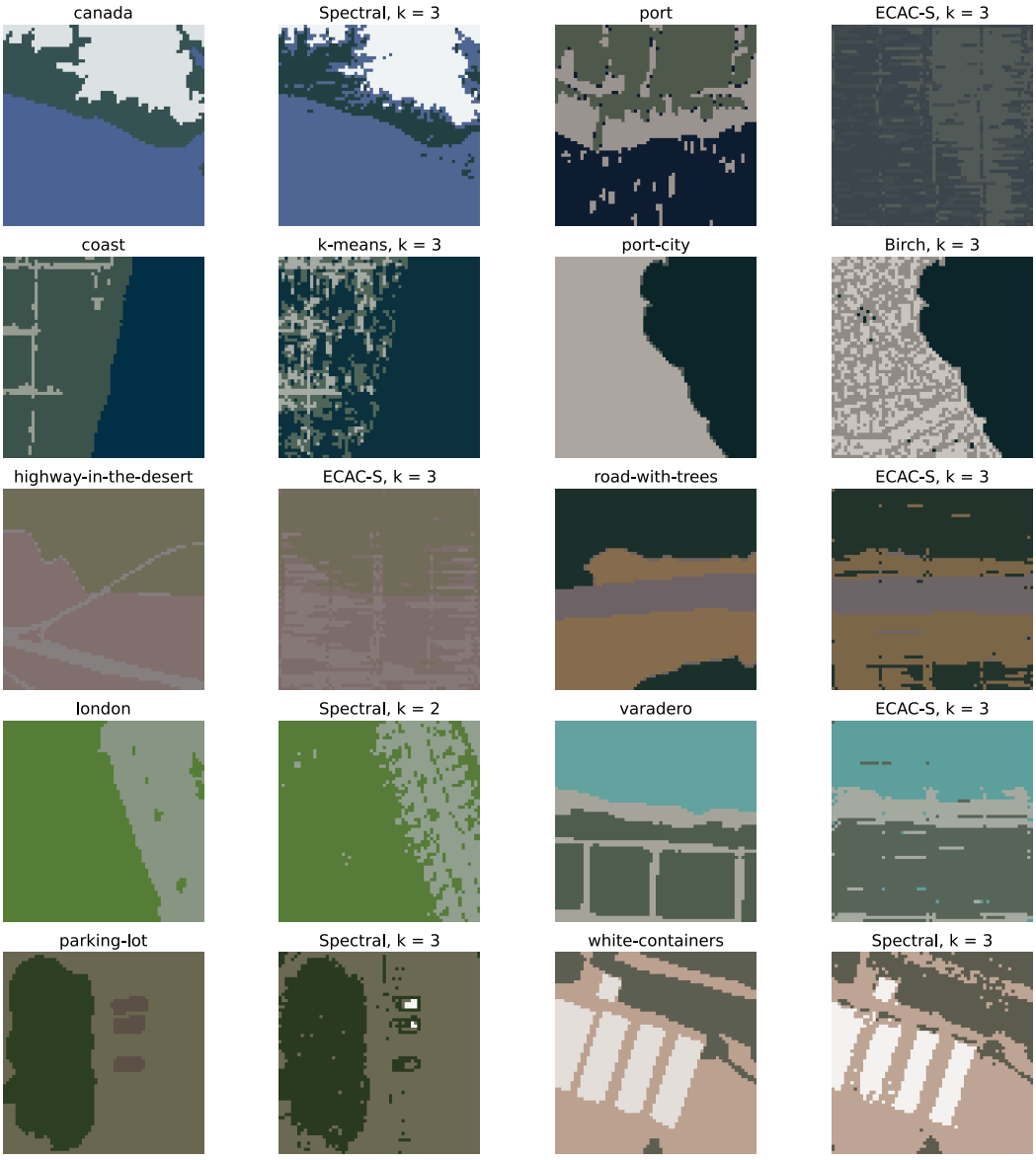

Figure 5.24: Ground truth masks and best segmentation per image in the real application benchmark of ECAC-S.

The 64x64 pixels ground truth masks can be found in Fig. 5.24 (columns one and three), along with the best segmentations generated by the algorithms that got the highest average ARI per image (columns two and four, where we also include the proposed number of clusters k for each capture). Spectral-clustering got the highest performance in the London image, which implied the challenging problem of correctly identifying the colors of cars and buildings. k-means performed best in the Coast image, capturing the ocean almost perfectly, but failed to represent the buildings and streets. ECAC-S outperformed the other algorithms in complex scenarios with unbalanced clusters containing slight color differentiation, as shown in the Highway-in-the-desert mask and solution. We confidently infer that ECAC-S offers comparable performance to available methods in this benchmark, which required processing of large datasets, and it can be applied in cross-domain tasks when looking for clusters that do not follow any predefined structure.

Chapter 6

Conclusion

Our series of Evolutionary Clustering Algorithms using Classifiers (ECAC) are novel methods that maximize the novel generalization cluster quality criterion following evolutionary search techniques. The proposed objective functions in our algorithms compute an approximation of the generalization capabilities of a solution using classifiers, unlike common approaches, which often rely on distance functions for measuring dissimilarity (introducing clustering bias). This is achieved by evaluating the capability of a partition to induce a set of well-trained classifiers, using its representative genotype as target labels. The algorithms in our ECAC series of contributions are recommended for clustering when looking for solutions that adapt, identify and capture the native relationships within the data without imposing nor making assumptions about their cluster structure.

We divided this thesis project into the development of three ECAC contributions, with one article per method. The first version is the original ECAC, an algorithm that takes a step forward and outperformed HG-means in a single-objective Evolutionary Clustering benchmark, in 6 out of 10 datasets. Our second proposal, F1-ECAC, provides a more interpretable pipeline by introducing the F_1 score instead of the Area Under the Curve (AUC) to test the classifiers and was proved competitive against traditional, single and multi-objective Evolutionary Clustering algorithms. F1-ECAC was 7 times faster than ECAC, and the quality of the solutions (in terms of Adjusted RAND Index) was improved by a margin of 83%. The experimental results of F1-ECAC mark its position in the literature as an algorithm with a distinctive objective function that offers high-performance in a wide variety of scenarios, returning no significant difference in Adjusted RAND Index against remarkable algorithms such as k-means, HG-means, and MOCLE. ECAC-S is our final contribution to this project, and returned the highest-quality solutions we were capable of achieving. ECAC-S reduced average computational runtime from 313 seconds to 60 seconds, without compromising solution quality (no significant difference in Adjusted RAND Index). Its efficiency improvements, compared to previous versions and mainly attributed to the inclusion of parallel computing and the modification of its objective function, allowed us to perform an image segmentation benchmark, achieving state-of-the-art performance against high-performing algorithms that have been applied in computer vision before. ECAC-S achieved better segmentation performance (average Adjusted RAND Index) than k-means, Spectral Clustering, Birch, and DBSCAN in 4 out of 10 images. We implemented our methods in algorithms ready to be deployed in any data mining task.

The ECAC series assists the clustering criterion selection dilemma by diminishing the need for a domain expert due to the emulated unweighted multi-expert voting system for evaluating thousands of solutions created along the evolutionary process of our algorithms with a quality criterion proven to be beneficial with data of varying nature. Our proposals offer a straightforward hyper-parameter setting, requiring no previous knowledge of the cluster structure and not presenting sensitivity towards the required number of clusters. Moreover, the number of generations and population size hyper-parameters are influential in augmenting or reducing the search space. However, neither of them influences the quality of the objective function's evaluation capabilities. Still, our ECAC family of contributions maintains the efficiency and intelligibility benefits of single-objective optimization by following the search strategy of the genetic algorithm. For future work, we plan to explore the application range of ECAC-S along with alternative optimization techniques and meta-heuristics. This project is the result of a continuous and collective learning process, and we hope the ECAC series represents a valuable asset for your data mining tasks.

Bibliography

[1] ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y., AND ZHENG, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] ANDERBERG, M. R. *Cluster analysis for applications: probability and mathematical statistics: a series of monographs and textbooks*, vol. 19. Academic press, 2014.

[3] BERKHIN, P. A survey of clustering data mining techniques. In *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[4] BRUSCO, M. J., SINGH, R., CRADIT, J. D., AND STEINLEY, D. Cluster analysis in empirical om research: survey and recommendations. *International Journal of Operations & Production Management* (2017).

[5] CHANG, H., AND YEUNG, D.-Y. Robust path-based spectral clustering. *Pattern Recognition 41*, 1 (2008), 191–203.

[6] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation 6*, 2 (2002), 182–197.

[7] DEMŠAR, J., CURK, T., ERJAVEC, A., ČRT GORUP, HOČEVAR, T., MILUTINOVIČ, M., MOŽINA, M., POLAJNAR, M., TOPLAK, M., STARIČ, A., ŠTAJDOHAR, M., UMEK, L., ŽAGAR, L., ŽBONTAR, J., ŽITNIK, M., AND ZUPAN, B. Orange: Data mining toolbox in python. *Journal of Machine Learning Research 14* (2013), 2349–2353.

[8] DERRAC, J., GARCÍA, S., MOLINA, D., AND HERRERA, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation 1*, 1 (2011), 3–18.

[9] DONATH, W. E., AND HOFFMAN, A. J. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development 17*, 5 (1973), 420–425.

[10] DUA, D., AND GRAFF, C. UCI machine learning repository, Accessed November, 2021 2017.

[11] ERTÖZ, L., STEINBACH, M., AND KUMAR, V. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the 2003 SIAM international conference on data mining* (2003), SIAM, pp. 47–58.

[12] ESPIN, A. Snn-clustering. https://github.com/albert-espin/snn-clustering, 2019.

[13] ESTER, M., KRIEGEL, H.-P., SANDER, J., XU, X., ET AL. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[14] ESTIVILL-CASTRO, V. Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter 4*, 1 (2002), 65–75.

[15] FACELI, K., DE CARVALHO, A. C., AND DE SOUTO, M. C. Multi-objective clustering ensemble. *International Journal of Hybrid Intelligent Systems 4*, 3 (2007), 145–156.

[16] FACELI, K., DE SOUTO, M. C., DE ARAUJO, D. S., AND DE CARVALHO, A. C. Multi-objective clustering ensemble for gene expression data analysis. *Neurocomputing 72*, 13-15 (2009), 2763–2774.

[17] FALKENAUER, E. *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, 1998.

[18] FERLIGOJ, A., AND BATAGELJ, V. Direct multicriteria clustering algorithms. *Journal of classification 9*, 1 (1992), 43–61.

[19] FORSYTH, D., AND PONCE, J. *Computer Vision: A Modern Approach*. Pearson, 01 2003.

[20] FRÄNTI, P., AND SIERANOJA, S. K-means properties on six clustering benchmark datasets, 2018.

[21] FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association 32*, 200 (1937), 675–701.

[22] GARZA-FABRE, M. Delta-mock. https://github.com/garzafabre/Delta-MOCK, 2017.

[23] GARZA-FABRE, M., HANDL, J., AND KNOWLES, J. An improved and more scalable evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation 22*, 4 (2017), 515–535.

[24] GHOSH, S., DAS, N., DAS, I., AND MAULIK, U. Understanding deep learning techniques for image segmentation. *ACM Comput. Surv. 52*, 4 (aug 2019).

[25] GIONIS, A., MANNILA, H., AND TSAPARAS, P. Clustering aggregation. *Acm transactions on knowledge discovery from data (tkdd) 1*, 1 (2007), 4–es.

[26] GOLDBERGER, J., HINTON, G. E., ROWEIS, S., AND SALAKHUTDINOV, R. R. Neighbourhood components analysis. *Advances in neural information processing systems 17* (2004), 513–520.

[27] GOOGLE. Google maps, Accessed November, 2021 Online.

[28] GRIBEL, D. Hg-means. https://github.com/danielgribel, 2019.

[29] GRIBEL, D., AND VIDAL, T. Hg-means: A scalable hybrid genetic algorithm for minimum sum-of-squares clustering. *Pattern Recognition 88* (2019), 569–583.

[30] HALKIDI, M., BATISTAKIS, Y., AND VAZIRGIANNIS, M. On clustering validation techniques. *Journal of intelligent information systems 17*, 2 (2001), 107–145.

[31] HANDL, J., AND KNOWLES, J. Evolutionary multiobjective clustering. In *International Conference on Parallel Problem Solving from Nature* (2004), Springer, pp. 1081–1091.

[32] HANDL, J., AND KNOWLES, J. An evolutionary approach to multiobjective clustering. *IEEE transactions on Evolutionary Computation 11*, 1 (2007), 56–76.

[33] HANSEN, P., AND MLADENOVIĆ, N. J-means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition 34*, 2 (2001), 405–413.

[34] HARALICK, R. M., AND SHAPIRO, L. G. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing 29*, 1 (1985), 100–132.

[35] HARTIGAN, J. A., AND WONG, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics) 28*, 1 (1979), 100–108.

[36] HERBOLD, S. Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software 5*, 48 (2020), 2173.

[37] HRUSCHKA, E. R., CAMPELLO, R. J., FREITAS, A. A., ET AL. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 39*, 2 (2009), 133–155.

[38] JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters 31*, 8 (2010), 651–666.

[39] JAIN, A. K., AND DUBES, R. C. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[40] JAIN, A. K., DUIN, R. P. W., AND MAO, J. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence 22*, 1 (2000), 4–37.

[41] JAIN, A. K., AND LAW, M. H. Data clustering: A user's dilemma. In *International conference on pattern recognition and machine intelligence* (2005), Springer, pp. 1–10.

[42] JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR) 31*, 3 (1999), 264–323.

[43] JAN, M. Z., MUNOZ, J. C., AND ALI, M. A. A novel method for creating an optimized ensemble classifier by introducing cluster size reduction and diversity. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[44] JIANG, Y., AND VERMA, N. Meta-learning to cluster. *arXiv preprint arXiv:1910.14134* (2019).

[45] KANUNGO, T., MOUNT, D. M., NETANYAHU, N. S., PIATKO, C. D., SILVERMAN, R., AND WU, A. Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence 24*, 7 (2002), 881–892.

[46] KAUFMAN, L., AND ROUSSEEUW, P. J. *Finding groups in data: an introduction to cluster analysis*, vol. 344. John Wiley & Sons, 2009.

[47] KHAN, H. A. Nsga-ii. https://github.com/haris989/NSGA-II, 2017.

[48] KHEDAIRIA, S., AND KHADIR, M. T. A multiple clustering combination approach based on iterative voting process. *Journal of King Saud University-Computer and Information Sciences* (2019).

[49] KULTZAK, A. Mcla. https://github.com/kultzak/MCLA, 2019.

[50] LE CESSIE, S., AND VAN HOUWELINGEN, J. C. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics) 41*, 1 (1992), 191–201.

[51] LIKAS, A., VLASSIS, N., AND VERBEEK, J. J. The global k-means clustering algorithm. *Pattern recognition 36*, 2 (2003), 451–461.

[52] MACQUEEN, J., ET AL. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), no. 14 in 1, Oakland, CA, USA, pp. 281–297.

[53] MALIK, J., BELONGIE, S., LEUNG, T., AND SHI, J. Contour and texture analysis for image segmentation. *International journal of computer vision 43*, 1 (2001), 7–27.

[54] MCKINNEY, W. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (2010), S. van der Walt and J. Millman, Eds., pp. 56–61.

[55] MINAEE, S., BOYKOV, Y. Y., PORIKLI, F., PLAZA, A. J., KEHTARNAVAZ, N., AND TERZOPOULOS, D. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–10.

[56] MITCHELL, T. M. *Machine learning*. McGraw-hill New York, 1997.

[57] MUKHOPADHYAY, A., MAULIK, U., AND BANDYOPADHYAY, S. A survey of multiobjective evolutionary clustering. *ACM Computing Surveys (CSUR) 47*, 4 (2015), 1–46.

[58] MUKHOPADHYAY, A., MAULIK, U., BANDYOPADHYAY, S., AND COELLO, C. A. C. Survey of multiobjective evolutionary algorithms for data mining: Part ii. *IEEE Transactions on Evolutionary Computation 18*, 1 (2013), 20–35.

[59] MURTHY, C. A., AND CHOWDHURY, N. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters 17*, 8 (1996), 825–832.

[60] NANDA, S. J., AND PANDA, G. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary computation 16* (2014), 1–18.

[61] NEMENYI, P. *Distribution-free Multiple Comparisons*. Princeton University, 1963.

[62] NG, A., JORDAN, M., AND WEISS, Y. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems* (2002), T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14, MIT Press, pp. 1–10.

[63] OROUSKHANI, M., SHI, D., AND OROUSKHANI, Y. Multi-objective evolutionary clustering with complex networks. *Expert Systems with Applications 165* (2021), 113916.

[64] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., DESMAISON, A., KOPF, A., YANG, E., DEVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., BAI, J., AND CHINTALA, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[65] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[66] PLATT, J., ET AL. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers 10*, 3 (1999), 61–74.

[67] PROVOST, F., AND FAWCETT, T. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. " O'Reilly Media, Inc.", 2013.

[68] REBACK, J., MCKINNEY, W., JBROCKMENDEL, DEN BOSSCHE, J. V., AUGSPURGER, T., CLOUD, P., HAWKINS, S., GFYOUNG, SINHRKS, ROESCHKE, M., KLEIN, A., PETERSEN, T., TRATNER, J., SHE, C., AYD, W., NAVEH, S., PATRICK, GARCIA, M., SCHENDEL, J., HAYDEN, A., SAXTON, D., JANCAUSKAS, V., GORELLI, M., SHADRACH, R., MCMASTER, A., BATTISTON, P., SEABOLD, S., DONG, K., CHRIS B1, AND H VETINARI. pandas-dev/pandas: Pandas 1.2.4, apr 2021.

[69] REY, D., AND NEUHAUSER, M. *Wilcoxon-Signed-Rank Test*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, ch. 1, pp. 1658–1659.

[70] RIPON, K. S. N., AND SIDDIQUE, M. N. H. Evolutionary multi-objective clustering for overlapping clusters detection. In *2009 IEEE Congress on Evolutionary Computation* (2009), IEEE, pp. 976–982.

[71] RODRÍGUEZ, J., MEDINA-PÉREZ, M. A., GUTIERREZ-RODRÍGUEZ, A. E., MONROY, R., AND TERASHIMA-MARÍN, H. Cluster validation using an ensemble of supervised classifiers. *Knowledge-Based Systems 145* (2018), 134–144.

[72] ROSENFELD, A., AND KAK, A. C. *Digital Picture Processing: Volume 1*, 2 ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[73] SAINZ-TINAJERO, B. M. Ecac. https://github.com/benjaminsainz/ecac, 2021.

[74] SAINZ-TINAJERO, B. M. F1-ecac. https://github.com/benjaminsainz/f1-ecac, 2021.

[75] SAINZ-TINAJERO, B. M. Mocle. https://github.com/benjaminsainz/mocle, 2021.

[76] SAINZ-TINAJERO, B. M. Ecac-s. https://github.com/benjaminsainz/ECAC-S, 2022.

[77] SAINZ-TINAJERO, B. M., GUTIERREZ-RODRIGUEZ, A. E., CEBALLOS, H. G., AND CANTU-ORTIZ, F. J. Evolutionary clustering algorithm using supervised classifiers. In *2021 IEEE Congress on Evolutionary Computation (CEC)* (June 2021), pp. 2039–2045.

[78] SAINZ-TINAJERO, B. M., GUTIERREZ-RODRIGUEZ, A. E., CEBALLOS, H. G., AND CANTU-ORTIZ, F. J. F1-ecac: Enhanced evolutionary clustering using an ensemble of supervised classifiers. *IEEE Access 9* (2021), 134192–134207.

[79] SAINZ-TINAJERO, B. M., OTERO-ARGOTE, D., OROZCO-MORA, C. E., AND GONZALEZ-MENDOZA, M. Facing a pandemic: A covid-19 time series analysis of vaccine impact. In *Advances in Computational Intelligence* (Cham, 2021), I. Batyrshin, A. Gelbukh, and G. Sidorov, Eds., Springer International Publishing, pp. 303–314.

[80] SANTOS, J. M., AND EMBRECHTS, M. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *Artificial Neural Networks – ICANN 2009* (Berlin, Heidelberg, 2009), C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, Eds., Springer Berlin Heidelberg, pp. 175–184.

[81] SCHUBERT, E., SANDER, J., ESTER, M., KRIEGEL, H. P., AND XU, X. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS) 42*, 3 (2017), 1–21.

[82] SHI, J., AND MALIK, J. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence 22*, 8 (2000), 888–905.

[83] SOKOLOVA, M., JAPKOWICZ, N., AND SZPAKOWICZ, S. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence* (2006), Springer, pp. 1015–1021.

[84] STEINLEY, D. Properties of the hubert-arable adjusted rand index. *Psychological methods 9*, 3 (2004), 386.

[85] STREHL, A., AND GHOSH, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research 3*, Dec (2002), 583–617.

[86] SYSWERDA, G. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd international conference on genetic algorithms* (1989), pp. 2–9.

[87] VAN ROSSUM, G., AND DRAKE, F. L. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[88] VAPNIK, V. N. An overview of statistical learning theory. *IEEE transactions on neural networks 10*, 5 (1999), 988–999.

[89] VEENMAN, C. J., REINDERS, M. J. T., AND BACKER, E. A maximum variance cluster algorithm. *IEEE Transactions on pattern analysis and machine intelligence 24*, 9 (2002), 1273–1280.

[90] VON LUXBURG, U. A tutorial on spectral clustering. *Statistics and computing 17*, 4 (2007), 395–416.

[91] VOORHEES, E. M. The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval. Tech. rep., Cornell University, 1985.

[92] WASKOM, M. L. seaborn: statistical data visualization. *Journal of Open Source Software 6*, 60 (2021), 3021.

[93] WIRTH, R., AND HIPP, J. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining* (2000), vol. 1, Springer-Verlag London, UK.

[94] XU, R., AND WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on neural networks 16*, 3 (2005), 645–678.

[95] YU, AND SHI. Multiclass spectral clustering. In *Proceedings Ninth IEEE International Conference on Computer Vision* (2003), pp. 313–319 vol.1.

[96] ZHANG, T., RAMAKRISHNAN, R., AND LIVNY, M. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec. 25*, 2 (jun 1996), 103–114.

[97] ZHU, S., XU, L., AND GOODMAN, E. D. Evolutionary multi-objective automatic clustering enhanced with quality metrics and ensemble strategy. *Knowledge-Based Systems 188* (2020), 105018.

Curriculum Vitae

Benjamin Mario Sainz Tinajero was awarded the Academic Talent Scholarship at Tecnologico de Monterrey, Mexico, where he received the B.S. degree in Mechatronics Engineering in 2020. Benjamin was awarded a Full-Tuition Scholarship for the excellence program at Tecnologico de Monterrey's Graduate School in 2020, where he received the M.S. degree in Computer Science in 2022. He has published scientific articles on machine learning, unsupervised learning, computer vision, and data mining. Benjamin's master's thesis was focused on the design of clustering algorithms that avoid the intrinsic bias of conventional methods by favoring solutions optimizing the novel cluster quality criterion of generalization, with objective functions modeled as supervised learning problems. Benjamin is an active reviewer at journals of the Institute of Electrical and Electronics Engineers (IEEE) and is an active member of the Mexican Society of Artificial Intelligence.

This document was typed in using LaTeX 2_ε[1] by Benjamin Mario Sainz Tinajero.

[1]The template `MCCi-DCC-Thesis.cls` used to set up this document was prepared by the Research Group with Strategic Focus in Intelligent Systems of Tecnológico de Monterrey, Monterrey Campus.