UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

MÉTODO DE CLUSTERING BASADO EN EL PRINCIPIO DE MÁXIMA ENTROPÍA

TESIS
QUE PARA OPTAR POR EL GRADO DE:
DOCTOR EN CIENCIAS (COMPUTACIÓN)

PRESENTA:
EDWYN JAVIER ALDANA BOBADILLA

TUTOR
DR. ANGEL FERNANDO KURI MORALES
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

MÉXICO, D.F. JUNIO 2015

Universidad Nacional Autónoma de México

Doctoral Thesis

A Clustering Method Based on the Maximum Entropy Principle

Author:
Edwyn Javier Aldana Bobadilla

Supervisor:
Dr. Angel Fernando Kuri Morales

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Sciences

in the

Graduate School of Computer Science and Engineering

June 2015

Declaration of Authorship

I, Edwyn Javier Aldana Bobadilla, declare that this thesis titled, 'A Clustering Method Based on the Maximum Entropy Principle' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________________

Date: ______________________________________

"In the mountains of truth you will never climb in vain: either you will get up higher today or you will exercise your strength so as to be able to get up higher tomorrow".

Friedrich Nietzsche

Abstract

Graduate School of Computer Science and Engineering

Doctor of Philosophy in Computer Sciences

A Clustering Method Based on the Maximum Entropy Principle

by Edwyn Javier ALDANA BOBADILLA

Clustering is an unsupervised process to determine which unlabeled objects in a set share interesting properties. The objects are grouped into k subsets (clusters) whose elements optimize a proximity measure. Methods based on information theory have proven to be feasible alternatives. They are based on the assumption that a cluster is one subset with the minimal possible degree of "disorder". They attempt to minimize the entropy of each cluster. We propose a clustering method based on the maximum entropy principle. Such a method explores the space of all possible probability distributions of the data to find one that maximizes the entropy subject to extra conditions based on prior information about the clusters. The prior information is based on the assumption that the elements of a cluster are "similar" to each other in accordance with some statistical measure. As a consequence of such a principle, those distributions of high entropy that satisfy the conditions are favored over others. Searching the space to find the optimal distribution of object in the clusters represents a hard combinatorial problem, which disallows the use of traditional optimization techniques. Genetic algorithms are a good alternative to solve this problem. We benchmark our method relative to the best theoretical performance, which is given by the Bayes classifier when data are normally distributed, and a multilayer perceptron network, which offers the best practical performance when data are not normal. In general, a supervised classification method will outperform a non-supervised one, since, in the first case, the elements of the classes are known *a priori*. In what follows, we show that our method's effectiveness is comparable to a supervised one. This clearly exhibits the superiority of our method.

Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Angel Fernando Kuri Morales for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Humberto Carrillo Calvet and Dr. Javier Garcia Garcia, for their encouragement, and insightful comments.

My sincere thanks also goes to Dr. Katya Rodríguez Vázquez and Dr. Nicolás Kemper Valverde, for reviewing this work and making critical suggestions.

I would especially like to thank the Graduate School of Computer Science and Engineering at Universidad Nacional Autónoma de México.

I also want to thank CONACYT (Consejo Nacional de Ciencia y Tecnología) for their financial support granted during my doctoral research.

Contents

List of Figures

List of Tables

Prologue

One of the basic tasks in Data Mining is the process of determining which unlabeled objects in a set do share interesting properties. Such process is denoted as "clustering", where the objects are grouped into k subsets or clusters whose elements optimize a proximity measure. Usually such proximity is defined by a metric or distance. The more traditional clustering methods are based on the minimization of such distance. This fact imposes important constraints on the geometry of the clusters found. Since each element in a cluster lies within a radial distance relative to a given center, the shape of the covering or hull of a cluster is hyper-spherical (convex) which sometimes does not encompass adequately the elements that belong to it.

The research described in this thesis was focused on finding a clustering method that overcomes such constraints. To reach this goal we explored different approaches: (1) We proposed a method in which the elements of a cluster were determined by finding a set of locus (one for each cluster), which encompass the data to be clustered. A locus is a curve or surface formed by all the data satisfying a particular equation. To find the feasible equations, we resorted to Gielis's Super-Formula (GSF) that allowed us to define a wide family of functions. To find the optimal locus, we used a Genetic Algorithm (GA). This approach was not hampered by distance considerations. A point to be made is that the generalization for data in $\Re^n$ $(n > 3)$ is a non-trivial problem. This limitation forced us to explore other alternatives. The reader can find details about this approach in Appendix D which was published in [1] and presented at *7th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Databases*(AIKED'08). (2) Other alternative, based on the identification of those elements of the dataset which optimize some *validity index* was proposed. In the usual process, the clusters are found via some arbitrary clustering method and then assigned a quality grade according to a certain validity index. We propose a novel method finding the elements of the clusters from the indices directly. Usually the purported indices are expressed mathematically with some complex function which is not prone to optimization by any of the traditional methods. We analyze clusters resulting by optimizing several validity indices with a GA. This approach and its results are presented in Appendix E, which were published in the book

New Fundamental Technologies in Data Mining [2]. We found that the optimization of validity indices is a promising approach, however such approach did not solve the problem about the constraints of the geometry of the clusters, since most indices are based on proximity relationships on the metric space of the dataset. (3) This fact led us to search a validity index on the probability space of the dataset rather than on the metric space of it. Such validity index was an information theoretical function. Our idea was to find a arrangement of clusters whose probability distributions optimize this new validity index. The results of this approach have partly been presented at international conferences in Holland, Germany and USA (see Appendix F).

Since exploring the feasible space of all probability distributions is a hard problem, the use of suitable optimization method was compulsory. Based on the assumption that under certain conditions the GAs always converge to the optimal solution [3], we consider the GAs as a good alternative to solve such problem. To determine the best GA, we conducted a study regarding the performance of a set of 4 structurally different GAs and a non-evolutionary algorithm to solve a wide reservoir of optimization problems. The results of this study were presented at *12th Mexican International Conference on Artificial Intelligence* and published in [4, 5]. We have decided to include this study in Appendix G.

The referred work gave rise to the following publications and presentations at international conferences:

1. Kuri-Morales, A., Aldana-Bobadilla, E., Clustering with an N-Dimensional Extension of Gielis Superformula, WSEAS, 343-350, Editor(s)): Zadeh et al., ISBN: 978-960-6766-4, ISSN: 1790-5109, 20/02/2008.

2. Kuri-Morales, A., Aldana-Bobadilla, E., Evolutionary Entropic Clustering: a new methodology for data mining, Sociedad Mexicana de Inteligencia Artificial, CD, Editor(s)): Alexander Gelbukh, ISBN: 978-607-95367, 09/11/2009.

3. Kuri-Morales, A., Aldana-Bobadilla, E., Finding irregularly shaped clusters based on Entropy, ICDM - 2010, Springer, 59-70, Editor(s)): Petra Perner, ISBN: 3-642-14399-7, ISSN: 0302-9743, 10/07/2010.

4. Aldana-Bobadilla, E., Kuri-Morales, A., A methodology to find clusters in the data based on Shannon's Entropy and Genetic Algorithms, WorldComp'11, CSRea Press, 21-27, Editor(s)): Arabnia, H., Hashemi, R., Solo, A., ISBN: 1-60132-182-1, 18/07/2011.

5. Aldana-Bobadilla, E., Kuri-Morales, A., A methodology to find clusters in the data based on Shannon's Entropy and Genetic Algorithms, Proceedings of the

10th WSEAS international conference on communications, WSEAS, 272-280, Editor(s)): Mastorakis, N., Mladenov, V., Savkovic-Stevanovic, J., ISBN: 978-960-474-28, 01/03/2011.

6. Kuri-Morales, A., Aldana-Bobadilla, E., The Best Genetic Algorithm Part I, A Comparative Study of Structurally Different Genetic Algorithms, Springer, 1-15, Editor(s)): Castro, Gelbukh, González, ISBN: 9783642451119, ISSN: 1611-3349, 27/11/2013. **Best Paper Award, Second Place**.

7. Kuri-Morales, A., Aldana-Bobadilla, E., López-Peña, J., The Best Genetic Algorithm Part II, A Comparative Study of Structurally Different Genetic Algorithms, Springer, 16-29, Editor(s)): Castro, Gelbukh, González, ISBN: 9783642451119, ISSN: 1611-3349, 27/11/2013.

8. Aldana-Bobadilla, E., Kuri-Morales, A., Unsupervised Classifier Based on Heuristic Optimization and Maximum Entropy Principle, Springer, 17-30, Editor(s)): Emmerich, M. et al, ISBN: 9783319011271, ISSN: 2194-5357, 10/07/2013

Other important publication was a chapter book dealing with different approaches of clustering. The complete reference is shown in what follows:

9. Kuri-Morales, A., Aldana-Bobadilla, E., The search for irregularly shaped clusters in data mining, New Fundamental Technologies in Data Mining, Intech, 323-354, Editor(s)): Funtasu, K., Hasegawa, K., ISBN: 9789533075471, 15/01/2011.

All these works are the basis and foundation of this dissertation. They allowed us to mature the final results which were submitted for publication in the international journal of *Entropy* (see [6]). The comments and suggestions of the reviewers allowed us to improve our research and validate the final results.

Chapter 1

Introduction

Pattern recognition is a scientific discipline whose methods allow us to describe and classify objects. The descriptive process involves the symbolic representation of these objects through a numerical vector $\vec{x}$:

$$\vec{x} = [x_1, x_2, \ldots x_n] \in \Re^n \tag{1.1}$$

where its n components represent the value of the attributes of such objects. Given a set of objects (“dataset”) X, there are two approaches to attempt the classification: (1) supervised; and (2) unsupervised.

In the unsupervised approach case, no prior class information is used. Such an approach aims at finding a hypothesis about the structure of X based only on the similarity relationships among its elements. These relationships allow us to divide the space of X into k subsets, called clusters. A cluster is a collection of elements of X, which are "similar" between them and "dissimilar" to the elements belonging to other clusters. Usually, the similarity is defined by a metric or distance function $d : X \times X \rightarrow \Re$ [7–9]. The process to find the appropriate clusters is typically denoted as a clustering method.

In the literature, many clustering methods have been proposed [10, 11]. Most of them begin by defining a set of k centroids (one for each cluster) and associating each element in X with the nearest centroid based on a distance d. This process is repeated, until the difference between centroids at iteration t and iteration $t-1$ tends to zero or when some other optimality criterion is satisfied. Examples of this approach are **k**-means [12] and fuzzy c-means [13–15]. Other methods not following the previous approach are: (1) hierarchical clustering methods that produce a tree or a graph in which, during the clustering process, based on some similarity measure, the nodes (which represent a possible cluster) are merged or split; in addition to the distance measure, we must also have a stopping criterion to tell us whether the distance measure is sufficient to split a node or to merge two existing nodes [16–18]; (2) density clustering methods, which consider clusters as dense regions of objects in the dataset that are separated by regions of low density; they assume an implicit, optimal number of clusters and make no restrictions with respect to the form or size of the distribution [19]; and (3) meta-heuristic clustering methods that handle the clustering problem as a general optimization problem; these imply a greater flexibility of optimization algorithms, but also longer execution times [20, 21].

We propose a numerical clustering method that lies in the group of meta-heuristic clustering methods, where the optimization criterion is based on information theory. Some methods with similar approaches have been proposed in the literature (see Section 1.5).

1.1 Determining the Optimal Value of k

In most clustering methods, the number k of clusters must be explicitly specified. Choosing an appropriate value has important effects on the clustering results. An adequate selection of k should consider the shape and density of the clusters desired. Although there is not a generally accepted approach to this problem, many methods attempting to solve it have been proposed [11, 22, 23]. In some clustering methods (such as hierarchical and density clustering), there is no need to explicitly specify k. Implicitly, its value is, nevertheless, still required: the cardinality or density of the clusters must be given. Hence, there is always the need to select a value of k. In what follows, we assume that the value of k is given *a priori*. We do not consider the problem of finding the optimal value of k. See, for example, [24–26].

1.2 Evaluating the Clustering Process

Given a dataset X to be clustered and a value of k, the most natural way to find the clusters is to determine the similarity among the elements of X. As mentioned, usually, such similarity is established in terms of proximity through a metric or distance function. In the literature, there are many metrics [27] that allow us to find a variety of clustering solutions. The problem is how to determine if a certain solution is good or not. For instance, in Figures 1.1 and 1.2, we can see two datasets clustered with $k = 2$. For such datasets, there are several possible solutions, as shown in Figures 1.1b,c and 1.2b,c, respectively.

Frequently, the goodness of a clustering solution is quantified through validity indices [10, 14, 28, 29]. The indices in the literature are classified into three categories: (1) external indices that are used to measure the extent to which cluster labels match externally-supplied class labels (F-measure [30], NMIMeasure [31], entropy [32], purity [33]); (2) internal indices, which are used to measure the goodness of a clustering structure with respect to the intrinsic information of the dataset ([34], [35],[36],[37], [14]); and (3) relative indices that are used to compare two different clustering solutions or particular clusters (the RAND index [38] and adjusted RAND index [39]).

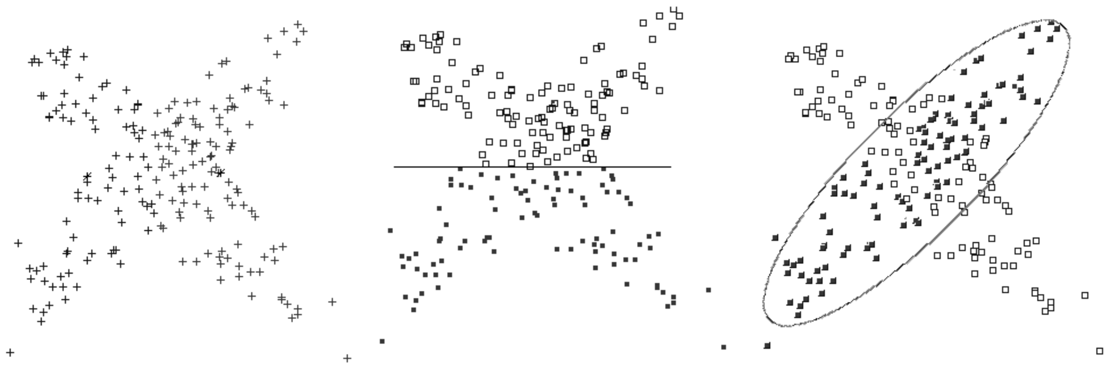

(A) (B) (C)

FIGURE 1.1: Example of a clustering problem. (**a**) Dataset X_1; (**b**) solution for $k = 2$; and (**c**) another solution for $k = 2$.

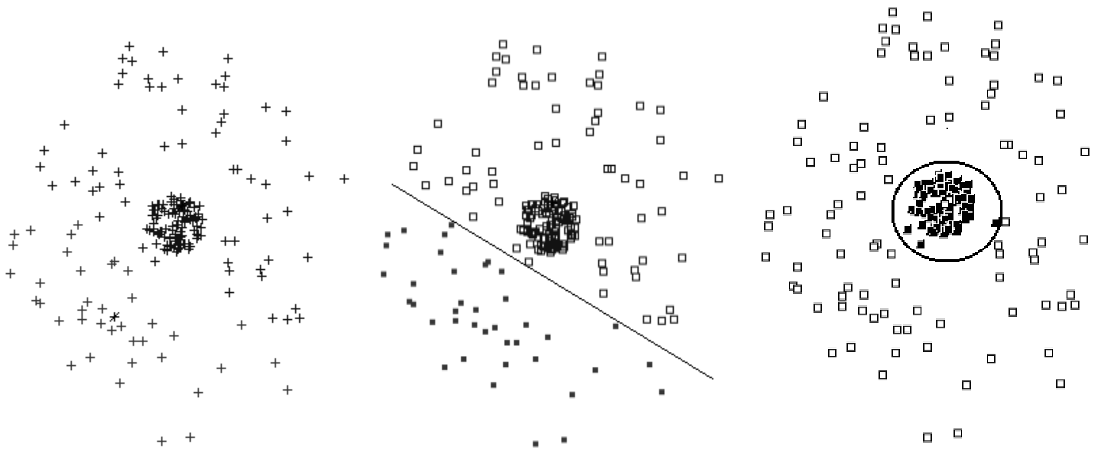

(A) (B) (C)

FIGURE 1.2: Another clustering problem. (**a**) Dataset X_2; (**b**) solution for $k = 2$; and (**c**) another solution for $k = 2$.

Clusters are found and then assigned a validity index. We bypass the problem of qualifying the clusters and, rather, define a quality measure and then find the clusters that optimize it. This allows us to define the clustering process as follows:

Definition 1. Given a dataset X to be clustered and a value of k, clustering is a process that allows the partition of the space of X into k regions, such that the elements that belong to them optimize a validity index q.

Let C_i be a partition of X and $\Pi = \{C_1, C_2, ...C_k\}$ the set of such partitions that represents a possible clustering solution of X. We can define a validity index q as a function of the partition set Π, such that the clustering problem may be reduced to an optimization problem of the form:

$$\begin{gathered} \textit{Optimize } q = f(\Pi) \\ \text{subject to:} \\ g_i(\Pi) \leq 0,\ i = 1, 2..., m \\ h_j(\Pi) = 0,\ j = 1, 2, ..., p \end{gathered} \tag{1.2}$$

where g_i and h_j are constraints, likely based on prior information about the partition set Π (e.g., the desired properties of the clusters).

We want to explore the space of those feasible partition sets and find one that optimizes a validity index instead, doing an *a posteriori* evaluation of a partition set (obtained by any optimization criterion) through such an index. This approach allows us to find "the best clustering" within the limits of a validity index. To tightly confirm such an assertion, we resort to the statistical evaluation of our method (see Chapter 5).

1.3 Finding the Best Partition of X

Finding the suitable partition set Π of X is a difficult problem. The total number of different partition sets of the form $\Pi = \{C_1, C_2, ...C_k\}$ may be expressed by the function $S(N, k)$ associated with the Stirling number of the second kind [40], which is given by:

$$S(N,k) = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} i^N \tag{1.3}$$

where $N = |X|$. For instance, for $N = 50$ and $k = 2$, $S(N, k) \approx 5.63 \times 10^{14}$. This number illustrates the complexity of the problem that we need to solve. Therefore, it is necessary to resort to a method that allows us to explore efficiently a large solution space. In the following section, we briefly describe some meta-heuristic searches.

1.4 Choosing the Meta-Heuristic

During the last few decades, there has been a tendency to consider computationally-intensive methods that can search very large spaces of candidate solutions. Among the many methods that have arisen, we mention tabu search [41–43], simulated annealing [44, 45], ant colony optimization [46], particle swarm optimization [47] and evolutionary computation [48]. Furthermore, among the many variations of evolutionary computation, we find evolutionary strategies [49], evolutionary programming [50], genetic programming [51] and genetic algorithms (GAs) [52]. All of these methods are used to find approximate solutions for complex optimization problems. It was proven that an elitist GA always converges to the global optimum [3]. Such a convergence, however, is not bounded in time, and the selection of the GA variation with the best dynamic behavior is very convenient. In this regard, we rely on the conclusions of previous analyses [4, 5], which showed that a breed of GA, called the eclectic genetic algorithm (EGA), achieves the best relative performance. These analyses have been include in Appendix G. Such an algorithm incorporates the following:

(1) Full elitism over a set of size n of the last population. Given that, by generation t, the number of individual tested is nt, the population in such a generation consists of the best n individuals.

(2) Deterministic selection as opposed to the traditional proportional selection operator. Such a scheme emphasizes genetic variety by imposing a strategy that enforces the crossover of predefined individuals. After sorting the individual's fitness from better to worse, the i-th individual is combined with the $(n - i + 1)$-th individual.

(3) Annular (two-point) crossover.

(4) Random mutation of a percentage of bits of the population.

In Appendix A, we present additional information and the pseudo-code of EGA, detailed information about it, can be found in [5].

1.5 Related Works

Our method is meta-heuristic. The main idea behind such method is to handle the clustering problem as a general optimization problem. In the literature there are various suitable meta-heuristic clustering methods. For instance, in [55–57], three different clustering methods based on differential evolution, ant colony and multi-objective programming, respectively, are proposed.

Both meta-heuristic or analytical methods (iterative, density-based, hierarchical) have objective functions, which are associated with a metric. In our method, the validity index becomes the objective function.

Our objective function is based on information theory. Several methods based on the optimization of information theoretical functions have been studied [58–63]. Typically, they optimize quantities, such as entropy [32] and mutual information [64]. These quantities involve determining the probability distribution of the dataset in a non-parametric approach, which does not make assumptions about a particular probability distribution. The term non-parametric does not imply that such methods completely lack parameters; they aim to keep the number of assumptions as weak as possible (see [60, 63]). Non-parametric approaches involve density functions, conditional density functions, regression functions or quantile functions in order to find the suitable distribution. Typically, such functions imply tuning parameters that determine the convergence of searching for the optimal distribution.

A common clustering method based on information theory is ENCLUS (entropy clustering) [65], which allows us to split iteratively the space of the dataset X in order to find those subspaces that minimize the entropy. The method is motivated by the fact that a subspace with clusters typically has lower entropy than a subspace without clusters. In order to split the space of X, the value of a resolution parameter must be defined. If such a value is too large, the method may not able to capture the differences in entropy in different regions in the space.

Our proposal differs from traditional clustering methods (those based on minimizing a proximity measure as k-means or fuzzy **c**-means) in that, instead of finding those clusters that optimize such a measure and then defining a validity index to evaluate its quality, our method finds the clusters that optimize a purported validity index.

As mentioned, such validity index is an information theoretical function. In this sense, our method is similar to those mentioned methods based on information theory. However, it does not use explicitly a non-parametric approach to find the distribution of the dataset. It explores the space of all possible probability distributions to find one that optimizes our validity index. For this reason, the use of a suitable meta-heuristic is compulsory. With the exception of the parameters of such a meta-heuristic, our method does not resort to additional parameters to find the optimal distribution and, consequently, the optimal clustering solution.

Our validity index involves entropy. Usually in the methods based on information theory, the entropy is interpreted as a "disorder" measure; thus, an obvious way is to minimize such a measure. In our work, we propose to maximize it due to *the maximum entropy principle*.

1.6 Organization of the Discussion

The rest of this work is organized as follows: In Chapter 2, we briefly discuss the concept of information content, entropy and the maximum entropy principle. Then, we approach such issues in the context of the clustering problem. In Chapter 3, we formalize the underlying ideas and describe the main line of our method (in what follows, clustering based on entropy (CBE)). In Chapter 4, we present a general description of the datasets. Such datasets were grouped into three categories: (1) synthetic Gaussian datasets; (2) synthetic non-Gaussian datasets; and (3) experimental datasets taken from the UCI Machine Learning Repository [67]. In Chapter 5, we present the methodology to evaluate the effectiveness of CBE regardless of a validity index. In Chapter 6, we show the experimental results. We use synthetically-generated Gaussian datasets and apply a Bayes classifier (BC) [66, 68, 69]. We use the results as a standard for comparison due to its optimal behavior. Next, we consider synthetic non-Gaussian datasets. We prove that CBE yields results comparable to those obtained by a multilayer perceptron network (MLP). We show that our (non-supervised) method's results are comparable to those of BC and MLP, even though these are supervised. The results of other clustering methods are also presented, including an information theoretic-based method (ENCLUS). Finally, in Chapter 7, we present our conclusions. We have included three appendices expounding important details.

Chapter 2

Maximum Entropy Principle and Clustering

The so-called entropy [32] appeals to an evaluation of the information content of a random variable Y with possible values $\{y_1, y_2, ...y_n\}$. From a statistical viewpoint, the information of the event $(Y = y_i)$ is inversely proportional to its likelihood. This information is denoted by $I(y_i)$, which can be expressed as:

$$I(y_i) = log\left(\frac{1}{p(y_i)}\right) = -log\,(p(y_i)) \tag{2.1}$$

From information theory [32, 70], the entropy of Y is the expected value of I. It is given by:

$$H(Y) = -\sum_{i=1}^{N} p(y_i) log\,(p(y_i)) \tag{2.2}$$

Typically, the log function may be taken to be $\log_2$, and then, the entropy is expressed in bits; otherwise, as ln, in which case the entropy is in nats. We will use $\log_2$ for the computations in this paper.

When $p(y_i)$ is uniformly distributed, the entropy of Y is maximal. This means that Y has the highest level of unpredictability and, thus, the maximal information content. The value of the entropy has many intuitive interpretations: the distributions of higher entropy represent more "disorder", they are "smoother", or they are "less predictable".

2.1 Maximum Entropy Principle

Since entropy reflects the amount of "disorder" of a system, many methods employ some form of such a measure in the objective function of clustering. It is expected that each cluster has a *low entropy*, because data points in the same cluster should look similar [59–63].

We consider the clustering problem a stochastic system with a set of states $S = \{\Pi_1, \Pi_2, ..., \Pi_n\}$ whose probabilities are unknown (recall that Π_i is a likely partition set of X of the form $\Pi = \{C_1, C_2, ..., C_k\}$). A possible assertion would be that the probabilities of all states are equal $(p(\Pi_1) = p(\Pi_2) = ... = p(\Pi_{n-1}) = p(\Pi_n))$, and therefore, the system has the maximum entropy. However, if we have additional knowledge, then we should be able to find a probability distribution that is better in the sense that it is less uncertain. This knowledge can consist of certain average values or bounds on the probability distribution of the states, which somehow define several conditions imposed upon such distribution. Usually, there is an infinite number of probability models that satisfies these conditions.

The question is: which model should we choose? The answer lies in the *maximum entropy principle*, which may be stated as follows [71]: when an inference is made on the basis of incomplete information, it should be drawn from the probability distribution that maximizes the entropy, subject to the constraints on the distribution. As mentioned, a condition is an expected value of some measure about the probability distributions. Such a measure is one for which each of the states of the system has a value denoted by $g(\Pi_i)$.

2.1.1 Numerical Example

Let $X = \{9, 10, 9, 2, 1\}$ be a discrete dataset to be clustered with $k = 2$. We assume that the "optimal" labeling of the dataset is the one that is shown in Table 2.1.

X	C
9	C_1
10	C_1
9	C_1
2	C_2
1	C_2

TABLE 2.1: Unidimensional dataset.

Such labeling defines a partition Π^* of the form $\Pi^* = \{C_1 = \{9, 10, 9\}, C_2 = \{2, 1\}\}$. The probability model of Π^* is given by the conditional probabilities $p(x|C_i)$ that represent the likelihood that, when observing cluster C_i, we can find object x. Table 2.2 shows such probabilities.

X	C	$p(x\|C_1)$	$p(x\|C_2)$
9	C_1	0.333	0.000
10	C_1	0.333	0.000
11	C_1	0.333	0.000
2	C_2	0.000	0.500
1	C_2	0.000	0.500

TABLE 2.2: Probability model of Π^*.

The entropy of X conditioned on the random variable C taking a certain value C_i is denoted as $H(X|C = C_i)$. Thus, we define the total entropy of Π^* as:

$$\begin{aligned} H(\Pi^*) &= \sum_{C_i \in \Pi^*} H(X|C = C_i) \\ H(\Pi^*) &= -\sum_{C_i \in \Pi^*} \sum_{x \in C_i} p(x|C_i) log(p(x|C_i)) \end{aligned} \tag{2.3}$$

Given the probability model of Π^* and Equation (2.3), the total entropy of Π^* is shown in Table 2.3. We may also calculate the mean and the standard deviation of the elements $x \in C_i$ denoted as $\sigma(C_i)$ in order to define a quality index:

$$g(\Pi^*) = \sum_{i=1}^{2} \sigma(C_i) \tag{2.4}$$

X	C	$p(x\|C_1)$	$p(x\|C_2)$	$p(x\|C_1)log\,(p(x\|C_1))$	$p(x\|C_2)log\,(p(x\|C_2))$
9	C_1	0.333	0.000	-0.528	0.000
10	C_1	0.333	0.000	-0.528	0.000
11	C_1	0.333	0.000	-0.528	0.000
2	C_2	0.000	0.500	0.000	-0.500
1	C_2	0.000	0.500	0.000	-0.500
				$H(\Pi^*)$	2.584
				$g(\Pi^*)$	1.316

TABLE 2.3: Probability model and the entropy of Π^*.

Alternative partition sets are shown in Tables 2.4 and 2.5.

X	C	$p(x\|C_1)$	$p(x\|C_2)$	$p(x\|C_1)log\,(p(x\|C_1))$	$p(x\|C_2)log\,(p(x\|C_2))$
9	C_1	0.250	0.000	−0.500	0.000
10	C_1	0.250	0.000	−0.500	0.000
11	C_1	0.250	0.000	−0.500	0.000
2	C_2	0.000	1.000	0.000	0.000
1	C_1	0.250	0.000	−0.500	0.000
				$H(\Pi_1)$	2.000
				$g(\Pi_1)$	3.960

TABLE 2.4: Probability model and entropy of Π_1.

X	C	$p(x\|C_1)$	$p(x\|C_2)$	$p(x\|C_1)log\,(p(x\|C_1))$	$p(x\|C_2)log\,(p(x\|C_2))$
9	C_2	0.000	0.333	0.000	−0.528
10	C_1	0.500	0.000	−0.500	0.000
11	C_1	0.500	0.000	−0.500	0.000
2	C_2	0.000	0.333	0.000	−0.528
1	C_2	0.000	0.333	0.000	−0.528
				$H(\Pi_2)$	2.584
				$g(\Pi_2)$	4.059

TABLE 2.5: Probability model and entropy of Π_2.

In terms of maximum entropy, partition $H(\Pi_2)$ is better than $H(\Pi_1)$. Indeed, we can say that $H(\Pi_2)$ is as good as $H(\Pi^*)$. If we assume that a cluster is a partition with the minimum standard deviation, then the partition set Π^* is the best possible solution of the clustering problem. In this case, the minimum standard deviation represents a condition that may guide the search for the best solution. We may consider any other set of conditions depending on the desired properties of the clusters. The best solution will be the one with the maximum entropy, which complies with the selected conditions.

2.2 Finding the Optimal Partition

In the example above, we knew the labels of the optimal class and the problem became trivial. In the clustering problems, there are no such labels, and thus, it is compulsory to find the optimal solution based solely on the prior information supplied by one or more conditions. We are facing an optimization problem of the form:

$$\begin{gathered}\text{Maximize:} H(\Pi)\\ \text{subject to:}\\ g_1(\Pi) \leq \epsilon_1\\ g_2(\Pi) \leq \epsilon_2\\ \ldots\\ g_n(\Pi) \leq \epsilon_n\\ \Pi \in S\end{gathered} \tag{2.5}$$

where ϵ_i is the upper bound of value of the i-th condition. We do not know the value of ϵ_i, and thus, we do not have enough elements to determine compliant values of g_i. Based on prior knowledge, we infer whether the value of g_i is required to be as small (or large) as possible. In our example, we postulated that $g_i(\Pi)$ (based on the standard deviation of the clusters) should be as small as possible. Here, $g_i(\Pi)$ becomes an optimization condition. Thus, we can redefine the above problem as:

$$\begin{gathered}\text{Optimize:}(H(\Pi), g_1(\Pi), g_2(\Pi), ..., g_n(\Pi))\\ \text{subject to:}\\ \Pi \in S\end{gathered} \tag{2.6}$$

which is a multi-objective optimization problem [72].

Without loss of generality, we can reduce the problem in Equation (2.6) to one of maximizing the entropy and minimizing the sum of the standard deviation of the clusters (for practical purposes, we choose the standard deviation; however, we may consider any other statistics, even higher-order statistics). The resulting optimization problem is given by:

$$\begin{gathered}\text{Maximize:} H(\Pi) \wedge \text{Minimize:} g(\Pi))\\ \text{subject to:}\\ \Pi \in S\end{gathered} \tag{2.7}$$

where $g(\Pi)$ is the sum of the standard deviation of the clusters. In a n-dimensional space, the standard deviation of the cluster C_i is a vector of the form $\vec{\sigma} = (\sigma_1, \sigma_2, \sigma_n)$, where σ_j is the standard deviation of each dimension of the objects $\vec{x} \in C_i$. In general, we calculate the standard deviation of a cluster as:

$$\sigma(C_i) = \sum_{j=1}^{n} \sigma_i \forall \sigma_j \in \vec{\sigma} \tag{2.8}$$

Then, we define the corresponding $g(\Pi)$ as:

$$g(\Pi) = \sum_{i=1}^{k} \sigma(C_i) \tag{2.9}$$

The entropy of the partition set Π denoted by $H(\Pi)$ is given by Equation (2.3).

The problem of Equation (2.7) is intractable via classical optimization methods [73, 74]. The challenge is to simultaneously optimize all of the objectives. The tradeoff point is a Pareto-optimal solution [75]. Genetic algorithms are popular tools used to search for Pareto-optimal solutions [76, 77].

Chapter 3

Solving the Problem through EGA

Most multi-objective optimization algorithms use the concept of dominance in their search to find a Pareto-optimal solution. For each objective function, there exists one different optimal solution. An objective vector constructed with these individual optimal objective values constitutes the ideal objective vector of the form:

$$z^* = (f_1^*, f_2^*, ..., f_n^*) \tag{3.1}$$

where f_i^* is the i-th objective function. Given two vectors z and w, it is said that z dominates w, if each component of z is less or equal to the corresponding component of w, and at least one component is smaller:

$$z \succeq w \leftrightarrow \forall i(z_i \leq w_i) \wedge \exists k(z_k < w_k) \tag{3.2}$$

We use EGA to solve the problem of Equation (2.7).

3.1 Encoding a Clustering Solution

We established that a solution (an individual) is a random sequence of symbols s from the alphabet $\sum = \{1, 2, 3...k\}$. Every symbol in s represents the cluster to which an object $\vec{x} \in X$ belongs. The length of s is given by the cardinality of X. An individual determines a feasible partition set Π of X. This is illustrated in Figure 3.1.

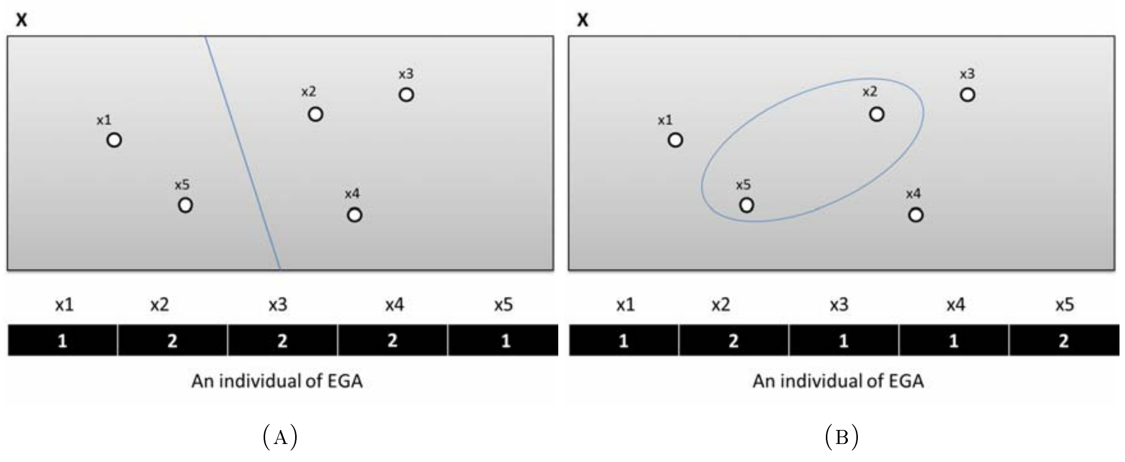


FIGURE 3.1: Feasible partition sets of $X \in \Re^2$ for $k = 2$. (**a**) Π_1 and (**b**) Π_2.

From this encoding, $\Pi_1 = \{C_1 = \{x_1, x_5\}, C_2 = \{x_2, x_3, x_4\}\}$ and $\Pi_2 = \{C_1 = \{x_1, x_3, x_4\}, C_2 = \{x_2, x_5\}\}$. EGA generates a population of feasible partition sets and evaluates them in accordance with their dominance. Evolution of the population takes place after the repeated application of the genetic operators, as described in [4, 5].

3.2 Finding The Probability Distribution of the Elements of a Cluster

Based on the encoding of an individual of EGA, we can determine the elements $\vec{x}$ that belong to C_i for all $i = 1, 2, ..., k$. To illustrate the way $p(\vec{x}|C_1)$ is calculated, we refer to the partition set shown in Figure 3.2.

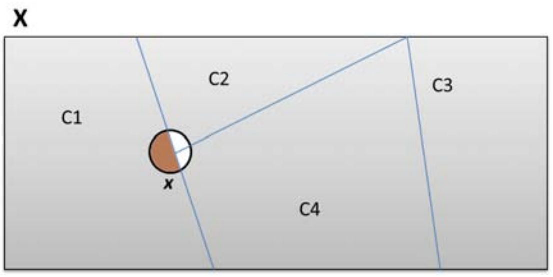


FIGURE 3.2: Representation of $p(\vec{x}|C_1)$.

The shaded area represents the proportion of $\vec{x}$, which belongs to cluster C_1. We divide the probability space of cluster C_1 into a fixed number of quantiles (see Figure 3.3). The conditional proportion of $\vec{x}$ in C_i is the proportion of the number of objects $\vec{x}$ in a given quantile.

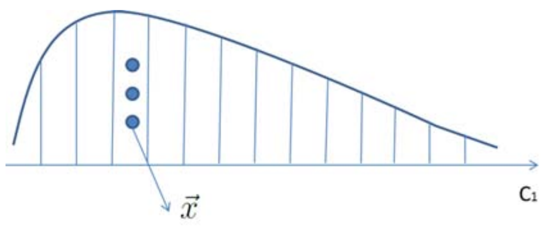


FIGURE 3.3: Example of the probability space of C_i in $\Re$.

An unidimensional case is illustrated in Figure 3.3. This idea may be extended to a multidimensional space, in which case, a quantile is a multidimensional partition of the space of C_i, as is shown in Figure 3.4. In general, the $p(\vec{x}|C_i)$ is the density of the quantile to which $\vec{x}$ belongs in terms of the percentage of data contained in it. We want to obtain quantiles that contain at most 0.0001 percent of the elements. Thus, we divide the space of C_i into 10,000 quantiles.

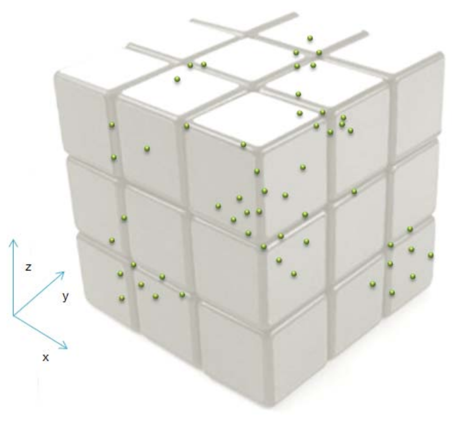


FIGURE 3.4: Example of the probability space of C_i in $\Re^3$.

3.3 Determining the Parameters of CBE

The EGA in CBE was executed with the following parameter values: $Pc = 0.90$, $Pm = 0.09$, $Population\ size = 100$, $Generations = 800$. It is based on a preliminary study, which showed that from a statistical view point, EGA converges to the optimal solution around such values when the problems are demanding (those with a non-convex feasible space) [4, 5]. The value of k in all experiments is known *a priori* from the dataset.

Chapter 4

Datasets

We present a general description of the datasets used in our work. They comprise three categories: (1) synthetic Gaussian datasets; (2) synthetic non-Gaussian datasets; and (3) experimental datasets taken from the UCI database repository, *i.e.*, real-world datasets. Important details about these categories are shown in Appendix B.

4.1 Synthetic Dataset

Supervised clustering is, in general, more effective than an unsupervised one. Knowing this, we make an explicit comparison between a supervised method and our own clustering algorithm. We will show that the performance of our method is comparable to the one of a supervised method. This fact underlines the effectiveness of the proposed clustering algorithm. It is known that, given a dataset X divided into k classes whose elements are drawn from a normal distribution, a BC achieves the best solution relative to other classifiers [69]. Therefore, we will gauge the performance of CBE relative to BC.

Without the normality assumption, we also want to measure the performance of CBE relative to a suitable classifier; in this regard, we use MLP, which has been shown to be effective without making assumptions about the dataset.

Our claim is that if CBE performs satisfactorily when it is compared against a supervised method, it will also perform reasonably well relative to other clustering (non-supervised) methods. In order to prove this, we generated both Gaussian and non-Gaussian synthetic datasets, as described in Appendix B. For each dataset, the class labels were known. They are meant to represent the expected clustering of the dataset. It is very important to stress that CBE is unsupervised, and hence, it does not require the set of class labels.

4.2 Real-World Datasets

Likewise, we also used a suite of datasets that represent “real-world” problems from the UCI repository, also described in Appendix B, Section B.3. We used an MLP as practical approximation to the best expected classification of these datasets.

Chapter 5

Methodology to Gauge the Effectiveness of a Clustering Method

In what follows, we present the methodology to determine the effectiveness of any clustering method (CM). We solve a large set of clustering problems by generating a sufficiently large supply of synthetic datasets. First, the datasets are made to distribute normally. Given the fact that BC will classify optimally when faced with such datasets, we will use them to compare a CM *vs.* BC. Afterwards, other datasets are generated, but no longer demanded to be Gaussian. We will use these last datasets to compare a CM *vs.* MLP. In both cases, we are testing the large number of classification problems to ensure that the behavior of a CM, relative to the best supervised algorithm (BC or MLP), will be statistically significant.

With the real-world datasets, the effectiveness of a CM is given by the number matching between cluster labels obtained by such a CM and the *a priori* class labels of the dataset.

5.1 Determining the Effectiveness Using Synthetic Gaussian Datasets

Given a labeled Gaussian dataset X_i^* $(i = 1, 2, ...n)$, we use BC to classify X_i^*. The same set not including the labels will be denoted X_i. The classification yields Y_i^*, which will be used as the benchmarking reference. We may determine the effectiveness of any CM as follows:

1. Obtain a sample X_i^*.
2. Train the BC based on X_i^*.

3. From the trained BC, obtain the "reference" labels for all $\vec{x^*} \in X_i^*$ (denoted by Y_i^*).
4. Train the CM with X_i to obtain a clustering solution denoted as Y_i.
5. Obtain the percentage of the number of elements of Y_i that match those of Y_i^*.

5.2 Determining the Effectiveness of Using Synthetic Non-Gaussian Datasets

Given a labeled non-Gaussian dataset X_i^* for $i = 1, 2, ...N$, we followed the same exact steps as described in Section 5.1, but we replaced the BC with an MLP. We are assuming (as discussed in [69]) that MLPs are the best practical choice as a classifier when data are not Gaussian.

5.3 Determining the Effectiveness Using Real-World Datasets

With the real-world datasets, the effectiveness of a CM is given by the percentage matching between cluster labels obtained by such a CM and the *a priori* class labels of the dataset.

5.4 Determining the Statistical Significance of the Effectiveness

We statistically evaluated the effectiveness of any CM using synthetic datasets (both Gaussian and non-Gaussian). In every case, we applied a CM to obtain Y_i. We denote the relative effectiveness as φ. We wrote a computer program that takes the following steps:

(1) A set of $N = 36$ datasets is randomly selected.

(2) Every one of these datasets is used to train a CM and BC or MLP, when data are Gaussian or non-Gaussian, respectively.

(3) φ_i is recorded for each problem.

(4) The average of all φ_i is calculated. Such an average is denoted as $\bar{\varphi}_m$.

(5) Steps 1–4 are repeated, until the $\bar{\varphi}_m$ are normally distributed. The mean and standard deviation of the resulting normal distribution are denoted by μ' and σ', respectively.

From the central limit theorem, $\bar{\varphi}_m$ will be normally distributed for appropriate M. Experimentally (see Appendix C), we found that an adequate value of M is given by:

$$M \approx \frac{b - \ln[\ln(\frac{a}{P})]}{c} \tag{5.1}$$

where P is the probability that the experimental χ^2 is less than or equal to 3.28, and there are five or more observations per quantile; $a = 0.046213555$, $b = 12.40231200$ and $c = 0.195221110$. For $p \leq 0.05$, from 5.1, we have that $M \geq 85$. In other words, if $M \geq 85$, the probability that $\bar{\varphi}$ is normally distributed is better that 0.95. Ensuring that $\bar{\varphi} \sim N(\mu', \sigma')$, we can easily infer the mean μ and the standard deviation σ of the distribution of φ from:

$$\sigma = \sigma' \sqrt{N} \tag{5.2}$$

$$\mu = \mu' \tag{5.3}$$

From Chebyshev's inequality [78], the probability that the performance of a CM φ lies in the interval $[\mu - \lambda\sigma, \mu + \lambda\sigma]$ is given by:

$$p(\mu - \lambda\sigma \leq \varphi \leq \mu + \lambda\sigma) \geq 1 - \frac{1}{\lambda^2} \tag{5.4}$$

where λ denotes the number of standard deviations. By setting $\lambda = 3.1623$, we ensure that the values of φ will lie in this interval with probability $p \approx 0.9$. A lower bound on φ (assuming the symmetric distribution of the φs) is given by $\mu + \lambda\sigma$.

Chapter 6

Results

In what follows, we show the results from the experimental procedure. We wanted to explore the behavior of our method relative to other clustering methods. We used a method based on information theory called ENCLUS and two other methods: k-means and fuzzy c-means. For all methods, we used the experimental methodology (with Gaussian and non-Gaussian datasets). All methods were made to solve the same problems.

6.1 Synthetic Gaussian Datasets

Table 6.1 shows the values of φ for Gaussian datasets. Lower values imply better closeness to the results achieved from BC (see Section 5.1). The Gaussian datasets were generated in three different arrangements: disjoint, overlapping and concentric. Figure 6.1 shows an example of such arrangements.

All four methods yield similar values for disjoint clusters (simple problem). However, important differences are found when tackling the overlapping and concentric datasets (hard problems). We can see that CBE is noticeable better.

Since BC exhibits the best theoretical effectiveness given Gaussian datasets, we have included the percentual behavior of the four methods relative to it in Table 6.2.

Algorithm	Disjoint	Overlapping	Concentric
k-means	0.018	0.42	0.52
fuzzy c-means	0.017	0.36	0.51
ENCLUS	0.019	0.04	0.12
CBE	0.020	0.03	0.04
p-*value* < 0.05			

TABLE 6.1: Average effectiveness (φ) for Gaussian datasets.

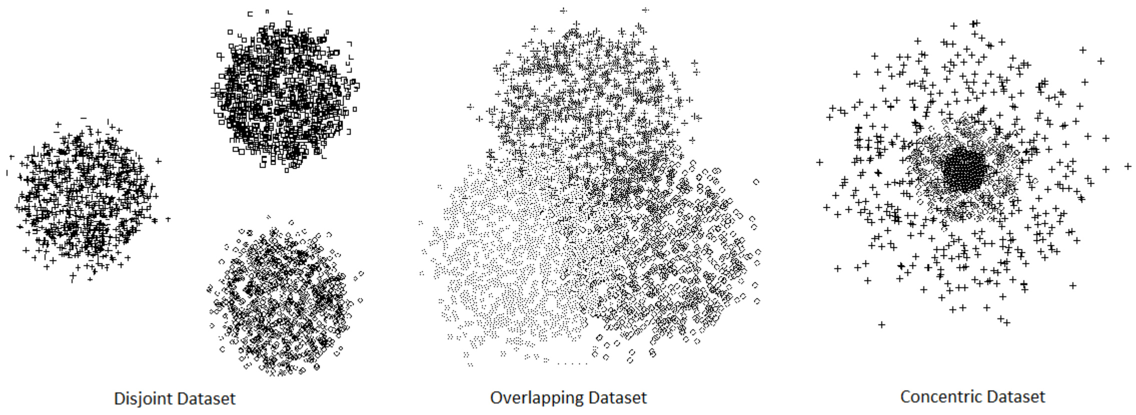


FIGURE 6.1: Example of possible arrangements of a Gaussian dataset.

Algorithm	Disjoint	Overlapping	Concentric
k-means	0.982	0.576	0.475
fuzzy c-means	0.983	0.636	0.485
ENCLUS	0.981	0.960	0.879
CBE	0.980	0.970	0.960
BC	1.000	1.000	1.000
p-*value* < 0.05			

TABLE 6.2: Performance relative to the Bayes classifier (BC) (%).

6.2 Synthetic Non-Gaussian Datasets

Table 6.3 shows the values of φ for non-Gaussian datasets. These do not have a particular arrangement (disjoint, overlapping and concentric). As before, lower values imply better closeness to the results achieved from MLP (see Section 5.2).

The values of CBE and ENCLUS are much better than the ones of traditional clustering methods. Nevertheless, when compared to ENCLUS, CBE is 62.5% better.

Algorithm	$\mu(\varphi)$
k-means	0.56
fuzzy c-means	0.54
ENCLUS	0.13
CBE	0.08
p-value < 0.05	

TABLE 6.3: Average effectiveness (φ) for non-Gaussian datasets. ENCLUS, entropy clustering; CBE, clustering-based on entropy.

Since MLP is our reference for non-Gaussian datasets, we have included the percentual behavior of the four methods relative to it in Table 6.4.

Algorithm	**Percentual Performance**
k-means	0.440
fuzzy c-means	0.460
ENCLUS	0.870
CBE	0.920
MLP	1.000
p-value < 0.05	

TABLE 6.4: Performance relative to MLP (%).

6.3 "Real-World" Datasets

Based on Section 5.3, we calculate the effectiveness for the same set of CMs. Here, we used MLP as a practical reference of the best expected classification. The results are shown in Table 6.5. Contrary to previous results, a greater value represents a higher effectiveness.

Set	**CBE**	**ENCLUS**	***k*-Means**	**Fuzzy *c*-Means**	**MLP**
Abalone	0.596	0.563	0.496	0.511	0.690
Cars	0.917	0.896	0.696	0.712	0.997
Census	0.855	0.812	0.774	0.786	1.000
Hepatitis	0.859	0.846	0.782	0.811	0.987
Yeast	0.545	0.496	0.470	0.486	0.750
Average	0.754	0.723	0.644	0.661	0.885

TABLE 6.5: Effectiveness of CBE and other clustering methods for "Real-World" Datasets

As expected, the best value was achieved by MLP. The relative performance is shown in Table 6.6.

Method	**Relative Effectiveness**
k-means	0.728
Fuzzy *c*-means	0.747
ENCLUS	0.817
CBE	0.853
MLP	1.000

TABLE 6.6: Performance relative to MLP.

Chapter 7

Conclusions

A new unsupervised classifier system (CBE) has been defined based on the entropy as a quality index (QI) in order to pinpoint those elements in the dataset that, collectively, optimize such an index. The effectiveness of CBE has been tested by solving a large number of synthetic problems. Since there is a large number of possible combinations of the elements in the space of the clusters, an eclectic genetic algorithm is used to iteratively find the assignments in a way that increases the intra- and inter-cluster entropy simultaneously. This algorithm is run for a fixed number of iterations and yields the best possible combination of elements given a preset number of clusters k. That the GA will eventually reach the global optimum is guaranteed by the fact that it is elitist. That it will approach the global optimum satisfactorily in a relatively small number of iterations (800) is guaranteed by extensive tests performed elsewhere (see [4, 5]).

We found that when compared to BC's performance over Gaussian distributed datasets, CBE and BC have, practically, indistinguishable success ratios, thus proving that CBE is comparable to the best theoretical option. Here, we, again, stress that BC corresponds to supervised learning, whereas CBE does not. The advantage is evident. When compared to BC's performance over non-Gaussian sets, CBE, as expected, displayed a much better success ratio. The conclusions above have been reached for a statistical p-value of 0.05. In other words, the probability of such results to persist on datasets outside our study is better than 0.95, thus ensuring the reliability of CBE.

The performance value φ was calculated by: (1) providing a method to produce an unlimited supply of data; (2) This method is able to yield Gaussian and non-Gaussian distributed data; (3) Batches of $N = 36$ datasets were clustered; (4) For every one of the N Gaussian sets, the difference between our algorithm's classification and BC's classification (φ) was calculated; (5) For every batch, $\overline{\varphi}$ was recorded; (6) The process described in Steps 3, 4 and 5 was repeated until the $\overline{\varphi}$ distributed normally; (7) Once the

$\overline{\varphi}$ are normal, we know the mean and standard deviation of the means; (8) Given these, we may infer the mean and standard deviation of the original pdf of the φs; (9) From Chebyshev's theorem, we may obtain the upper bound of φ with probability 0.95. From this methodology, we may establish a worst case upper bound on the behavior of all of the analyzed algorithms. In other words, our conclusions are applicable to any clustering problem (even those outside of this study) with a probability better than 0.95 .

For completeness, we also tested the effectiveness of CBE by solving a set of "real world" problems. We decided to use an MLP as the comparison criterion to measure the performance of CBE based on the nearness of its effectiveness with regard to it. Of all of the unsupervised algorithms that were evaluated, CBE achieved the best performance.

Here, two issues should be underlined: (1) Whereas BC and MLP are supervised, CBE is not. The distinct superiority of one method over the others, in this context, is evident; (2) As opposed to BC, CBE was designed not to need the previous calculation of the conditional probabilities of BC; a marked advantage of CBE over BC. It is important to mention that our experiments include tight tests comparing the behavior of other clustering methods. We compared the behavior of k-means, fuzzy c-means and ENCLUS against BC and MLP. The results of the comparison showed that CBE outperforms them (with 95% reliability).

For disjoint sets, which offer no major difficulty, all four methods (k-means, fuzzy c-means, ENCLUS and CBE) perform very well relative to BC. However, when tackling partially overlapping and concentric datasets, the differences are remarkable. The information-based methods (CBE and ENCLUS) showed the best results; nevertheless, CBE was better than ENCLUS. For non-Gaussian datasets, the values of CBE and ENCLUS also outperform the other methods. When CBE is compared to ENCLUS, CBE is 62.5% better.

In conclusion, CBE is a non-supervised, highly efficient, universal (in the sense that it may be easily utilized with other quality indices) clustering technique whose effectiveness has been proven by tackling a very large number of problems (1,530,000 combined instances). It has been, in practice, used to solve complex clustering problems that other methods were not able to solve.

Appendix A

Eclectic Genetic Algorithm

For those familiar with the methodology of genetic algorithms, it should come as no surprise that a number of questions relative to the best operation of the algorithm immediately arise. The simple genetic algorithm [52] frequently mentioned in the literature leaves open the optimal values of, at least, the following parameters:

(1) Probability of crossover (P_c).

(2) Probability of mutation (P_m).

(3) Population size.

Premature and/or slow convergence are also of prime importance. For this, the EGA includes the following characteristics:

(1) The best (overall) n individuals are considered. The best and worst individuals $(1-n)$ are selected; then, the second best and next-to-the-worst individuals $(2-[n-1])$ are selected, *etc.*

(2) Crossover is performed with a probability P_c. Annular crossover makes this operation position independent. Annular crossover allows for unbiased building block search, a central feature to GA's strength. Two randomly selected individuals are represented as two rings (the parent individuals). Semi-rings of equal size are selected and interchanged to yield a set of offspring. Each parent contributes the same amount of information to their descendants.

(3) Mutation is performed with probability P_m. Mutation is uniform and, thus, is kept at very low levels. For efficiency purposes, we do not work with mutation probabilities for every independent bit. Rather, we work with the expected number of mutations, which, statistically is equivalent to calculating mutation probabilities for every bit.

Hence, the expected number of mutations is calculated from $\ell * n * p_m$, where ℓ is the length of the genome in bits and n is the number of individuals in the population.

In what follows, we present the pseudo-code of EGA:

Algorithm 1: Eclectic genetic algorithm.

```
Data:
$n$ = Number of individuals
$p_c$ = Crossover probability
$p_m$ = Mutation probability
$\ell$ = Length of the Individual
$b2m = \ell * n * p_m$ number of bits to mutate
Result: The top $n$ individuals
Generate a population $P$ of size $n$ whose $n\ell$ bits are randomly set:
initialize($P$);
evaluate($P$);
Sort individuals from best to worst based on their fitness:
sort($P$);
while convergence criteria are not met do
    duplicate($P$);
    for $i = 1$ to $n$ do
        Generate a random number $R$;
        if $R > p_c$ then
            Generate a random integer locus $\in [1, \ell]$;
            Interchange the semi-ring starting at locus for individuals $i$ and $2n - i + 1$:
            crossover($P(i)$, $P(2n - i + 1)$);
        end
    end
    Mutate the population in $b2m$ randomly-selected bits:
    mutate($P$);
    Evaluate the population again:
    evaluate($P$);
    Sort individuals from best to worst based on their fitness:
    sort($P$);
    Eliminate the worst $n$ individuals from $P$
    remove($P$);
end
Return $P$
```

A.1 Performance of EGA

To choose EGA as the best optimization method to solve our problem, we rest on the conclusions of our previous analysis regarding the performance of a set of algorithms (which included evolutionary and non-evolutionary algorithms) [4, 5]. In the following subsections we show the set of algorithms analyzed and what we consider the most important results. The reader can find details of this results in Appendix G

A.1.1 Set of Algorithms

We selected a set A of 4 structurally different GAs and a non-evolutionary algorithm (NEA) in order to solve, in principle, an unlimited supply of functions in $\Re \times \Re$ systematically generated. From the results obtained, a ranking of the algorithms in A was determined. For completeness, an extended set of functions in $\Re \times \Re$, $\Re \times \Re^2$ and $\Re \times \Re^3$ was generated and subsequently solved. A similar behavior of A (within statistical limits) was found in all cases. This fact allowed to hypothesize about the behavior of A for functions in $\Re \times \Re^n$. As supplement, a suite of problems was selected which includes hard unconstrained problems (which traditionally have been used for benchmarking purposes) [94] and constrained problems [95]. Lastly, atypical GA-hard functions were analyzed [89, 90]. The set A included the following GAs:

1. An elitist canonical GA (in what follows referred to as TGA [eliTist GA]) [3].
2. A Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation algorithm (CHC algorithm) [91].
3. An Eclectic Genetic Algorithm (EGA) [92].
4. A Statistical GA (SGA) [93, 96].

The set A also includes a non-evolutionary algorithm called RMH algorithm[88].

A.1.2 Global Performance

The results were grouped into three categories: 1) Unbiased Problems (set of functions systematically generated), 2) Selected Suite Problems (Unconstrained and Constrained Functions) and 3) Atypical Problems. Table A.1 shows the relative global performance of such functions. The best algorithm in the table is EGA

Table A.1: Global Performance of all algorithms for different categories of problems

A_i	Unbiased	Suite	Atypical	Global Performance	Relative
EGA	9.64	8.00	4.48	7.37	100.00%
RMH	6.24	0.012	2.04	2.76	37.49%
TGA	1.35	1.16	4.77	2.43	32.91%
SGA	1.33	0.036	3.33	1.57	21.23%
CHC	2.12	0.08	2.10	1.43	19.44%

Appendix B

Datasets

B.1 *Synthetic Gaussian Datasets*

To generate a Gaussian element $\vec{x} = [x_1, x_2, ..., x_n]$, we use the acceptance-rejection method [79, 80]. In this method, a uniformly distributed random point $(x_1, x_2, ..., x_n, , y)$ is generated and accepted iff $y < f(x_1, x_2, ..., x_n)$ where f is the Gaussian pdf. For instance, on the assumption of $\vec{x} \in \Re^2$, in Figure B.1, we show different Gaussian sets obtained by applying this method.

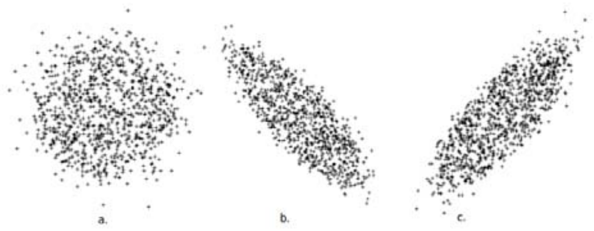
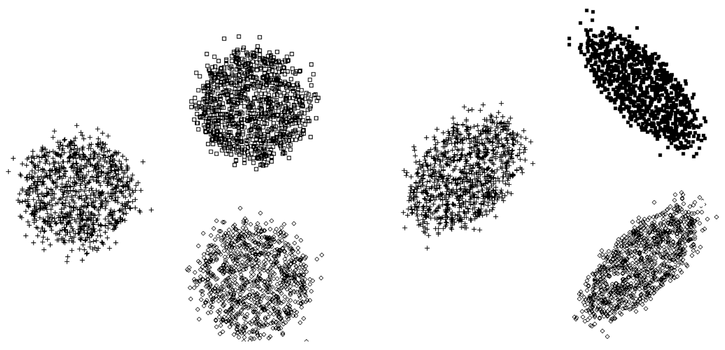


FIGURE B.1: Gaussian sets in $\Re^2$.

In the context of some of our experiments, X is a set of k Gaussian sets in $\Re^n$. We wanted to test the effectiveness of our method with datasets that satisfy the following definitions:

Definition 2. Given n and k, a dataset X is a disjoint set if $\forall C_i \subset X$, it holds that $C_i \cap C_j = \phi \; \forall i, j = 1, 2, ..k$ and $i \neq j$.

In Figure B.2, we illustrate two possible examples of such datasets for $n = 2$ and $k = 3$. The value ρ is the correlation coefficient.

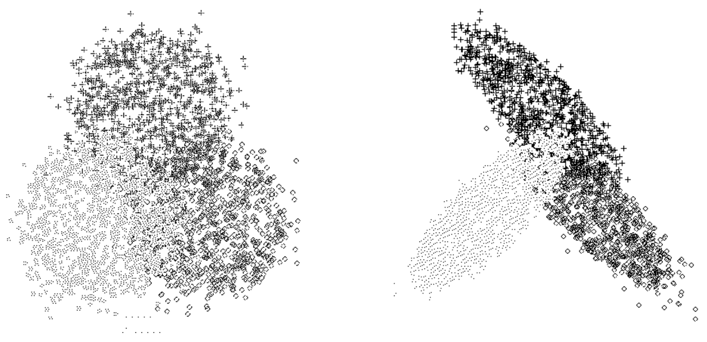


(A) (B)

FIGURE B.2: Disjoint Gaussian sets for $n = 2$ and $k = 3$. (**a**) $\rho = 0$ and (**b**) $\rho = 0.4$, $\rho = -0.7$, $\rho = 0.7$.

Definition 3. Given n and k, a dataset X is an overlapping set if $\exists C_i, C_j \subset X$, such that $C_i \cap C_j \neq \phi$ and $C_i \nsubseteq C_j$ $\forall i, j = 1, 2, ..k$ and $i \neq j$.

In Figure B.3 we illustrate two possible examples of such datasets for $n = 2$ and $k = 3$.

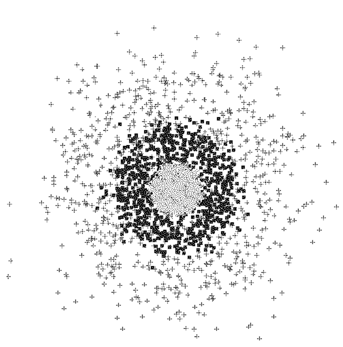


(A) (B)

FIGURE B.3: Overlapping Gaussian sets for $n = 2$ and $k = 3$. (**a**) $\rho = 0$ and (**b**) $\rho = 0.8$, $\rho = -0.8$.

Definition 4. Given n and k, a dataset X is a concentric set if $\forall C_i \subseteq X$, its mean value, denoted by μ_{C_i}, is equal to $\mu_{C_j} \forall i, j = 1, 2, ..k$.

In Figure B.4 we illustrate two possible examples of such datasets for $n = 2$ and $k = 3$.


(A)

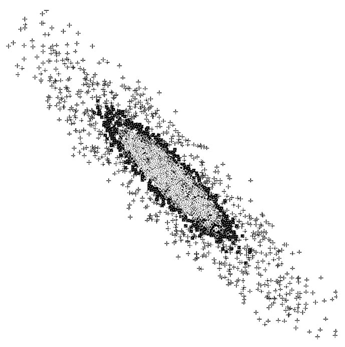

(B)

FIGURE B.4: Concentric Gaussian sets for $n = 2$ and $k = 3$. (**a**) $\mu_{C_i} = (0.5, 0.5)$ $\forall i$ and $\rho = 0$ and (**b**) $\mu_{C_i} = (0.5, 0.5)$ $\forall i$ and $\rho = -0.9$.

We generated 500 datasets for each type (1500 total), randomly choosing the values of n and k, $n \sim U[2-20]$ and $k \sim U[2-15]$. In Table B.1, we show the parameters of this process.

Parameter	**Value**
Type of dataset	Disjoint, Overlapping, Concentric
Clusters per dataset (k)	$U \sim [2-15]$
Dimensionality (n)	$U \sim [2-20]$
Cardinality of cluster ($\|C\|$)	1000 elements
Maximum size of a dataset	15,000 elements
Dataset per type	500
Total number of problems	1500

TABLE B.1: Parameters of the generation process of datasets.

B.2 *Synthetic Non-Gaussian Datasets*

To generate Non-Gaussian patterns in $\Re^n$, we resort to polynomial functions of the form:

$$f(x_1, x_2, ..., x_n) = a_{m1}x_1^m + ... + a_{mn}x_n^m + ... + a_{11}x_1 + a_{1n}x_n \tag{B.1}$$

Given that such functions have larger degrees of freedom, we can generate many points uniformly distributed in $\Re^n$. As reported in the previous section, we wrote a computer program that allows us to obtain a set of 500 different problems. These problems were generated for random values of n and k ($U \sim [2-20]$ and $U \sim [2-15]$, respectively) with $|C_i| = 200$.

B.3 *"Real World" Datasets*

In order to illustrate the performance of our method when faced with “real world” problems, we selected five datasets (Abalone [81], Cars [82], Census Income[83], Hepatitis [84] and Yeast [85]) from the UCI Machine Learning repository, whose properties are shown in Table B.2.

Problem’s Name	**Number of Variables**	**Number of Classes**	**Sample Size**	**Missing Values**
Abalone	8	4	3133	no
Cars	22	4	1728	no
Census Income	14	2	32,561	yes
Hepatitis	20	2	155	yes
Yeast	10	8	1486	no

Table B.2: Properties of the selected datasets.

We chose datasets that represent classification problems where the class labels for each object are known. Then, we can determine the performance of any clustering method as an effectiveness percentage. The selection criteria of these datasets were based on the following features:

- Multidimensionality.
- Cardinality.
- Complexity (non-linearly separable problems).
- Categorical data.
- Data with missing values.

Some of these features involve preprocessing tasks to guarantee the quality of a dataset. We applied the following preprocessing techniques:

(1) Categorical variables were encoded using dummy binary variables [86].

(2) We scaled the dataset into $[0, 1)$.

(3) In order to complete missing information, we interpolate the unknown values with natural splines (known to minimize the curvature of the approximant) [87].

Appendix C

Ensuring Normality in an Experimental Distribution

We wish to experimentally find the parameters of the unknown probability density function (pdf) of a finite, but unbounded, dataset X. To do this, we must determine the minimum number of elements M that we need to sample to ensure that our experimental approximation is correct with a desired probability P. We start by observing that a mean value is obtained from X by averaging N randomly-selected values of the x_i, thus: $\bar{x}_i = \frac{1}{N}\sum x_i$. We know that any sampling distribution of the $\bar{x}_i$ (sdm) will be normal with parameters $\mu_{\bar{x}}$ and $\sigma_{\bar{x}}$ when $N \to \infty$. In practice, it is customary to consider that a good approximation will be achieved when $N > 20$; hence, we decided to make $N = 36$. Now, all we need to determine the value of M is to find sufficient $\bar{x}_i$'s $(i = 1, 2, \ldots, M)$ until they distribute normally. Once this occurs, we immediately have the parameters $\mu_{\bar{x}}$ and $\sigma_{\bar{x}}$. The parameters of the unknown pdf are then easily calculated as $\mu = \mu_{\bar{x}}$ and $\sigma = \sqrt{n}\sigma_{\bar{x}}$. To determine that normality has been reached, we followed the following strategy. We used a setting similar to the one used in the χ^2 test in which: (1) We defined ten categories d_i (deciles) and the corresponding 11 limiting values (v_i) assuming a normal distribution, such that one tenth of the observations are expected per decile: $\int_{v_i}^{v_i+1} N(\mu, \sigma) \approx 0.1$, $i = 0, 1, \ldots, 9$. For this to be true, we make, $v_0 = -5.000$, $v_1 = -1.285$, $v_2 = -0.845$, $v_3 = -0.530$, $v_4 = -0.255$, $v_5 = 0.000$ and the positive symmetrical values; (2) We calculated $\chi^2 = \sum(o_i - e_i)^2/e_i$ for every x_i, where o_i is the observed number of events in the i-th decile and e_i is the expected number of such events. For the k-th observed event, clearly, the number of expected events per decile is $k/10$; (3) We further require, as is usual, that there be at least $o_{min} = 5$ events per decile. Small values of the calculated χ^2 indicate that there is a larger similarity between the hypothesized and the true pdfs. In this case, a small χ^2 means that the observed behavior of the $\bar{x}_i$'s is closer to a normal distribution. The question we want to answer is: How small should χ^2 be in order for us to ascertain normality? Remember the test χ^2 is designed to verify whether two distributions differ significantly, so that one may reject the null hypothesis, *i.e.*, the two populations are not statistically equivalent. This happens for large values of χ^2 and is a function of the degrees of freedom (*df*). In our case, *df* = 7. Therefore, if we wanted to be 95% certain that the observed $\bar{x}_i$'s were not normally distributed, we would demand that $\chi^2 \geq 14.0671$. However, this case is different. We want to ensure the likelihood that the observed behavior of the $\bar{x}_i$'s is normal. In order to do this, we performed a Monte Carlo experiment along the following lines. We set a desired probability P that the $\bar{x}_i$'s are normal.

We establish a best desired value of χ^2, which we will call χ_{best}. We make the number of elements in the sample $NS = 50$. We then generate NS instances of $N(0,1)$ and count the number of times the value of the instance is in every decile. We calculate the value of the corresponding χ^2 and store it. We thusly calculate $100,000$ combinations of size NS. Out of these combinations, we count those for which $\chi^2 \leq \chi_{best}$ and there are at least o_{min} observations per decile. This number divided by $100,000$, which we shall call p, is the experimental probability that, for NS observations, the sample "performs" as required. We repeat this process increasing NS up to 100. In every instance, we test whether $p > P$. If such is the case, we decrement the value of χ_{best} and re-start the process. Only when $p \leq P$, does the process end.

In Figure C.1, we show the graph of the size of sample M vs. the experimental probability p that $p \leq P$ and $o_i \leq o_{min}$ for χ_{best} (from the Monte Carlo experiment).

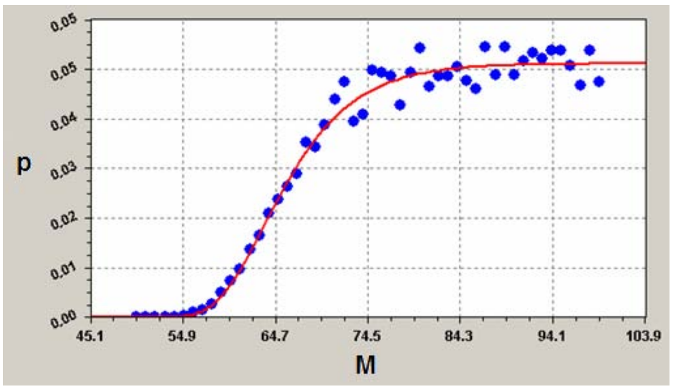


FIGURE C.1: Size of sample M vs. p

Every point represents the proportion of combinations satisfying the required conditions per 100,000 trials. For this experiment, $\chi_{best} = 3.28$. We obtained an approximation to a Gompertz model with $S = 0.0023$ and $r = 0.9926$. It has the form $p = ae^{-e^{b-cM}}$; where $a = 0.046213555$, $b = 12.40231200$, $c = 0.195221110$. From this expression, we solve for M, to get:

$$M \approx \frac{b - \ln[\ln(\frac{a}{P})]}{c} \tag{C.1}$$

As may be observed, $p < 0.05$ for $M \geq 85$, which says that the probability of obtaining $\chi^2 \leq 3.28$ by chance alone is less than five in one hundred. Therefore, its is enough to obtain 85 or more $\bar{x}_i$'s (3060 x_i's) to calculate μ and σ and be 95% sure that the values of these parameters will be correct.

Appendix D

Clustering with an N-Dimensional Extension of Gielis Superformula

Clustering with an N-Dimensional Extension of Gielis Superformula

ANGEL KURI MORALES,EDWIN ALDANA BOBADILLA
Department of Computation
Instituto Tecnológico Autónomo de México
Río Hondo No. 1
MEXICO
akuri@itam.mx, ealdana@mcc.unam.mx

Abstract:- Clustering is the task which allows us to identify groups, distributions or patterns over a set of data. To achieve it we must, first, adopt a measure of likelihood which allows us to group elements of similar characteristics into two or more such clusters. Presently there are several clustering algorithms which appeal to various metrics among which there outstand the ones based on Minkowski's and Mahalanobis' distances. In general this approach yields hyperspherical forms which restrict the fact that some clusters may be represented by irregular N-dimensional bodies. In this paper we use a formula originally developed by Johan Gielis which has been called the "superformula". It allows the generation of N-dimensional bodies of arbitrary shape by modifying certain parameters. This approach allows us to represent a cluster as the set of data contained in a given body without resorting to a metric of distance. Therefore, we replace the idea of nearness by one of membership. To determine the more adequate values of the parameters in the superformula we applied a Genetic Algorithm (GA); the so called Vasconcelos' GA.

Key-Words:- Clustering, Minkowsky's Distance, Mahalanobis' Distance, Gielis superformula, Genetic algorithm, Vasconcelos GA.

1 Introduction

The goal of Clustering is to partition a set of data into subsets each of which is called a cluster. In one approach we must begin by determining the optimum number of clusters.
This is not an easy task. In fact, it may be proven that no optimum assignment algorithm exists for arbitrary conditions. Most methodologies require this issue to be solved at the outset. Typical methods such as K-means fall into this category. In [2] and [3], for instance, some alternatives to solve this issue are discussed.
Secondly, it is necessary to determine a measure of likelihood such that it is possible to establish a criterion of membership for any data point. A very popular measure is the proximity or distance between the elements of the data set whenever these data are expressed in like units. The most used metrics are, probably, Minkowski's and Mahalanobis' which are described in sections 2.1 and 2.2.
The disadvantage of using these metrics is that they restrict the form of the clusters to, in general, hyper-spherical shapes in an N-dimensional space.
In figure 1 we show a set of data in a bidimensional space which are to be grouped into three clusters.

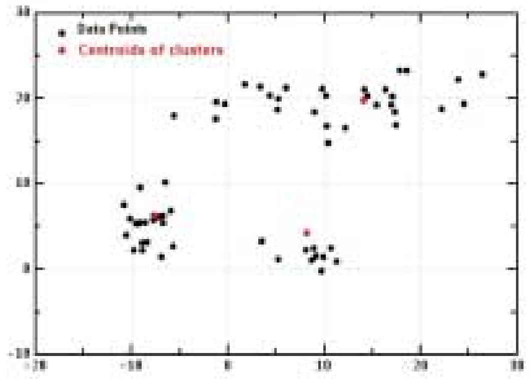


Fig. 1: A bi-dimensional data set

If we use a metric distance we get a clustering such as the one shown in figure 2.

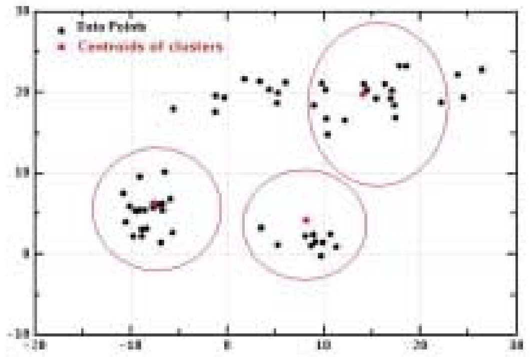


Fig. 2: Clustering with metric distance.

Ideally we would like to be able to find clusters which encompass all data from the original set as shown in figure 3.

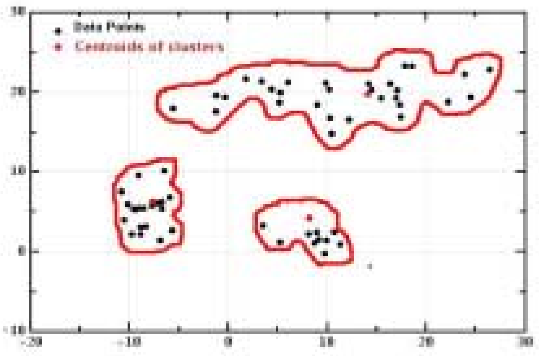


Fig. 3: Idealized clustering

In order to be able to achieve this we propose to use a formula originally developed by Johan Gielis [1], which is the generalization of a hyper-ellipse and incorporates a set of parameters which yield the corresponding degrees of freedom which, when modified, may conform such a hyper-ellipse allowing us to find a virtually unlimited number of forms as described in section 2.2.
This implies that we must optimize the values of the parameters such that the forms of the clusters adapt themselves to accommodate all the elements of the data set. In order to do this we appeal to a non-traditional Genetic Algorithm (GA) called VGA (Vasconcelos' Genetic Algorithm).

2 Generalities

2.1 Minkowski's Distance

Let **x** = (x_1, …, x_n) and **y** = (y_1,…y_n) be vectors in R_n. Then Minkowsky's distance between **x** and **y** is given by:

$$d = \left(\sum_{i=1}^{n} | x_i - y_i |^p \right)^{1/p} \qquad (1)$$

Where *p* is not necessarily an integer but must be larger than 1 to comply with the triangle's inequality. When *p*=2 we have the Euclidean distance.

2.2 Mahalanobis' Distance

Mahalanobis' distance is defined as a measure of likelihood between the vectors **x** = (x_1, …, x_n) and **y** = (y_1,…y_n). It is given by

$$d = \sqrt{\sum_{x=1}^{N} \sum_{y=1}^{N} b_{xy}(x_i - x_j)(y_i - y_j)} \qquad (2)$$

where $a_{\mathbf{xy}}$ is the covariance between **x** and **y**, and is given by

$$a_{\mathbf{xy}} = \frac{1}{K}\sum_{k=1}^{K}(x_k - \bar{x})(y_k - \bar{y}) \qquad (3)$$

$$\bar{x} = \frac{1}{K}\sum_{k=1}^{K} x_k ; \quad \bar{y} = \frac{1}{K}\sum_{k=1}^{K} y_k \qquad (4)$$

and

$b_{\mathbf{xy}}$ is the corresponding element of the inverse of the covariance matrix; K is the number of objects in the data. If the covariance matrix is the identity then Mahalanobis' distance reduces to the Euclidean distance.

2.3 Gielis Formula

Gielis superformula (GSF) generalizes the equation of a hyper-ellipse by introducing some parameters which increase the degrees of freedom of the resulting figures when geometrically interpreted. It is given by

$$r(\phi) = \left[\left| \frac{\cos\left(\frac{m\phi}{4}\right)}{a} \right|^{n_2} + \left| \frac{\sin\left(\frac{m\phi}{4}\right)}{b} \right|^{n_3} \right]^{-\frac{1}{n_1}} \qquad (5)$$

Where *r* is the radius and Φ is the angle. In figure 4 we show some forms generated from *a=b=1* and several values assigned to *m, n_1, n_2, n_3* . It is possible to generalize the formula to 3 or more dimensions. Figure 5 shows some forms obtained for a 3D space.

2.4 Genetic Algorithms

Genetic Algorithms (GA) (a very interesting introduction to genetic algorithms and other evolutionary algorithms may be found in [5]) are optimization algorithms which are frequently cited as "partially simulating the process of natural

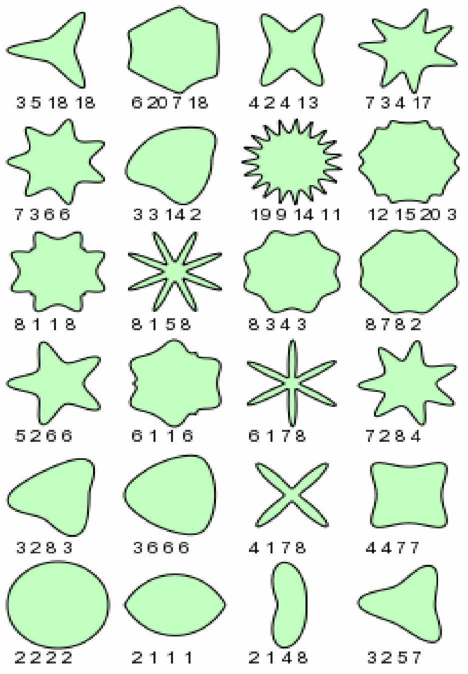


Fig. 4: Some forms generated with the superformula for a=b=1, m, n1, n2, n3

evolution". Although this a suggestive analogy behind which, indeed, lies the original motivation for their inception, it is better to understand them as a kind of algorithms which take advantage of the implicit (indeed, unavoidable) granularity of the search space which is induced by the use of the finite binary representation in a digital computer. In such finite space numbers originally thought of as existing in $\Re^n$ actually map into ***B*** space. Thereafter it is simple to establish that a genetic algorithmic process is a Markov chain (MC) whose states are the populations arising from the so-called genetic *operators*: (typically) selection, crossover and mutation. As such they display all of the properties of a MC. Therefore one may conclude the following mathematical properties of a GA: 1) The results of the evolutionary process are independent of the initial population; 2) A GA preserving the best individual arising during the process will converge to the global optimum (albeit the convergence process is not bounded in time).

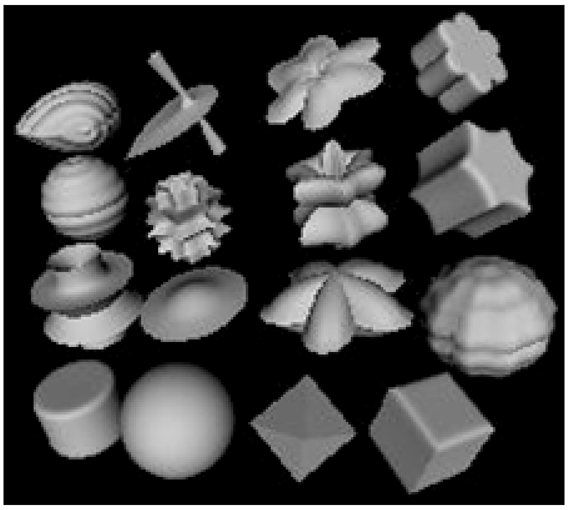


Fig. 5: Some forms gotten from the extension of Gielis formula to 3D

For a proof of these facts the interested reader may see [6]. Their most outstanding feature is that, as opposed to other more traditional optimization techniques, the GA iterates simultaneously over *several* possible solutions to the problem. Thereafter, other plausible solutions are obtained by combining (*crossing over*) the *codes* of these solutions to obtain hopefully better ones. The solution space (SS) is, therefore, traversed stochastically searching for increasingly better plausible solutions. In order to guarantee that the SS will be globally considered some bits are randomly selected and changed (a process called *mutation*). The main concern of GA-practitioners (given the fact that the GAs, in general, will find the best solution) is to make the convergence as efficient as possible. The work of Forrest et al. (see [7]) has determined the characteristics of the so-called *Idealized GA* (IGA) which is impervious to GA-hard problems (see [8]).

2.4.1 Vasconcelos' Genetic Algorithm

Clearly the implementation of the IGA is unattainable. However, a practical approximation called the Vasconcelos' GA (VGA) has been repeatedly tested and proven to be highly efficient [9]. The VGA, therefore, turns out to be an optimization algorithm of broad scope of application and demonstrably high efficiency.

2 The problem

Considering that the use of metric distances limits the forms of the clusters to hyper-spherical forms in N-space, we use GSF in order to find (possibly) highly irregular forms which represent in a more general way such clusters. This implies that we abandon the idea that a cluster is a set of vectors which are *close* to each other (in some sense) by the one of a cluster as the set of vectors which lie *within* an N-dimensional object (NDO). In this case we have chosen to select the cluster forms from the family of NDOs arising from the N-dimensional generalization of GSF. In order to implement this idea we must optimize the parameters in GSF. The idea is to work in reverse and find the parameters which are to be put in GSF such that all data vectors may be found within the NDOs. To achieve this we must define K NDOs, where K is the number of clusters. Every NDO must have a precise definition in terms of its position and its geometric characteristics. Of course the problem resides in finding the appropriate combination of values which maximizes the membership of all vectors into all of the clusters and other criteria, as will be discussed in the sequel. This, of course, is a complex problem of optimization that we have tackled with VGA. One crucial issue is the precise formulation of the fitness function for VGA.

3 Solution

The solution we describe assumes that the number of clusters is already known. To illustrate our method we consistently assumed, without loss of generality, that there are 4 clusters and we are working on 3D.
Firstly, we generate 4 NDOs (one per cluster)

as shown in figures 6 and 7.

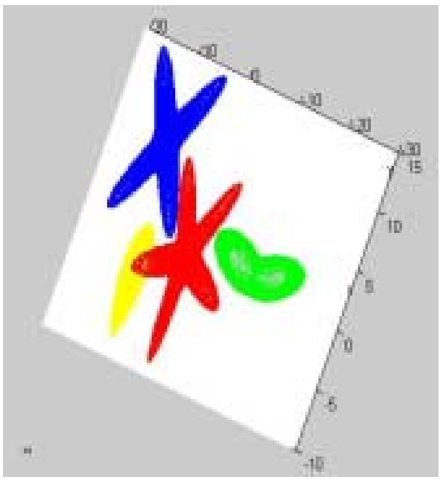


Fig. 6. NDOs projected into (x,y)

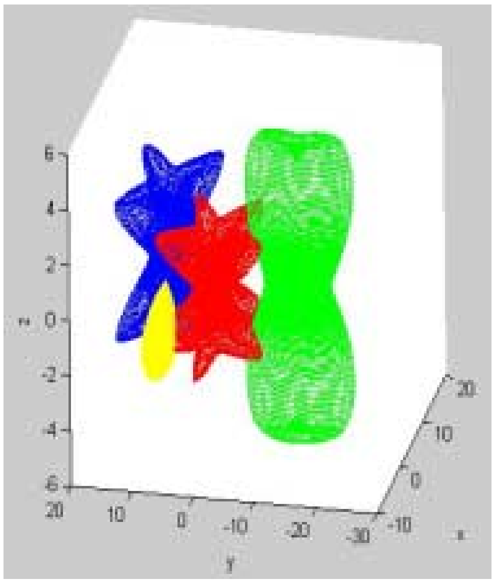


Fig. 7. NDOs represented in (x,y,z)

For every cluster we calculated 2,500 coordinates. Therefore we got a data set (3D coordinates) of size 10,000.
Hence, there are 10,000 elements that we know, a priori, belong to the 4 clusters defined by the NDOs in figures 6 and 7. The goal was to use this set as an input for VGA in a way such that we may verify that the values of the parameters in GSF correspond to those of figures 6 and 7.

3.1 Encoding the problem for the Genetic Algorithm

The variables to optimize are *m*, n_1, n_2, n_3, *a*, *b* for each of the clusters. Therefore, the encoding of one cluster for VGA is shown in figure 8.

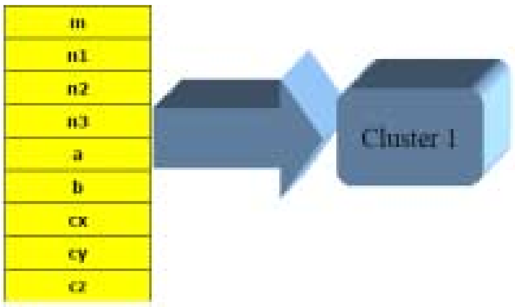


Fig. 8: Part of the genome encoding a cluster

We have introduced variables *cx*, *cy*, *cz* which correspond to the centers of the clusters. For illustration purposes we found the initial coordinates for these centers by applying a fuzzy-c means algorithm. This allows us to compare the results obtained from VGA and those used to define the clusters we used. Figure 9 shows the 3D positions of the centers.

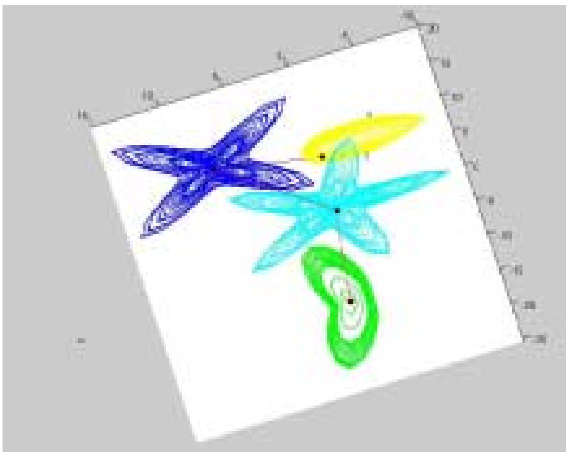


Fig. 9: Centers for Gielis EBs gotten from fuzzy-c means

Since, in accordance to Gielis Superformula

$m, n_1, n_2, n_3 \in \Re_+$ and $a, b \in \Re_+ \neq 0$ we used three bits to represent the integer part of the parameter and 32 for the fractional part. Figure 10 illustrates the full chromosome used for VGA.

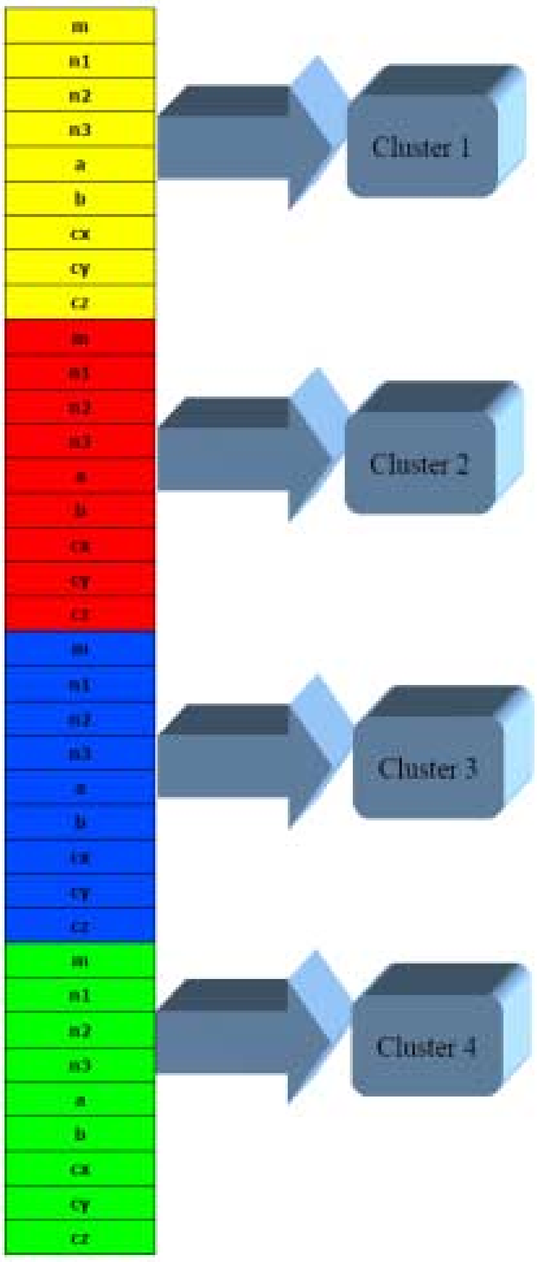


Fig. 10. Full chromosome for VGA

3.2 Fitness function

The fitness for the individuals in the population is given by the correct membership assignment of every vector in the data set to the NDOs given the parameters encoded in the individual. For instance, in a set of size N assumed to lie within 4 clusters a "good" individual is one in which the parameters in GSF yield 4 NDOs which include all N vectors. Hence, the fitness of an individual is given by the number of vectors lying within the clusters encoded in the individual's chromosome. In figure 11 we may see an individual whose genome corresponds to 4 NDOs which include 5 vectors out of 10. hence, its fitness is 5.

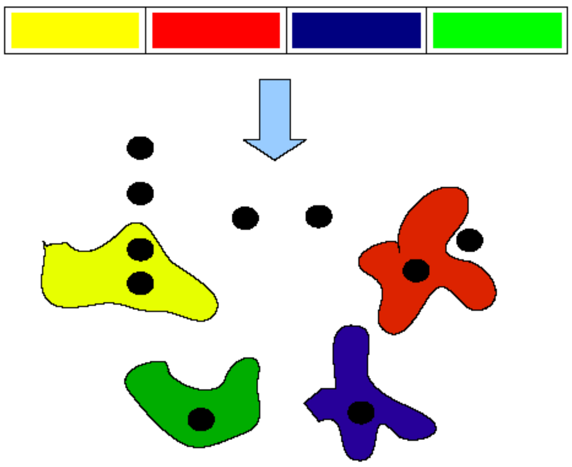


Fig. 11. Individual whose genome achieves partial inclusion for a data set.

It may happen that an individual includes all vectors in a single NDO as shown in figure 12. In such case, even if the inclusion is total, the individual is less than optimal and this representation must be penalized. To this effect we introduced an index given by the ratio of the number of clusters induced by the individual and the target number of clusters. In the example these quotient would evaluate to ¼ or 0.25. If we now multiply this index times the included number of vectors the individual induces we get $10 \times 0.25 = 2.5$ which is the proposed fitness function. Equation (6) generalizes this criterion.

$$f(n) = n \times \frac{C_{vga}}{C} \qquad (6)$$

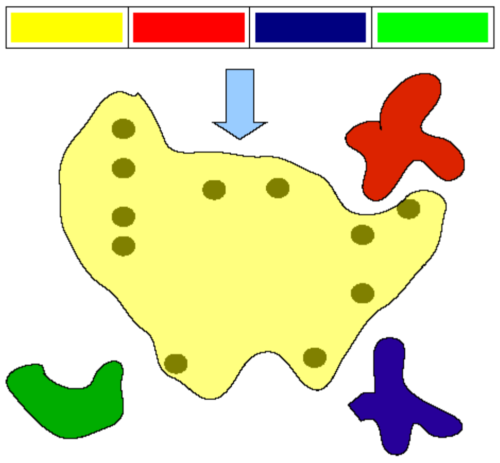


Fig. 12. Full inclusion in a single set

In equation (6) C_{vga} is the number of clusters induced by an individual and C is the desired number of clusters.

3.3. Cluster membership for an NDO

A very basic subproblem one has to cope with is how to determine whether a given vectors lies inside a given NDO. To this effect we first generate a grid over the surface of the NDO.

We applied a triangulation algorithm called *ear clipping* [4] which, for every 3 non-consecutive points generates a triangle belonging to the grid. In the end the algorithm yields a number of triangles leading to a body's characterization as shown in figure 13.

To test whether a point lies within the NDO we make, for every triangle, a projection which has an angle of spread. This defines an orthogonal cone on the grid. One then tests algebraically whether the point under analysis is within the cone.

4 Results and future research

The results of the execution of VGA when a sample of size 10,000 whose memberships are known for 4 clusters are shown in table 1.

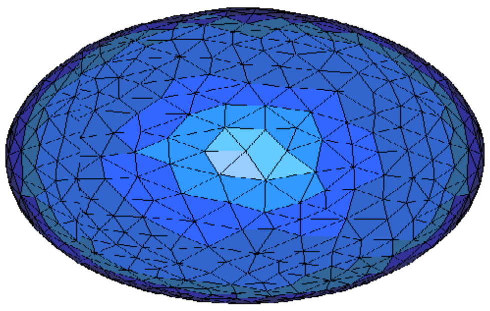


Fig. 13. Surface grid on a 3D NDO.

Likewise, in table 2 we include a simple comparison between alternative methods.

The results gotten so far are preliminary. Although we have achieved reasonable clustering for synthetic data where traditional clustering techniques perform poorly, several issues remain to be solved.

Table 1. Number of elements correctly assigned by VGA on a NDO obtained from GSF.

Cluster 1	2430
Cluster 2	2455
Cluster 3	2463
Cluster 4	2475
Total	9823
% Error	1,77

Table 2. Comparison between different classification methods

Method	**Errors**	**% Error**
Kohonen Maps	2,654	25.51
Fuzzy C-Means	389	3.74
Perceptron Network	310	2.98
VGA-Gielis	177	1.7

a) The fitness function has to be refined. It is not clear whether a simple strategy as outlined above will be adequate in most cases.
b) The computation involved in the determination of the membership of the vectors to an NDO has to be improved, given the computationally intensive nature of the evolutionary algorithm.
c) The best spread angle has to be determined a priori. We would like to relieve the user from this task.

In the near future we will report on these issues.

All in all we know, from initial experiments, that our method will perform adequately where others simply will not do because of conceptual constraints, as stressed in the introduction. In the past it has been customary to verify the goodness of the clusters gotten by a given method through tests which emphasize one or several criteria. For a discussion of this matter the reader is referred to [10], [11]. In our method the goodness of the cluster is a necessary condition for the clusters to be found. Therefore, there is no need for “outside” validity measures. In fact, the method may be seen as a dynamic measure of such validity, which is another interesting contribution. That is, the validity is defined as the one maximizing the set of vectors lying in the NDOs.

References:

[1] Gielis, J., *A generic geometric transformation that unifies a wide range of natural and abstract shapes,* American Journal of Botany, 2003.

[2] Tsunenori, I., *An expansion of X-means for automatically determining the optimal number of clusters,*2005,http://www.rd.dnc.ac.jp/~tunenori/doc/487-053.pdf *access* 26/01/2008.

[3] Xin L., Wai M., Chi K.L., *Determining the Optimal Number of Cluster by an Extended RPCL Algorithm*, Hong Kong: Polytechnnic University, 1999.

[4] Eberly, D., *Triangulation by ear clipping*, 2002, http://www.geometrictools.com/Documentation/TriangulationByEarClipping.pdf, *access 25/01/2008.*

[5] Bäck, T., *Evolutionary Algorithms in Theory and Practice,* Oxford University Press, 1996.

[6] Rudolph, G., *Convergence Analysis of Canonical Genetic Algorithms*, IEEE Transactions on Neural Networks, **5**(1):96-101, January, 1994.

[7] Forrest, S. and Mitchell, M., *What makes a problem hard for a genetic algorithm?* Machine Learning, **13**:285-319, 1993.

[8] Mitchell, M. *An introduction to genetic algorithms*, MIT Press, pp. 132-134, 1996.

[9] Kuri, A., *A Methodology for the Statistical Characterization of Genetic Algorithms*, LNAI 2313, pp. 79-88, Springer-Verlag, 2002.

[10] Domeniconi, C., Gunopulos,D., Ma, S., Papadopoulos, D., Yan , B., *Locally Adaptive Metrics for Clustering High Dimensional Data*, Data Mining and Knowledge Discovery , Springer, Volume 14, Number 1, 2007.

[11] Kovács, F., Iváncsy, R., *A novel cluster validity index: variante of the nearest neighbor distance*, WSEAS Transactions on Computers, Issue 3, Vol. 5, pp. 477-483, 2006.

Appendix E

Clustering based on optimization of validity index (CVI)

We propose an alternative to solve the clustering problem based on the identification of those elements of X which optimize some quality measures or *validity index* [10]. In the usual process, the clusters are found via some arbitrary clustering method and then assigned a quality grade according to a certain validity index. We propose a novel alternative finding the elements of the clusters from the indices directly. Usually the purported indices are expressed mathematically with some complex function which is not prone to optimization by any of the traditional methods. We analyze clusters resulting by optimizing a validity index with EGA.

E.1 Validity Index

The validity index are used for measure the quality of a clustering results obtained through of the some clustering method. If we work "backwards" and optimize the purported index, we should come up with a "good" clustering. The following indices was selected with this purpose.

E.1.1 Dunn and Dunn (DD) index

This validity index for clustering attempts to identify “compact and well separated clusters” [15]. The index is defined by following equation for a specific number of clusters.

$$D = min_{i=1,..,k} \left\{ min_{j=i+1,...,k} \left\{ \frac{d(C_i, C_j)}{max_{m=1,...,k} diam(C_m)} \right\} \right\} \tag{E.1}$$

where k is the number of clusters, $d(C_i, C_j)$ is the dissimilarity function between two clusters C_i and C_j defined as

$$d(C_i, C_j) = min_{x \in C_i, y \in C_j} d(x, y) \tag{E.2}$$

and $diam(c_m)$ is diameter of the m-th cluster which may be considered as a measure of dispersion of the clusters and defined as follows:

$$diam(C_m) = max_{x,y \in C_m} \{d(x, y)\} \tag{E.3}$$

If the data set contains well-separated clusters, the distance between clusters is usually large and the diameter of the clusters is expected to be small. Therefore large value means better clustering results.

E.1.2 Davies-Bouldin (DB) index

A similarity measure R_{ij} between the clusters C_i and C_j is defined based on a measure of dispersion of a cluster denoted as s_i and a dissimilarity measure between two clusters denoted as d_{ij}. The R_{ij} index is defined to satisfy the following conditions:

1. $R_{ij} \geq 0$
2. $R_{ij} = R_{ji}$
3. if $s_i = 0$ and $s_j = 0$ then $R_{ij} = 0$
4. if $s_j > s_k$ and $d_{ij} = d_{ik}$ then $R_{ij} > R_{ik}$
5. if $s_j = s_k$ and $d_{ij} < d_{ik}$ then $R_{ij} > R_{ik}$

These conditions state that R_{ij} is non-negative and symmetric. It is given by:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{E.4}$$

The dissimilarity measure d_{ij} and the dispersion s_i are defined as follows:

$$d_{ij} = d(v_i, v_j) \tag{E.5}$$

$$s_i = \frac{1}{|C_i|} \sum_{x \in C_i} d(x, v_i) \tag{E.6}$$

where v_i is the centroid of the cluster C_i . The DB index measures the average of similarity between each cluster an its most similar one [36]. The lower value of DB means better clustering result due to the clusters have to be compact and separated. The value of the index is calculated as follows:

$$DB = \frac{1}{k} \sum_{i=1}^{k} R_i \tag{E.7}$$

where $R_i = \underset{j=1,\ldots,k,\, j\neq i}{max}\{R_{ij}\}$, $i = 1, \ldots k$

E.1.3 Scattering and Dispersion (SD) index

The Scattering and Dispersion validity index is defined based on the concepts of the average scattering for clusters and total separation between clusters [10]. Let $\sigma(C_i)$ be the vector variance of a i-th cluster and $\sigma(X)$ the vector variance of the set X such that the average scattering is given by:

$$Scat = \frac{1}{k} \sum_{i=1}^{k} \frac{\parallel \sigma(C_i) \parallel}{\parallel \sigma(X) \parallel} \tag{E.8}$$

The dispersion or total separation between clusters is given by following equation:

$$Dis = \frac{D_{max}}{D_{min}} \sum_{i=1}^{k} \left[\sum_{j=1}^{k} |v_i - v_j| \right]^{-1} \tag{E.9}$$

where $D_{max} = max(|v_i - v_j|)$ and $D_{min} = min(|v_i - v_j|)$ $\forall i,\ j \in \{1, 2, 3, \ldots, k\}$ are the maximum and minimum distance between cluster centroids v_i respectively. Now, we can define a validity index based on equations above, as follows

$$SD = \alpha Scat + Dis \tag{E.10}$$

where α is a weighting factor equal to Dis value in case of maximum number of clusters. Lower SD index means better cluster configuration, it means that the clusters are compact and separated.

E.1.4 Variance of the Nearest Neighbor Distance (VNND) index

A disadvantage of the introduced validity index is that they do not investigate the local environment of the elements of a cluster. A novel approach is to investigate the local environment through the deviation of the nearest neighbor distances in every cluster [97]. This can be defined as:

$$d_{min}(x_i) = min_{y \in C_i} \{d(x_i, y)\} \tag{E.11}$$

$$\overline{d_{min}(C_i)} = \frac{\sum\limits_{x \in Ci} d_{min}(x_i)}{|C_i|} \tag{E.12}$$

$$V(C_i) = \frac{1}{|C|_i - 1} \sum_{x_i \in C_i} d_{min}(x_i) - d_{min}(C_i) \tag{E.13}$$

Based on above equations the VNND index is defined as:

$$VNND = \sum_{i=1}^{k} V(C_i) \tag{E.14}$$

This validity index measures the homogeneity of the clusters. Lower index value means more homogeneous clustering. The principal advantage is that VNND index do not use global references points for calculate its value (e.g centroids) thus it can measure clusters with non-convex hulls.

E.2 Experiments and Results

To measure the effectiveness of this method, we ran 200 times the EGA (with 150 individuals and $G = 300$), using different validity index and a given type of Gaussian dataset (disjoint, partial and total overlap). In Table E.1 we show the average performance of CVI for the disjoint pattern sets generated. The “performance value” is defined as the success ratio based on the class labels of the objects (elements in a dataset) known a priori. We also show the performance displayed by the Bayesian Classifier for the same set of problems (see Table E.2 and E.3). We show the result obtained with partial and total overlap respectively.

Considering that CVI displays the best performance with the index SD(with an average performance value of 70.3), we show the global results in Table E.4. In general we can see that CVI yields "good" results relative to a Bayesian classifier.

Table E.1: Performance of CVI for disjoint Gaussian dataset

Index	Type Optimization	Performance
Dunn and Dunn	Maximize	99.00
SD Validity Index	Minimize	99.00
Davies-Boulding	Minimize	99.00
Nearest Neighbor Distance	Minimize	99.00
Bayesian Classifier Efectiveness		99.00

Table E.2: Performance of CVI for Gaussian dataset with partial overlap

Index	Type Optimization	Performance
Dunn and Dunn	Maximize	36.00
SD Validity Index	Minimize	58.00
Davies-Boulding	Minimize	38.00
Nearest Neighbor Distance	Minimize	57.00
Bayesian Classifier Efectiveness		71.70

Table E.3: Performance of CVI for Gaussian dataset with total overlap

Index	Type Optimization	Performance
Dunn and Dunn	Maximize	40.00
SD Validity Index	Minimize	54.00
Davies-Boulding	Minimize	38.00
Nearest Neighbor Distance	Minimize	53.00
Bayesian Classifier Efectiveness		66.50

Table E.4: Performance of CVI and BC for different Gaussian datasets

Algorithm	Disjoint	Partial Overlap	Total Overlap	Global	Relative
CVI	99.0	58.0	54.0	70.3	92.0%
BC	99.9	71.7	56.5	76.0	100%

Appendix F

Clustering Based on Entropy Optimization

Unsupervised Classifier Based on Heuristic Optimization and Maximum Entropy Principle

Edwin Aldana-Bobadilla and Angel Kuri-Morales

Universidad Nacional Autónoma de México, Mexico City, Mexico,
Instituto Tecnológico Autónomo de México, México City, Mexico
ealdana@uxmcc2.iimas.unam.mx,
akuri@itam.mx

Abstract. One of the basic endeavors in Pattern Recognition and particularly in Data Mining is the process of determining which unlabeled objects in a set do share interesting properties. This implies a singular process of classification usually denoted as "clustering", where the objects are grouped into k subsets (clusters) in accordance with an appropriate measure of likelihood. Clustering can be considered the most important unsupervised learning problem. The more traditional clustering methods are based on the minimization of a similarity criteria based on a metric or distance. This fact imposes important constraints on the geometry of the clusters found. Since each element in a cluster lies within a radial distance relative to a given center, the shape of the covering or hull of a cluster is hyper-spherical (convex) which sometimes does not encompass adequately the elements that belong to it. For this reason we propose to solve the clustering problem through the optimization of Shannon's Entropy. The optimization of this criterion represents a hard combinatorial problem which disallows the use of traditional optimization techniques, and thus, the use of a very efficient optimization technique is necessary. We consider that Genetic Algorithms are a good alternative. We show that our method allows to obtain successfull results for problems where the clusters have complex spatial arrangements. Such method obtains clusters with non-convex hulls that adequately encompass its elements. We statistically show that our method displays the best performance that can be achieved under the assumption of normal distribution of the elements of the clusters. We also show that this is a good alternative when this assumption is not met.

Keywords: Clustering, Genetic Algorithms, Shannon's Entropy, Bayesian Classifier

1 Introduction

Pattern recognition is a scientific discipline whose purpose is to describe and classify objects. The descriptive process involves a symbolic representation of these objects called *patterns.* In this sense, the most common representation is through a numerical vector $\boldsymbol{x}$:

$$\boldsymbol{x} = [x_1, x_2, \dots x_n] \in \Re^n \tag{1}$$

where the n components represent the value of the properties or attributes of an object. Given a pattern set X, there are two ways to attempt the classification: a) *Supervised Approach* and b) *Unsupervised Approach.*

In the supervised approach, $\forall \boldsymbol{x} \in X$ there is a class label $y \in \{1, 2, 3, ..., k\}$. Given a set of class labels Y corresponding to some observed patterns $\boldsymbol{x}$ ("training" patterns), we may postulate a hypothesis about the structure of X that is usually called the *model.* The model is a mathematical generalization that allows us to divide the space of X into k decision regions called *classes.* Given a model M, the class label y of an unobserved (unclassified) pattern $\boldsymbol{x}'$ is given by:

$$y = M(\boldsymbol{x}') \tag{2}$$

On the other hand, the unsupervised approach consists in finding a hypothesis about the structure of X based only on the similarity relationships among its elements. The unsupervised approach does not use prior class information. The similarity relationships allow to divide the space of X into k subsets called *clusters.* A cluster is a collection of elements of X which are "similar" between them and "dissimilar" to the elements belonging to other clusters. Usually the similarity is defined by a *metric* or distance function $d : X \times X \rightarrow \Re$.

In this work we discuss a clustering method which does not depend explicitly on minimizing a distance metric and thus, the shape of the clusters is not constrained by hyper-spherical hulls. Clustering is a search process on the space of X that allows us to find the k clusters that satisfy an optimization criteria. Mathematically, any criterion involves an objective function f which must be optimized. Depending on the type of f, there are several methods to find it. Since our clustering method involves an objective function f where its feasible space is, in general, non-convex and very large, a good optimization algorithm is compulsory. With this in mind, we made a comprehensive study [13] which dealt with the relative performance of a set of structurally different GAs and a non-evolutionary algorithm over a wide set of problems. These results allowed us to select the statistically "best" algorithm: the EGA [20]. By using EGA we may be sure that our method will displays high effectiveness for complex arrangements of X.

The paper is organized as follows: In Section 2, we briefly show the results that led us to select the EGA. Then we present the different sets of patterns X that will serve as a the core for our experiments. We use a Bayesian Classifier[2,4,8] as a method of reference because there is theoretical proof that its is optimal given data stemming from normal distributions. In this section we discuss the issues which support our choice. In Section 3 we discuss the main characteristics of our method and the experiments which show that it is the best alternative. In Section 4 we present our general conclusions.

2 Preliminaries

As pointed out above, a "good" optimization algorithm must be selected. We rest on the conclusions of our previous analysis regarding the performance of a set of GAs[13] . Having selected the best GA, we prove the effectiveness of our clustering method by classifying different pattern sets. To this effect, we generated pattern sets where, for each pattern, the class of the objects is known. Hence, the class found by our clustering method may be compared to the true ones. To make the problems non-trivial we selected a non-linearly separable problems. We discuss the process followed to generate these sets. Finally, we resort to a *Bayesian Classifier* [4] in order to show that the results obtained by our method are similar to those obtained with it.

2.1 Choosing the Best Optimization Algorithm

This section is a very brief summary of the most important results found in [13]. A set A of 4 structurally different GAs and a non-evolutionary algorithm (NEA) was selected in order to solve, in principle, an unlimited supply of systematically generated functions in $\Re \times \Re$(called unbiased functions). An extended set of such functions in $\Re \times \Re^2$ and $\Re \times \Re^3$ was generated and solved. Similar behavior of all the GAs in$A$ (within statistical limits) was found. This fact allowed us to hypothesize that the expected behavior of A for functions in $\Re \times \Re^n$ will be similar. As supplement, we tackled a suite of problems (approximately 50) which includes hard unconstrained problems (which traditionally have been used for benchmarking purposes) [19,3] and constrained problems [11]. Lastly, atypical GA-hard functions were analyzed [18,16].

Set of Algorithms The set A included the following GAs: a)An elitist canonical GA (in what follows referred to as TGA [eliTist GA]) [21], b) A Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation algorithm (CHC algorithm) [5], c) An Eclectic Genetic Algorithm (EGA) [20], d) A Statistical GA (SGA) [23,12] and e) A non-evolutionary algorithm called RMH [17].

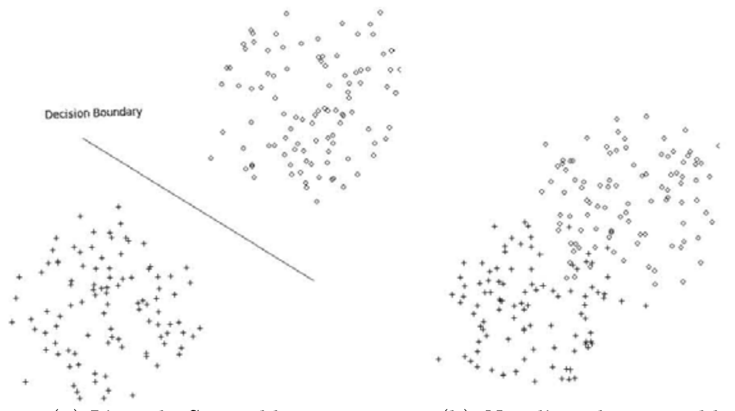
Table 1 shows the relative global performance of all algorithms for the functions mentioned. The best algorithm in the table is EGA.

Table 1: Global Performance

A_i	Unbiased	Suite	Atypical	Global Performance	Relative
EGA	9.64	8.00	4.48	7.37	100.00%
RMH	6.24	0.012	2.04	2.76	37.49%
TGA	1.35	1.16	4.77	2.43	32.91%
SGA	1.33	0.036	3.33	1.57	21.23%
CHC	2.12	0.08	2.10	1.43	19.44%

2.2 The pattern set

Given a set of patterns to be classified, the goal of any classification technique is to determine the decision boundary between classes. When these classes are unequivocally separated from each other the problem is separable; otherwise, the problem is non-separable. If the problem is linarly separable, the decision consists of a hyperplane. In Figure 1 we illustrate ths situation.



(a) Linearly Separable patterns (b) Non-linearly separable patterns

Fig. 1: Decision boundary

When there is overlap between classes some classification techniques (e.g. Linear classifiers, Single-Layer Perceptrons [8]) may display unwanted behavior because decision boundaries may be highly irregular . To avoid this problem many techniques has been tried (e.g. Support Vector Machine [9], Multilayer Perceptrons [22]). However, there is no guarantee that any of this methods will perform adequately. Nevertheless, there is a case which allows us to test the appropriateness of our method. Since it has been proven that if the classes are normally distributed, a Bayesian Classifier yields the best possible result (in Section 2.3 we discuss this fact) and the error ratio will be minimized. Thus, the Bayesian Classifier becomes a good method with which to compare any alternative clustering algorithm.

Hence, we generated Gaussian pattern sets considering singular arrangements in which determining the decision boundaries imply non-zero error ratios. Without loss generality we focus on patterns defined in $\Re^2$. We wish to show that the results obtained with our method are close to those obtained with a Bayesian Classifier; in Section 3, the reader will find the generalization of our method for $\Re^n$.

Gaussian Patterns in $\Re^2$ Let X_j be a pattern set defined in $\Re^2$ and $C_i \subset X_j$ a pattern class. A pattern $\boldsymbol{x} = [x_1, x_2] \in C_i$ is drawn from a Gaussian distribution if its joint probability density function (pdf) is given by:

$$f(x_1, x_2) = \frac{1}{2\pi\sigma_{x_1}\sigma_{x_2}\sqrt{1-\rho^2}} e^{\left(-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x_1-\mu_{x_1}}{\sigma_{x_1}}\right)^2 - 2\rho\frac{(x_1-\mu_{x_1})(x_2-\mu_{x_2})}{\sigma_{x_1}\sigma_{x_2}} + \left(\frac{x_2-\mu_{x_2}}{\sigma_{x_2}}\right)^2\right]\right)} \tag{3}$$

where $-1 < \rho < 1$, $-\infty < \mu_{x_1} < \infty$, $-\infty < \mu_{x_2} < \infty$, $\sigma_{x_1} > 0$, $\sigma_{x_2} > 0$. The value ρ is called the correlation coefficient.

To generate a Gaussian pattern $\boldsymbol{x} = [x_1, x_2]$, we use the *acceptance-rejection* method [1,10] which allows us to generate random observations (x_1, x_2) that are drawn from $f(x_1, x_2)$. In this method, a uniformly distributed random point (x_1, x_2, y) is generated and accepted iff $y < f(x_1, x_2)$. In Figure 2.1 we show different pattern sets obtained by applying this method with distinct statistical arguments in (3)

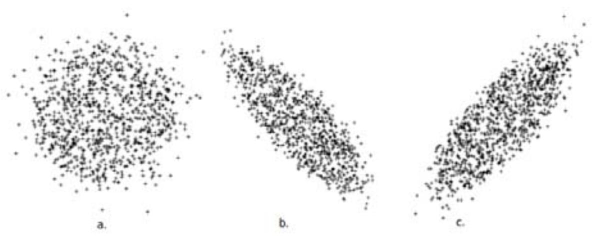


Fig. 2: Different Gaussian pattern sets with $\mu_{x_1} = \mu_{x_2} = 0.5$,$\sigma_{x_1} = \sigma_{x_2} = 0.09$. Each set was generated with different correlation coefficient: a. $\rho = 0.0$, b. $\rho = -0.8$, c. $\rho = 0.8$.

The degrees of freedom in (3) allow us to generate Gaussian pattern sets with varied spatial arrangements. In principle, we analyze pattern sets with the following configurations:

- Sets with disjoint pattern classes.
- Sets with pattern classes that share some elements (partial overlap between classes)
- Sets with pattern classes whose members may share most elements (total overlap).

We proposed these configurations in order to increase gradually the complexity of the clustering problem and analyze systematically the performance of our method. In the following subsections, we make a detailed discussion regarding the generation of sets with these configurations.

Gaussian pattern set with disjoint classes.

Definition: Let X_1 be a pattern set with classes $C_i \subset X_1 \forall i = 1, 2...k$ which are drawn from a Gaussian distribution. X_1 is a set with disjoint classes if $\forall C_i,\ C_j \subset X_1,\ C_i \cap C_j = \phi$.

Based on the above definition, we generate two different sets where $\boldsymbol{x} \in [0,1]^2$ (in every set there are three classes and $|X| = 1000$). In Figure 3 we illustrate the spatial arrangement of these sets.

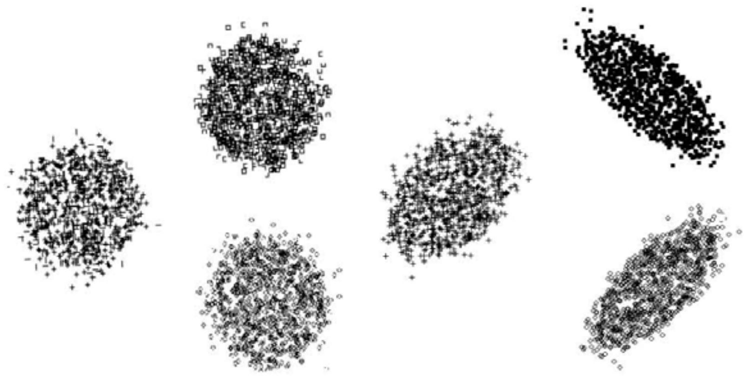
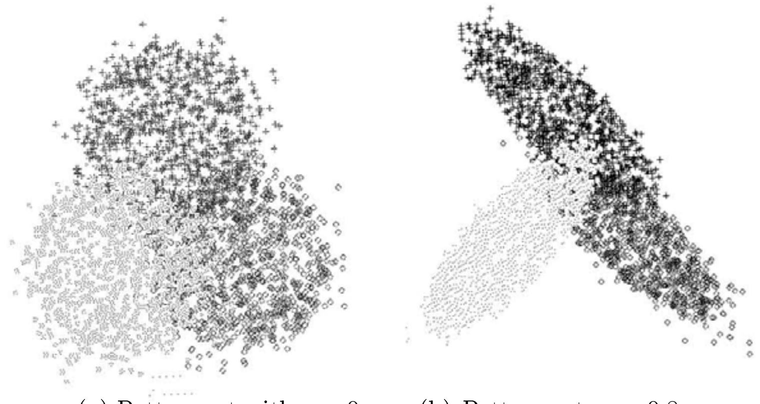

(a) Pattern set with $\rho = 0$ (b) Pattern set with $\rho = 0.4$,$\rho = -0.7$, $\rho = 0.7$

Fig. 3: Gaussian pattern sets with disjoint classes

Gaussian pattern set with Partial Overlap

Definition: Let X_2 be a pattern set with classes $C_i \subset X_2 \forall i = 1, 2...k$ which are drawn from a Gaussian distribution. X_2 is a set with partial overlap if $\exists C_i,\ C_j \subset X_2$ such that $C_i \cap C_j \neq \phi$ and $C_i \nsubseteq C_j$.

Based on the above definition, we generate two different sets where $\boldsymbol{x} \in [0,1]^2$ (in every set there are three classes and $|X| = 1000$). In Figure 4 we illustrate the spatial arrangement of these sets.



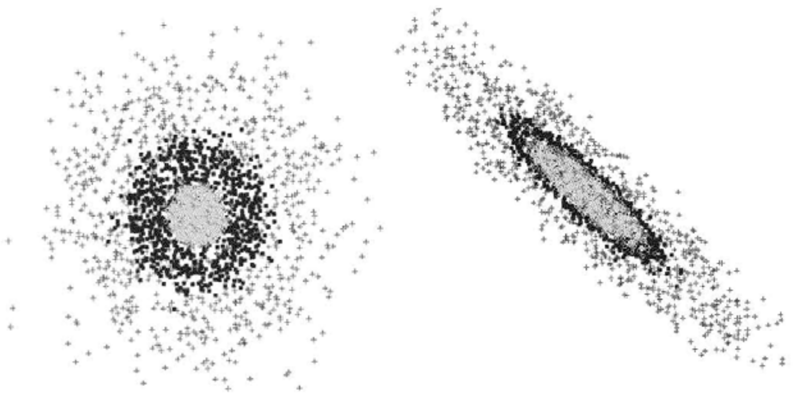
(a) Pattern set with $\rho = 0$ (b) Pattern set $\rho = 0.8$, $\rho = -0.8$, $\rho = -0.8$

Fig. 4: Pattern sets with overlap classes

Gaussian pattern set with Total Overlap

Definition: Let X_3 be a pattern set with classes $C_i \subset X_3 \forall i = 1, 2 \ldots k$ which are drawn from a Gaussian distribution. X_3 is a set with total overlap if $\exists C_i$, $C_j \subset X_3$ such that $C_i \subseteq C_j$.

Based on the above definition, we generate two different sets where $\boldsymbol{x} \in [0, 1]^2$ (in every set there are three classes and $|X| = 1000$). In Figure 5 we illustrate the spatial arrangement of these sets.



(a) Pattern set with$\rho = 0$ (b) Pattern set with $\rho = -0.9$

Fig. 5: Pattern sets with total overlap classes

2.3 Bayesian Classifier

If the objects in the classes to be clustered are drawn from normally distributed data, the best alternative to determine the decision boundary is using a Bayesian Classifier. The reader can find a extended discussion in [4,8].

Given a sample of labeled patterns $\boldsymbol{x} \in X$, we can hypothesize a partitioning of the space of X into k classes C_i. The classification problem can be reduced to find the probability that given a pattern $\boldsymbol{x}$, it belongs to C_i. From Bayes's theorem this probability is given by:

$$p(C_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_i)p(C_i)}{p(\boldsymbol{x})} \tag{4}$$

where $p(C_i)$ is usually called the *prior* probability. The term $p(\boldsymbol{x}|C_i)$ represents the *likelihood* that observing the class C_i we can find the pattern $\boldsymbol{x}$. The probability to find a pattern $\boldsymbol{x}$ in X is denoted as $p(\boldsymbol{x})$ which is given by:

$$p(\boldsymbol{x}) = \sum_{i=0}^{k} p(\boldsymbol{x}|C_i)p(C_i) \tag{5}$$

The prior probability $p(C_i)$ is determined by the training pattern set (through the class labels of every pattern). From 4 we can note that the product $p(\boldsymbol{x}|C_i)p(C_i)$ is the most important term to determine $p(C_i|\boldsymbol{x})$ (the value of $p(\boldsymbol{x})$ is merely a scale factor). Therefore, given a pattern $\boldsymbol{x}$ to be classified into two classes C_i or C_j, our decision must focus on determining $max\,[p(\boldsymbol{x}|C_i)p(C_i), p(\boldsymbol{x}|C_j)p(C_j)]$. In this sense, if we have a pattern $\boldsymbol{x}$ for which $p(\boldsymbol{x}|C_i)p(C_i) \geq p(\boldsymbol{x}|C_j)p(C_j)$ we will decide that it belongs to C_i, otherwise, we will decide that it belongs to C_j. This rule is called *Bayes's Rule.*

Under gaussian assumption, the Bayesian classifier outperforms other classification techniques, such as those based on *linear predictor functions* [8]. We discuss our method assuming normality so as to measure its performance relative to that of a Bayesian Classifier. Our claim is that, if the method performs satisfactorily when faced with Gaussian patterns, it will also perform reasonably well when faced with other possible distributions.

3 Clustering based on Shannon's Entropy (CBE)

We can visualize any clustering method as a search for k regions (in the space of the pattern set X), where the *dispersion* between the elements that belong to them is minimized. This dispersion can be optimized via a distance metric [15,7], a quality criterion or a membership function [14]. In this section we discuss an alternative based on *Shannon's Entropy* [24] which appeals to an evaulation of the *information content* of a random variable Y with possible values $\{y_1, y_2, ...y_n\}$. From a statistical viewpoint, the information of the event $(Y = y_i)$ is proportional to its likelihood. Usually this information is denoted by $I(y_i)$ which can be expressed as:

$$I(y_i) = log\left(\frac{1}{p(y_i)}\right) = -log\left(p(y_i)\right) \tag{6}$$

From information theory [24,6], the information content of Y is the expected value of I. This value is called Shannon's Entropy which is given by:

$$H(Y) = -\sum_{i=1}^{n} p(y_i) log\left(p(y_i)\right) \tag{7}$$

When $p(y_i)$ is uniformly distributed, the entropy value of Y is maximal. It means that all events in the probability space of Y have the same ocurrence probability and thus Y has the highest level of unpredictability. In other words, Y has the maximal information content.

In the context of the clustering problem, given an unlabeled pattern set X, we hypothesize that a cluster is a region of the space of X with a large information content. In this sense, the clustering problem is reduced to a search for k regions where the entropy is maximized. Tacitly, this implies finding k optimal probability distributions (those that maximize the information content for each regions). It is a hard combinatorial optimization problem that requires an efficient optimization method. In what follows we discuss the way to solve such problem through EGA. In principle, we show some evidences that allow us to think that our method based on entropy is succesful. We statistically show this method is the best.

3.1 Shannon's Entropy in clustering problem

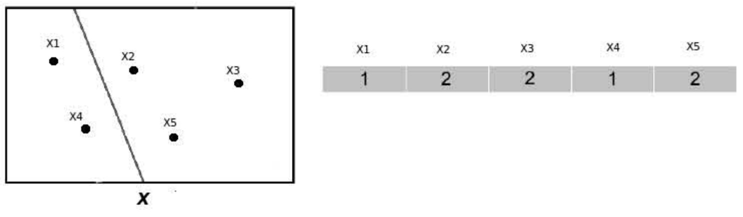
Given an unlabeled pattern set X, we want to find a division of the space of the X into k regions denoted by C_i, where Shannon's Entropy is maximized. We consider that the entropy of C_i depends on the probability distribution of all possible patterns $\boldsymbol{x}$ that belong to it. In this sense, the entropy of C_i can be expressed as:

$$H(C_i) = \sum_{\boldsymbol{x} \in C_i} p(\boldsymbol{x}|C_i) log(p(\boldsymbol{x}|C_i) \tag{8}$$

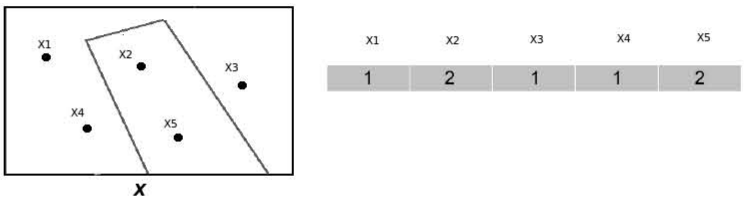
Since we want to find k regions C_i that maximize such entropy, the problem is reduced to an optimization problem of the form:

$$\begin{array}{c} \text{Maximize: } \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} p(\boldsymbol{x}|C_i) log(p(\boldsymbol{x}|C_i)) \\ \text{subject to:} \\ p(\boldsymbol{x}|C_i) > 0 \end{array} \tag{9}$$

To find the C_i's that minimize (9) we resort to the EGA. We encoded an individual as a random sequence of symbols L from the alphabet $\sum = \{1, 2, 3...k\}$. Every element in L represents the class label of a pattern $\boldsymbol{x}$ in X such that the length of L is $|X|$. Tacitly, this encoding divides the space of X into k regions C_i as is illustrated in Figure 5.1.



(a)



(b)

Fig. 6: Possible divisions of the space of a pattern set X in $\Re^2$ based on the encoding of an individual

Given this partititon, we can determine some descriptive parameters θ_i of the probability distribution of C_i (e.g. the mean or the standard deviation). Having determined θ_i and under the assumption that the probability distribution of C_i is known, the entropy of C_i can be determined. Complementarily, in Subection 3.3.2 we discuss a generalization making this assumption unnecessary.

3.2 Effectiveness of the Entropic Clustering for Gaussian Patterns in $\Re^2$

Based on the above, given an encoding solution L of a clustering problem (where X is a pattern set in $\Re^2$) we can determine k regions C_i with parameter $\theta_i = [\mu(C_i), \sigma(C_i)]$. Assuming that C_i is drawn from a Gaussian distribution, the value of $p(\boldsymbol{x}|C_i)$ with $\boldsymbol{x} = [x_1, x_2]$ is given by:

$$\int_{x_1}\int_{x_2} f(x_1, x_2) \tag{10}$$

where $f(x_1, x_2)$ is the bivariate Gaussian density function (see Equation 3). Given such probability the entropy for each C_i can be determined, the fitness of every L is given by $\sum_{i=1}^{k} H(C_i)$. The optimal solution will be the individual with the best fitness value after G iterations. To measure the effectiveness of our method, we ran 100 times the EGA (with 150 individuals and $G = 300$), selecting randomly different Gaussian pattern sets (disjoint, partial overlap and total overlap) in $\Re^2$. In Table 2 we show the performance of CBE for these problems. The "performance value" is defined as the success ratio based on the class labels of the patterns known a priori. We also illustrate the performance displayed by the Bayesian Classifier for the same set of problems.

Table 2: Performance of CBE and BC for different Gaussian pattern sets in $\Re^2$

Algorithm	Disjoint	Partial Overlap	Total Overlap	Global	Relative
CBE	99.0	70.8	57.6	75.8	99.7%
BC	99.9	71.7	56.5	76.0	100%

We see that there is not an important difference between results of our method and the results of a Bayesian Classifier. This result allows us to ascertain that our method is as good as the BC. Recall that the BC displays the best possible performance under Gaussian assumption. Furthermore, it is very important to stress that our method is unsupervised and, hence, the pattern set X is unlabeled. CBE allows us to find the optimal value of θ_i for all C_i that maximize the objective function (see Equation 9). These results are promising but we want to show that our method performs successfully, in general, as will be shown in the sequel.

3.3 Comprehensive Effectiveness Analysis

In order to evaluate the general effectivennes of CBE, we generated systematically a set of 500 clustering problems in $\Re^n$ assuming normality. The number of clusters for each problem and the dimensionality were randomly selected (the number of clusters $k \sim U(2, 20)$ and the dimensionality $n \sim U(2, 10)$). Thus, we obtained an unbiased set of problems to solve through CBE and BC. Similarly, we also propose a method to generate systematically a set of clustering problems in $\Re^n$ without *any assumption regarding the pdf* of the patterns to be classified.

Effectiveness for Gaussian Patterns in $\Re^n$ We wrote a computer program that generates Gaussian patterns $\boldsymbol{x} = [x_1, x_2, ...x_{n,}]$ through the *acceptance-rejection* method [1,10] given a value of n. Here, a uniformly distributed random point $(x_1, x_2, ...x_n, y)$ is generated and accepted iff $y < f(x_1, x_2, ..., x_n)$ where $f(x_1, x_2, ..., x_n)$ is the Gaussian density function with parameters μ and σ. Our program determines randomly the values of $\mu = [\mu_{x_1}, \mu_{x_2} ..., \mu_{x_n}]$ and $\sigma = [\sigma_{x_1}, \sigma_{x_2} ..., \sigma_{x_n}]$ such that $\mu_{x_i} \in [0, 1]$ and $\sigma_{x_i} \in [0, 1]$. In this way a cluster is a set of Gaussian patterns with the same values of μ and σ. and a clustering problem is a set of such clusters. The cardinality of a cluster is denoted by $|C_i|$ whose value was established as 200. It is important to note that the class label of every generated pattern was recorded in order to determine the performance or effectiveness of a classification process. We obtained a set of 500 different Gaussian clustering problems. To evaluate the performance of any method (CBE or BC) for such problems, we wrote a computer program that executes the following steps:

1. A set of $N = 36$ clustering problems are randomly selected.
2. A effectiveness value y_i is recorded for each problem.
3. For every N problems $\bar{y}_i$ is calculated.
4. Steps 1-3 are repeated until the values $\bar{y}_i$ are approximately normally distributed with parameter μ' and σ' (from *the central limit theorem*).

Dividing the span of means $\bar{y}_i$ in deciles, normality was considered to have been reached when:

1. $\chi^2 \leq 4$ and
2. The ratio of observations in the i-th decil $O_i \geq 0.5 \; \forall i$.

From Chebyshev's Inequality [25] the probability that the performance of an method denoted by τ lies in the interval $[\mu' - \lambda\sigma',\, \mu' + \lambda\sigma']$ is given by:

$$p(\mu' - \lambda\sigma' \leq \tau \leq \mu' + \lambda\sigma') \geq 1 - \frac{1}{\lambda^2} \tag{11}$$

where λ denotes the number of standard deviations. By setting $\lambda = 3.1623$ we ensure that the values of τ will lie in this interval with probability $p \approx 0.9$. Hence, the largest value of performance found (with $p \approx 0.95$ if we assume a symmetric distribution) by any method (CBE and BC) is $\mu + \lambda\sigma$. The results are shown in Table 3. These results allow us to prove statistically (with significance level of 0.05) that in $\Re^n$our method is as good as the BC under normality assumption.

Table 3: Comparative average success ratio for Gaussian Problems

Algorithm	μ	σ	$\mu + \lambda\sigma$	Relative
CBE	75.53	3.22	85.71	99%
BC	76.01	3.35	86.60	100%

Effectiveness for Non-Gaussian Patterns $\Re^n$ To generate a Non-Gaussian patterns in $\Re^n$we resort to polynomial functions of the form:

$$f(x_1, x_2, ..., x_n) = a_{m1}x_1^m + ... + a_{mn}x_n^m + ... + a_{11}x_1 + a_{1n}x_n \tag{12}$$

Given that such functions have larger degrees of freedom, we can generate many points uniformly distributed in $\Re^n$. As reported in the previous section, we wrote a computer program that allows us to obtain a set of 500 different problems.

These problems were generated for random values of n and k with $|C_i| = 200$. As before, a set of tests with $N = 36$ was performed. As before, the number of samples of size N dependend on the distribution reaching normality. The results of this set of experiments is shown in Table 4. These results show that our method outperforms BC with a significant level of 0.05.

Table 4: Comparative average success ratio for non-Gaussian Problems

Algorithm	μ	σ	$\mu + k\sigma$	Relative
CBE	74.23	2.12	80.93	100%
BC	61.76	2.65	70.14	86%

4 Conclusions

The previous analysis, based on the solution of an exhaustive sample set, allows us to reach the following conclusions:

Entropic clustering (CBE) is reachable via an efficient optimization algorithm. In this case, based on previous work by the authors, one is able to take advantage of the proven efficiency of EGA. The particular optimization function (defined in (9)) yields the best average success ratio. We found that CBE is able to find highly irregular clusters in pattern sets with complex arrangements. When compared to BC's performance over Gaussian distributed data sets, CBE and BC have, practically, indistinguishable success ratios. Thus proving that CBE is comparable to the best theoretical option. Here we, again, stress that while BC corresponds to supervised learning whereas CBE does not. The advantage of this characteristic is evident. When compared to BC's performance over non-Gaussian sets CBE, as expected, displayed a much better success ratio. Based on comprehensive analysis (subsection 3.3), the conclusions above have been reached for statistical p values of $O(0.5)$. In other words, the probability of such results to persist on data sets outside our study is better than 0.95. Thus ensuring the reliability of CBE. Clearly above older alternatives.

References

1. Casella, G., Robert, C.P.: Monte carlo statistical methods (1999)
2. De Sa, J.M.: Pattern recognition: concepts, methods, and applications. Springer Verlag (2001)
3. Digalakis, J., Margaritis, K.: An experimental study of benchmarking functions for genetic algorithms (2002)
4. Duda, R., Hart, P., Stork, D.: Pattern classification. New York: John Wiley, Section 10, 6 (2001)
5. Eshelman, L.: The chc adaptive search algorithm. how to have safe search when engaging in nontraditional genetic recombination (1991)
6. Gallager, R.G.: Information theory and reliable communication (1968)

7. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17(2), 107–145 (2001)
8. Haykin, S.: Neural Networks: A Comprehensive Foundation. 2nd Edition (1998)
9. Hsu, C.W., Chang, C.C., Lin, C.J., et al.: A practical guide to support vector classification (2003)
10. Johnson, J.L.: Probability and statistics for computer science. Wiley Online Library (2003)
11. Kim, J.H., Myung, H.: Evolutionary programming techniques for constrained optimization problems (1997)
12. Kuri-Morales, A.: A statistical genetic algorithm (1999)
13. Kuri-Morales, A., Aldana-Bobadilla, E.: A comprehensive comparative study of structurally different genetic algorithms. Sent for publication (2013)
14. Kuri-Morales, A., Aldana-Bobadilla, E.: The Search for Irregularly Shaped Clusters in Data Mining. Kimito, Funatsu and Kiyoshi, Hasegawa (2011)
15. 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. vol. 1, p. 14. California, USA (1967)
16. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1996)
17. Mitchell, M., Holland, J., Forrest, S.: When Will a Genetic Algorithm Outperform Hill Climbing? Advances of Neural Information Processing Systems, No. 6, Morgan Kaufmann, pp. 51-58 (1994)
18. Molga, M., Smutnicki, C.: Test functions for optimization needs. Retrieved March 11, 2012 from http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf, pp. 41-42 (2005)
19. Pohlheim, H.: Geatbx: Genetic and evolutionary algorithm toolbox for use with matlab documentation (2012)
20. Rezaee, J., Hashemi, A., Nilsaz, N., Dezfouli, H.: Analysis of the strategies in heuristic techniques for solving constrained optimisation problems (2012)
21. Rudolph, G.: Convergence Analysis of Canonical Genetic Algorithms. IEEE Transactions on Neural Networks, 5(1):96-101, January (1994)
22. Rumelhart, D.E., Hintont, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
23. Sánchez-Ferrero, G., Arribas, J.: A statistical-genetic algorithm to select the most significant features in mammograms (2007)
24. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review 5(1), 3–55 (2001)
25. Steliga, K., Szynal, D.: On markov-type inequalities (2010)

Finding irregularly shaped clusters based on Entropy

Angel Kuri-Morales[1], Edwin Aldana-Bobadilla[2]
[1] Department of Computation,
Autonomous Technological Institute of Mexico,
Rio Hondo No. 1,
Mexico City, Mexico

[2] Institute of Research in Applied Mathematics and Systems,
Autonomous University of Mexico,
University City,
Mexico City, Mexico
akuri@itam.mx, ealdana@uxmcc2.iimas.unam.mx

Abstract. In data clustering the more traditional algorithms are based on similarity criteria which depend on a metric distance. This fact imposes important constraints on the shape of the clusters found. These shapes generally are hyperspherical due to the fact that each element in a cluster lies within a radial distance relative to a given center. In this paper we propose a clustering algorithm that does not depend on such distance metrics and, therefore, allows us to find clusters with arbitrary shapes in n-dimensional space. Our proposal is based on some concepts stemming from Shannon's information theory and evolutionary computation. Here each cluster consists of a subset of the data where entropy is minimized. This is a highly non-linear and usually non-convex optimization problem which disallows the use of traditional optimization techniques. To solve it we apply a rugged genetic algorithm (the so-called Vasconcelos' GA). In order to test the efficiency of our proposal we artificially created several sets of data with known properties in a tridimensional space. The result of applying our algorithm has shown that it is able to find highly irregular clusters that traditional algorithms cannot. Some previous work is based on algorithms relying on similar approaches (such as ENCLUS' and CLIQUE's). The differences between such approaches and ours are also discussed.

Keywords: clustering, data mining, information theory, genetic algorithms

1 Introduction

Clustering is an unsupervised process that allows the partition of a data set X in k groups or *clusters* in accordance with a similarity criterion. This process is unsupervised because it does not require a priori knowledge about the clusters. Generally the similarity criterion is a distance metrics based in *Minkowsky Family* [1] which is given by:

$$d_{mk}(P,Q) = \sqrt[p]{\sum_{i=1}^{n} |P_i - Q_i|^p} \qquad \textbf{(1)}$$

where *P* and *Q* are two vectors in an n-dimensional space. From the geometric point of view, these metrics represent the spatial distance between two points. However, this distance is sometimes not an appropriate measure for our purpose. For this reason sometimes the clustering methods use statistical metrics such as *Mahalanobis'* [2], *Bhattacharyya's* [3] or *Hellinger's* [4], [5]. These metrics statistically determine the similarity of the probability distribution between random variables *P* and *Q*. In addition to a similarity criterion, the clustering process typically requires the specification of the number of clusters. This number frequently depends on the application domain. Hence, it is usually calculated empirically even though there are methodologies which may be applied to this effect [6].

1.1 A Hierarchy of Clustering Algorithms

A large number of clustering algorithms has been proposed which are usually classified as follows:

Partitional. Which discover clusters relocating iteratively elements of the data set between subsets. These methods tend to build clusters of proper convex shapes. The most common methods of this type are k-means [7], k-medoids or PAM (Partitioning Around Medoids) and CLARA (Clustering Large Applications) [8].

Hierarchical. In which large clusters are merged successively into smaller clusters. The result is a tree (called a *dendrogram*) whose nodes are clusters. At the highest level of the *dendrogram* all objects belong to the same cluster. At the lowest level each element of the data set is in its own unique cluster. Thus, we must select the adequate cut level such that the clustering process is satisfactory. Representative methods in this category are BIRCH [9], CURE and ROCK [10].

Density Based. In this category a cluster is a dense (in some pre-specified sense) region of elements of the data set that is separated by regions of low density. Thus, the clusters are identified as areas highly populated with elements of the data set. Here each cluster is flexible in terms of their shape. Representative algorithms of this category are DBSCAN [11] and DENCLUE [12].

Grid Based. Which use space segmentation through a finite number of cells and from these performs all operations. In this category are STING (Statistical Information Grid-based method) described by Wang et al. [13] and Wave Cluster [14].

Additionally, there are algorithms that use tools such as fuzzy logic or neural networks giving rise to methods such as Fuzzy C-Means [15] and Kohonen Maps [16], respectively. The performance of each method depends on the application domain. However, Halkidi [17] present several approaches that allow to measure the quality of the clustering methods via the so-called "quality indices".

1.2 Desired Properties of Clustering Algorithms

In general a good clustering method must:

- Be able to handle multidimensional data sets.
- Be independent of the application domain.
- Have a reduced number of settings.
- Be able to display computational efficiency.
- Be able to yield irregular shaped clusters.

With respect to last point, the great majority of the clustering methods restrict the shape of the clusters to hyperspherical shapes (in the space of the metric) owing to the use of some sort of distance as a similarity criterion. Thus, necessarily the distance between each point inside a cluster and its center is smaller than the radius of an n-dimensional sphere. This is illustrated in Figure 1 for the simplest case where n=2 (and, thus, yields a circle).

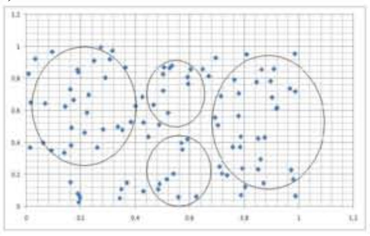


Fig. 1. Clusters with circular shapes

An ideal case would allow us to obtain arbitrary shapes for the clusters that adequately encompass the data. Figure 2 illustrates this fact.

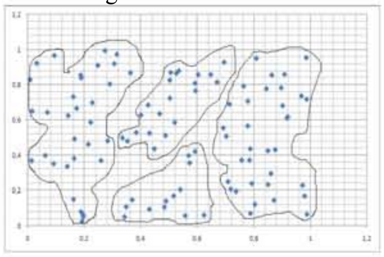


Fig. 2. Clusters with arbitrary shapes

Therefore, we propose a clustering algorithm that allows us to find clusters with irregular shapes that represent the data set better than clustering approaches that use distance metrics.

2 Related Works

Our proposal is based on maximizing density in terms of the entropy of the area of the space that represents a cluster. There exist previous works with similar approaches. Cheng et al [18], for example, present an algorithm called ENCLUS (Entropic Clustering) in which they link the entropy with two concepts that the authors call *coverage* and *density*. These are determined by the space's segmentation. Segmentation is made iteratively. Henceforth, several conditions have to be satisfied for every iteration of the algorithm. The space segmentation is a partition on non-overlapping rectangular units based on CLIQUE (Clustering in Quest) algorithm where a unit is dense if the fraction of the elements contained in the unit is greater than a certain threshold. A cluster is the maximum set of connected dense units. Another work is the so-called COOLCAT algorithm [19] which also approaches the clustering problem on entropic considerations but is mainly focused on categorical sets of data. The difference of our proposal is that the space is quantized through a hypercube that encapsulates all elements of the data set. The hypercube is composed of units of quantization that called “hypervoxels” or, simply, “voxels”. The number of voxels determines the resolution of the hypercube. Contrary to ENCLUS, our algorithm does not iterate to find the optimal space quantization. Here the hypercube is unique and its resolution is given a priori as a parameter. The units of quantization become the symbols of the source's alphabet which allow an analysis through information theory. Our working hypothesis is that areas with high density have minimum entropy with respect to areas with low density.

3 Generalities

In what follows we make a very brief mention of most of the theoretical aspects having to do with the proper understanding of our algorithm. The interested reader may consult the references.

3.1 Information Theory

Information theory addresses the problem of collecting and handling data from a mathematical point of view. There are two main approaches: the statistical theory of communication (proposed by Claude Shannon [20] and the so-called algorithmic complexity (proposed by Andrei Kolmogorov [21]. In this paper we rely on the statistical approach in which information is a series of symbols that comprise a *message,* which is produced by an *information source* and is received by a *receiver* through a *channel*.

Where:
Message. It is a finite succession or sequence of symbols.

Information Source. It is a mathematical model denoted by *S* which represents an entity which produces a sequence of symbols (message) randomly. The space of all possible symbols is called source alphabet and is denoted as Σ [22].
Receiver. It is the end of the communication's channel which receives the message.
Channel. It is the medium used to convey a *message* from an *information source* to a *receiver*.

In this document we apply two key concepts which are very important for our proposal.

Self Information. It is the information contained in a symbol s_i, which is defined as[1]:

$$I(s_i) = -\log_2 p(s_i) \qquad \textbf{(2)}$$

Where $p(s_i)$ is the probability that the symbol s_i is generated by the source *S*. We can see that the information of a symbol is greater when its probability is smaller. Thus, the self information of a sequence of statistically independent symbols is:

$$I(s_1 s_2 \dots s_n) = I(s_1) + I(s_2) + \dots + I(s_n) \qquad \textbf{(3)}$$

Entropy. The entropy is the expected value of the information of the symbols generated by the source *S*. This value may be expressed as:

$$H(S) = \sum_{i=1}^{n} p(s_i) I(s_i) = -\sum_{i=1}^{n} p(s_i) \log_2 p(s_i) \qquad \textbf{(4)}$$

Where *n* is the size of the alphabet Σ. Therefore, we see that entropy is greater the more uniform the probability distribution of symbols is.

3.2 Genetic Algorithms

Genetic Algorithms (GA) (a very interesting introduction to genetic algorithms and other evolutionary algorithms may be found in [23]) are optimization algorithms which are frequently cited as "partially simulating the process of natural evolution". Although this a suggestive analogy behind which, indeed, lies the original motivation for their inception, it is better to understand them as a kind of algorithms which take advantage of the implicit (indeed, unavoidable) granularity of the search space which is induced by the use of the finite binary representation in a digital computer.

In such finite space numbers originally thought of as existing in $\Re^n$ actually map into $\boldsymbol{B}^m$ space. Thereafter it is simple to establish that a genetic algorithmic process is

[1] The base for the logarithms is arbitrary. When (as above) we choose base 2 the information is measured in "bits".

a finite Markov chain (MC) whose states are the populations arising from the so-called genetic *operators*: (typically) selection, crossover and mutation. As such they display all of the properties of a MC. From this fact one may infer the following mathematical properties of a GA: 1) The results of the evolutionary process are independent of the initial population and 2) A GA preserving the best individual arising during the process will converge to the global optimum (albeit the convergence process is not bounded in time). For a proof of these facts the interested reader may see [24]. Their most outstanding feature is that, as opposed to other more traditional optimization techniques, the GA iterates simultaneously over *several* possible solutions. Thereafter, other plausible solutions are obtained by combining (*crossing over*) the *codes* of these solutions to obtain hopefully better ones. The solution space (SS) is, therefore, traversed stochastically searching for increasingly better plausible solutions. In order to guarantee that the SS will be globally explored some bits of the encoded solution are randomly selected and changed (a process called *mutation*). The main concern of GA-practitioners (given the fact that well designed GAs, in general, will find the best solution) is to make the convergence as efficient as possible. The work of Forrest et al. has determined the characteristics of the so-called *Idealized GA* (IGA) which is impervious to GA-hard problems [25].

3.3 Vasconcelos' Genetic Algorithms

The implementation of the IGA is unattainable in practice. However, a practical approximation called the Vasconcelos' GA (VGA) has been repeatedly tested and proven to be highly efficient [26]. The VGA, therefore, turns out to be an optimization algorithm of broad scope of application and demonstrably high efficiency.
A statistical analysis was performed by minimizing a large number of functions and comparing the relative performance of six optimization methods[2] of which five are GAs. The ratio of every GA's absolute minimum (with probability $P=0.95$) relative to the best GA's absolute minimum may be found in Table 1 under the column "Relative Performance". The number of functions which were minimized to guarantee the mentioned confidence level is shown under "Number of Optimized Functions".

[2] VGA: Vasconcelos' GA; EGA: Eclectic GA; TGA: Elitist GA; SGA: Statistical GA; CGA: Canonical (or Simple) GA; RMH: Random Mutation Hill Climber.

Table 1. Relative Performance of Different Breeds of Genetic Algorithms

Algorithm	Relative Performance	Number of Optimized Functions
VGA	1.000	2,736
EGA	1.039	2,484
TGA	1.233	2,628
SGA	1.236	2,772
CGA	1.267	3,132
RHC	3.830	3,600

It may be seen that the so-called Vasconcelos' GA (VGA) in this study was the best of all the analyzed variations. Interestingly the CGA (the classical or "canonical" genetic algorithm) comes at the bottom of the list with the exception of the random mutation hill climber (RHC) which is not an evolutionary algorithm. According to these results, the minima found with the VGA are, on the average, more than 25% better than those found with the CGA. Due to its tested efficiency, we now describe in more detail the VGA.

Outline of Vasconcelos' Genetic Algorithm (VGA)

1. Generate random population of *n* individuals (suitable solutions for the problem).
2. Evaluate the fitness *f(x)* of each individual *x* in the population.
3. Order the *n* individuals from best (top) to worst (bottom) for $i=1, 2, \dots, n$ according to their fitness.
4. Repeat steps A-D (see below) for $i = 1,2,\dots,\lfloor n/2 \rfloor$.

   A. *Deterministically* select the *i-th* and the $(n-i+1)$*-th* individuals (the *parents*) from the population.
   B. With probability *Pc* cross over the selected parents to form two new individuals (the *offspring*). If no crossover is performed, offspring are an exact copy of the parents.
   C. With probability *Pm* mutate new offspring at each locus (position in individual).
   D. Add the offspring to a new population
5. Evaluate the fitness *f(x)* of each individual *x* in the new population
6. Merge the newly generated and the previous populations
7. If the end condition is satisfied, stop, and return the best solution.
8. Order the *n* individuals from best to worst ($i=1, 2, \dots, n$) according to their fitness
9. Retain the top *n* individuals; discard the bottom *n* individuals
10. Go to step 4

As opposed to the CGA, the VGA selects the candidate individuals deterministically picking the two extreme (ordered according to their respective fitness) performers of the generation for crossover. This would seem to fragrantly

violate the survival-of-the-fittest strategy behind evolutionary processes since the genes of the more apt individuals are mixed with those of the least apt ones. However, the VGA also retains the best *n* individuals out of the *2n* previous ones. The net effect of this dual strategy is to give variety to the genetic pool (the lack of which is a cause for slow convergence) while still retaining a high degree of elitism. This sort of elitism, of course, guarantees that the best solutions are not lost. On the other hand, the admixture of apparently counterpointed plausible solutions is aimed at avoiding the proliferation of similar genes in the pool. In nature as well as in GAs variety is needed in order to ensure the efficient exploration of the space of solutions[3]. As stated before, all GAs will eventually converge to a global optimum. The VGA does so in less generations. Alternatively we may say that the VGA will outperform other GAs given the same number of generations. Besides, it is easier to program because we need not to simulate a probabilistic process. Finally, the VGA is impervious to negative fitness's values.
We, thus, have a tool which allows us to identify the best values for a set of predefined metrics possibly reflecting complementary goals. This metric(s) may be arbitrary and this suits us well because one way to establish which of a set of software systems is better than the other is to: a) Define a metric or set of metrics. b) Determine the one system whose combination is best for the systems. For these reasons we use in our work the VGA as the optimization method. In what follows we explain our proposal based in the concepts mentioned above.

4 Evolutionary Entropic Clustering

Let X be a data set of elements x_i such that $x_{i=}\{x_{i1}, x_{i2}, \dots, x_{in}\}$, let D be an n-dimensional space such that $xi \in D$ and let c_j be a subset of D called cluster. Then we must find a function that associates each element of X to the *j-th* cluster c_j as:

$$f(x_i) = c_j; \forall X \ and \ 2 \le j \le k \qquad \textbf{(5)}$$

Where k is the number of clusters and $f(x_i)$ is called the membership function. Now we describe a method which attempts to identify those elements within the data set which share common properties. These properties are a consequence of (possibly) high order relationships which we hope to infer via the entropy of a quantized vector space. This space, in what follows, will be denoted as the *Hypercubic Wrapper*.

4.1 Hypercubic Wrapper

A *Hypercubic Wrapper* denoted as is an n-dimensional subspace of D such that:

[3] The Latin American philosopher José Vasconcelos proposed that the admixture of all races would eventually give rise to a better one he called the *cosmic* race; hence the algorithm's name.

$$x_i \in HW \quad \forall x_i \in X \qquad \textbf{(6)}$$

HW is set of elements v_m called voxels, which are units in n-dimensional that can contain zero or more elements of the set *X*. The cardinality of *HW* is given by the number the voxels that we specify in each dimension of the space *D* such that:

$$|HW| = \prod_{i=1}^{n} L_i \qquad \textbf{(7)}$$

Where L_i is the number of voxels in the *i-th* dimension and *n* is the dimension number of *D*. From equation (7) it follows that $\forall v_m \in HW$:

$$0 < m \leq \prod_{i=1}^{n} L_i \qquad \textbf{(8)}$$

Figure 4 shows a graphical representation of *HW* in a tridimensional space, where the cardinality is equal to 27 (because there are three voxels in each dimension, i.e. $L_i = 3$).

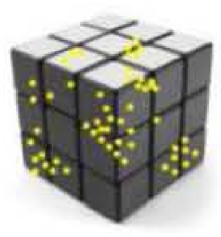


Fig. 3. Hypercubic Wrapper in a tri-dimensional space

However, in general, $L_i \neq L_j \; \forall i, j$ and it is, therefore, possible to define different *HWs* by changing the values of the L_i's as shown in Figure 4

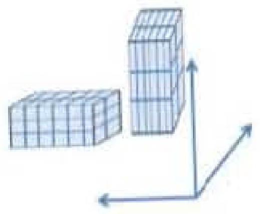


Fig. 4. Hypercubes with different lengths per dimension

Once this wrapper's characteristics are defined, we may use a clustering method based in the concept of *Self Information*, *Entropy*, and *VGA*. Now we describe our proposal which we call the Fixed Grid Evolutionary Entropic Algorithm (FGEEA).

4.2 Fixed Grid Evolutionary Entropic Algorithm

Definition 1. Every voxel that includes at least one element of the data set *X* is called a non-empty voxel; otherwise it is called an empty voxel

Definition 2. For the purpose of entropy calculation, any non-empty voxel is identified with the i-th symbol and will be denoted by s_i.

Definition 3. The space of all symbols is an alphabet denoted by Σ.

Definition 4. The data set is equivalent the source of information S. Such source only produces symbols of Σ.

Definition 5. The probability that the symbol s_i is produced by *S* is the number of its elements divided by the cardinality of the data set *X*. This probability is denoted as $p(s_i)$.

Corollary. The density of a symbol s_i is proportional to its probability $p(s_i)$.

It follows that:

$$|\Sigma| \leq |HW| \tag{9}$$

$$H(S) = \sum_{i=1}^{n} p(s_i) I(s_i) = -\sum_{i=1}^{n} p(s_i) \log_2 p(s_i) \tag{10}$$

Our idea is to use the entropy for determine the membership of every symbol in the *j-th* cluster, based on the follows assumptions:

Assumption 1. The source *S* always produces the same symbols for a given data set *X*.

Assumption 2. The spatial position of each symbol (voxel) is invariant for a given data set *X*. Therefore, H(S) is constant.

According to the working hypothesis, the areas with high density have minimum entropy with respect to areas with low density. Then the areas with minimum entropy possibly identify a cluster.

To determine the entropy of a possible cluster we introduce a term we call *intracluster entropy*, defined as:

$$H(c_i) = -\sum p(s_j) \log_2 p(s_j) \quad \forall \ s_j \in c_i \tag{11}$$

Where $H(c_i)$ is the intracluster entropy of *i-th* cluster. In order to determine that s_j belongs to c_i we use a genetic algorithm, as discussed in what follows.

Application of Vasconcelos' Genetic Algorithm

Our aim is to find the areas of the subspace *HW* where the entropy is minimal. This is to find groups of voxels such as each group have minimal entropy (intracluster entropy).

Clearly this is an optimization problem. For reasons explained above we use a VGA. The individuals of the algorithm have been encoded as follows. a) The length of the genome is equal to the cardinality of Σ. [It is composed by all symbols (or g*enes*)]. b) Each gene is assigned a label that represents the cluster to which it belongs. c) It has a sequential index. Such index will allow mapping all symbols to subspace HW. Fig.5 exemplifies a genome for *k=3*.

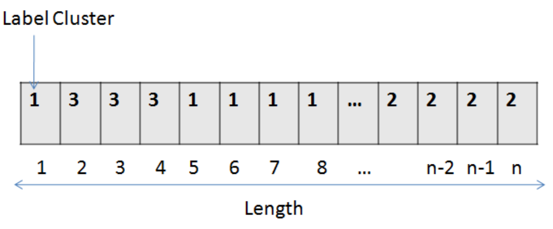


Fig. 5. Genome of the individual (*k=3*).

Now, we defined the fitness function as:

$$f(individual_j) = min \sum_{i=0}^{k} H(c_i) \quad for\ j \leq N \tag{12}$$

Subject to:

$$\sum_{i=0}^{k} H(c_i) \geq H(S) \tag{13}$$

$$\left|\sum_{i=0}^{k} H(c_i) - H(S)\right| \geq \Delta_1 \tag{14}$$

Where *N* is the size of the population and Δ_1 is a parameter that represents a threshold of the difference between the sum of intracluster entropies and the entropy of source *S*. Additionally we have introduced a constraint called *intracluster density* defined as:

$$dc_i \leq \varepsilon \tag{15}$$

where ε is the threshold density. One last constraint is the intracluster density (*dc*$_i$). It is the number of elements of data set *X* which belong to the symbols of *i-th* cluster:

$$dc_i = \frac{\alpha}{\beta} \tag{16}$$

where α is the number the elements that belong to the data set *X* and β is the number of symbol within cluster *i*. This constraint ensures that entropy is minimal within any given cluster. The algorithm yields a best individual which represents a set of clusters of symbols that are map into sets of voxels in the subspace *HW*, as shown in Fig. 6.

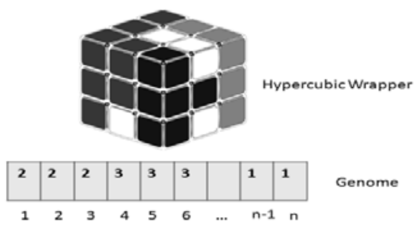


Fig. 6. Possible clustering delivered by the VGA. Different intensities in the cube represent a different cluster. White voxels are empty.

In what follows we show some experiments that allow us to test the effectiveness of the algorithm presented previously

5 Experimental Results

Our algorithm was tested with a synthetic data set which consists of a set of points contained by three disjoint spheres. The features and parameters of the first test are given in Table 2. The values of the parameters were determined experimentally.

Table 2. Features and parameters first test.

Feature	Value	Parameter	Value
Sample size	192	N (Number of individuals)	500
Elements per cluster	64	G (Generations)	1000
Dimensions	3	Pm (Mutation Probability)	0.001
Data Distribution	Disjoint sphere	Pc (Crossover Probability) Δ_i	0.99 $3.5<\Delta_i<3.6$
Cardinality of ∑	26	ε	5

The VGA was run 20 times (with different seeds of the pseudo random number generator) yielding an average effectiveness of 98%. Notice that no information other than the number of clusters is fed to FGEEA.

The same data set was tested with other algorithms such as *Kohonen Maps* and *Fuzzy C-Means*. The results obtained are shown in Table 3.

Table 3. Results obtained with Kohonen Maps and Fuzzy C-Means.

Algorithm	Average Effectiveness
Kohonen Maps	0.99
Fuzzy C- Means	0.98

This us allow see that the result of our proposal is similar to result given by some traditional algorithms. The high effectiveness in all cases is probably due to the spatial distribution of data set.

Next, we test with other data set whose spatial distribution yields presents overlapping clusters as is shown in Fig. 11. For clarity we show a bi-dimensional example. The actual runs consisted of three dimensional data.

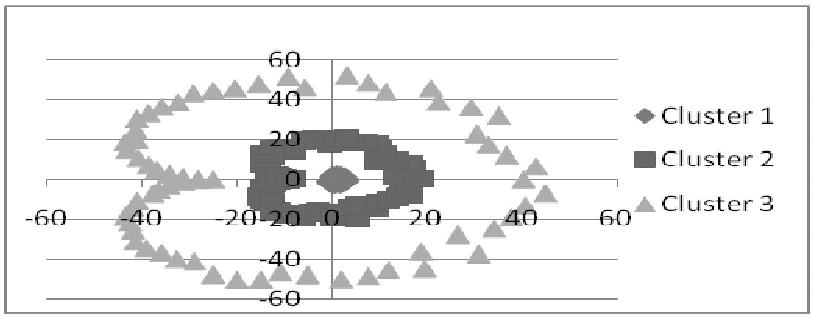


Fig. 7. Overlapping clusters.

In this case also the size sample is 192 elements distributed in three clusters a priori whose cardinality is 64 elements. The results obtained are shown in Table 4.

Table 4. Results of FGEEA with overlapping data set

Algorithm	Average Effectiveness
Kohonen Maps	0.62
Fuzzy C- Means	0.10
FGEEA	0.73

Here the effectiveness decreases significantly in general. But FGEEA showed the better results.

Finally we tested our algorithm with a data set in tridimensional space with an unknown spatial distribution. For *k = 3* (number of clusters) the algorithm found a solution that is shown in Fig. 8. Here the clusters are irregularly shaped.

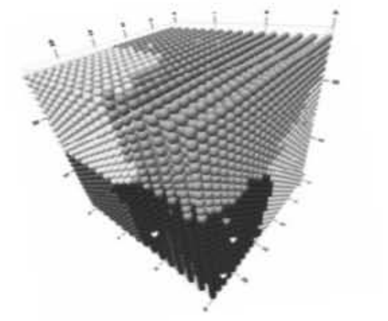


Fig. 8. Irregular clusters. The number of voxels is 15625 (25 voxels per dimension). The white voxels are empty.

These last results were not compared with other clustering algorithms. However, we can see that in principle our approach is feasible.

6 Conclusions and Future Work

These results allow us to test the feasibility of our algorithm. This is not enough, however, to assume its effectiveness in general. To achieve this proof we require testing with several data sets and applying more solid clustering validation techniques. Computationally, the analysis of the geometric and spatial membership relation between elements of a multidimensional data set is hard. Our approach showed that in principle, membership relations in a data set can be found through of its entropy without an excessive demand on computational resources. Even though the results obtained are limited (since they correspond to particular cases and in tri-dimensional data) they are promissory. Therefore, future work requires to generalize our method for a data set in n-dimensional space (with $n>3$), to analyze its computational complexity and to test its detailed mathematical formulation. We will report on these issues shortly.

References

1. Cha, Sung Hyuk: Taxonomy of Nominal Type Histogram Distance Measures, Massachusetts (2008).
2. Mahalanobis, Prasanta Chandra: On the genaralized distance in statistics. (1936).
3. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by probability distributions, Calcutta (1943).
4. Pollard, David E.: A user's guide to measure theoretic probability. Cambridge University Press, Cambridge (2002).
5. Yang, Grace Lo y Le Cam, Lucien M.: Asymptotics in Statistics: Some Basic Concepts. Springer, Berlin(2000).
6. Li, Xin, Wai, Man y Kwong Li, Chi: Determining the Optimal Number of Clusters by an Extended RPCL Algorithm. Hong Kong Polytechnic University, Hong Kong (1999).

7. MacQueen, J.B.: Some Methods for Classification and Analysis of Multivariate Observations. In :Proceedings of 5th Berkley Sysmposium on Mathematical Statiscs and Probability, pp. 281-297, Berkley (1967).
8. Ng, R y Han, J.: Effecient and Effective Clustering Methods for Spatial Data Mining, Santiago de Chile (1994).
9. Zhang, T, Ramakrishnman, R y Linvy, M.: BIRCH: An Efficient Method for Very Large Databases, Montreal, Canada (1996).
10. Guha, S, Rastogi, R y Shim, K: An efificient Clustering Algorithm for Large Databases, (1998).
11. Ester, M, Kriegel, H., Sander, J., Xu X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, pp. 226-223, Portland (1996).
12. Hinneburg, A y Keim, D.: An Efficient Approach to Clustering in Large Multimedia Databases with noise, (2000).
13. Wang, Wei, Yang, Jiong y Muntz, Richard. STING : A Statistical Information Grid Approach to Spatial Data. In: Proceedings of the 23rd VLDB Conference, Athens (1997).
14. Sheikholeslami, Gh., Chatterjee, S. y Zhang, A.: Wavecluster: A multi-resolution clustering. In : Proceedings of the 24th VLDB conference, (1998).
15. Dunn, J. C.: A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, pp. 32-57, (1973)
16. Kohonen, T.: Self-Organizing Maps. In: Series in Information Sciences, (1995).
17. Halkidi, M, Batistakis, Y. y Vzirgiannis, M.: On Clustering Validation Techniques, pp. 107-145, (2001)
18. Cheng, C., Fu, Ada W. y Zhang, Y.: Entropy- based Subspace Clustering for Mining Numerical Data, (1998).
19. Barbará, D., Julia, C. y Li, Y.: COOLCAT: An entropy-based algorithm for categorical clustering, George Mason University (2001).
20. Shannon, Claude E. A mathematical theory of communication, pp. 379-423, (1948)
21. Kolmogorov, A. N.: Three approaches to the quantitative definition of information, pp. 1-7, (1948).
22. Gray, Robert M.: Entropy and Information Theory. In : Springer Verlag, (2008).
23. Bäck, T.: Evolutionary Algorithms in Theory and Practice. In: Oxford University Press, (1996).
24. Rudolph, G. Convergence Analysis of Canonical Genetic Algorithms. In. : IEEE Transactions on Neural Networks, (1994).
25. Forrest, S. and Mitchell, M. What makes a problem hard for a genetic algorithm? In : Machine Learning, (1993).
26. Kuri, Angel. A Methodology for the Statistical Characterization of Genetic Algorithms. In : Springer-Verlag, págs. 79-88, (2002).

A methodology to find clusters in the data based on Shannon's Entropy and Genetic Algorithms

Edwin Aldana-Bobadilla[1], **Angel Kuri-Morales**[2]
[1]Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas, Universidad Nacional Autónoma de México, Mexico City, Mexico
[2]Departamento Académico de Computación, Instituto Tecnológico Autónomo de México, Mexico City, Mexico

Abstract - *The most common clustering methods are based on metrics that allow the determination of the similarity between elements of a given data set. This similarity allows us to divide the data set into subsets (clusters) that contain "highly similar" elements. The use of a metric imposes two constraints. First, the shape of the found clusters is generally hyper-spherical (in the space of the metric) due to the fact that each element in a cluster lies within a radial distance relative to a given center. Second, the metric may be sensitive to the probability density function of the data set. Following this fact several methods based on statistical approaches have become an attractive and powerful option. These involve the estimation of the probability density function (pdf) of the data set which minimizes an optimality criterion. Generally this is a highly non-linear and usually non-convex optimization problem which disallows the use of traditional optimization techniques. In this paper we propose a statistical method based on Shannon's Conditional Entropy which uses a rugged genetic algorithm to find the optimal pdf. Each individual of the Genetic Algorithm is a possible solution of a clustering problem. The fitness of an individual is determined by Shannon's entropy encoded in its genome and an additional constraint related to the "quality" of this solution. The "quality" is measured through a validity index of the clustering process. A novel and important aspect of our method is the form of representation of the objects of the data set in order to reduce the computational complexity due to the high dimensionality. We show that our proposal has high effectiveness relative to methods as k-means, fuzzy c-means and Kohonen Maps with a synthetic data set.*

Keywords: Clustering, Information Theory, Genetic Algorithms, Bayesian Classifier, Data Mining.

1 Introduction

The clustering process is an optimization problem that maximizes the similarity between objects or elements that belong to same cluster and minimizes the similarity between elements of different clusters. The effectiveness of a clustering method is given by several factors such as the metric and the desired number of clusters.

Particularly, the use of a metric imposes some constraints on the shape of clusters found. These shapes generally are hyperspherical (in the space of the metric) due to the fact that each element in a cluster lies within a radial distance relative to a given center. In other words the elements of a cluster tend to group around a single mean value (center) which sometimes disallows the extraction of hidden patterns in the data set.

In this paper we propose an alternative method based on a statistical approach. Our proposal does not use explicitly a metric to determine the elements that belong to given cluster. Overall, this proposal is an iterative search of a partition model of the data set in which the entropy (uncertainty) is minimized. In order to determine the entropy of the data set for a particular partition model, the estimation of its probability density function (pdf) is necessary. This estimation can be achieved statistically from three different methods: parametric, semi-parametric and non-parametric [15]. Unlike parametric and semi-parametric methods, the non-parametric methods do not make any assumption about of the pdf of the data set. The Parzen window [5] is among the most widely-used non-parametric density estimation method.

Different clustering methods have been proposed around these non-parametric methods and minimum entropy principle [9], [15],[16]. These methods can be seen as an iterative search of an optimal pdf of the data set such that the entropy is minimal. However, depending on the dataset the search may be unfeasible or may yield local optimal solutions. Thus, this is a highly non-linear and usually non-convex optimization problem which disallows the use of traditional optimization techniques or pdf estimation methods.

We propose a method which uses a rugged genetic algorithm (the so-called Vasconcelos's GA [12]). Each individual of GA is a possible solution of a clustering problem which represents a pdf of the data set. The fitness of an individual is based on the minimum entropy principle and an additional constraint related to the "quality" of the solution. The "quality" is measured through an validity indices of the clustering process. Several validity indices have been developed and introduced [4],[8], [11]. A novel and important aspect of our proposal is the form of representation of the objects of the data set. Generally the properties of each object are represented as real values of vector in a Euclidean space. The dimensionality of this vector is given by the number of such properties. Its value is an important element of the computational complexity of a clustering algorithm.

In order to reduce the dimensionality, statistical techniques as such as Pearson's correlation analysis [3] and/or principal components' analysis [17] have been used. In many cases, however, these techniques are sensitive to the data distribution and impair the effectiveness of the clustering process. To avoid this fact we map the *n*-dimensional vector space of the data set to the space of all possible strings (words) that can be built using the symbols of an alphabet $\sum$. This transformation allows us to represent an object of data set as a word of length *n* (for a *n*-dimensional space) and a cluster as a subset of words with some "degree of similarity". The entropy of a cluster is determined by the probability distribution of all words that belong to that cluster.

Our work begins with an account of several concepts which are needed to expose our method. Then, we expound the fundamental process of our proposal. Finally we show several numerical results and the respective conclusions.

2 Theoretical Aspects

In what follows we make a very brief mention of the theoretical aspects having to do with the proper understanding of our proposal. The reader may find more details in the references.

2.1 Minimum Entropy Principle

Shannon's entropy [20] allows us to measure the uncertainty associated with a random variable *X*. Mathematically, Shannon's entropy of *X* with a probability mass function *p(x)* is defined as:

$$H(X) = -\sum p(x)\log(p(x)) \tag{1}$$

The possible values of a random variable *X* occur with certain probability *p(X=x)* or simply *p(x)*. When *p(x)* is uniformly distributed we say that the uncertainty is greatest or that the process represented by the random variable *X* has a highest degree of "disorder". Figure 1 represents the entropy for two possible values of *X* with probabilities *p* and *1- p*; when *p=0.5* the entropy is maximum.

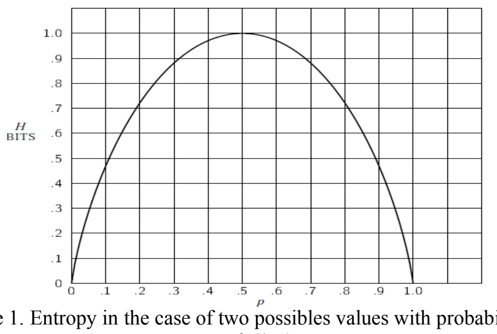


Figure 1. Entropy in the case of two possibles values with probabilities *p* and *(1-p)*

In the context of the clustering problem we assume that a cluster is a subset of the data set which has minimum entropy. It means that a cluster is a partition of data set with minimum degree of "disorder". The entropy of a cluster is directly related to its elements. In terms of probability, the entropy of the cluster depends of the pdf of its elements. In what follows we expound on this fact.

Let *D* be the data set with *K* partitions (clusters) and *x* an element that belong to *D*. Then the pdf of *x* is given by:

$$p(x) = \sum_{i}^{K} p(x|i)\,p(i) \tag{2}$$

where *p(i)* is the prior probability for the *i-th* partition and *p(x|i)* is the prior probability of *x* given the *i-th* partition. In Figure 2 we show an intuitive representation of the probabilities *p(x|1)* and *p(x|3)*, the probability *p(x|2)* is zero.

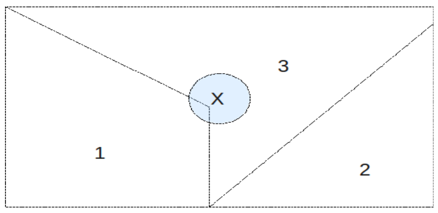


Figure 2. Probability space of a data set with three partitions. The element *x* belongs to partition 1 and 2 with a probability greater than zero.

However we would like to know the dependence of pdf of the *i-th* partition with respect to *x*. This dependence is given by Bayes Theorem [10] :

$$p(i|x) = \frac{p(x|i)\,p(i)}{p(x)} \tag{3}$$

When *p(i|x)* is uniformly distributed for all i, we can say that the element *x* belongs to any partition and thus the uncertainty is maximum (see Figure 3a.). On the other hand if all *p(i|x)* but one are zero (one having the value unity) then we are certain of the partition to which *x* belongs (see Figure 3b).

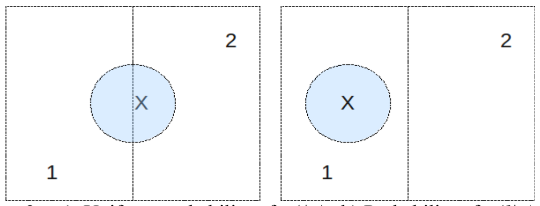


Figure 3. a) Uniform probability of *p(i|x)* . b) Probability of *p(1|x)*

Now, let *C* be a random variable whose possible values are *1,2,..K* which represent the partitions of *D*. Let *X* be a random variable whose possible values are all elements *x* that belong to *D*. Then the entropy of *C* given *X* is:

$$H(C|X) = -\sum_{i=1}^{K} p(i|x)\log(p(i|x)) \tag{4}$$

where *p(i|x)* is a posteriori pdf. Thus, our goal is to find this function such that *H(C|X)* is minimum. For reasons already mentioned we use a genetic algorithm. The entropy given by Equation 4 is called *Conditional Entropy* [1].

2.2 Genetic Algorithms

Genetic Algorithms (an interesting introduction to GA's and other evolutionary algorithms may be found in [2]) are optimization algorithms which are frequently cited as "partially simulating the process of natural evolution". Although this a suggestive analogy behind which, indeed, lies the original motivation for their inception, it is better to understand them as a kind of algorithms which take advantage of the implicit (indeed, unavoidable) granularity of the search space which is induced by the use of the finite binary representation in a digital computer. In such finite space, numbers originally conceived as existing in R^n actually map into B^m space. Thereafter it is simple to establish that a genetic algorithmic process is a finite Markov chain (MC) whose states are the populations arising from the so called genetic operators: (typically) selection, crossover and mutation [19]. As such they display all of the properties of a MC. From this fact one may prove that:

1. The final results of the evolutionary process are independent of the initial population and

2. A GA preserving the best individual arising during the process will converge to the global optimum (albeit the convergence process is not bounded in time).

Their most outstanding feature is that, as opposed to other more traditional optimization techniques, the GA iterates simultaneously over several possible solutions. Then, other plausible solutions are obtained by combining (crossing over) the codes of these solutions to obtain hopefully better ones. The solution space (SS) is, therefore, traversed stochastically searching for increasingly better plausible solutions. In order to guarantee that the SS will be globally explored some bits of the encoded solution are randomly selected and changed (a process called mutation). The main concern of GA-practitioners (given the fact that well designed GAs, in general, will find the best solution) is to make the convergence as efficient as possible. The work of Forrest et al. has determined the characteristics of the so-called Idealized GA (IGA) which is impervious to GA-hard problems [6].

2.2.1 Vasconcelos's Genetic Algorithm

The implementation of the IGA is unattainable in practice. However, a practical approximation called the Vasconcelos's GA (VGA) has been repeatedly tested and proven to be highly efficient [12]. The VGA, therefore, turns out to be an optimization algorithm of broad scope of application and demonstrably high efficiency. A statistical analysis was done by minimizing a large number of functions and comparing the relative performance of six optimization methods of which five are GAs[1]. The ratio of every GAs absolute minimum (with probability $p = 0.95$) relative to the best GAs absolute minimum may be found in Table 1 under the column "Relative Performance". The number of functions which were minimized to guarantee the mentioned confidence level is shown under "Number of Optimized Functions". It may be seen that VGA, in this study, was the best of all the analyzed variations. Interestingly the CGA (the classical or "canonical" genetic algorithm) comes at the bottom of the list with the exception of the random mutation hill climber (RHC) which is not an evolutionary algorithm. According to these results, the minimal found with VGA are, in the worst case, more than 25% better than those found with the CGA. Due to its tested efficiency, we now describe in more detail VGA.

As opposed to the CGA, VGA selects the candidate individuals deterministically picking the two extreme (ordered according to their respective fitness) performers of the generation for crossover. This would seem to fragrantly violate the survival-of-the-fittest strategy behind evolutionary processes since the genes of the more apt individuals are mixed with those of the least apt ones. However, VGA also retains the best *n* individuals out of the *2n* previous ones.

Table 1: Relative Performance of Different Breeds of Genetic Algorithms

Algorithm	**Relative Performance**	**Number of Optimized Functions**
VGA	1.000	2,736
EGA	1.039	2,484
TGA	1.233	2,628
SGA	1.236	2,772
CGA	1.267	3,132
RHC	3.830	3,600

The net effect of this dual strategy is to give variety to the genetic pool (the lack of which is a cause for slow convergence) while still retaining a high degree of elitism. This sort of elitism, of course, guarantees that the best solutions are not lost. On the other hand, the admixture of apparently counterpointed plausible solutions is aimed at avoiding the proliferation of similar genes in the pool. In nature as well as in GAs variety is needed in order to ensure the efficient exploration of the space of solutions. As stated before, all elitist GAs will eventually converge to a global optimum. The VGA does so in less generations. Alternatively we may say that VGA will outperform other GAs given the same number of generations. Besides, it is easier to program because we need not simulate a probabilistic process. Finally, VGA is impervious to negative fitness's values. We, thus, have a tool which allows us to identify the best values for a set of predefined metrics possibly reflecting complementary goals. For these reasons we use in our work VGA as the optimization method. In what follows we explain our proposal based in the concepts mentioned above.

[1]VGA: Vasconcelos' GA; EGA: Eclectic GA; TGA: Elitist GA; SGA: Statistical GA; CGA: Canonical (or Simple) GA; RMH: Random Mutation Hill Climber.

3 Methodology

We begin our explanation by discussing the preprocessing of the data set. It will allow us to change the vector representation of the data in order to facilitate subsequent calculations. Second, we show the details of the genome's encoding in the context of the clustering problem. Finally we show the way to evaluate each solution or individual in order to find the best.

3.1 Preprocessing of the data set.

Let Σ be an alphabet and w a string that contains symbol of Σ. Let D be a data set. Let $x_i = \{a_1, a_2, \ldots a_n\}$ be an n-dimensional vector such that $x_i \in D$ where $a_i \in \boldsymbol{R}$ and $D \in \boldsymbol{R}^n$.

Let $\perp a_k$, $\top a_m$ be the minimal and maximal value $\forall\, a_i \in D$. Let Δ be the difference between $\top a_m$ and $\perp a_k$, then we assign to every symbol of Σ an interval value as following:

Table 2: Assigning values to symbols of Σ

Symbol	Interval Value
s_0	$\left[\perp a_k, \perp a_k + \frac{\Delta}{\lvert\Sigma\rvert}\right]$
s_1	$\left[s_{0\,max}, s_{0\,max} + \frac{\Delta}{\lvert\Sigma\rvert}\right]$
…	…
s_m	$\left[s_{m-1\,max}, s_{m-1\,max} + \frac{\Delta}{\lvert\Sigma\rvert}\right]$

Where $s_{i\,max}$ is the maximum interval value of S_i and $|\Sigma|$ is the cardinality of Σ ($m=|\Sigma|$). Now we assume that Σ is conformed by the letters of the English alphabet and $\top a_m = 1$ and $\perp a_k = 0$. In accordance with Table 2 we can determine the interval values of Σ as shown in Table 3.

Table 3: Possible assignment of values for letters of the English alphabet

Symbol	Symbol Value
A	$\left[0, 0+\frac{1}{26}\right]$
B	$\left[\frac{1}{26}, \frac{1}{26}+\frac{1}{26}\right]$
…	…
Z	$\left[\frac{25}{26}, 1\right]$

Moreover, if we assume a data set D in R^3 such that some x=[0.038,0.022,0.99]. Then x may be represented by w=AAZ. Thus, $\forall x \in D$, $\exists w \in \Sigma^*$. We represent the set of all strings or words w as D'. For practical reasons we use the English Alphabet although the method described does not depend on any particular symbol set. However this method will be affected by the cardinality of Σ. For example, if $|\Sigma| = 1$ we have that all elements of the data set are represented by the same word regardless of their degree of similarity. Otherwise when the value of $|\Sigma|$ is higher we will have more precision but the performance will be affected.

3.2 Encoding of the genome.

The individuals of the algorithm have been encoded as follows. a) The length of the genome is equal to the cardinality of D'. b) Each gene is associated with a word of D'. The value of each gene corresponds to a label (for practical purposes we use $1,2,\ldots K$) of the cluster to which the word belongs. Thus, the i-th gene represent the cluster to which the i-th word belongs. Figure 4 exemplifies a genome for $K=3$.

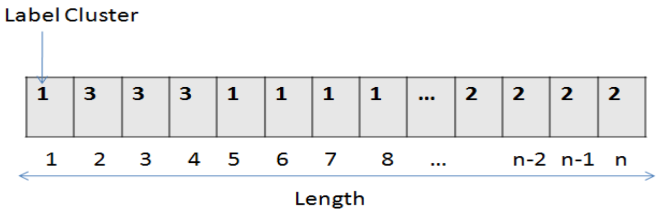


Figure 4. Genome of the individual (K=3)

3.3 Fitness

Each individual is a possible solution of a clustering problem which is evaluated through a fitness function. In what follows we explain how this function is defined in the context of our method.

Suppose that D'={AAA, ACA, MOM, NPM, ADE, UVT, VXT, NQP, VWV} and $K=3$. Let I_i be the i-th individual of the population whose genome are shown in Figure 5.

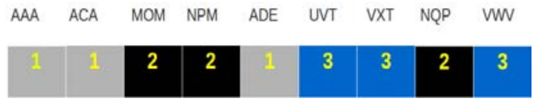


Figure 5. Possible solution given by an Individual for K=3. Here are shown the words associated to each gene.

As discussed above we use the Minimum Entropy Principle. In Equation 4 X is a random variable whose set of possibles values belongs to D. Thus, if the data set D is transformed to set D' (conformed by words w) then Equation 4 may be rewritten as:

$$H(C|W) = -\sum_{i=1}^{K} p(i|w)\log(p(i|w)) \qquad (5)$$

Where W is a random variable whose possibles values are strings of the Σ alphabet. We can calculate $H(C|W)$ for all individuals based on their genomes. This entropy may be expressed as the sum of entropies for each cluster as follows:

$$H(C|W)=\sum_{i=1}^{K} H(i|W) \qquad (6)$$

Where *H(i|W)* is the entropy of cluster *i.* The idea is to minimize the entropy for each cluster. However, this fact involves a multi-objective optimization problem because minimizing the entropy of a cluster affects the entropy of any other. To resolve this problem we apply Pareto's Efficiency [18]. Our objective function may be written as:

$$min[H(1|W),H(2|W)\ldots H(K|W)] \qquad (7)$$

So, the GA must find the individuals that minimize this function which is represented as a vector of K dimensions. In what follows this vector is called *Entropy Vector*. Each individual has a Entropy Vector whose values are given by its genome. In order to determine the individual with the best vector, we apply the principle of *Pareto Dominance* [18]. The Pareto Dominance says that a Y vector dominates to Y^* if $\forall y_i \in Y$, $y_i \le y_i^*$ and $\exists y_p$ such that $y_p < y_p^*$. In the context of VGA, a solution vector X of an Individual will dominate other solution vectors. The number of vectors dominated by X are called the *dominance value.* Thus, individuals with higher dominance value will be the best. The result of the evolutionary process yields a *Pareto Front*[18]. The fitness function for *i-th* individual (I_i) may be written as:

$$f(I_i)=dom_i \qquad (8)$$

Where dom_i is the dominance value of the i^{th} individual. However this function does not always assure that an individual with maximal dominance value is the best solution to the clustering problem. We, therefore propose a quality measure.

Our quality measure is based on the concept of *Mutual Information (MI)* [21]. It is a symetric measure that quantifies the mutual dependence between two random variables or the information that these share. In the context of our problem, the MI between two cluster u and v is given by:

$$I(u,v)=\sum_{i=1}^{R}\sum_{j=1}^{S} p(w_i,w_j)\log\frac{p(w_i,w_j)}{p(w_i)p(w_i)} \qquad (9)$$

where R and S are $|u|$ and $|v|$ respectively and $p(w_i, w_j)$ is the probability that the words w_i and w_j are similar. This probability is given by:

$$p(w_i,w_j)=\frac{|w_i \cap w_j|}{length(w_i)} \qquad (10)$$

where the intesection between two word is given by their common symbols. Clearly, all words of D' have the same length.

If $u \neq v$ then the value of *I(u,v)* will be called *Mutual Information Intercluster* (MI_{Inter}) . Otherwise this value will be called *Mutual Information Intracluster* (MI_{Intra}). A lower value of MI_{Inter} and higher value of MI_{Intra} means better clusters. So, we propose a quality measure given by:

$$Q=\frac{\sum_{i=1}^{K} MI_{Intra}(i,i)}{\sum_{i,j\le K,i\neq j} MI_{Inter}(i,j)} \qquad (11)$$

An individual with higher value of Q means a better solution. Therefore the fitness function of the i^{th} individual may be defined as:

$$f(I_i)=dom_i Q_i \qquad (12)$$

However, we observe that an individual with a “good” fitness value does not always represent a global optimum. Thus, we assume that each individual must be subject to the following constraint : *The probability for all partition (cluster) of D must be greater than zero. Mathematically* $p(i)>0\ \forall\ i=1,2,..K$ *(see Equation 2 and Equation 3).*

This constraint ensures that the individuals consists of non-empty clusters whose entropy is minimal. Otherwise the solutions will be outside of the feasible region. To encourage reproduction of feasible individuals (which represents feasible solutions) in every generation of VGA, we appeal to an penalty method [14] whose goal is to punish unfeasible individuals.

Here the penalty for unfeasible individual I_i is given by:

$$P(I_i)=J-\sum_{i=1}^{s}\frac{J}{m} \qquad (13)$$

where J is a large constant [$O(10^9)$], m is the number of constraints and s is the number of these which have been satisfied.

4 Numerical Experiment

In what follows we briefly describe how the test data set was generated. Subsequently we show several parameters and features of the performed tests. Finally we show the results. We call our proposal has been called *Entropic Evolutionary Clustering* (EEC).

4.1 The data set

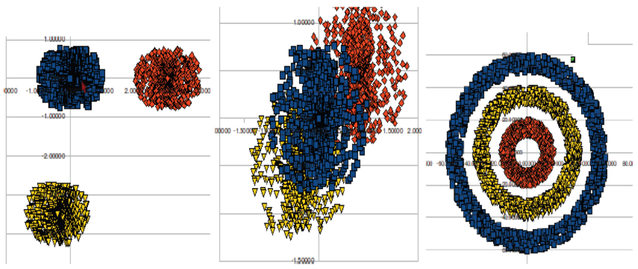
Three data sets are analyzed in this work. We shall call them "*A*", "*B*" and "*C*" respectively. Every set is composed of vectors (in a *3D* space) that belong to three different spheres which we call sphere 1, 2 and 3 respectively. There are 10,000 vectors in each one of the spheres. They were generated from.

$$x=x_0+r\sin\theta\cos\phi \qquad (14)$$

$$y=y_0+r\sin\theta\sin\phi \qquad (15)$$

$$z=z_0+r\cos\theta \qquad (16)$$

from uniformly distributed values for $r \in [0,1)$, $(0 \le \phi \le 2\pi$ and $0 \le \theta \le \pi)$. For set *A* the three centers of the spheres were chosen so that the spheres would not intersect. In set *B*, the chosen centers yield partially overlapping data. Finally, in set *C*, the spheres shared a common center. However, in the last set for sphere 1 $r \in [0,1)$; for sphere 2 $r \in [0, 0.666)$; for sphere 3 $r \in [0, 0.333)$. In this case, then, spheres 1, 2 and 3 share the same space where the density of 2 is larger than that of 1 and the density of 3 is larger than the other two. Our intent is to choose vectors in set A, B and C whose distribution is not uniform but Gaussian. To achieve this, we determined to divide the space of probabilities of a Gaussian curve in 20 equally spaced intervals. The area under the curve for a normal distribution with $\mu = 0$ and $\sigma = 1$ between -4 and +4 is very closely equal to one. Therefore, it is easy to see that 5%, of the observations will be between −4 and −1.654; 5%, will be between −1.654 and −1.280, etc. The required normal behavior may be approximated by selecting 50 of the uniformly distributed values from the interval $[-4,-1,654)$; another 50 from the interval $[-1.654,-1.280)$, etc. In all we will end up with 1000 vectors for every sphere. These vectors will now be very closely Gaussian. When data is normally distributed, a Bayesian classifier is optimal. The behavior of one such classifier will serve as a base point. To stress: when the distribution of the data set to classify is Gaussian, a Bayesian classifier yields the best theoretical results (by minimizing the probability of classification error independently of the degree of overlap between the distributions of the clusters) [7]. Hence, we resorted to Gaussian distributed data in order to establish a behavior relative to the best theoretical one when measuring the performance of non-traditional methods. Our claim is that, if the methods perform satisfactorily when faced with Gaussian data, they will also perform reasonably well when faced with other possible distributions. That is, we wish to show that the results obtained with non-traditional methods are close to those obtained with a Bayesian classifier for the same data set. This would mean that these results correspond to an efficient algorithm. The data sets are illustrated in Figure 6.



(a) Disjoint data set (b) Overlapped data set (c) Overlapped data set

Figure 6. Types of data set

5 Results

The values of the parameters of VGA are given in Table 4. These values were determined experimentally. As mentioned above we use the English Alphabet to transform the original data set. The VGA was run 20 times (with different seeds of the pseudo random number generator) per data set. The same data sets was tested with *K-Means* [22], *Kohonen Maps* [23] and *Fuzzy C-Means* [4]. Since it may be proven that a Bayesian Classifier is optimal when the data's pdf is Gaussian [7], we include a comparison with such a Bayesian Classifier. The results obtained with disjoint clusters are shown in Table 5. This allows us to see that the results of EEC are similar to those given by some alternative algorithms. The high effectiveness in all cases is due to the spatial distribution of data set. The results obtained with overlapping clusters are shown in Table 6 where we can see that the effectiveness decreases significantly in general.

Table 4: Parameters Test

Parameter Name	**Values**
N (Number of Individuals)	50
G (Generations)	4000
pm (Mutation probability)	0.00
pc (Crossover Probability) 0.99	0.99

However EEC showed better results than traditional methods and close results to Bayesian Classifier. The results obtained in the two last cases (overlapping and concentric clusters) are due to the fact that it is not possible to find a simple separable boundary. Therefore, the boundary decision is unclear and the vast majority of the clustering methods yield poor solutions. The closeness of the results obtained so far relative to a Bayesian Classifier, tells us that our approach is quite efficient. In future works we will report on experiments encompassing a wider range of data sets.

Table 5: Results obtained with disjoint clusters data set

Algorithm	**Average Effectiveness**
EEC	0.99
K-Means	0.98
Kohonen Maps	0.99
Fuzzy C-Means	0.98
Bayesian Classifier Effectiveness	0.99

Table 6: Results obtained with overlapping clusters data set

Algorithm	**Average Effectiveness**
EEC	0.87
K-Means	0.45
Kohonen Maps	0.72
Fuzzy C-Means	0.15
Bayesian Classifier Effectiveness	0.89

Table 7: Results obtained with concentric clusters data set

Algorithm	**Average Effectiveness**
EEC	0.71
K-Means	0.36
Kohonen Maps	0.38
Fuzzy C-Means	0.15
Bayesian Classifier Effectiveness	0.72

6 Conclusion

Following the minimum entropy principle we employed a genetic algorithm so that we were able to explore the solution space of the clustering problem. This approach resulted a better effectiveness with different data sets respect to *K-Means*, *Kohonen Maps* and *Fuzzy C-Means.* If we consider that Bayesian Classifier represents a theoretical limit then the most interesting result is the nearness of EEC respect this classifier. Our method promises to be a feasible alternative to find non-spherical clusters due to the results obtained with the concentric clusters of the data set C. However, we require testing several data sets that allow us to statistically ascertain that our method is good. We will report on these issues shortly. Additionally, data preprocessing proved to be a good alternative to reduce the computational complexity when the dimensionality of the data set is fairly high.

7 References

[1] Arndt, C., Information measures: information and its description in science and engineering, *Springer Verlag,* 2001

[2] Bäck, Th., Evolutionary Algorithms in Theory and Practice, *Oxford University Press*, 1996

[3] Cohen, J., Applied multiple regression/correlation analysis for the behavioral sciences, *Lawrence Erlbaum*, 2003

[4] Dunn, J. C., A Fuzzy Relative of the ISODATA Process and Its Use in Detecting CompactWell-Separated Clusters, *Journal of Cybernetics 3*, volume 3, 32–57, 1973

[5] Parzen E.: On the estimation of a probability density function and the mode. *Annals of Math. Stats.*, 33:1065-1076, 1962.

[6] Forrest, and Mitchell, What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation, MACHLEARN: *Machine Learning 13*, volume 13, 1993

[7] Haykin, Simon, *Neural networks: A comprehensive foundation*, MacMillan,1994

[8] Halkidi, Maria, Batistakis, Yannis, and Vazirgiannis, Michalis, On Clustering Validation Techniques, J. Intell. *Inf. Syst. 17(2-3)*, volume 17, 107–145, 2001

[9] Jenssen, R., Hild, KE, Erdogmus, D., Principe, J.C., and Eltoft, T., Clustering using Renyi's entropy, Neural Networks, 2003. *Proceedings of the International Joint Conference on*, volume 1, 523–528, 2003

[10] Joyce, James, *Bayes Theorem, The Stanford Encyclopedia of Philosophy*, Fall 2008 edition, Eds: Zalta, Edward N., 2008

[11] Kovacs, Ferenc, and Ivancsy, Renata, *A novel cluster validity index: variance of the nearest neighbor distance*, *WSEAS Transactions on Computers*, volume 3, 477-483, March 2006

[12] Kuri-Morales, Angel, A Methodology for the Statistical Characterization of Genetic Algorithms, MICAI, volume 2313, *Springer*, 79–88, Eds: Coello, Carlos A. Coello, de Albornoz, Alvaro, Sucar, Luis Enrique, and Battistutti, Osvaldo Cairó, 2002

[13] Kuri-Morales, Angel, and Aldana-Bobadilla, Edwin, Finding Irregularly Shaped Clusters Based on Entropy, ICDM, volume 6171, *Springer*, 57–70, Eds: Perner, Petra, 2010

[14] Kuri-Morales, Angel and Gutiérrez-García, Jesús, Penalty Function Methods for Constrained Optimization with Genetic Algorithms: A Statistical Analysis, *MICAI, volume 2313, Springer*, 108–117, Eds: Coello, Carlos A. Coello, de Albornoz, Alvaro, Sucar, Luis Enrique, and Battistutti, Osvaldo Cairó, 2002

[15] Lee, Y., and Choi, S., Minimum entropy, k-means, spectral clustering, Neural Networks, 2004. *Proceedings IEEE International Joint Conference on*, volume 1, 2005.

[16] Li, H., Zhang, K., and Jiang, T., Minimum entropy clustering and applications to gene expression analysis, *IEEE Computer Society*, 2004.

[17] Pearson, K., Principal components analysis, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 6(2)*, volume 6, 559, 1901

[18] Podinovskii, VV, and Nogin, VD, Pareto-Optimal Solutions of Multicriteria Problems, *Nauka*, Moscow, 1982

[19] Rudolph, G., Convergence Analysis of Canonical Genetic Algorithms, *IEEE Transactions on Neural Networks 5(1)*, volume 5, 96–101, January 1994

[20] Shannon, C. E., and Weaver, W., The Mathematical Theory of Communication, *Scientific American*, July 1949

[21] Vinh, N.X., Epps, J., and Bailey, J., Information theoretic measures for clusterings comparison: is a correction for chance necessary?, *Proceedings of the 26th Annual International Conference on Machine Learning*, 1073–1080, 2009

[22] McQueen, J. B., Some Methods of Classification and Analysis of Multivariate Observations, *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 281–297, Eds: Cam, L. M. Le, and Neyman, J., 1967

[23] Kohonen Teuvo, Self-organizing maps, *Springer-Verlag,* New York, Inc., 199

Appendix G

Choosing The Best Genetic Algorithm

The Best Genetic Algorithm I

A Comparative Study of Structurally Different Genetic Algorithms

Angel Kuri-Morales

Instituto Tecnológico Autónomo de México
Río Hondo No. 1
México 01000, D.F.
México
akuri@itam.mx

Edwin Aldana-Bobadilla

Universidad Nacional Autónoma de México
IIMAS
México 04510, D.F.
México
edwynjavier@yahoo.es

Abstract. Genetic Algorithms (GAs) have long been recognized as powerful tools for optimization of complex problems where traditional techniques do not apply. However, although the convergence of elitist GAs to a global optimum has been mathematically proven, the number of iterations remains a case-by-case parameter. We address the problem of determining the best GA out of a family of structurally different evolutionary algorithms by solving a large set of unconstrained functions. We selected 4 structurally different genetic algorithms and a non-evolutionary one (NEA). A schemata analysis was conducted further supporting our claims. As the problems become more demanding, the GAs significantly and consistently outperform the NEA. A particular breed of GA (the Eclectic GA) is superior to all other, in all cases

Keywords. Global optimization; Genetic algorithms; Unconstrained functions; Schemata analysis.

1. Introduction.

Optimization is an all pervading problem in engineering and the sciences. It is, therefore, important to rely on an optimization tool of proven efficiency and reliability. In this paper we analyze a set of optimization algorithms which have not been analyzed exhaustively before and achieve interesting conclusions which allow us to recommend one such algorithm as applicable to a large number complex problems. When attempting to assess the relative efficiency of a set of optimization algorithms one may take one of two paths: a) Either one obtains closed models for the algorithms thus allowing their parametric characterization [1], [2], [3] or b) One selects a set of problems considered to be of interest and compares the performance of the algorithms when measured vs. such a set. Modeling an algorithm is frequently a complex task and, more importantly, even slight changes in the algorithm lead to basically different models [4], thus making the purported characterization impractical. The second option, therefore, seems better suited for practical purposes. However, although there are many examples of such an approach (for instance see [5], [6], [7]) it is always true that a) The nature of the algorithms under study and their number are necessarily limited and b) The selection of the benchmarking functions obeys to subjective criteria. In this paper we choose to establish the relative efficiency of a set of genetic algorithms (GAs) which are structurally different from one another as will be discussed in the sequel. We have selected a set of such GAs and, for completeness, we have also included a particular non-evolutionary algorithm (the Random Mutation Hill Climber or RMH) whose efficiency has been reported in previous works [8], [9]. Many GAs are variations (i.e. different selection criteria [10], crossover strategies [11], population size [12], 13] relationship between Pc and Pm, [14], [15], etc.) of the initial one proposed by Holland (the so-called “Simple” or “Canonical” Genetic Algorithm [CGA] [16]) which do not significantly improve on CGA’s overall performance. For benchmarking purposes the mentioned variations are not useful since they all share the same basic algorithmic structure. However there are GAs where the strategies to a) select, b) identify and c) recombine candidate solutions differ from the CGA’s substantially. The purported changes impose structural differences between these algorithms which have resulted in remarkable performance implications. We have selected four GAs with this kind of diverse characteristics. We begin, in Section 2, by introducing the necessary notation; then presenting some concepts and definitions. In Section 3 we describe the five algorithms in our work. In section 4 we present the functions and results for a suite of problems that traditionally have been used for benchmarking purposes of optimization algorithms [17] [18]. In Section 5 we present our general conclusions.

2. Preliminaries

Throughout we use the following notation and definitions: A: Set of selected optimization algorithms; A_i: The *i-th* optimization algorithm (i.e. $A_i \in A$); $\bar{x}$: Vector in $\Re^n$; Ω: Feasibility region of the space $\Re^n$; B: Set defined as $B = \{0,1\}$; t: Iteration number such that $1 \le t \le G; t \in N;$ G: Upper bound on the number of iterations of A_i. Without loss of generality our discussion will be focused on numerical optimization problems. One such problem f is defined as:

$$\begin{array}{lll} \textit{Minimize} & f(x) & \\ \textit{Subject to} & h_i(\bar{x}) = 0 & i = 1,...m \\ & g_i(\bar{x}) \le 0 & i = m+1,...p \end{array} \qquad (1)$$

where $f(\bar{x}) : \Re^n \to \Re$ is the objective function, $h_i(\bar{x}) = 0$ and $g_i(\bar{x}) \le 0$ are constraint functions defining Ω. This means that if a vector $\bar{x}$ complies with all constraints it belongs to Ω. In a problem without constraints, such as the ones discussed here, all vectors $\bar{x}$ lie within Ω.

We briefly pause to define what we understand as a genetic algorithm. Elsewhere [20], it has been argued that an algorithm is "genetic" when it exhibits implicit parallelism. Instead, we list the characteristics an iterative algorithm must have to be considered "genetic". Implicit parallelism is a consequence of these.

Definition 1:
A genetic algorithm is one which satisfies the following conditions:

1. It works on an *n*-dimensional discrete space D defined in $\mathbb{N}^n$ rather than in $\mathbb{R}^n$.
2. It traverses D searching an approximation of the optimum vector $\bar{x}$ of (1) by simultaneously analyzing a finite set $S(t) \in D$ of candidate solutions.
3. The elements of $S(t) = \{s_1(t), s_2(t),..., s_n(t)\}$ are explicitly encoded in some suitable way.
4. The information regarding the partial adequacy of the elements in $S(t)$ is extracted by solving the optimization problem for all $s_i(t)$.
5. The qualified elements of $S(t)$ are analyzed to select an appropriate subset in order to improve the search in the problem's space.
6. Selected sections of the codes of $s_i(t)$ are periodically combined.
7. Selected elements of the codes of the $s_i(t)$ are periodically and randomly altered.
8. A subset of the best solutions of $S(t)$ is preserved for all the future steps of the algorithms.
9. The algorithm cycles through steps 4-8 until a stopping condition is met.

The algorithms selected for this study satisfy all of the characteristics above and, therefore, may be aptly considered to be genetic in a broader sense then the one implied by the frequently cited "bio-inspired" analogy. In fact, this analogy, attractive as it may seem, frequently distracts the attention of the user from the basic efficiency elements which any optimization algorithm should incorporate. These issues must supersede other considerations when determining the desirability of one algorithm over others.

Consequently, set *A* includes the following GAs:

a) An elitist canonical GA (in what follows referred to as TGA [eliTist GA]) [21].
b) A Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation algorithm (CHC algorithm) [22].
c) An Eclectic Genetic Algorithm (EGA) [23].
d) A Statistical GA (SGA) [24] [25].

3. Selected Genetic Algorithms

It is frequent to cite the variations of the GAs by their "popular name". However, in so doing one incurs in the risk of not being precise on the details of the algorithm. One of the basic tenets of this paper is that even small variations lead to potentially important differences in their behaviors. For this reason, we now include the pseudo-codes of the algorithms in our study. Keep in mind that our results refer to their precise implementation and no others. As a case in point, when discussing SGA (the Statistical GA) it may be easy to confuse it with EDA (Estimation of Distribution Algorithm). However, in EDA no mutation is explicitly included, whereas in SGA it is (see the code below)

In the description of the algorithms which follows a) We denote the arguments $\bar{x} = (x_1,...,x_k)$ with x_i and the corresponding fitness function $f(\bar{x}) = f(x_1,...,x_k)$ with $f(x_i)$, b) The function $f(x_i)$ to be optimized is numerical, c) We aim to minimize $f(x_i)$, and d) The arguments x_i of the fitness function $f(x_i)$ are encoded in binary.

Let $G \equiv$ number of generations; $n \equiv$ number of individuals; $I(n) \equiv$ *the n-th individual;* $L \equiv$ length of the chromosome; $P_C \equiv$ probability of crossover; $P_M \equiv$ probability of mutation.

By *"Generation of a random population"* we mean that, for a population of n individuals each of whose chromosome's length is L we make

```
for i = 1 to n
    for j=1 to L
        Generate a uniform random number $0 \le r < 1$.
        If $r < 0.5$ make $bit_j \leftarrow 0$; else make $bit_j \leftarrow 1$.
    endfor
endfor
```

3.1. Elitist Canonical GA (TGA)

This is the classical CGA with two provisions: a) The best individual is kept along the whole process forming part of the evolving population and b) In step 3 of the algorithm

$$j\,(x_i) = f(x_i) + |\,min(f\,(x_i))\,| + avg(|\,f\,(x_i)\,|) \qquad \text{(A.1)}$$

is used. These two steps ensure that no fitness value is negative making the proportional selection scheme always feasible (see [28, 29, 30]).

```
0. Make $k \leftarrow 1$.
1. Generate a random population
2. Select randomly an individual from the population (call it best).
      Make $f(best) \leftarrow \infty$ .
3. Evaluate.
      for i=1 to n
            Evaluate $f(x_i)$ .
            Make $f(x_i) \leftarrow j\,(f(x_i))$.
            If $f(x_i) < f(best)$ make $best \leftarrow x_i$ and $f(best) \leftarrow f(x_i)$
      endfor
4. If $k = G$ return best and stop.
5. Selection
      Make $F = \sum_{i=1}^{n} f(x_i)$
      for i = 1 to n; $PS_i = \frac{f(x_i)}{F}$; Endfor
      for i =1 to n; Select I(i) with probability $PS_i$.; endfor
6. Crossover
for i = 1 to n step 2
      Randomly select two individuals (say I(X) and I(Y)) with probabilities $PS_X$
      and $PS_Y$, respectively.
      Generate a uniform random number $0 \le r < 1$.
      If $r \le P_C$ do
         • Randomly select a locus $\ell$ of the chromosome; Pick the leftmost L-$\ell$ bits of I(X)
           and the rightmost $\ell$ bits of I(Y) and concatenate them to form the new chromosome
           of I(X); Pick the leftmost L-$\ell$ bits of I(Y) and the rightmost $\ell$ bits of the
           previous I(X) and concatenate them to form the new chromosome of I(Y)
      Make $I(i) \leftarrow I(X)$; $I(i+1) \leftarrow I(Y)$.
endfor
7. Mutation
for i = 1 to n
      Select I(i)
```

```
        for j=1 to L
              Generate a uniform random number $0 \le r < 1$.
              If $r \le P_M$ make $bit_j \leftarrow \overline{bit_j}$.
        endfor
endfor
8. Make $k \leftarrow k+1$ and go to step 3.
```

3.2. Cross Generational elitist selection, Heterogeneous recombination and Cataclysmic mutation GA (CHC)

This algorithm focuses on maintaining diversity while retaining the characteristics of the best individuals. Inter-generational survival-of-the-fittest is attempted by unbiased parent selection. Furthermore it tries to maintain diversity implementing the so-called HUX crossover (Half, Uniform X-over) and introducing cataclysmic mutations when the population's diversity falls below a pre-defined threshold (see [20]).

```
0. Make
      $\ell \leftarrow 0$
      $threshold \leftarrow L/4$
1. Generate a random population.
2. Evaluate $f(x_i)$ $\forall i$
3. $f(best) \leftarrow min(f(x_i))\forall i$
   $best \leftarrow I(i \mid f(x_i)$ is best)
4. $\ell \leftarrow \ell + 1$
   If $\ell = G$ return best and stop.
5. Copy all individuals from population P into set C
6. [HUX Crossover]
Let $Bit_{xy}$ denote the y-th bit of individual x
for i=1 to n/2
      Randomly select individuals $I_X$ and $I_Y$ from C
      $hammingXY \leftarrow 0$
      for $j \leftarrow 1$ to L
            if bit j $(I_X) \neq$ bit j$(I_Y)$; DiffXY[j]=true;
              $hammingXY \leftarrow hammingXY+1$; elseDiffXY[j]=false; f
      endfor
      if ($hammingXY/2 \le threshold$)
            eliminate C(X) and C(Y) from C
      else
```

```
                mutated $\leftarrow 0$
                while (mutated<hammingXY/2)
                        j $\leftarrow$ random number between 1 and L
                        if DiffXY[j]
                                Interchange the j-th bit of $I_X$ and $I_Y$
                                mutated $\leftarrow$ mutated+1; DiffXY[j] $\leftarrow$ false
                endwhile
endfor
Evaluate $f(x_i)$ in C( i) $\forall i$
Make P'= $P \cup C$
Sort P' from best to worst
P' $\leftarrow$ Best n individuals from P' ; $f(best) \leftarrow f(x_1)$; best $\leftarrow P_1'$
if P = P'
        threshold $\leftarrow$ threshold-1
        if threshold=0
                for i=2 to n
                        Select the i-th individual
                        for j=1 to L
                                Generate a uniform random number $0 \le r < 1$
                                If $r \le 0.35$ make $bit_j \leftarrow \overline{bit_j}$
                threshold $\leftarrow$ L/4
P $\leftarrow$ P'; go to 4
```

3.3 Eclectic GA (EGA)

This algorithm uses deterministic selection, annular crossover, uniform mutation and full elitism (a strategy akin to $m+l$ selection of evolutionary strategies [31]. The probabilistic nature of EGA is restricted to parameters P_C and P_M. In EGA avoidance of premature convergence is achieved by a two-fold strategy. First, the 2*n* individuals from the last two generations are ordered from best to worst and only the best *n* are allowed to survive. Then the individuals are deterministically selected for crossover by mixing the best with the worst (*1* with *n*), the second with the second worst *(2* and *n-1*, . . .,), and so on. In this way *n* new individuals are generated. As the iterations proceed, the surviving individuals become the top elite of size *n* of the whole process. Annular crossover (equivalent to two-point crossover) is preferred because it makes the process less dependent on a particular encodings. This algorithm was first reported in [26] and included self-adaptation and periodic cataclysmic mutation. Later studies [27] showed that neither of the two mechanisms was necessary. EGA is relatively simple, fast and easy to program.

```
0. B2M $\leftarrow \lceil nL \times P_M \rceil$            ( Expected number of mutations per generation)
1. i $\leftarrow$ 1
2. Generate a random population
```

```
3. Evaluate the population.
4. [ Duplicate Population]
for j = 1 to n
        I(n+j) $\longleftarrow$ I(j)
        fitness(n+j) $\longleftarrow$ fitness(j)
endfor
5. [ Deterministic Selection Annular Crossover]
for j=1 to n/2
        Generate a uniform random number $0 \le r < 1$
        If $r \le P_c$
                Generate a random number $1 \le r < L/2$
                Interchange the semi-ring starting at locus $r$ between I(j)
                 and I(n-j-1)
        endif
endfor
5. [Mutation]
for j=1 to B2M
        Generate uniform random numbers $0 \le r_1, r_2 < 1$
        Mutate Bit $\lceil r_2 L \rceil$ of I($\lceil r_1 n \rceil$)
endFor
6. [Evaluate the New Individuals]
Calculate fitness($x_i$) for i=1,…,n
7. [ $m + l$ Selection]
Sort the 2n individuals by their fitness, ascending
8. i $\leftarrow$ i+1
  if i = G return I(1) and stop
 Go to 3
```

3.4. Statistical GA (SGA)

In this case the algorithm takes advantage of the fact that the average behavior of traditional TGA may be achieved without actually applying the genetic operators but, rather, statistically simulating their behavior [24]. SGA starts by generating the initial population's (P_0) individuals randomly. The fitness for each individual is calculated as per (A.1). It is then easy to determine its relative fitness $\Phi(x) \leftarrow f(x_j) / \sum_j f(x_j)$ which, immediately, induces a partial ordering in the population according to the value of $\Phi(x)$. Once this is done, the so-called *probabilistic genome* (PG) is calculated. In this genome, the probability that the *k-th* bit of a genome attains a value of 1 is derived from

$$P_k = \sum_j \Phi_j b_{jk} \quad k = 1, ..., L \tag{A.2}$$

where b_{jk} denotes the *k-th* bit of the *j-th* individual. Notice that P_k actually represents the weighted expected number of times that bit k will take the value *1* as a function of the fitness of the *i-th* population. This is equivalent to defining a set of probability distribution functions (*pdfs*); one for each of the L bits in the genome. These *pdfs* are Bernoulli distributed and, initially, may have rather large variances (s^2). Every new population is generated by sampling from the *j-th* distribution to compose its new N individuals. The *i-th* population consists of individuals that respond to the average behavior of the *(i-1)-th*. Every new population is also Bernoulli distributed but with an increasingly small s. Eventually the *pdfs* of the final population will have a Bernoulli distribution with $s \approx 0$, implying approximate convergence. In a strict sense, the SGA avoids the need to include explicit mutation provisions. Preliminary tests showed that premature convergence is avoided if such provisions are made. The whole process may be seen as a search for a crisp encoding of the solution with a set of fuzzy bits. Each bit is progressively de-fuzzyfied in consecutive generations.

```
0. Make $k \leftarrow 1$;
   $B2M \leftarrow \lceil nL \times P_M \rceil$          ( Expected number of mutations per generation)
1. Generate a random population
2. Select randomly an individual from the population (call it best). Make f(best) $\leftarrow \infty$.
3. [Get probabilistic genome]
      $PopFit \leftarrow 0$
      for i=1 to n
            Evaluate $f(x_i)$
            If $f(x_i)<f(best)$
                  $best \leftarrow I(i);\ f(best) \leftarrow f(x_i)$
            endif
            Make $f(x_i) \leftarrow j(f(x_i))$; $PopFit \leftarrow PopFit + f(x_i)$
      Endfor
      for i=1 to n
            $RelFit_i \leftarrow f(x_i)/PopFit$;
      endfor
      for i=1 to L
            $PG_i \leftarrow 0$;
            for j=1 to n
                  if $bit_{ji} = 1 \rightarrow PG_i \leftarrow PG_i + RelFit_j$
            endFor
      endFor
4. [Get new population]
      [Probabilistic Individuals]
      for i = 1 to n
            for j = 1 to L
                  Generate a uniform random number $0 \le r < 1$
                  if $r \le PG_j \rightarrow Bit_{ij} \leftarrow 1$; else $Bit_{ij} \leftarrow 0$
            endFor
      endFor
      [Mutate Individuals]
      for i=1 to B2M
            Generate uniform random numbers $0 \le r_1, r_2 < 1$
            Mutate Bit $\lceil r_2 L \rceil$ of I($\lceil r_1 n \rceil$)
      endFor
  5. $k \leftarrow k+1$
    If $k = G$ return best and stop; else Go to step 3
```

3.5. Random Mutation Hill-Climber (RMH)

This algorithm is the only non-evolutionary one considered in this study. In general, a "hill-climber" is an algorithm which attempts, iteratively, to improve on its best found value by refining the last attempt.

```
    1. [Generate the individual]
    for i=1 to L
          Generate a uniform random number $0 \le r < 1$.
        If $r < 0.5$ make $bit_i \leftarrow 0$; else make $bit_i \leftarrow 1$.
    endfor
    Make best $\leftarrow$ I(0)
        BestFit $\leftarrow \infty$
    2. [Iterate]
    for i = 1 to G
        [Evaluate the individual]
        f(i) $\leftarrow$ fitness ($x_i$)
        if fitness(i)<BestFit $\rightarrow$ best $\leftarrow$ I($x_i$); BestFit $\leftarrow$ fitness($x_i$)
        [Mutate]
         Generate a uniform random number $1 \le k \le L$; Make $bit_k \leftarrow \overline{bit_k}$
    endfor
```

For the case of RMH, $S(t)$ is a unitary set such that $S(t) = \{s(t)\}$ where $s(t)$ is also a binary encoded candidate solution which is chosen at random and whose fitness is evaluated. We explored the behavior of the *A_i*'s taking snapshots of their progress in steps of 50 generations up to 800. A GA works with several candidate solutions that allow it to explore different regions of Ω in parallel. On the other hand, the RMH works with a single candidate solution that allows it to explore a single region of Ω. The number of iterations of RMH needed for convergence, for this reason, differs significantly from that of a GA. For benchmarking purposes, therefore, we established the following standard.

1) Let M be the number of candidate solutions for a GA. Thus, for *A_i* it holds that $|S(t)| = M$

2) The upper bound on the number of iterations of a RMH is set to $M \times G$.
3) The upper bound on the number of iterations of any GA is set to G.

Any A_i will, therefore, approach the solution to a problem f in at most G iterations. For a detailed discussion of the algorithms see Appendix A.

4. Selected Functions

In this section we discuss the behavior of the algorithms for selected functions whose minima are known and, therefore, allow us to establish a measure of effectiveness relative to the best value found by the algorithm. The evaluation of all algorithms in A is based on a set of unconstrained functions (UF) which have some properties (multimodality, non-convexity, non-linearity, etc.) that make them good choices for benchmarking purposes,

For reasons of space we may only succinctly present 6 of the 23 functions considered in this study. 1) **Hansen Function**. Unimodal; it is defined in a n-dimensional space $\forall n \in \mathbb{N}$: $f(n,x) = \sum_{i=0}^{4}(i+1)\cos(ix_0 + i + 1)\sum_{j=0}^{4}(j+1)\cos((j+2)x_1 + j + 1)$. O is $|x_i| \leq 10$; the known minimum is -176.54. 2) **De Jong's Function**. Continuous, convex and unimodal: $f(x) = \sum_{i=1}^{n} x_i^2$. O is $-5.12 \leq x_i \leq 5.12,\quad i = 1,\ldots,n$; the known minimum is 0: 3) **Rotated hyper-ellipsoid function**. Continuous, convex and unimodal: $f(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$; O is -65536 <= x_i <= 65536; the known minimum is 0. 4) **Rosenbrock's valley function.** The global optimum lies inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. However convergence to the global optimum is difficult and hence this problem has been frequently used to test the performance of optimization algorithms: $f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$. O is $-2.048 \leq x_i \leq 2.048,\quad i = 1,\ldots,n$; the known minimum is 0. 5) **Rastrigin's function.** It is based on the function of De Jong with the addition of cosine modulation in order to produce frequent local minima. The function is highly multimodal: $f(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2px_i)$. O is $-5.12 \leq x_i \leq 5.12,\quad i = 1,\ldots,n$; the known minimum is 0. 6) **Schwefel's function.** It is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction: $f(x)=\sum_{i=1}^{n}[-x_i \sin(\sqrt{|x_i|}]$. O is $-500 \le x_i \le 500$, $i=1,\ldots,n$; the known minimum is -418.9828n.

All 23 functions have known optima. This allows us to define a relative measure of performance for A_i as follows:

$$Q(A_i, f_j) = \frac{y_j * - y_j}{y_j *} \qquad (2)$$

where $y_j *$ is the known optimum of f_j and y_j is the best value found by A_i. We ran every algorithm 100 times for every problem and obtained its average performance. We obtained this average performance for all A_i with G = 800. We got snapshots of the process every 50 generations. In Figure 1 we show the corresponding results.

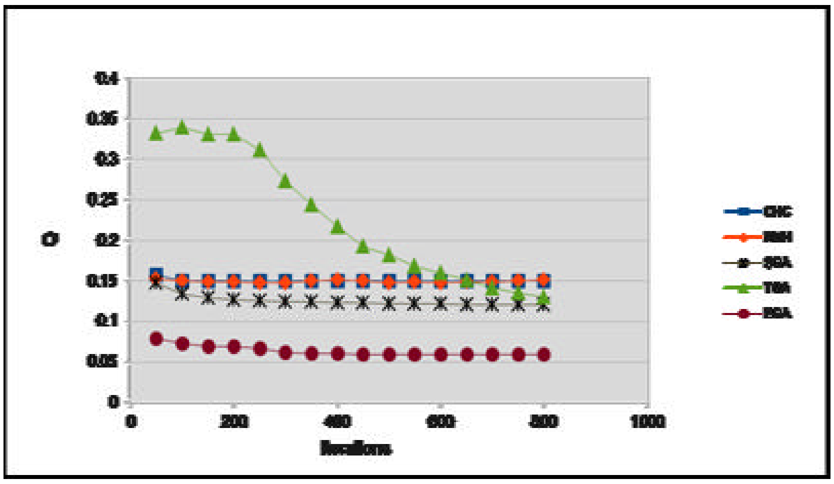


Fig. 1. Average Performance for UF

Notice that in these functions all of the GAs outperform the RMH only marginally, with the exception of EGA which is considerably better. Also notice that TGA (the canonical GA) is able to reach an acceptable value only after a large number of generations. From a further analysis we determined how the algorithms identify larger order schemata. We indicate that A_i is better than A_j when the order of the schemata of A_i is larger than the order of the schemata of A_j. (see Fig. 2). Consistent with the previous results, TGA is the slowest algorithm to identify schemata of higher order. That is, the sluggish nature of TGA is due to the fact that it spends many more generations in identifying the better individuals in terms of their schemata order.

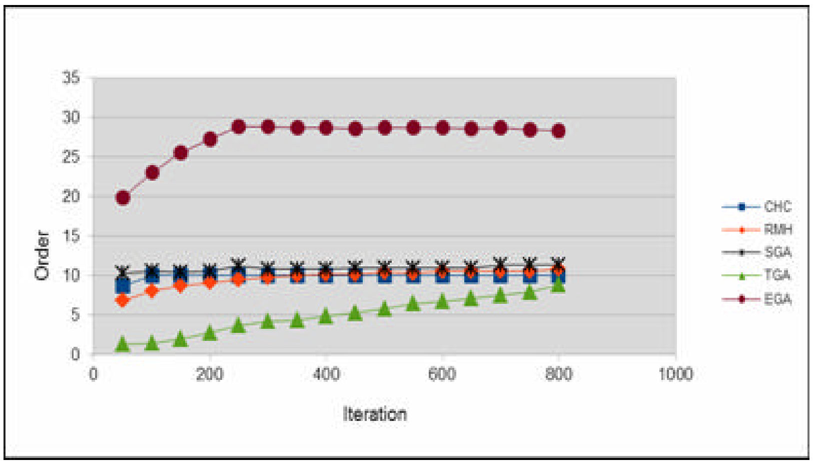


Fig. 2. Average order of the schemata for UF.

5. Conclusions

The table for the suite of unconstrained problems (see Table 1) show that EGA outperforms the rest of the algorithms; in this case, notably so. TGA is, by far, the worst of the algorithms. Again RMH's behavior is close to SGA's and CHC's.

Table 1. Average minimum for the suite of un constrained problems

Algorithm	Average Minimum	Relative Efficiency
EGA	0.0635	100.00%
SGA	0.1260	50.43%
RMH	0.1491	42.60%
CHC	0.1501	42.32%
TGA	0.2272	27.96%

The best algorithm is EGA. Considering the wide size and variety of the set of problems we can say that, with high probability, the EGA is the best global optimization algorithm in our study.

In concluding:

1. In all experiments, the EGA exhibited the best performance. We know that EGA is a good alternative in problems with hard search spaces (e.g. non-convex, constrained or highly dimensional spaces) .
2. From a large set of runs it is possible to obtain a practical estimate of the

schemata found by the algorithms.

3. The analysis of schemata order of the algorithms leads to results consistent with the previous one.
4. A similar analysis including other optimization techniques (e.g. Simulated annealing, Evolution strategies, Ant colony optimization, Particle swarm optimization, Differential evolution, etc.) may be easily implemented.

References

1. MacKay, B., College, C., "Mathematics and Statistics Models", http://serc.carleton.edu/introgeo/mathstatmodels/index.html.
2. Mooney D., Swift, R., Mooney, D., A Course in Mathematical Modeling, Cambridge University Press, 1999.
3. Anna Kolesárová; and Radko Mesiar, "Parametric characterization of aggregation functions", Fuzzy Sets Syst. 160, 6, 816-831, 2009.
4. Back, T., Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms, Ch.4, p. 149-159, 1996.
5. Coello, C., "A comprehensive survey of evolutionary-based multiobjective optimization techniques", Knowledge and Information Systems, 1, 269-308, 1998.
6. De Jong, K., "An analysis of the behavior of a class of genetic adaptive systems", Diss. PhD thesis, Dept. of Computer and Comm. Sciences, Univ. of Michigan, Ann Arbor, MI, 1975.
7. Endre, A., Hinterding, R., Michalewicz, Z., "Parameter control in evolutionary algorithms.", Evolutionary Computation, IEEE Transactions on 3.2: 124-141, 1999.
8. Mitchell, M., An Introduction to Genetic Algorithms, MIT Press, 1996.
9. Mitchell, M., J. Holland y S. Forrest, "When Will a Genetic Algorithm Outperform Hill Climbing?", Advances of Neural Information Processing Systems, No. 6, Morgan Kaufmann, pp. 51-58, 1994.
10. Baker, J., "Adaptive selection methods for genetic algorithms". In Grefenstette, J., editor. Proceedings of the 1st International Conference on Genetic Algorithms and their Applications. Lawrence Earlbaum Associates, N.J., pp. 101-111 1985.
11. Anand, V., and Spears, W., "A Study of Crossover Operators in Genetic Programming", Massachusetts Institute of Technology, Proceedings of the International Symposium on Methodologies for Intelligent Systems, eds. Z. W. Ras and M. Zemankova, Berlin: Springer-Verlag, 542: 409-418, 1991.
12. Bäck, T., "Self-Adaptation in Genetic Algorithms". In Francisco Varela and Paul Bourgine, eds. Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, pp. 263-271, MIT Press, 1991.
13. Bäck, T., "Evolutionary Algorithms in Theory and Practice", Oxford University Press, 1996.
14. De Jong, K., "An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Doctoral Dissertation, University of Michigan, 1975.
15. De Jong, K., and Spears, W., "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms". In Hans-Paul Shwefel and Reihnard Manner, eds. Parallel Problem Solving from Nature, 1st. Workshop, PPSN 1, volume 496, of Lecture Notes in Computer Science, pp. 38-47, Berlin, Germany, Springer-Verlag, 1991.
16. Holland, J. "Adaptation in Natural and Artificial Systems", Ann Arbor, MI; University of Michigan Press, 1975.

17. Pohlheim, Hartmut, “GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with MATLAB Documentation”, http://www.geatbx.com/docu/in dex.html.Version 3.80 (released December 2006).
18. Digalakis, J. and Margaritis, K., “An experimental study of Benchmarking functions for genetic algorithms”, Intern. J. Computer math., vol. 79(4), pp. 403–416, 2002.
19. Vose, D., (1991), "Generalizing the notion of schema in genetic algorithms". Artificial Intelligence, 50(3):385-396, 1991.
20. Eshelman, L., “The CHC Adaptive Search Algorithm. How to Have Safe Search When Engaging in Nontraditional Genetic Recombination”. In Rawlins, G., editor, FOGA -1, pages 265–283. Morgan Kaufmann., 1991.
21. Rudolph, G., "Convergence Analysis of Canonical Genetic Algorithms", IEEE Transactions on Neural Networks, 5(1):96-101, January, 1994.
22. Eshelman, op. cit., 1991.
23. Rezaee Jordehi, A., Hashemi, N, Nilsaz Dezfouli, H., “Analysis of the Strategies in Heuristic Techniques for Solving Constrained Optimisation Problems”, Journal of American Science, 8(10), 2012.
24. Sánchez-Ferrero, G., Arribas, JI, “A Statistical-Genetic Algorithm to Select the Most Significant Features in Mammograms”, Lecture Notes in Computer Science, Volume 4673/2007, 189-196, DOI: 10.1007/978-3-540-74272-2_24, 2007.
25. Kuri-Morales, A., “A statistical genetic algorithm”, Proc. Of the 3d. National Co mputing Meeting, ENC '99, pp. 215-228, Hgo., México, 1999.
26. Kuri-Morales, A., Villegas-Quezada, C. (1998). *A universal eclectic genetic algorithm for constrained optimization*, Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing. Vol. 1
27. Kuri-Morales, A. (2002). *A methodology for the statistical characterization of genetic algorithm s*, MICAI 2002: Advances in Artificial Intelligence : 77-96
28. Back, T. (1996). *Evolutionary algorithms in theory and pactice*: evolution strategies, evolutionary programming, genetic algorithms Ch.4, p. 149-159
29. Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, MIT Press
30. Vose, D. (1996). *The Walsh Transform and the Theory of the Simple Genetic Algorithm*, Pal, S. and Wang, P., editors. Genetic Algorithms for Pattern Recognition, CRC Pres
31. Rowhanimanesh, A., Sohrab, E. (2012). *A Novel Approach to Improve the Performance of Evolutionary Methods for Nonlinear Constrained Optimization*, Advances in Artificial Intelligence

The Best Genetic Algorithm II

A Comparative Study of Structurally Different Genetic Algorithms

Angel Kuri-Morales

Instituto Tecnológico Autónomo de México
Río Hondo No. 1
México 01000, D.F.
México
akuri@itam.mx

Edwin Aldana-Bobadilla

Universidad Nacional Autónoma de México
IIMAS
México 04510, D.F.
México
edwynjavier@yahoo.es

Ignacio López-Peña

Universidad Nacional Autónoma de México
IIMAS
México 04510, D.F.
México
ignalp@gmail.com

Abstract. Genetic Algorithms (GAs) have long been recognized as powerful tools for optimization of complex problems where traditional techniques do not apply. In [1] we reported the superior behavior, out of 4 evolutionary algorithms and a hill climber, of a particular breed: the so-called Eclectic Genetic Algorithm (EGA). EGA was tested vs. a set (TS) consisting of large number of selected problems most of which have been used in previous works as an experimental testbed. However, the conclusions of the said benchmark are restricted to the functions in TS. In this work we extend the previous results to a much larger set (U) consisting of $? \approx \sum_{i=1}^{31} (2^{64})^i \approx 11 \times 10^{50}$ unconstrained functions. Randomly selected functions in U were minimized for 800 generations each; the minima were averaged in batches of 36 each yielding $\overline{X}_i$ for the i-th batch. This process was repeated until the $\overline{X}_i$'s displayed a Gaussian distribution with parameters $m_{\overline{x}}$ and $s_{\overline{x}}$. From these, the parameters m and s describing the probabilistic behavior of each of the algorithms for U were calculated with 95% reliability. We give a sketch of the proof of the convergence of an elitist GA to the global optimum of any given function.

We describe the methods to: a) Generate the functions; b) Calculate m and s for U and c) Evaluate the relative efficiency of all algorithms in our study. EGA's behavior was the best of all algorithms.

Keywords. Genetic algorithms, Unbiased functions, Statistical validation.

1. Introduction.

Optimization is an all pervading problem in engineering and the sciences. It is, therefore, important to rely on an optimization tool of proven efficiency and reliability. In this paper we compare a set of optimization algorithms which were analyzed in [1] over a set of unconstrained selected functions in $\Re \times \Re$. Here we extend our study in two ways: First, we consider a, for all practical purposes, unlimited reservoir of unconstrained functions. Second, we use the same basic reservoir so that the functions correspond to $\Re \times \Re$, $\Re \times \Re^2$ and $\Re \times \Re^3$. The results may be generalized for $\Re \times \Re^n$. In analogous comparative studies in the past (for instance see [2], [3], [4]) it is always true that a) The nature of the algorithms under study and their number are necessarily limited and b) The selection of the benchmarking functions obeys to subjective criteria. We know that any elitist GA will find the global optimum. The time (iterations) the GA has to spend is, however, not bounded. Therefore it seems appropriate to seek the fastest GA, in general. We analyze a set whose functions are a) Representative, b) Large enough, c) Automatically generated and d) Randomly selected. We may apply statistical methodologies to extract the probabilistic behavior of the algorithms under study with arbitrary reliability. The results from an analysis following the previous guidelines will enable us to ascertain which of the GAs is fastest, i.e. the best, for most functions likely to be found in practice. In [5] it is shown that elitist GAs always converge to a global optimum. The basic idea hinges on the following: a) Because GAs perform the search in a discrete space, the number of possible points to examine is finite; b) Any combination of individuals in the population may be thought of as a state in a Markov chain (MC), c) Via mutation, there is a non-zero probability that the GA will reach all possible states in the MC and d) If the best individual is retained throughout the process, when the GA is stopped, the best individual will correspond to the best possible solution to the problem. This is true iff there is no sink state (i.e. if there is always a non-zero probability of exiting a given state of the MC). Holland's original GA [6] did not include elitism. But most practical implementations do. In fact, any elitist algorithm, even if it is not evolutionary, satisfying the condition of exhaustive visits to all possible states, will, by the same token, reach a global optimum. A case in point is the so-called Random Mutation Hill Climber or RMH (for which see [1, 7, 8]). If we keep track of the best value, however, the behavior of the algorithm may be illustrated as in Figure 1. In this case, even if the process looses its aim in the final stages of the evolutionary search, the best value is retained and, eventually, the best overall value will be reached.

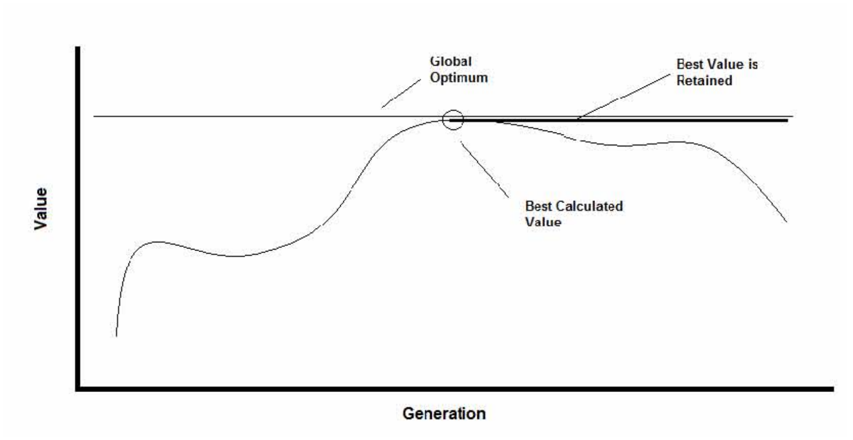


Fig. 1. Convergence with Elitism

It is easy to see that, given the above, the only basic difference in speed between a RMH (for example) and a GA has to reside in the crossover operator. The crossover component of a GA is actually responsible for the convergence speed of the process. Because of this we want to analyze several possible alternative GAs in trying to determine which is most effective. The GAs considered were the following:

a) An elitist canonical GA (in what follows referred to as TGA [eliTist GA]) [7].
b) A Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation algorithm (CHC algorithm) [9].
c) An Eclectic Genetic Algorithm (EGA) [12].
d) A Statistical GA (SGA) [10] [11].

For a detailed description and the pseudo-code of all the algorithms see [1]. Because we found that EGA was the best, we consider of interest to reproduce it here. When this algorithm was first reported it included self-adaptation and periodic cataclysmic mutation. Later studies [13] showed that neither of the two mechanisms was compulsory. EGA is relatively simple, fast and easy to program. In what follows, the next variables are used: $n \rightarrow$ number of individuals; $L \rightarrow$ length of the chromosome in bits; $P_M \rightarrow$ Probability of mutation; $Pc \rightarrow$ probability of crossover; $I(i) \rightarrow$ the i-th individual; $G \rightarrow$ number of generations.

Pseudo-Code of the Eclectic GA (EGA)

```
0. $B2M \leftarrow \lceil nL \times P_M \rceil$            ( Expected number of mutations per generation)
1. $i \leftarrow 1$
2. Generate a random population
3. Evaluate the population.
4. [ Duplicate Population]
for $j = 1$ to $n$
```

```
        I(n+j) $\longleftarrow$ I(j)
        fitness(n+j) $\longleftarrow$ fitness(j)
endfor
5. [ Deterministic Selection Annular Crossover]
for j=1 to n/2
        Generate a uniform random number $0 \le r < 1$
        If $r \le P_c$
                Generate a random number $1 \le r < L/2$
                Interchange the semi-ring starting at locus $r$ between I(j)
                 and I(n-j-1)
        endif
endfor
5. [Mutation]
for j=1 to B2M
        Generate uniform random numbers $0 \le r_1, r_2 < 1$
        Mutate Bit $\lceil r_2 L \rceil$ of I($\lceil r_1 n \rceil$)
endFor
6. [Evaluate the New Individuals]
Calculate fitness($x_i$) for i=1,…,n
7. [ $m + l$ Selection]
Sort the 2n individuals by their fitness, ascending
8. i $\leftarrow$ i+1
   if i = G return I(1) and stop
  Go to 3
```

The rest of the paper is organized as follows: in Section 2 we show how to extract the mean value μ and the standard deviation s from the minima of the functions in U. In Section 3 we describe how the functions in U may be generated and evaluated in $\Re \times \Re$, $\Re \times \Re^2$ and $\Re \times \Re^3$. In Section 4 we present our conclusions.

2. Statistical Determination of the Best Algorithm in U

A thorough experimental test of a given set of algorithms (*A*) implies running a large series of minimization trials. The probability that A_i reaches some minimum value (which we denote by k) is unknown. These k will vary for every problem and will distribute with mean µ and standard deviation s which are also unknown. We shall approximate these values by sampling *U*. It is, therefore, of utmost importance that we select sample *S* adequately; both in its nature and its size. Typically, the size of *S* is determined from assumptions (directly or indirectly) depending on the form of the population's (k 's distribution). We followed a method which does not necessitate from such assumptions. It relies on the knowledge that any sampling distribution of means (*sdom*) will eventually become Gaussian. Therefore, we generate a succession of problems to optimize (i.e. minimizing, every time, 36 problems of *U*) and calculate the corresponding mean $\overline{X}$. The iterations will be stopped only after the k's are distributed normally. Normality was considered to have been reached after dividing the results in deciles when a) $c^2 \le 3.28$ and b) O_i, the number of observations in the *i-th* decil, is 5 or more. We rely on the following theorems.

Theorem 1:

Any sampling distribution of means (*sdom*) is distributed normally for a large enough sample size *n*. [The Central Limit Theorem].

Remark: It is considered that any *n*>20 is satisfactory. We have chosen *n*=36.

Theorem 2:

In a normal distribution (with mean $m_{\overline{X}}$ and standard deviation $s_{\overline{X}}$) approximately one tenth of the observations lie in the intervals: $m_{\overline{X}}$–5 $s_{\overline{X}}$ to $m_{\overline{X}}$-1.29 $s_{\overline{X}}$; $m_{\overline{X}}$-1.29 $s_{\overline{X}}$ to $m_{\overline{X}} - 0.85 s_{\overline{X}}$; $m_{\overline{X}}$-0.85 $s_{\overline{X}}$ to $m_{\overline{X}}$–0.53 $s_{\overline{X}}$; $m_{\overline{X}}$-0.53 $s_{\overline{X}}$ to $m_{\overline{X}}$–0.26 $s_{\overline{X}}$; $m_{\overline{X}}$-0.26 $s_{\overline{X}}$ to $m_{\overline{X}}$ and the positive symmetrical.

Remark: These deciles divide the area under the normal curve in 10 unequally spaced intervals where the expected number of observed events will be one tenth.

Theorem 3:

The relation between the population distribution's parameters m and s and the *sdom*'s parameters $m_{\overline{X}}$ and $s_{\overline{X}}$ is given by $m = m_{\overline{X}}$ and $s = \sqrt{n} \cdot s_{\overline{X}}$.

Theorem 4:

The proportion of any distribution found within *k* standard deviations of the mean is, at least, *1-1/k²*. That is, $p(m - ks \le y_i \le m + ks) \ge 1 - 1/k^2$.

We selected k = *3.1623*, which guarantees that our observations will lie within the selected interval with $p \ge 0.90$.

The question we want to answer is: How small should c^2 be in order for us to ascertain normality? Remember the c^2 test is designed to verify whether two distributions *differ* significantly so that one may reject the null hypothesis, i.e. the two populations are statistically NOT equivalent. This corresponds to large values of c^2 and is a function of the degrees of freedom. In this case, if we wanted to be 95% certain that the observed $\overline{x_i}$'s were NOT normally distributed, we would demand that $c^2 \ge 14.0671$. However, this case is different. We want to ensure the likelihood that the observed behavior of the $\overline{x_i}$'s IS normal. In order to do this we performed the following Monte Carlo experiment. We set a desired probability *P* that the $\overline{x_i}$'s are normal. We establish a best desired value of c^2 which we will call c_{best}. We make *NS* ? 50. We then generate *NS* instances of *N(0,1)* and count the number of times the value of the instance is in every decile. We calculate the value of the corresponding c^2 and store it. We thusly calculate 100,000 combinations of size *NS*. Out of these combinations we count those for which $c^2 \le c_{\text{best}}$ AND there are at least o_{min}=5 observations per decile. This number divided by 100,000 (which we shall call *p*) is the experimental probability that, for NS observations, c^2 "performs" as *p*) is the experimental probability that, for NS observations, c^2 "performs" as required. We repeat this process increasing NS up to 100. In every instance we test whether $p > P$. If such is the case we decrement the value of c_{best} and re-start the process. Only when $p \leq P$ does the process end. The probability that c^2 exceeds c_{best} as a function of the number of problems being minimized (M) is shown in Figure 2. Every point represents the proportion of combinations satisfying the required conditions per 100,000 trials. For this experiment, c_{best} = 3.28. We obtained an approximation to a Gompertz model with $S=0.0023$ and $r=0.9926$. It has the form $p = ae^{-e^{b-cM}}$; where $a = 0.046213555$, $b = 12.40231200$, $c = 0.195221110$. From this expression we solve for M, to get $M = \{b - \ln[\ln(a/p)]\}/c$. As may be observed, $p<0.05$ for $M \geq 85$, which says that the probability of obtaining $c^2 \leq 3.28$ by chance alone is less than five in one hundred. Therefore, it is enough to obtain 85 or more $\overline{x_i}$'s to calculate m and s with 95% reliability.

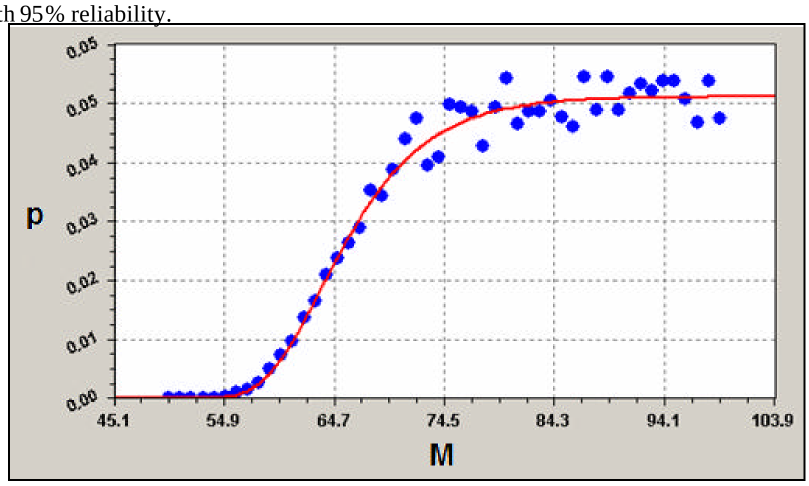


Fig. 2. Probability of c^2 and O_i>5 as a function of the number of problems solved.

In what follows we describe the algorithm which results from all the foregoing considerations.

Algorithm for the Determination of the Distribution's Parameters

Select an optimization algorithm *A*.

1. Make $G \leftarrow$ number of generations.
2. Generate a random binary string. This is one possible $f_i(x)$.
3. Minimize $f_i(x)$ iterating A, G times.
4. Store the best minimum value k_i.

5. Repeat steps (2-4) 36 times.
6. Calculate the average best value $? _j = (1/36)\sum_{i=1}^{36} k_i$.
7. Repeat steps (5-6) 50 times.
8. Calculate $m_{\bar{K}}$ and $s_{\bar{K}}$.
9. Repeat step (7) 85 times. The *sdom*'s distribution is normal with *p=0.95*.
10. Calculate $m = m_{\bar{K}}$ and $s = 6 s_{\bar{K}}$. We have inferred the expected best value k and the standard deviation for this algorithm.

From T4:

$$P(m - 3.1623\,s \le k \le m + 3.1623\,s) \ge 0.90 \qquad (1)$$

We have found a quantitative, unbiased measure of A_i's performance in $\Re \times \Re$. These values for the different A_i's allow us to make a fair unbiased assessment of their behavior.

3. Generation of U for $\Re \times \Re^n$

Once we have determined how to extract the parameters of the *pdf* for the algorithms we need an unbiased and automated way to obtain the problems to solve. We started by using Walsh's polynomials for $\Re \times \Re$. Next we used a monomial basis to, likewise, do so for $\Re \times \Re$, $\Re \times \Re^2$, $\Re \times \Re^3$. The behavior of the algorithms for all three cases was analyzed. The distributions were statistically equivalent. An induction principle leads to the conclusion that the observed behavior will be similar for $\Re \times \Re^n$.

3.1 Generation of Unbiased Functions using Walsh Polynomials

A reservoir of 250,000 randomly generated binary strings of length L ($32 \le |L| \le 1024$) may be interpreted as a set of 250,000 functions in $\Re \times \Re$. Call this set "*U*". By "unbiased" we mean that, because the functions in U are randomly generated, there is no bias in their selection. To generate functions automatically we resort to Walsh functions $y_j(x)$ which form an orthogonal basis for real-valued functions defined on $(0,1)^\ell$, where x is a bit string and ℓ is its length. Henceforth, any function *f(x)* thusly defined can be written as a linear combination of the y_j's yielding a Walsh polynomial.

$$f(x) = \sum_{j=0}^{31} w_j y_j(x) \qquad (2)$$

where $$y_j(x) = \begin{cases} +1 & if\ p(x \wedge j) = 0 \\ -1 & if\ p(x \wedge j) = 1 \end{cases} \tag{3}$$

$x \wedge j$ is the bitwise AND of x and j; $p(x)$ denotes the parity of x; and $w_j \in \Re$. Therefore, the index j and argument x of $y_j(x)$ must be expressed in comparable binary. We, therefore, used 16 bits to represent x in a $P_{0,15}(x)$ format. This means that we used one sign bit, 0 integers and 15 bits in a fixed point format for every term in the x's of (2). Consequently, we also used 16 bits for the indices j of (2). That implies that $-0.99996948\,2421875 \le x \le +0.99996948\,2421875$ and $0 \le j \le 65{,}535$. For example, consider $y_{61,680}(0.00048828\,125) = -1$. To see why, notice that $j=61{,}680$, in binary, is 1111000011110000. Also, x= 0.00048828 125 (with $P_{0,15}(x)$ format), corresponds to 0000000000010000. And $x \wedge j$ = *0000000000010000* for which $p(0000000000\,010000) = 1$. The length of the binary strings for the coefficients was also made 16 and, hence, $-0.99996948\ 2421875\ \le w_j \le +0.99996948\ 2421875$. Therefore, any Walsh monomial $w_j y_j$ is uniquely represented by a binary string of length 32. Finally, we allow at least one but no more than 32 non-zero terms in (2). This last conditions is mathematically expressed by including an a_j term which may only take two values (1 or 0) depending on whether the term appears. Given this, we have

$$g(x) = \sum_{j=1}^{32} a_j w_j y_j(x) \tag{3}$$

where $$a_j = \begin{cases} 1 & if\ the\ j-th\ term\ is\ present \\ 0 & if\ the\ j-th\ term\ is\ not\ present \end{cases}$$

Denoting with t the number of non-zero terms in 3 we see that a full (t =32) function's binary representation is 1,024 bits long. We denote the space of all possible functions defined by (3) with U and its cardinality with χ. It is easy to see that $? \approx \sum_{i=1}^{31} (2^{64})^i \approx 11 \times 10^{50}$. The method outlined provides us with an unlimited reservoir of functions in $\Re \times \Re$. Equally importantly, the random selection of a number t and the further random selection of t different indices and t different w_j's yields a uniquely identifiable function from such reservoir. The pool of Walsh functions was randomly generated at the beginning; the $f(x)$'s which the algorithms were required to minimize were all gotten from the same pool, thus allowing us to test the algorithms in a homogeneous functional environment.

3.2. Generation of Unbiased Functions from a Monomial Basis

Although it is possible to extend Walsh polynomials to higher dimensions, we found it more convenient to appeal to a monomial basis for the remaining cases, as follows.

3.2.1 The case y=f(x)

For the same functions in U we generated 150 random values $0 \le x_i < 1$ and calculated $y_i = f(x_i)$ for *i=1,…,150.* The sample consists of 150 binary strings of length 16. We stored these binary strings in a set we shall call **B**. Likewise, we stored the resulting y_i's in a set we shall call **F**. Notice that $x_i, y_i \in \Re$. Then we obtained the least squares approximating polynomial of degree 7. We will denote this set of approximated polynomial functions as U_2.

We minimized enough polynomials in U_2 for the distribution of the means to be normal. We did this for each of the algorithms in our study. The results of the minimization process are shown in Figure 3.

A c^2 goodness-of-fit test did not justify us to reject the null hypothesis H0: "The distributions of the Walsh basis functions (WBF) and the monomial basis functions (MBF) are similar". Hence, we conclude that the statistical behaviors of the algorithms when faced with problems defined with WBF and MBF are analogous. A quality index Q = *mean value of the minima* with $p = 0.95$ was defined for all the algorithms in our study. To visually enhance the difference between algorithms, we represent the values of Q in a logarithmic scale. Since some of the Q's are negative, we first scaled them into the interval $[d,1)$, where $d << 1$. Q^* is defined as follows:

$$Q^* = \log_{10}\{[Q_i - \min(Q)]/[\max(Q) - \min(Q)](1-d) + d\}$$

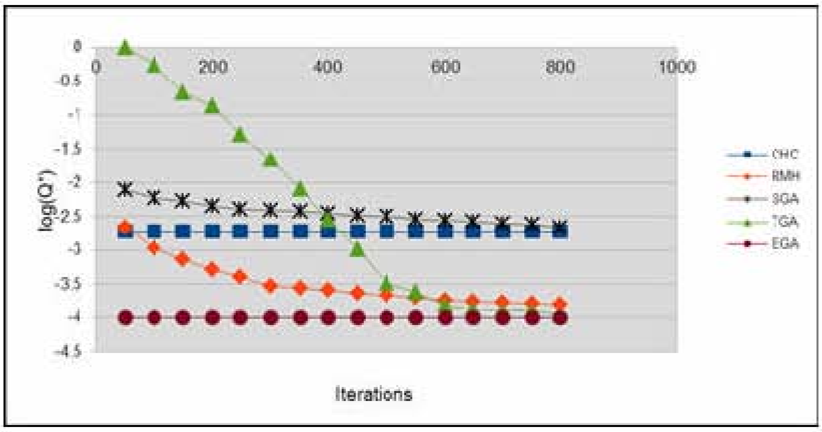


Fig. 3. Behavior of the algorithms minimizing unbiased polynomial *y=f(x)* functions.

3.2.2 The case z=f(x,y)

In this case we considered the binary strings of set **B.** They were split into two binary string sets of length 8 each, with $P_{0,7}$. Then the leftmost 8 bits were mapped into $\Re$ (which we now call x) and the rightmost 8 bits were also mapped into $\Re$ (which we shall call y). These (x_i,y_i) pairs were stored in matrix *XY*. The values of the independent variable z were those of set **F**. Our aim is to find polynomials of the form *z=f(x,y)*. In general, the problem is to find a set of (m) coefficients on a set of (n) independent variables expressed as a linear combination of monomials of the n variables of degree up to $d_1,...,d_n$ such that the absolute difference between an approximating function and the observed data is minimized. This problem is considerably more complex than the case $y=f(x)$. Furthermore, m grows exponentially as n and d_i do. For instance, if $n=2$ and $d_1=d_2=7$, $m=64$; likewise, for $n=3$, $d_1 = d_2 = d_3 = 7$ we have that $m=512$. This is the so-called called *curse of dimensionality*. Both problems were circumvented by applying the Ascent Algorithm (AA) [14]. The purpose of this algorithm is to express the behavior of a dependent variable (y) as a function of a set of n independent variables (v).

$$\begin{aligned} y &= f(v_1,v_2,...,v_n) \\ y &= f(\mathbf{v}) \end{aligned} \tag{4}$$

The approximant is <u>defined</u> to have the following form:

$$y = c_1 X_1 + c_2 X_2 + ... + c_m X_m \tag{5}$$

X_i denotes a combination of the independent variables. That is, $X_i = f_i(\mathbf{v})$. According to the way these combinations are defined one may obtain different approximants.
Now, from the universal approximation theorem [15], any function of n variables may be approximated with at most

$$T = \sum_{i=1}^{k} \left[\frac{\left(2i - 1 + \left(\sum_{j=1}^{k} \frac{(2j-1+n)!}{(2j-1)!n!} \right) \right)!}{(2i-1)! \left(\left[\sum_{j=1}^{k} \frac{(2j-1+n)!}{(2j-1)!n!} \right] \right)!} \right] \tag{6}$$

terms of degree k. The expression of T yields numbers of the order of 10^{12} even for small n. Obviously it makes no sense to try to approximate any function with a polynomial of these many terms. Therefore, we use a GA to select the best subset of the terms we decide to consider to make the problem reasonably expressible.

Genetic Polynomials

The basic reason to choose AA is that it is not dependent on the origin of the X_i in (5). We decided them to be the monomials of a full polynomial $y = \sum_{i_1=0}^{d_1} ... \sum_{i_n}^{d_n} c_{i1}...c_{in} v_1^{i1}...v_n^{in}$. But it makes no difference to the AA whether the X_i are gotten from a set of monomials or they are elements of arbitrary data vectors. To avoid the problem of the coefficient's explosion we define the number (say ß) of desired monomials of the approximant and then focus on slecting which of the p

possible ones these will be. There are *C(p, ß)* possible combinations of monomials and even for modest values of *p* and ß an exhaustive search is out of the question. This optimization problem may be tackled using a genetic algorithm (GA), as follows.

The genome is a binary string of size *p*. Every bit in it represents a monomial. These monomials are ordered as per the sequence of the consecutive powers of the variables. If the bit is '1' it means that the corresponding monomial remains while if it is a '0' it means that such monomial is not to be considered. All one has to ensure is that the number of 1's is equal to ß. Assume, for example, that $y = f(v_1, v_2, v_3)$ and that $d_1=1$, $d_2=d_3=2$. In such case the powers assigned to the $2 \times 3 \times 3 = 18$ positions of the genome are

000,001,002,010,011,012,020,021,022,100,101,102,110,111,112,120,121,122.

For the case where ß=*6* the genome 110000101010000001 corresponds to the polynomial in (7).

$$P(v_1, v_2, v_3) = c_{000} + c_{001}v_3 + c_{020}v_2^{\ 2} + c_{022}v_2^{\ 2}v_3^{\ 2} + c_{101}v_1v_3 + c_{122}v_1v_2^{\ 2}v_3^{\ 2} \tag{7}$$

The initial population of the GA consists of a set of binary strings of length *p* in which there are only ß 1's. The RMS error

$$e_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2} \tag{8}$$

is calculated for each tested polynomial and, at the end of the process, the one exhibiting the smallest such error is selected as the best approximant for the original data set. That is, for every genome the terms corresponding to the 1's are calculated. These take the place of the *X* in (5). Then the AA is applied to get the corresponding coefficients. To each combination of ß 1's there corresponds a set of ß coefficients minimizing $e_{MAX} = max(|f_i - y_i|)\ \forall i$. For this set of coefficients e_{RMS} is calculated. This is the fitness function for the GA. In the end, we retain the coefficients which best minimize e_{RMS} (from the GA) out of those which best minimize e_{MAX} (from the AA).

In our experiments, we set $d_1=d_2=4$ and ß =*6*. We obtained an expression of the form :

$$z = c_1X_1 + c_2X_2 + c_3X_3 + c_4X_4 + c_5X_5 + c_6X_6 \tag{9}$$

where V_{ij}'s value is either 0 or 1 as determined by the GA and

$$X_i = \sum_{i=0}^{4}\sum_{j=0}^{4} V_{ij}c_{ij}x^i y^j \tag{10}$$

Following the above procedure we found a polynomial for each of the functions in *xyz*. We denote this new set of approximated polynomial functions as U_3. Once the distribution of the means is normal, as before, we inferred the mean *m* and the standard deviation s of the *pdf* of the minimum values reached by each of the algorithms in our study. The results of the minimization process are shown in Figure 4.

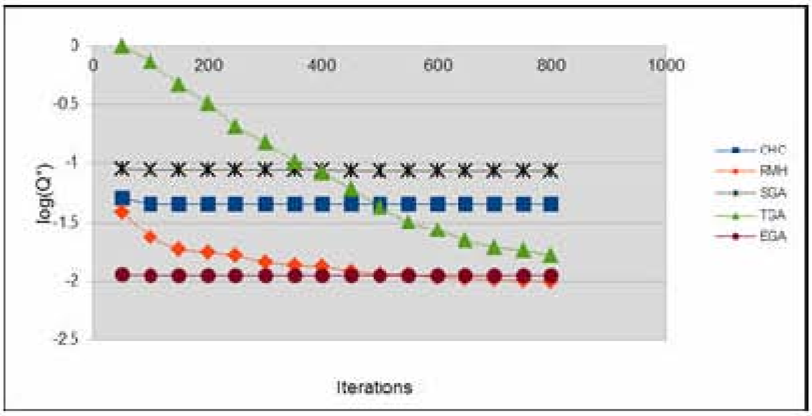


Fig. 4. Behavior of algorithms minimizing unbiased polynomial *z=f(x,y)* functions.

3.2.3. The case w=f(x,y,z)

In this case we considered the binary strings of set **B** but they were now split into three binary sets of lengths 5-5-6 with $P_{0,4}$, $P_{0,4}$, $P_{0,5}$, respectively. Then the leftmost 5 bits, the middle 5 bits and the rightmost 6 bits were, likewise, mapped into $\Re$. We call the corresponding variables *x*, *y* and *z*. These (x_i, y_i, z_i) triples were stored in matrix *XYZ*. The values of the independent variable *w* were those of set **F**. Our aim is to find a polynomial of the form *w=f(x,y,z)*. Following a process entirely similar to the one described above, we now defined *d1=d2=d3=4* with ß=*6* and obtained

$$w = c_1 X_1 + c_2 X_2 + c_3 X_3 + c_4 X_4 + c_5 X_5 + c_6 X_6 \tag{11}$$

but now

$$X_i = \sum_{i=0}^{4} \sum_{j=0}^{4} \sum_{k=0}^{4} V_{ij} c_{ij} x^i y^j z^k \tag{12}$$

Again we found a polynomial for each of the functions now in *wxyz*. We denote this new set of approximated polynomial functions with U_4. We minimized enough polynomials in U_4 for the distribution of the means to be normal. Again we inferred the mean *m* and the standard deviation *s* of the *pdf* of the minimum values reached by each of the algorithms. The results of the minimization process are shown in Figure 5.

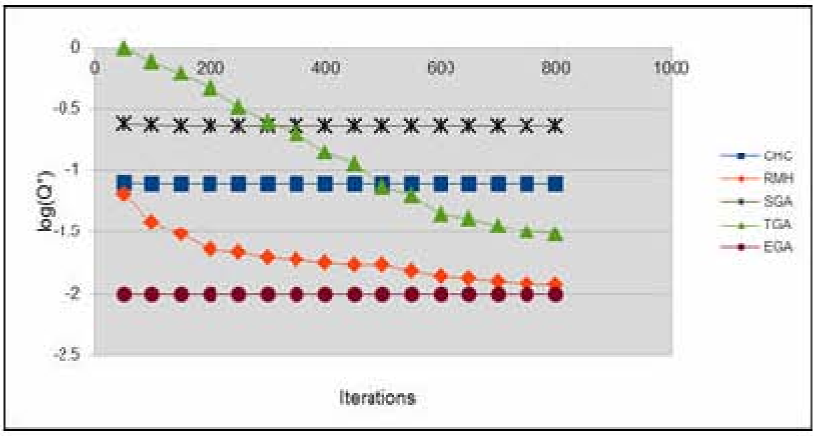


Fig. 5. Behavior of algorithms minimizing unbiased polynomial *w=f(x,y,z)* functions.

4. Conclusions and Future Work

The results of our study show:

1) All the algorithms have a very similar behavior. We had to use a logarithmic scale on the quality (Q*) of the results to make them apparent.

2) Remarkably, the RMH turns out to be as efficient as any of the GAs except for EGA.

3) Their behavior as the search space grows from $\Re\times\Re$, to $\Re\times\Re^2$ to $\Re\times\Re^3$ is statistically indistinguishable.

4) We may expect, from an induction principle, that the algorithms behave similarly in $\Re\times\Re^n$.

5) Even though all algorithms eventually approach similar minima, they do so with evidently different rates. For example, TGA does not reach adequate values until the very last generations.

6) SGA turns out to be the worst algorithm albeit it is the fastest (in CPU time) of all the A_i.

7) As in [1], where the minimized functions were hand-picked, EGA is the best algorithm of all.

8) EGA reaches its best minima in a relatively short number of generations. Therefore, it is guaranteed to reach the best solution without having to specify a large G.

In a paper to appear soon, we show that EGA works above par even when faced with constrained problems. Intuitively this should be the case since even constrained problems have to be, somehow, transformed into unconstrained ones. In the end, it appears that, for very simple problems, RMH is enough to reach acceptable solutions given enough time. However, when faced with more demanding ones, EGA seems to be the best alternative: it is better and faster.

References

1. Kuri-Morales, A., Aldana-Bobadilla, E., “The Best Genetic Algorithm I: A Comparative Study of Structurally Different Genetic Algorithms”, *sent for publication*.
2. De Jong K., “An analysis of the behavior of a class of genetic adaptive systems”, Diss. PhD thesis, Dept. of Computer and Comm. Sciences, Univ. of Michigan, Ann Arbor, MI, 1975.
3. Endre, A., Hinterding, R., Michalewicz, Z., "Parameter control in evolutionary algorithms.", Evolutionary Computation, IEEE Transactions on 3.2: 124-141, 1999.
4. Molga, M., Smutnicki, C. , “Test functions for optimization needs”, Retrieved March 11, 2012from http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf
5. Rudolph, G., "Convergence Analysis of Canonical Genetic Algorithms", IEEE Transactions on Neural Networks, 5(1):96-101, January, 1994.
6. Holland, J. "Adaptation in Natural and Artificial Systems", Ann Arbor, MI; University of Michigan Press, 1975.
7. Mitchell, M., An Introduction to Genetic Algorithms, MIT Press, 1996.
8. Mitchell, M., J. Holland y S. Forrest, “When Will a Genetic Algorithm Outperform Hill Climbing?”, Advances of Neural Information Processing Systems, No. 6, Morgan Kaufmann, pp. 51-58, 1994.
9. Eshelman, L., “The CHC Adaptive Search Algorithm. How to Have Safe Search When Engaging in Nontraditional Genetic Recombination”. In Rawlins, G., editor, FOGA -1, pages 265–283. Morgan Kaufmann., 1991.
10. Sánchez-Ferrero, G., Arribas, JI, “A Statistical-Genetic Algorithm to Select the Most Significant Features in Mammograms”, Lecture Notes in Computer Science, Volume 4673/2007, 189-196, DOI: 10.1007/978-3-540-74272-2_24, 2007.
11. Kuri-Morales, A., “A statistical genetic algorithm”, Proc. Of the 3d. National Computing Meeting, ENC '99, pp. 215-228, Hgo., México, 1999.
12. Kuri-Morales, A., Villegas-Quezada, C. (1998). *A universal eclectic genetic algorithm for constrained optimization*, Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing. Vol. 1
13. Kuri-Morales, A. (2002). *A methodology for the statistical characterization of genetic algorithms*, MICAI 2002: Advances in Artificial Intelligence : 77-96
14. Henryk UGOWSKI, "Remarks On The Ascent Algorithm For The Linear Minimax Problem", COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, Vol. 8 Iss: 3, pp.181 – 184, 1989.
15. Cybenko., G. "Approximations by superpositions of sigmoidal functions", Mathematics of Control, Signals, and Systems, 2 (4), 303-314, 1989.

References

[1] Kuri-Morales, A., Aldana-Bobadilla, E. Clustering with an N-Dimensional Extension of Gielis Superformula. WSEAS, 343-350, Editor(s)): Zadeh et al., ISBN: 978-960-6766-4, ISSN: 1790-5109, 20/02/2008

[2] Kuri-Morales, A., Aldana-Bobadilla, E. The search for irregularly shaped clusters in data mining. New Fundamental Technologies in Data Mining, Intech, 323-354, Editor(s): Funtasu, K., Hasegawa, K., ISBN: 9789533075471, 2011

[3] Rudolph, G. Convergence Analysis of Canonical Genetic Algorithms. *IEEE Trans. Neural Networks, 1994*, *5*, 96–101.

[4] Kuri-Morales, A.; Aldana-Bobadilla, E. The best genetic algorithm I. In Proceedings of the 12th Mexican International Conference on Artificial Intelligence, Mexico City, Mexico, 24–30 November 2013; pp. 1–15.

[5] Kuri-Morales, A.; Aldana-Bobadilla, E.; López-Peña, I. The best genetic algorithm II. In Proceedings of the 12th Mexican International Conference on Artificial Intelligence, Mexico City, Mexico, 24–30 November 2013; pp. 16–29.

[6] Aldana-Bobadilla, E., Kuri-Morales A. A Clustering Method Based on the Maximum Entropy Principle. Entropy 2015, 17, 151-180

[7] Bhattacharyya, A. On a Measure of Divergence between Two Statistical Populations. *The Indian Journal of Statistics* **1946**. *7*, 401–406.

[8] Kruskal, J.B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* **1964**, *29*, 1–27.

[9] Mahalanobis, P.C. On the generalized distance in statistics. In Proceedings of the National Institute of Sciences of India, Calcutta, India, 16 April 1936; Volume 2, pp. 49–55.

[10] Halkidi, M.; Batistakis, Y.; Vazirgiannis, M. On clustering validation techniques. *J. Intell. Inf. Syst.* **2001**, *17*, 107–145.

[11] Rokach, L.; Maimon, O. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*; Springer: New York, NY, USA, 2005; pp. 321–352.

[12] MacQueen, J. Some methods for classification and analysis of multivariate observations. In *The Fifth Berkeley Symposium on Mathematical Statistics and Probability*; Le Cam, L.M., Neyman, J., Eds.; University of California Press: Berkeley, CA, USA, 1967; Volume 1, pp. 281–297.

[13] Bezdek, J.C. *Pattern Recognition with Fuzzy Objective Function Algorithms*; Springer: New York, NY, USA, 1981.

[14] Dunn, J.C. A fuzzy relative of the ISODATA process and its use in detecting vompact well-separated clusters. *J. Cybern.* **1973**, *3*, 32–57.

[15] Dunn, J.C. Well-separated clusters and optimal fuzzy partitions. *J. Cybern.* **1974**, *4*, 95–104.

[16] Friedman, J.; Hastie, T.; Tibshirani, R. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed.; Springer: Standord, CA, USA, 2001.

[17] Guha, S.; Rastogi, R.; Shim, K. Cure: An efficient clustering algorithm for large databases. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, WA, USA, 1–4 June 1998; pp. 73–84.

[18] Livny, T.Z.R.R.M. Birch: An efficient data clustering method for very large databases. In Proceedings of the ACM SIGMOD international Conference on Management of Data, Montreal, QC, Canada, 1996; pp. 103–114.

[19] Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996; pp. 226–231.

[20] Caruana, R.; Elhaway, M.; Nguyen, N.; Smith, C. Meta clustering. In Proceedings of the Sixth International Conference on Data Mining, Hong Kong, China, 18–22 December 2006; pp. 107–118.

[21] Das, S.; Abraham, A.; Konar, A. *Metaheuristic Clustering*; Springer: Berlin/Heidelberg, Germany, 2009.

[22] Milligan, G.W.; Cooper, M.C. An examination of procedures for determining the number of clusters in a data set. *Psychometrika* **1985**, *50*, 159–179.

[23] Tibshirani, R.; Walther, G.; Hastie, T. Estimating the number of clusters in a data set via the gap statistic. *J. R. Stat. Soc. Ser. B* **2001**, *63*, 411–423.

[24] Li, X.; Mak, M.-W.; Li, C.-K. Determining the optimal number of clusters by an extended rpcl algorithm. *J. Adv. Comput. Intell. Intell. Inf.* **1999**, *3*, 467–473.

[25] Peck, R.; Fisher, L.; van Ness, J. Approximate confidence intervals for the number of clusters. *J. Am. Stat. Assoc.* **1989**, *84*, 184–191.

[26] Yan, M. Methods of Determining the Number of Clusters in a Data Set and a New Clustering Criterion. Ph.D. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, November 2005.

[27] Cha, S.H. Taxonomy of nominal type histogram distance measures. In Proceedings of the American Conference on Applied Mathematics, Harvard, MA, USA, 24–26 March 2008.

[28] Kim, D.-J. A novel validity index for determination of the optimal number of clusters. *IEICE Trans. Inf. Syst.* **2001**, *84*, 281–285.

[29] Liu, Y.; Li, Z.; Xiong, H.; Gao, X.; Wu, J. Understanding of internal clustering validation measures. In Proceedings of the 10th International Conference on Data Mining (ICDM), Sydney, Australia, 13–17 December 2010; pp. 911–916.

[30] Larsen, B.; Aone, C. Fast and effective text mining using linear-time document clustering. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 15–18 August 1999; pp. 16–29.

[31] Strehl, A.; Ghosh, J. Cluster ensembles—A knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **2003**, *3*, 583–617.

[32] Shannon, C.E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423.

[33] Zhao, Y.; Karypis, G. *Criterion Functions for Document Clustering: Experiments and Analysis*; Technical Report #01-40; Available online: http://glaros.dtc.umn.edu/gkhome/node/165 (accessed on 5 January 2015).

[34] Raftery, A.E. A note on bayes factors for log-linear contingency table models with vague prior information. *J. R. Stat. Soc. Ser. B* **1986**, *48*, 249–250.

[35] Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65.

[36] Davies, D.L.; Bouldin, D.W. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *PAMI-1*, 224–227.

[37] Rendón, E.; Garcia, R.; Abundez, I.; Gutierrez, C.; Gasca, E.; del Razo, F.; Gonzalez, A. Niva: A robust cluster validity. In Proceedings of the WSEAS International Conferenceon Communications, Heraklion, Greece, 23–25 July 2008.

[38] Rand, W.M. Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* **1971**, *66*, 846–850.

[39] Hubert, L.; Arabie, P. Comparing partitions. *J. Classif.* **1985**, *2*, 193–218.

[40] Graham, R.L.; Knuth, D.E.; Patashnik, O. *Concrete Mathematics: A Foundation for Computer Science*; Addison-Wesley: Boston, MA, USA, 1989.

[41] Glover, F. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **1986**, *13*, 553–549.

[42] Glover, F.; Laguna, M. *Tabu Search*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1997.

[43] Glover, F.; McMillan, C. The general employee scheduling problem. An integration of MS and AI. *Comput. Oper. Res.* **1986**, *13*, 563–573.

[44] Kirkpatrick, S. Optimization by simulated annealing: Quantitative studies. *J. Stat. Phys.* **1984**, *34*, 975–986.

[45] Kirkpatrick, S.; Gelatt, C.; Vecchi, M. Optimization by simulated annealing. *Science* **1983**, *220*, 671–679.

[46] Dorigo, M.; Blum, C. Ant colony optimization theory: A survey. *Theor. Comput. Sci.* **2005**, *344*, 243–278.

[47] Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.

[48] Bäck, T.; Schwefel, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.* **1993**, *1*, 1–23.

[49] Beyer, H.G.; Schwefel, H.P. *Evolution Strategies–A Comprehensive Introduction*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2002; Volume 1.

[50] Fogel, L.J. The future of evolutionary programming. In Proceedings of the Twenty-Fourth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 5–7 November 1990; Volume 2, pp. 1036–1038.

[51] Banzhaf, W.; Nordin, P.; Keller, R.E.; Francone, F.D. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications (The Morgan Kaufmann Series in Artificial Intelligence)*; Morgan Kaufmann Publishers: Burlington, MA, USA, 1997.

[52] Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, 2nd ed.; MIT Press: Cambridge, MA, USA, 1992.

[53] Pandey, Hari Mohan and Chaudhary, Ankit and Mehrotra, Deepti A comparative review of approaches to prevent premature convergence in GA. Applied Soft Computing, 2014; Volume 24, pp. 1047–1077.

[54] Kuri-Morales, A.; Villegas, C.Q. A universal eclectic genetic algorithm for constrained optimization. In Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, September 1998; Volume 1, pp. 518–524.

[55] Abudalfa, S.I. Metaheuristic Clustering Algorithm. Ph.D. Thesis, The Islamic University of Gaza, Gaza, Palestine, 2010.

[56] Caballero, R.; Laguna, M.; Martí, R.; Molina, J. *Multiobjective Clustering with Metaheuristic Optimization Technology*; Available online: http://www.uv.es/sestio/TechRep/tr02-06.pdf (accessed on 5 January 2015).

[57] Shelokar, P.S.; Jayaraman, V.K.; Kulkarni, B.D. An ant colony approach for clustering. *Anal. Chim. Acta* **2004**, *509*, 187–195.

[58] Faivishevsky, L.; Goldberger, J. A nonparametric information theoretic clustering algorithm. In Proceedings of the 27th International Conference on Machine Learning, Israel, Israel, 21–24 June 2010.

[59] Gokcay, E.; Principe, J.C. Information theoretic clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 158–171.

[60] Hino, H.; Murata, N. A nonparametric clustering algorithm with a quantile-based likelihood estimator. *Neural Comput.* **2014**, *26*, 2074–2101.

[61] Jenssen, R.; Hild, K.E.; Erdogmus, D.; Principe, J.C.; Eltoft, T. Clustering using Renyi's entropy. In Proceedings of the International Joint Conference on Neural Networks, Portland, OR, USA, 20–24 July, 2003; pp. 523–528.

[62] Slonim, N.; Atwal, G.S.; Tkačik, G.; Bialek, W. Information-based clustering. *Proc. Natl. Acad. Sci. USA* **2005**, *102*, 18297–18302.

[63] Sugiyama, M.; Niu, G.; Yamada, M.; Kimura, M.; Hachiya, H. Information-maximization clustering based on squared-loss mutual information. *Neural Comput.* **2014**, *26*, 84–131.

[64] Cover, T.M.; Thomas, J.A. *Elements of Information Theory*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2006.

[65] Cheng, C.-H.; Fu, A.W.; Zhang, Y. Entropy-based subspace clustering for mining numerical data. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 15–18 August 1999; pp. 84–93.

[66] De Sa, J.P.M. *Pattern Recognition: Concepts, Methods, and Applications*; Springer: Berlin/Heidelberg, Germany, 2001.

[67] M. Lichman. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences,2013

[68] Duda, R.O.; Hart, P.E.; Stork, D.G. *Pattern Classification*; Wiley: New York, NY, USA, 2000.

[69] Haykin, S. *Neural Networks: A Comprehensive Foundation*, 2nd ed.; Prentice Hall: Upper Saddle River, NY, USA, 1999.

[70] Gallager, R.G. *Information Theory and Reliable Communication*; Wiley: Hoboken, NJ, USA, 1968.

[71] Jaynes, E.T. Information theory and statistical mechanics. *Phys. Rev.* **1957**, *106*, 620–630.

[72] Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*; Wiley: Chichester, UK, 2001.

[73] Snyman, J. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*; Springer: New York, NY, USA, 2005.

[74] Thomas, G.B.; Finney, R.L.; Weir, M.D. *Calculus and Analytic Geometry*; Addison-Wesley: Boston, MA, USA, 1988.

[75] Censor, Y. Pareto optimality in multiobjective problems. *Appl. Math. Optim.* **1977**, *4*, 41–59.

[76] Sindhya, K.; Sinha, A.; Deb, K.; Miettinen, K. Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In Proceedings of the IEEE Congress on Evolutionary Computation, Trondheim, Norway, 18–21 May 2009; pp. 2919–2926.

[77] Zitzler, E.; Laumanns, M.; Thiele, L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*; TIK-Report 103; Available online: http://www.kddresearch.org/Courses/Spring-2007/CIS830/Handouts/P8.pdf (accessed on 5 January 2015).

[78] Steliga, K.; Szynal, D. On Markov-Type Inequalities. *Int. J. Pure Appl. Math.* **2010**, *58*, 137–152.

[79] Casella, G.; Robert, C.P. *Monte Carlo Statistical Methods*; Springer: New York, NY, USA, 1999.

[80] Johnson, J.L. *Probability and Statistics for Computer Science*; Wiley: Hoboken, NJ, USA, 2003.

[81] Abalone Data Set. Available online: http://archive.ics.uci.edu/ml/datasets/Abalone (accessed on 30 December 2014).

[82] Cars Data Set. Available online: http://archive.ics.uci.edu/ml/datasets/Car+Evaluation (accessed on 30 December 2014).

[83] Census Income Data Set. Available online: http://archive.ics.uci.edu/ml/datasets/Census+Income (accessed on 30 December 2014).

[84] Hepatitis Data Set. Available online: http://archive.ics.uci.edu/ml/datasets/Hepatitis (accessed on 30 December 2014).

[85] Yeast Data Set. Available online: http://archive.ics.uci.edu/ml/datasets/Yeast (accessed on 30 December 2014).

[86] Agresti, A. *Categorical Data Analysis*; Wiley: Hoboken, NJ, USA, 2002.

[87] Shampine, L.F.; Allen, R.C.; Pruess, S. *Fundamentals of Numerical Computing*; Wiley: New York, NY, USA, 1997.

[88] M. Mitchell An Introduction to Genetic Algorithms; MIT Press, pp 129–130,1996

[89] M. Mitchell and J. Holland and S. Forrest; When Will a Genetic Algorithm Outperform Hill Climbing?; Advances of Neural Information Processing Systems, No. 6, Morgan Kaufmann, pp. 51-58,1994

[90] M. Molga and C. Smutnicki; Test functions for optimization needs; Retrieved March 11, 2012 from http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf, pp. 41-42

[91] L. Eshelman; The CHC Adaptive Search Algorithm. How to Have Safe Search When Engaging in Nontraditional Genetic Recombination; In Rawlins, G., editor, FOGA -1, pages 265–283. Morgan Kaufmann, 1991

[92] J. Rezaee and A. Hashemi and N. Nilsaz and H. Dezfouli; Analysis of the Strategies in Heuristic Techniques for Solving Constrained Optimisation Problems; Journal of American Science, 8(10), 2012

[93] G. Sánchez-Ferrero and J. Arribas; A Statistical-Genetic Algorithm to Select the Most Significant Features in Mammograms; Lecture Notes in Computer Science, Volume 4673/2007, 189-196, DOI: 10.1007/978-3-540-74272-2_24, 2007

[94] J. Digalakis and K. Margaritis; An experimental study of Benchmarking functions for genetic algorithms; Intern. J. Computer math., vol. 79(4), pp. 403–416, 2002

[95] J-H. Kim and H. Myung; Evolutionary Programming Techniques for Constrained Optimization Problems; IEEE Transactions on Evolutionary Computation, pp. 129, July 1997

[96] A. Kuri-Morales; A statistical genetic algorithm; Proc. Of the 3d. National Computing Meeting, ENC '99, pp. 215-228, Hgo., México", 1999

[97] A novel cluster validity index: variance of the nearest neighbor distance; F. Kovacs and R. Ivancsy; WSEAS Transactions on Computers, Volume 5, Nummber 3, pp. 477-483, 2006