E2PAMEA: A fast evolutionary algorithm for extracting fuzzy emerging patterns in big data environments

Ángel Miguel García-Vico[a,*], Francisco Charte[a], Pedro González[a], David Elizondo[b], Cristóbal José Carmona[a,c]

[a]*Interuniversity Andalusian Institute on Data Science and Computation Intelligence, University of Jaén, 23071 Jaén, Spain*
[b]*School of Computer Science and Informatics, De Monfort University, Leicester LE1 9BH, United Kingdom*
[c]*Leicester School of Pharmacy, De Montfort University, LE1 9BH Leicester, United Kingdom*

Abstract

In this paper, a cooperative-competitive multi-objective evolutionary fuzzy system called E2PAMEA is presented for the extraction of emerging patterns in big data environments. E2PAMEA follows an adaptive schema to automatically employ different genetic operators according to the learning needs, which avoid the tuning of some parameters. It also employs a token-competition-based procedure for updating an elite population where the best set of patterns found so far is stored. In addition, a novel MapReduce procedure for an efficient computation of the evaluation function employed for guiding the search process is proposed. The method, called Bit-LUT employs a pre-evaluation stage where data is represented as a look-up table made of bit sets. This look-up table can be employed later in the chromosome evaluation by means of bitwise operations, reducing the computational complexity of the process.

The experimental study carried out shows that E2PAMEA is a promising alternative for the extraction of high-quality emerging patterns in big data. In addition, the proposed Bit-LUT evaluation shows a significant improvement on efficiency with a great scalability capacity on both dimensions of data, which enables the processing of massive datasets faster than other alternatives.

*Corresponding author. Tel:+34-953213356

Email addresses: agvico@ujaen.es (Ángel Miguel García-Vico), fcharte@ujaen.es (Francisco Charte), pglez@ujaen.es (Pedro González), elizondo@dmu.ac.uk (David Elizondo), ccarmona@ujaen.es (Cristóbal José Carmona)

Keywords: Emerging pattern mining, Evolutionary fuzzy systems,
Multi-objective evolutionary algorithm, Big data

1. Introduction

The amount of data generated everyday has grown exponentially in recent decades due to the explosion of devices that generate data. Valuable insights can be extracted from these large quantities of data. It is well-known that nowadays big data is a hot topic in business and academia [1]. However, classic machine learning approaches are not suitable for dealing with these large amounts of data efficiently. There are currently many definitions for big data [2; 3; 4]. Usually, a problem is considered as a big data one when the amount of information exceeds the computing capacities of a single machine due to several constraints such as volume, arrival speed or heterogeneity of data.

Metaheuristics [5] are optimisation methods that combine local improvement procedures and high-level strategies. These methods are characterised by their ability to exploit information about an unknown initial search space in order to bias the subsequent search towards useful subspaces. Due to its capacity to find good solutions (the optimal one is not guaranteed) in a reasonable amount of time, metaheuristics have been widely used in many machine learning tasks such as classification [6] and clustering [7], amongst others, as they can be seen as optimisation problems.

Within machine learning, there is a set of tasks whose main aim is to characterise relationships in data with respect to a variable of interest. This is known as supervised descriptive rule discovery (SDRD) [8]. These relationships should be understandable by the expert in order to help them take decisions [9]. Throughout the SDRD literature, exact algorithms have been developed for the extraction of all possible relationships that fulfils these requirements. However the main drawback of these methods is the high computational cost when data grows and the need to discretise numerical data. In this way, metaheuristics, especially evolutionary algorithms (EAs) [10], have been employed within SDRD in order to improve the trade-off between the quality of the solution and its computational time [11; 12; 13].

Unfortunately, the application of EAs in machine learning tasks usually involve the use of an evaluation function that needs to traverse the whole dataset each time a chromosome of the population is evaluated throughout the evolutionary process. Therefore, when the amount of data grows as in big data problems,

the application of these methods becomes unfeasible. One of the most common strategies performed in big data analysis is distributed computing. Here the data and the processing are distributed amongst the computing nodes of a cluster. Each node only processes a subset of the data. The final result is then extracted by means of an aggregation of all the outputs of the computing nodes. This strategy is well-suited for big data analysis on large computing clusters or cloud computing environments [14]. The most popular paradigm that follows this strategy is MapReduce [15], where their open-source frameworks Apache Hadoop [28] and Apache Spark [16] are mainly employed for both distributed storage and the processing of data.

This paper is focused on the extraction of high-quality emerging patterns by means of a multi-objective EA (MOEA) called Extraction of Emerging Patterns through an Adaptive Multi-objective Evolutionary Algorithm (E2PAMEA). The main aim of this algorithm is twofold: firstly, to improve the quality of the extracted knowledge while parameters tuning is reduced with respect to similar alternatives. For this purpose, the algorithm employs an adaptive approach for the extraction of interesting emerging patterns, where different genetic operators are automatically applied according to the learning needs. Secondly, to reduce the execution time in big data environments avoiding data-division problems. Following similar approaches in the literature, the evaluation process is performed in a distributed way by means of a novel MapReduce procedure called Bit Look-Up Table (Bit-LUT). This procedure is based on the transformation of the data into a look-up table made of bit sets where the coverages of the different attribute-value pairs of the problem are stored. This allows the method to reduce the evaluation time and the use of physical memory due to the use of bits and bitwise operations. In addition, the scalability is also improved with respect to the number of variables, which is a major drawback on other approaches.

The paper is organised as follows: Section 2 presents the main concepts employed in this paper. In Section 3 the Bit-LUT evaluation approach is shown. Section 4 presents the E2PAMEA algorithm. Section 5 shows the experimental analysis carried out to determine the quality of the proposed method. In Section 6 a discussion of the results is presented. Finally, the conclusions extracted from this work are depicted in Section 7.

2. Background

The main concepts employed throughout this paper are described in this section. Firstly, the big data problem is defined in Section 2.1. Secondly, a revision

of metaheuristics employed in big data analysis is depicted in Section 2.2. Finally, the EPM task is presented in Section 2.3.

2.1. Big Data

Big data can be defined as large volumes of data, arriving at high speed into the systems from a variety of sources [2]. These characteristics of big data impose new challenges upon us for the extraction of knowledge in these kinds of environments. For example, traditional algorithms cannot handle these large amounts of data as they assume that data fit in memory. Nowadays, big data analysis is applied in almost every field of application, including: education [17], smart cities [18; 19], security [20] and many machine learning tasks [21; 22; 23] amongst others. For this reason, there are several frameworks developed for dealing with big data. One of the most popular is the MapReduce paradigm [24; 25]. It is a framework for distributed computing based on a divide-and-conquer strategy. One of the main advantages of this framework is that it can be easily deployed within large computing centres as it can automatically handle all the necessary mechanisms involved in distributed computation in a way which is transparent to the developer. Therefore, experts are able to focus on how the data will be processed by their algorithms.

As implied by the name, MapReduce defines two main functions [26]: map and reduce. These functions must be implemented by the developer. The definition of these functions is given below [26]:

- Map phase. Data is split into several partitions. These data partitions are automatically sent to the computing nodes of the cluster optimising data locality. Each partition is identified with a tuple $(key, value)$. After that, all the nodes process their partitions concurrently. Finally, the output of this procedure is another tuple $(key', value')$ with the result of the map operation. These pairs are then shuffled and ordered by key if necessary and become the output of the reduce function.

- Reduce phase. This aggregates the intermediate results produced by the map function. The reduce function is executed on each key' key, where all the values are aggregated following the reduce function. Thus, it returns a new tuple $(key', value'')$, which corresponds to the final result for each key.

Despite the advantages of MapReduce, their jobs are loaded from disk each time they are applied. Therefore, the performance on iterative jobs is decreased

[27]. Fortunately, alternative solutions have been developed. Currently, Apache Spark [16], amongst others, solve this issue. Spark uses a structure called resilient distributed dataset (RDD) [28] which provides a set of parallel transformations and actions over data. The key point of RDDs is that intermediate results produced by several transformations or map procedures can persist in the main memory, so the re-execution of the whole MapReduce pipeline across different jobs is not necessary.

2.2. Metaheuristics for big data analysis

Metaheuristics are usually nature-inspired methods aimed at solving optimisation problems. The main advantage of these methods is their capacity for finding a solution close enough to the optimal one within an affordable amount of time when dealing with complex problems, e.g., in [29; 30; 31; 32; 33]. Moreover, they are able to find solutions without describing in detail the problem, although it is possible to add expert knowledge to the search process in order to improve it. In fact, a learning task can be seen as an optimisation problem as well, so it is well known that metaheuristics have been widely applied to many machine learning tasks [34]. In concrete, many metaheuristics have been widely applied on classification tasks, especially for feature selection, where EAs [35; 36], particle swarm optimisation (PSO) algorithms [37], GRASP [38], Tabu Search [39], amongst others have been used. Also, metaheuristics have been widely employed for optimising the parameters of neural networks [40], support vector machines [41], or ensembles [42]. They have also been employed for clustering [7], where swarm intelligence methods such as PSO, ant colony optimization, artificial bee colony, and others are the most employed [43]. For association rule mining, methods such as cuckoo search [44], bee colony [45], firefly algorithm [46] amongst others have been employed. Finally, for the SDRD task, the majority of approaches presented apply an EA [47; 48; 49; 50].

Despite the good results obtained with the use of metaheuristics, their application to machine learning tasks over big data environments is a challenge due to scalability issues. This problem is mainly associated with the computation of an evaluation function for determining the quality of each candidate solution. This evaluation usually involves a full traversal of the dataset, which is costly. Therefore, it is necessary to create strategies for the development of scalable metaheuristics. In this case, a MapReduce-based approach is the most widely employed strategy for the implementation of these techniques. In particular, two main approaches have been proposed [26]:

- A local approach (also known as approximate model). A baseline algorithm is executed within the map phase, extracting insights related to the partition of the data. Next, all the pieces of knowledge are combined in the reduction phase. The final result of the algorithm depends on the partitions employed.
- A global approach (also known as exact model). The entire method, or one of its most expensive tasks, is designed to work in a distributed way. With this kind of algorithm, the result is always the same regardless of the partitions employed.

In data mining tasks, the whole dataset must be within each map in order to properly evaluate the individual, which is not possible in big data by definition. Therefore, pioneering metaheuristics developed for data mining tasks in big data environments were mainly based on local approaches, which approximates the value of the evaluation function [51; 52; 53; 54]. These approaches can efficiently scale-up with respect to the amount of data. However, there are several drawbacks that should be taken into consideration: firstly, the aggregation performed in the reduce phase is not trivial and must be carefully designed. Secondly, as knowledge is extracted from a random partition of data, knowledge extracted may suffer from data-division problems such as skewed class distributions, lack of training data, and so on [55]. This could produce the extraction of less accurate models.

These problems can be avoided by means of a global approach where the evaluation function can be exactly computed regardless the partitions employed. Examples of this approach can be found in [56; 57; 58; 59]. In general, global methods are desirable over local ones as they are not affected by data-division problems. However, the design of this kind of methods is usually complex as it could require several MapReduce stages or additional processing with respect to the original algorithm in order to verify the exact computation. Additionally, this kind of approaches usually applies the same MapReduce procedure on each iteration. As a result, global approaches are usually slower than local ones. In this paper, the proposed Bit-LUT approach encourage the employment of global approaches by means of an easy method for an exact evaluation of the individuals of a population, reducing its computational cost and improving its scalability with respect to previous approaches.

2.3. Emerging pattern mining

EPM [60; 13] is a data mining task that belongs to the SDRD [8] framework. It searches for those patterns containing a significant change in their support from

one dataset to another one or from a given class with respect to the remaining classes in a single dataset.

Within this context, a dataset D which corresponds to a specific problem is defined as follows: let $V = \{v_1, v_2, ..., v_n\}$ be the set of variables of the problem. These can be categorical or numerical. One of them is the class or target variable of the problem, denoted as v_c. For this task, it is assumed that the class is always a categorical variable. $X_i = \{x_{i1}, x_{i2}, ..x_{in}\}$ is defined as the set of different categories for the categorical variable V_i, the domain of a numerical variable V_i, or a set of linguistic labels (LLs) [61] for representing numeric variables by means of fuzzy logic. Let $e = \{(v_1, x_{1j}), (v_2, x_{2j}), ..., (v_n, x_{nj})\}$ be an example (or instance) of the problem. Finally, $D = \{e_1, e_2, ..., e_n\}$ is defined as a dataset, which is a set of examples of the problem.

A selector [62] is defined as a tuple $(v_i, x_{i,j})$ where $v_i \in V$ and $x_{i,j} \in X_i$ that connects a variable to its values. Let $I = (i_1, i_2, ..., i_n)$ be the set of selectors of a given problem. A pattern P is defined as a subset of I. The selectors in P are usually joined by conjunctions or in disjunctive normal form. In this way, P covers e_i, or e_i contains P if and only if all the selectors in P are satisfied by e_i. Finally, P is defined as an EP if its growth rate (GR) is higher than $\rho > 1$. The GR is calculated as in Eq. 1.

$$GR(P) = \begin{cases} 0, & IF\ Sup_{D_1}(P) = Sup_{D_2}(P) = 0, \\ \infty, & IF\ Sup_{D_1}(P) \neq 0 \wedge Sup_{D_2}(P) = 0, \\ \frac{Sup_{D_1}(P)}{Sup_{D_2}(P)}, & otherwise \end{cases} \tag{1}$$

where $Sup_{D_i}(P)$ is the support of the pattern P in the dataset D_i. The support is calculated as $Sup_{D_i}(P) = \frac{count_{D_i}(P)}{|D_i|}$ where $count_{D_i}(P)$ is the number of instances covered by P on D_i while $|D_i|$ is the number of instances in D_i.

The main objectives of this task are the description of the discriminative characteristics between classes or the description of emerging tendencies in data. In this study the first objective is tackled in order find the discriminative characteristics of the classes of a given problem. In fact, it is important to find a trade-off between the discriminative power of the patterns extracted and their capacity to easily explain the underlying phenomena to the experts. Therefore, several quality measures should be computed in order to achieve this purpose. All these measures, employed within the SDRD framework, can be easily calculated by means of a contingency table where the number of correctly/incorrectly cov-

ered/uncovered instances are shown for each pattern. This kind of table is presented in Table 1.

Table 1: Contingency table of a pattern.

	Class	No class
Covered	tp	fp
Not covered	fn	tn

where tp is the number of instances correctly covered by the pattern, fp is the number of incorrectly covered instances, fn is the number of incorrectly uncovered instances and tn is the number of correctly uncovered instances. The most widely used quality measures within EPM are presented in Table 2 [63], where T is equal to the total number of instances.

Table 2: Quality measures used in EPM for the determination of the quality of a pattern.

Name	Abbreviation	Formula
Confidence [64]	Conf	$\frac{tp}{tp+fp}$
Weighted Relative Accuracy [65]	WRAcc	$\frac{tp+fp}{T}\left(\frac{tp}{tp+fp} - \frac{tp+fn}{T}\right)$
Growth Rate[60]	GR	$\frac{p(fp+tn)}{n(tp+fn)}$
True Positive Rate [66]	TPR	$\frac{tp}{tp+fn}$
False Positive Rate [67]	FPR	$\frac{fp}{fp+tn}$

Classic deterministic algorithms have been developed for EPM such as DeEPS [68]. There are also works based on FP-trees such as Tree-based JEP-C for EPM [69]. Moreover, there are developments based on efficient data structures such as the CP-tree for the StrongJEP-C algorithm [70] and DGCP-Tree [71], which imposes some constraints in order to find subsets of relevant emerging patterns. Evolutionary fuzzy systems [72] have been presented in [50; 59] for EPM. These methods improve the quality and the interpretability of the results by making use of fuzzy logic. In addition, the execution time is significantly reduced with respect to the classic approaches. Nevertheless, it is necessary to tune many parameters. In fact, the work in [73] is oriented towards big data, but its execution time is usually high.

3. The Bit-LUT evaluation approach

The use of EAs in learning tasks usually involves the application of an evaluation function that must traverse the whole dataset each time a chromosome is eval-

uated. As can be observed, the performance of an EA for learning tasks is highly dependent on the amount of data. The proposed evaluation approach has been developed using the Apache Spark framework following a MapReduce approach. In a nutshell, the new proposed evaluation procedure is based on an optimised look-up table calculated before the beginning of the evolutionary process in order to avoid the full traversal of the raw dataset for each chromosome evaluation. This table can be employed on an EA to improve its performance at the chromosome evaluation stage. By means of this method, the fitness calculation procedure:

1. Improve its processing time.
2. Reduce memory consumption.
3. Properly scale up on both dimensions of data: number of instances and number of variables.

The proposed chromosome evaluation approach by means of MapReduce is firstly performed by means of a pre-evaluation stage, where raw data is transformed in order to be efficiently processed. This is carried out before the evolutionary process and it is independent of the representation employed. Next, the chromosome evaluation stage will determine the quality of the chromosomes by means of the look-up table. This will be carried out throughout the evolutionary process and depends on the chromosome representation employed. This section describes the pre-evaluation stage as it is independent of the underlying EA.

3.1. Bit-LUT pre-evaluation

It is assumed that training data are already split into p partitions across the computing cluster. In addition, each instance e is identified with an unique index k. For the pre-evaluation stage it is assumed that there is a fixed, immutable amount of selectors. Thus, the membership degree of a given instance is always the same with respect to a given selector throughout the whole process. For the sake of understanding, the term membership degree will be employed to refer to all categorical, numerical or fuzzy selectors. According to this idea, a look-up table can be created where the membership degree of all the instances with respect to a given selector is stored. Within Bit-LUT, this look-up table contains, for each selector, a bit set $BS_i^j = \{x_1, x_2, ..., x_n\}$ where n is the total number of instances and x_k is a binary value, calculated as in Eq. 2, where a value of one means that the selector covers the instance k or zero otherwise. Using this approach, we can assert that only the necessary membership degree computations are undertaken, thus improving the performance, while physical memory consumption is reduced due to the use of a single bit instead of a 64-bit value for each instance and selector.

Algorithm 1 Bit-LUT pre-evaluation stage. Map procedure

Input:
A dataset D composed of tuples (k, e_k)
Output:
A matrix M with $|I|$ rows and n columns.
1: **for** $i \in 1, \ldots, |V|$ and $j \in 1, \ldots, |X_i|$ **do**
2: **for all** $(k, e_k) \in D$ **do**
3: $M_{ij,k} \leftarrow cov\,(k, i, j)$
4: **end for**
5: **end for**

Algorithm 2 Bit-LUT pre-evaluation stage. Reduce procedure

Input:
M_p, a set of matrices with $|I|$ rows and n columns
Output:
$Data'$, a matrix with $|I|$ rows and n columns.
1: $Data' \leftarrow M_1 \mid \ldots \mid M_p$

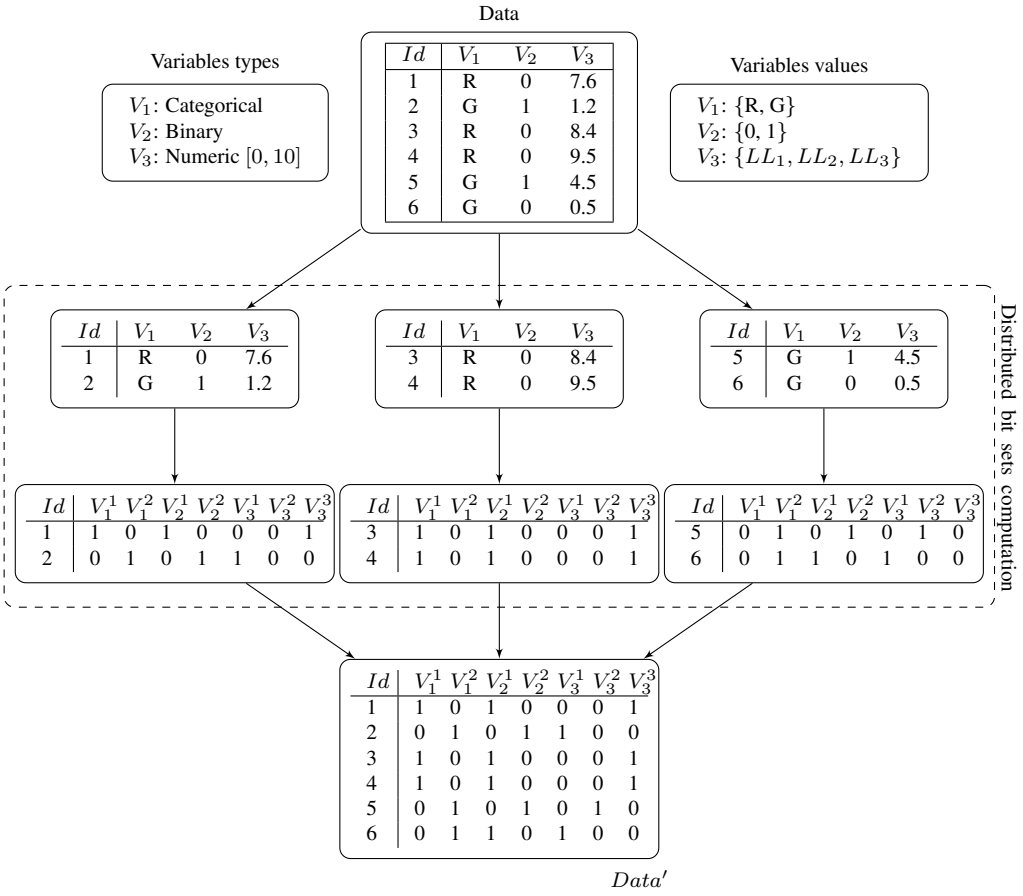
Bit-LUT calculates its look-up table by means of a single MapReduce job, presented in Algorithm 1 and 2 for the map and reduce procedure, respectively. On each data partition or map, the binary value for the instance k that determines whether it is covered by a given selector or not is determined by the coverage (cov) function presented in Eq. 2 (Algorithm 1, line 3).

$$cov(k,i,j) = \begin{cases} 1, & If\ TC\left(\mu_i^1\left(k,i\right), ..., \mu_i^{l_i}\left(k,i\right)\right) = \mu_i^j\left(k,i\right), \\ 0, & otherwise \end{cases} \tag{2}$$

where TC is the maximum t-conorm and $\mu_i^j\,(k,i)$ is the membership degree of the value in instance k for variable i with respect to the value j for variable i. In case of a non-fuzzy selector, $\mu_i^j\,(k,i) = 1$ if the value in instance k for variable i satisfies the selector or zero otherwise. Then it is determined that the selector covers the instance if and only if its belonging degree with respect to the given selector is the maximum with respect to all other possible values for that variable. This operation is carried out over all selectors for each instance in the partition.

Finally, the map phase returns for each partition a partial matrix of bits according to the data it owns. After that, all these partial results are collected in order to extract a final matrix of bits $Data'$ at the reduce phase, where each row represents the coverage of a selector. The aggregation is performed by means of a bitwise OR operation on those rows referring to the same selector (Algorithm 2, line 2). As can be observed, the number of rows of $Data'$ is equal to the number of selectors of the problem, i.e., $|I|$ while the number of columns depends on the

number of instances. In this way, if $Data'$ is stored in a distributed data structure such as Apache Spark RDDs [28], the algorithm is able to properly scale up with respect to the number of selectors of the problem as well. An example of this pre-evaluation stage, where $Data'$ is calculated on a 6-instance dataset using 3 partitions, is presented in Figure 1. The coverage of the selectors is presented in columns instead of rows to facilitate the understanding.



Id	V_1	V_2	V_3
1	R	0	7.6
2	G	1	1.2
3	R	0	8.4
4	R	0	9.5
5	G	1	4.5
6	G	0	0.5

Id	V_1	V_2	V_3
1	R	0	7.6
2	G	1	1.2

Id	V_1	V_2	V_3
3	R	0	8.4
4	R	0	9.5

Id	V_1	V_2	V_3
5	G	1	4.5
6	G	0	0.5

Id	V_1^1	V_1^2	V_2^1	V_2^2	V_3^1	V_3^2	V_3^3
1	1	0	1	0	0	0	1
2	0	1	0	1	1	0	0

Id	V_1^1	V_1^2	V_2^1	V_2^2	V_3^1	V_3^2	V_3^3
3	1	0	1	0	0	0	1
4	1	0	1	0	0	0	1

Id	V_1^1	V_1^2	V_2^1	V_2^2	V_3^1	V_3^2	V_3^3
5	0	1	0	1	0	1	0
6	0	1	1	0	1	0	0

Id	V_1^1	V_1^2	V_2^1	V_2^2	V_3^1	V_3^2	V_3^3
1	1	0	1	0	0	0	1
2	0	1	0	1	1	0	0
3	1	0	1	0	0	0	1
4	1	0	1	0	0	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	1	0	0

Figure 1: Example of a Bit-LUT pre-evaluation stage using a dataset with six instances and three partitions. The coverage of selectors is presented in columns instead of rows to facilitate the understanding.

3.2. Computational complexity

The computational complexity of the Bit-LUT evaluation is analysed in this section. The complexity of the fitness calculation stage without the Bit-LUT ap-

proach is $O(ISN)$, where I is the number of individuals in the population, S is the number of selectors, and N is the number of instances of the problem. The reason for this is that, for each instance of the problem, it is necessary to calculate its fuzzy belonging degree with respect to the selectors represented by the individuals. In the big data scenario, it is assumed that $N >> S >> I$, so the evaluation stage is a very heavy process.

Bit-LUT softens this complexity by means of a pre-evaluation stage, whose complexity is $O(SN)$, in order to create a look-up table. Thanks to this table, the computational complexity of the fitness calculation within the evolutionary process is reduced to $O(IS)$, as checking the coverage of the instances by a given selector has a constant cost. Therefore, the proposed Bit-LUT evaluation approach reduces, by an order of magnitude, the computational cost of evaluating the chromosomes.

4. The E2PAMEA algorithm

In this section an EA which employs the proposed evaluation approach is presented. In particular, E2PAMEA is an EFS that looks for emerging patterns with a good trade-off between its reliability and descriptive capacity in the context of a big data problem. The method is able to extract patterns in a disjunctive normal form (DNF). For this purpose, an adaptive multi-objective evolutionary approach called NSGA-II adaptive [74] is employed. This is due to the complexity of finding EPs with high reliability, interestingness and understandability, as multiple contradictory objectives must be optimised at the same time. In summary, E2PAMEA presents the following main characteristics:

1. Use of fuzzy logic for representing numeric variables.
2. Codification of chromosomes by means of a "chromosome = rule" approach.
3. Evolutionary process based on NSGA-II adaptive with specific operators for EPM.
4. Cooperative-competitive approach within an elite population. The elite population will keep the best set of patterns found so far throughout the evolutionary process.
5. Employment of the Bit-LUT evaluation approach to improve the processing time and scalability in both dimensions of data.

The main elements of the proposed algorithm are described in the following subsections. First, the pattern representation within E2PAMEA is described in

Subsection 4.1. After that, the proposed evaluation process is depicted in Subsection 4.2. Next, the genetic operators and their application are described in Subsection 4.3. The reinitialisation procedure is depicted in Subsection 4.4. Finally, the operational schema is shown in Subsection 4.5.

4.1. Pattern representation

Patterns extracted by E2PAMEA are presented in DNF form and they use fuzzy logic for the representation of numerical variables such as LLs [61]. The use of LLs allows us both to improve its robustness [55; 75] and to extract more interpretable knowledge than other representations [76]. Each fuzzy set that corresponds to a LL can be defined by the user when expert knowledge is available or by means of triangular membership functions otherwise.

In Fig 2 an example of a chromosome in E2PAMEA is presented. A DNF pattern can activate several values for each variable, which are joined by disjunctions. Therefore, an individual is represented by means of a binary vector. Its size is the number of selectors of the problem, i.e., $|I|$. Moreover, an integer value is added into this representation in order to codify the class of the pattern. This value represents the i-th nominal value of the class. Using this representation, a selector participates in the pattern if its corresponding value in the binary vector is equal to one. It is important to remark that a variable does not participate in the pattern if all its values are zero or one. This representation allows us the extraction of emerging patterns for all the classes of the problem.

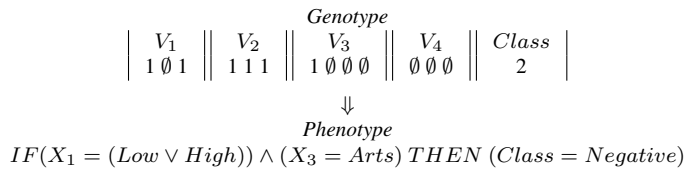


Figure 2: Representation of a fuzzy DNF pattern with continuous and categorical variables in E2PAMEA.

Using this representation, an instance is covered if and only if its Antecedent Part Compatibility (APC) is higher than zero [50]. This value is the degree of compatibility of the example with respect to the antecedent part of the pattern. It is calculated by means of an initial application of a fuzzy t-conorm which represents the fuzzy disjunction together with a fuzzy t-norm afterwards which represents the fuzzy conjunction. Therefore, if the instance has a membership degree higher than zero in the fuzzy subspace delimited by the antecedent part of the pattern, then the instance is covered.

4.2. Chromosome evaluation with Bit-LUT

The evaluation of chromosomes is carried out efficiently by means of $Data'$, extracted previously during the pre-evaluation process of Bit-LUT. According to the representation employed and $Data'$, the coverage for a given individual P can be calculated as depicted in Eq. 3.

$$Cov(P_i) = \left(BS_1^1|BS_1^2|...|BS_1^j\right) \,\&\, \left(BS_2^1|BS_2^2|...|BS_2^j\right) \,\&...\&\, \left(BS_n^1|BS_n^2|...|BS_n^j\right) \tag{3}$$

where $\&$ and $|$ are the bitwise AND and OR operators, respectively. It is important to remark that, using this representation, if a selector j for variable i does not participate in the pattern, then $BS_i^j = 0$, as it is the neutral element for the bitwise OR operator. Moreover, if a whole variable i does not participate in the pattern, then $\left(BS_i^1|BS_i^2|...|BS_i^j\right) = 1$, as this is the neutral element for the bitwise AND operator.

As can be observed, by means of Bit-LUT this evaluation procedure can be easily parallelised by means of a MapReduce job, where the map phase calculates all possible OR operations between elements that belongs to the same variable, while the reduce phase performs the AND operations on the previous result in order to obtain the final coverage of the pattern.

Finally, the contingency table for an individual P using Bit-LUT is calculated as shown in Table 3, where the symbol ! means the complement of that bit set and BS_{class}^j is the bit set representing the class of the pattern.

Table 3: Contingency table of a pattern by means of the Bit-LUT evaluation process.

	Class	No class
Covered	$tp = \|Cov(P)\ \&\ BS_{class}^j\|$	$fp = \|Cov(P)\ \&\ !BS_{class}^j\|$
Not covered	$fn = \|!Cov(P)\ \&\ BS_{class}^j\|$	$tn = \|!Cov(P)\ \&\ !BS_{class}^j\|$

Returning to the example in Fig. 1, a pattern $P : V_1 = R \wedge V_3 = (LL_1 \vee LL_3) \rightarrow V_2 = 0$ is analysed in Fig. 3.

4.3. Genetic operators

Firstly, the population of E2PAMEA is initialised by means of an oriented initialisation procedure. With this method, 75% of the individuals are generated with at most 25% of their variables randomly initialised, while the remaining individuals are generated at random. The idea is to perform an initialisation that promotes generality within the patterns of the population.

Within E2PAMEA, a binary tournament selection [77] is employed in the selection process. The different genetic operators employed are presented below:

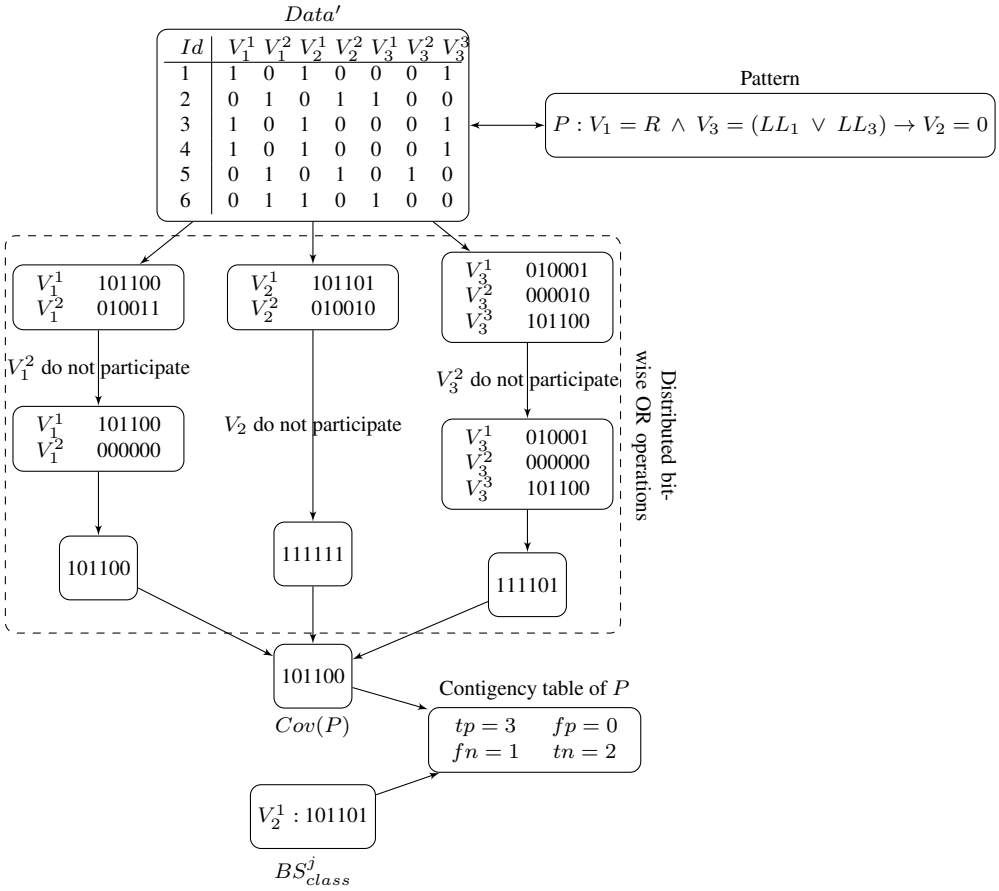


Figure 3: Chromosome evaluation in E2PAMEA by means of MapReduce using Bit-LUT data.

- Two-point crossover [78] for the intensification of promising areas of the search space.
- HUX crossover [79] for diversification in order to avoid stagnation.
- Mutation operator that completely removes a variable of a pattern, by setting all its genes to zero. This allows generalisation of the patterns.
- Mutation based on a random change in a single gene for diversification in the searching process.

E2PAMEA uses an adaptive strategy for the application of different genetic operators according to how the population is evolving. This approach avoids tuning the parameters for the operator's probabilities at the beginning of the process. Initially, the application probabilities of all operators are uniformly distributed in order to avoid any bias in the application of the operators. Next, these probabilities change according to the operator's success in the last generation. This success is the number of individuals generated by the operator. In the case of the success of an operator being lower than two, it is set to two in order to avoid its removal.

4.4. Reinitialisation procedure

It is possible for the population to stagnate, i.e., it is not possible to improve any more because a local optimum has been reached. A reinitialisation procedure is applied in E2PAMEA in order to avoid falling into a local optimum. The procedure is executed when the population is not able to cover new examples for 25% of the total evaluations. Therefore, if the population is not able to improve its coverage for a very long period of time, it is assumed that the population is stagnated.

The reinitialisation procedure works as follows: first, individuals in the Pareto front are joined together with the current elite population, creating J. After that, a token-competition-based procedure [59] is triggered wherein only those non-repeated, high-coverage, interesting patterns are kept. Then, the resulting population will overwrite the current elite one if and only if its average WRAcc is higher than the current elite. Finally, the result of this procedure is introduced into the population of the next generation P_{t+1}. Remaining individuals in P_{t+1} are generated by means of a coverage-based initialisation [47] in order to cover other areas of the search space.

4.5. Operational schema

In this section the proposed operational schema for E2PAMEA is presented. As mentioned previously, the proposed method has been developed for the Apache Spark framework as it is particularly suitable for iterative, distributed computations.

Algorithm 3 The E2PAMEA algorithm with distributed chromosome evaluation.

Input:
A dataset D
m: maximum evaluations to perform
Output:
A set of patterns P

1: $Data' \leftarrow$ BIT-LUT_PREEVALUATION(D) ▷ See Alg. 1 and Alg. 2.
2: P $\leftarrow$ ORIENTEDINITIALISATION
3: BIT-LUT_EVALUATE(P, $Data'$) ▷ See Sect. 4.2 and Fig. 3.
4: **while** $evaluations \leq m$ **do**
5: $P' \leftarrow$ GENETICOPERATORS(P) ▷ See Sect. 4.3
6: BIT-LUT_EVALUATE(P', $Data'$)
7: $Q \leftarrow$ FASTNONDOMINATEDSORTING($P \cup P'$) ▷ See [80]
8: **if** Reset Criterion **then**
9: $P \leftarrow$ REINITIALISATIONPROCEDURE($Elite$, Q) ▷ See Sect. 4.4.
10: **else**
11: $P \leftarrow$ FILLBYCROWDINGDISTANCE(Q) ▷ See [80]
12: **end if**
13: **end while**
14: $P \leftarrow$ TOKENCOMPETITION(P) ▷ See [59]
15: **return** Population with best average WRAcc between P and $Elite$

After the Bit-LUT pre-evaluation stage (Algorithm 3, line 1), the evolutionary algorithm is triggered for the extraction of EPs. Algorithm 3 presents the pseudo-code of the main method. First, the initial population is generated by means of the oriented initialisation operator and then it is evaluated by means of a MapReduce procedure using Bit-LUT data (Algorithm 3, lines 2-3). After that, the evolutionary process is executed until a maximum number of evaluations is reached (Algorithm 3, lines 4-13). Within this process an offspring population P' is first generated by applying the genetic operators following the proposed adaptive schema in Section 4.3 (Algorithm 3, line 5). Next, P' is evaluated using Bit-LUT (Algorithm 3, line 6). Once P' is evaluated, both populations are joined together and then shorted by dominance fronts by means of the fast non-dominated sorting algorithm [80] (Algorithm 3, line 7). Finally, the reinitialisation procedure presented in Section 4.4 is checked and applied, updating the elite population, if necessary. Otherwise, the population is updated by introducing the first n fronts directly in P or the first n individuals of a front, sorted by crowding distance, until P is full (Algorithm 3, line 11). At the end of the evolutionary process, a

token-competition-based procedure is triggered on the current population in order to remove redundancies as much as possible (Algorithm 3, line 14). Finally, if the average WRAcc of the current population P is higher than the elite one, P is returned. Otherwise, the elite population is returned (Algorithm 3, line 15). In this procedure, the key element for performance is the chromosome evaluation by means of MapReduce using the Bit-LUT data.

4.6. Computational complexity

The computational complexity of an algorithm is governed by its most complex component. Therefore, in this section the computational complexity of the different components of E2PAMEA are analysed:

- In the worst case, the computational complexity of the oriented initialisation (line 2), crossover and mutation operators (line 5) is $O\left(IS\right)$, where I is the number of individuals and S is the number of selectors, as all genes of the population are explored and modified.
- The evaluation function within the evolutionary process (line 6) has a complexity of $O(IS)$. At the pre-evaluation stage, this complexity becomes $O(SN)$, where N corresponds the number of instances (line 3).
- The fast non-dominated sorting procedure (line 7) has a complexity of $O(MI^2)$ [80], where M is the number of objectives.
- The checking of the reset criterion (line 8) is constant for each individual. Therefore, its complexity is of $O(I)$.
- The complexity of the reinitialisation procedure, at line 11, is of $O(MI\ log\ I)$. This is because the crowding distance sorting [80] if the reset criterion is not fulfilled. Otherwise, the token competition (line 9) has a complexity of $O(I\ log\ I)$. This is due to its sorting stage.

According to these elements, the entire evolutionary process has a complexity of $O(MI^2)$ thanks to the reduction brought by the Bit-LUT procedure. As the algorithm is designed for big data environments, it assumed that $N >> I$ and $S > I$. Therefore, it can be noticed that the pre-evaluation stage and even the Bit-LUT evaluation within the evolutionary process are very heavy processes that need to be distributed in order to improve efficiency.

5. Experimental study

In this section, the experimental study for the determination of the quality of the proposal is carried out. First, the experimental framework is presented. Next, a comparison of the quality of the knowledge extracted from different algorithms is shown. Finally, a scalability analysis is performed.

5.1. Experimental framework

The experimental framework used for the evaluation of E2PAMEA is presented in this section. Its characteristics are described below:

- Datasets. Six datasets from the UCI repository [81] were employed for comparing the quality of the proposed method. In addition, artificial datasets were created using the MOA framework [82] for the scalability analysis. For further details on these datasets, please refer to our repository at the website (https://github.com/SIMIDAT/e2pamea-bd). Table 4 presents the characteristics of data, where the number of instances in millions (# Instances), the number of variables (# Variables) separated into real, integer and nominal (R/I/N), the size of selectors (# Selectors), and the number of classes are shown.

Table 4: Properties of the datasets used in the experiments.

Name	# Examples (in millions)	# Variables (R/I/N)	# Selectors	# Classes
census	0.299	41 (1/12/28)	427	2
kddcup	0.494	41 (26/0/15)	544	23
rlcp	5.749	11 (11/0/0)	33	2
susy	5.000	18 (18/0/0)	54	2
higgs	11.000	28 (28/0/0)	84	2
hepmass	10.500	29 (29/0/0)	87	2

- Experiment evaluation. As EPM tries to describe the underlying phenomena in data, an evaluation becomes necessary of the patterns extracted using unseen data. Therefore, this experimental study follows a five-fold stratified cross-validation schema in order to avoid as much as possible bias when creating the training-test partitions.

- Algorithms and parameters. The E2PAMEA algorithm is compared in this paper against BD-EFEP as it is the most promising EA for EPM in the big data context up to now [59]. The computational complexity of BD-EFEP is governed by its evaluation procedure, which is $O(ISN)$, i.e., each individual must traverse the whole dataset once for each selector. It is important

to remark that it is assumed that $N >> S > I$ as it is focused on big data environments. A pseudo-code for BD-EFEP is shown in Alg. 4, where the computational complexity of each procedure is analysed. The parameters configuration is depicted in Table 5. The parameters employed for BD-EFEP were taken from [59]. The parameters chosen for E2PAMEA were selected in order to be as similar as possible to the BD-EFEP and provide a fair comparison. This way, it is not necessary to tune the parameters of the E2PAMEA. In addition, it is important to mention that due to the adaptive nature of the E2PAMEA, the crossover and the mutation probabilities are not available.

Algorithm 4 Pseudo-code of the BD-EFEP algorithm [59]. Computational complexity of each procedure is shown as a comment, where N is the number of instances, M the number of objectives, S the number of selectors and I the number of individuals.

Input:
A dataset D
m: maximum evaluations to perform
Output:
A set of patterns P

$P \leftarrow$ GENERATEINITIALPOPULATION ▷ $O(I)$
EVALUATE(P,D) ▷ $O(ISN)$
while $evaluations \leq m$ **do**
  $Q \leftarrow$ GENETICOPERATORS(P) ▷ $O(IS)$
  EVALUATE(Q,D) ▷ $O(ISN)$
  $R \leftarrow$ FASTNONDOMINATEDSORTING($P \cup Q$) ▷ $O(MI^2)$
  if RESETCRITERION(R) **then**
    $P \leftarrow$ RANDOMINITIALISATION(F_0, P) ▷ $O(IS)$
  else
    $P \leftarrow$ FILLBYCROWDINGDISTANCE(R) ▷ $O(MI\ log\ I)$
  end if
end while
$P \leftarrow$ TOKENCOMPETITION(P) ▷ $O(I\ log\ I)$
$P \leftarrow$ FILTERBYCONFIDENCE(P) ▷ $O(I)$

- Quality measures. The quality measures analysed in this study were presented in Table 2. These measures are key for the determination of the quality of the patterns extracted regarding the different aspects of EPM. In addition, the number of patterns (n_p) and the average number of variables (n_v) are analysed in order to determine the model complexity. It is important to remark that the value shown for GR represents the percentage of patterns whose GR in test is greater than one. This is because the domain of GR is $[0, \infty]$, so the average cannot be computed properly.

Table 5: Algorithms and their parameters used in this experimental study.

Algorithm	Parameters
BD-EFEP [59]	Population length (per class) = 51 Number of labels = 3 Number of evaluations = 10000 Objectives = Jaccard,FPR Crossover probability = 0.6 Mutation probability = 0.1
E2PAMEA	Population length (per class) = 51 Number of labels = 3 Number of evaluation = 10000 Objectives = Jaccard,FPR

- Run environment. The experimental study was carried out using a computation cluster composed of 16 nodes with two Intel Xeon E5-2670v2 each, 10 cores at 2.50 Ghz and 64 GB of RAM. The cluster is based on RedHat Enterprise Linux (relase 7.3). The experiments were performed using Apache Spark version 2.1.

5.2. Analysis of the results

This section presents the analysis of the results obtained with these algorithms. Both methods use a global MapReduce approach, so they extract the same results independently of the number of distributed training partitions employed across the nodes of the cluster. Therefore, the number of partitions is not shown in this study.

Table 6: Average results obtained by the big data algorithms for emerging pattern mining.

Dataset	Algorithm	n_p	n_v	WRAcc	CONF	GR	TPR	FPR
census	E2PAMEA	**18.0000**	**3.0710**	**0.6460**	**0.9670**	**1.0000**	**0.6660**	**0.3740**
	BD-EFEP	21.4000	4.0923	0.6211	0.9365	**1.0000**	0.6212	0.3790
kddcup	E2PAMEA	29.0000	**4.7800**	**0.7050**	0.5160	0.7720	**0.4750**	**0.0220**
	BD-EFEP	**13.0000**	5.2971	0.4510	**0.5480**	**0.7857**	0.1864	0.1532
rlcp	E2PAMEA	5.6000	**2.0070**	**0.9220**	**1.0000**	**1.0000**	**0.8590**	**0.0160**
	BD-EFEP	**5.2000**	6.6700	0.5161	0.7010	**1.0000**	0.1097	0.0776
susy	E2PAMEA	**9.4000**	3.5050	0.5540	0.7360	**1.0000**	0.2620	**0.1530**
	BD-EFEP	14.0000	**1.9727**	**0.6695**	**0.8284**	**1.0000**	**0.5106**	0.1717
higgs	E2PAMEA	**2.8000**	3.1330	0.5040	0.6790	**1.0000**	0.0450	0.0360
	BD-EFEP	9.6000	**2.6911**	**0.9203**	**0.9827**	0.9833	**0.8710**	**0.0303**
hepmass	E2PAMEA	**11.6000**	2.6050	**0.6450**	**0.7860**	**1.0000**	**0.5170**	0.2270
	BD-EFEP	19.0000	**1.9934**	0.5659	0.4098	0.6738	0.2488	**0.0884**
Median	E2PAMEA	**10.5000**	**3.1020**	**0.6455**	0.7610	**1.0000**	**0.4960**	**0.0945**
	BD-EFEP	13.5000	3.3916	0.5934	**0.7648**	0.9916	0.3769	0.1207

The average results extracted for the analysed algorithms on each dataset is presented in Table 6. In addition, the median results for each algorithm on all

datasets are presented in the last two rows in order to ease the analysis. An analysis of the different quality measures is presented below:

- n_p. The number of patterns extracted by both algorithms is, in general, low. Therefore, the models are simple. It is important to remark that, in general, E2PAMEA extracts a lower number of rules than BD-EFEP.

- n_v. The average number of variables extracted by both algorithms allows a fast, easy analysis of each individual rule. For E2PAMEA, the average number of variables is lower than BD-EFEP, which produces a simpler model.

- WRAcc. In both cases, the WRAcc extracted is high. However, it is important to remark that the median WRAcc extracted with E2PAMEA is higher than the one extracted with BD-EFEP.

- CONF. Both algorithms contain a very similar confidence, with a slightly better confidence for BD-EFEP. The use of the elite population and its update policy in E2PAMEA, together with the coverage-based reinitialisation allows the extraction of individuals with high reliability. Nevertheless, this is a great improvement as generality is improved (as can be seen on WRAcc), while reliability remains stable.

- GR. Both algorithms extract a high number of patterns that are EPs on test data. However, it is important to remark that in general, the knowledge extracted by E2PAMEA fits better to the real underlying phenomena, as it has a higher percentage of patterns that are EPs on test.

- TPR and FPR. In general, it can be observed that E2PAMEA is able to extract a set of patterns with higher TPR, together with lower FPR than BD-EFEP. This means that the knowledge extracted is more general and more reliable with the proposed method than other alternatives, which is a great improvement on the quality of the insights extracted.

The knowledge extracted in EPM searches for a good trade-off between three main objectives: simplicity, generality and reliability. It is important to remark that reliability is the most relevant objective as descriptions should always be accurate. According to the results extracted, E2PAMEA improves all the aspects analysed with respect to BD-EFEP, except for confidence where its quality is similar. This means that the pattern model of E2PAMEA is simpler, allowing an easier analysis by the experts. The knowledge is more general as it covers a higher

amount of positive examples, while the reliability is improved as there are less incorrectly covered examples while the confidence is kept. These properties allow the extraction of better conclusions about the underlying phenomena from these insights than the ones extracted with BD-EFEP. Therefore, the proposed algorithm extracts EPs with a better trade-off between simplicity, generality and reliability than other alternatives proposed up to now for EPM in big data environments.

5.3. Performance comparison

In this section, an execution time and memory consumption comparison between the different algorithms analysed is carried out. The results presented have been obtained using 512 distributed training partitions on each method.

Table 7: Performance comparison between E2PAMEA and BD-EFEP on the analysed datasets. The last row presents the statistical signficance value according to the Wilcoxon test.

Dataset	Algorithm	Exec. time (s)	Memory (MB)
census	E2PAMEA	**7.00**	**14.00**
	BD-EFEP	135.00	37.00
kddcup	E2PAMEA	**26.64**	35.10
	BD-EFEP	196.00	**4.70**
rlcp	E2PAMEA	**37.17**	**21.90**
	BD-EFEP	2535.00	356.08
susy	E2PAMEA	**43.81**	**36.20**
	BD-EFEP	2772.00	562.30
higgs	E2PAMEA	**105.55**	**111.40**
	BD-EFEP	6796.00	1909.00
hepmass	E2PAMEA	**95.93**	**107.40**
	BD-EFEP	5701.00	1759.00
Median	E2PAMEA	**40.49**	**35.65**
	BD-EFEP	2653.50	459.55
	p-value	**0.0138**	**0.0374**

Table 7 presents the time comparison and memory consumption of the algorithms analysed. It is important the remark that the last row is the significance values according to the Wilcoxon statistical test [83], with $\alpha = 0.05$. In addition, Figure 4 graphically presents this comparison using a logarithmic scale. The results presented are the average execution time in seconds of the different datasets analysed. The results show that both algorithms present really good execution

times on those datasets with the lowest number of instances, i.e., *census* and *kddcup*. However, when the amount of data grows to approximately five million instances on *rlcp* and *susy*, the execution time of BD-EFEP drastically increases up to more than 2000 seconds, while the execution time of E2PAMEA is less than 50 seconds. Finally, for the largest datasets with more than 10 million instances, the execution time of E2PAMEA is approximately 100 seconds, while on BD-EFEP this time is more than 5000 seconds, which could be unacceptable in some scenarios. Therefore, according to the results of the Wilcoxon test, E2PAMEA significantly surpasses BD-EFEP with regard to execution time.

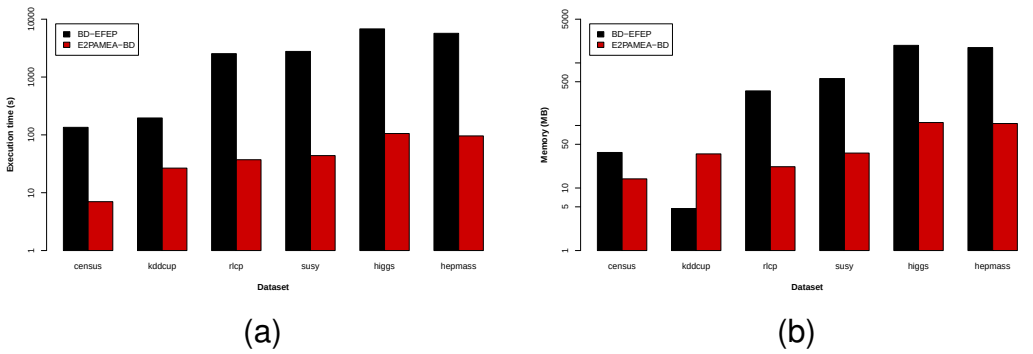


Figure 4: Performance comparison between E2PAMEA and BD-EFEP on the analysed datasets. (a) Execution time comparison, (b) Memory consumption comparison.

On the other hand, memory consumption on the proposed method is drastically reduced, especially on the largest datasets. However, it is important to mention the results obtained using the *kddcup* dataset, where the amount of memory employed by E2PAMEA is significantly higher than BD-EFEP. This is due to the high ratio between the number of selectors and the number of instances. Nevertheless, this is not a problem on bigger datasets as this ratio is usually much lower, i.e., millions of instances against thousands of selectors. In addition there is a high number of classes, which significantly increases the population size and memory consumption. In general, the average memory gain using E2PAMEA with respect to BD-EFEP is about 92%. Therefore, the proposed evaluation approach significantly outperforms the classical one in terms of memory consumption, which is also supported by the results extracted from the Wilcoxon test.

5.4. Scalability analysis

In big data analysis the methodologies should be prepared for handling the future amounts of data in a reasonable time as well. For this purpose, a scalability analysis for both the number of instances and the number of variables has been carried out. For the scalability analysis with respect to the number of instances, a set of artificial datasets generated by the MOA framework [82] ranging from $1 \cdot 10^7$ to $1 \cdot 10^8$ instances with a fixed amount of 20 variables each one was analysed. For the scalability analysis with respect to the number of variables, it is important to remark that the complexity is related to the amount of selectors of the problem. For this purpose a set of artificial datasets ranging from 100 to 15000 variables was analysed, where half of them were categorical variables with five categories, and the other half numeric variables. For numeric variables, five LLs were employed. Therefore, the number of selectors ranges from 500 $(50 \cdot 5 + 50 \cdot 5)$ to 75000 $(7500 \cdot 5 + 7500 \cdot 5)$. The number of instances in these datasets was fixed at $1 \cdot 10^6$.

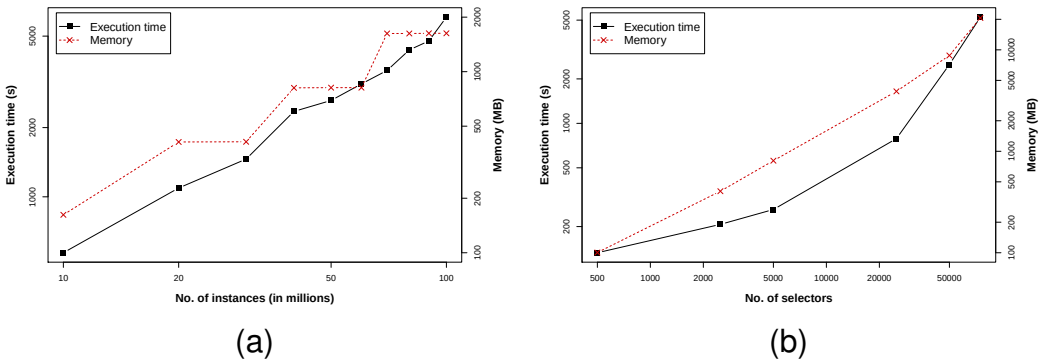


Figure 5: Scalability analysis of E2PAMEA by (a) number of instances (b) number of selectors. (b) is shown using a logarithmic scale.

In particular, Fig 5a presents the scalability analysis with respect to the number of instances, where in logarithmic scale are presented the number of instances on the x axis, the execution time on the y axis on the left-hand side, and the memory consumption on the right-hand side. As can be observed, the execution time scales up linearly as the number of instances increases. This is a desirable behaviour in big data as more nodes can be added in order to maintain an acceptable execution time. Therefore, the execution time of the proposed algorithm properly scales up with respect to the number of instances. In addition, the amount of

memory required is very low. In fact, the amount of memory required suggests that the method can fit on a single machine even when processing 100 million instances, although parallel processing should be employed to improve execution time. Moreover, the memory scales up linearly following a stepped shape. This shape is produced due to the memory management of the bit set data structure employed in Bit-LUT that doubles its size when it needs to allocate more memory. Thus, the memory scalability with respect to the number of instances is appropriate for big data analysis.

Finally, Fig. 5b presents the scalability analysis with respect to the number of selectors of the problem. On both axes the logarithmic scaling was employed, where the number of selectors is presented on the x axis and the execution time on the y axis. It can be observed that the execution time scales up linearly when the number of variables increases. Thus, the proposed algorithm is able to properly scale up with respect to the number of variables. However, it is remarkable that the slope of the scalability is higher with respect to the number of instances. This could be produced by several elements such as higher complexity in individuals, which produce a higher data transfer for each evaluation process, together with a significant increase of reduce operations which slow down the process due to data transfers. Finally, it is important to remark that the amount of memory required is low and it linearly scales up with respect to the number of selectors. Therefore, the memory scalability with respect to the number of selectors of the problem is appropriate.

According to these results, the proposed Bit-LUT evaluation process is a promising alternative for speeding up the chromosome evaluation process with respect to BD-EFEP. Bit-LUT is also able to handle big data as it properly scales up with respect to the number of instances. Moreover, it is also able to linearly scale up with respect to the number of variables/selectors of the problem. Therefore, Bit-LUT is a promising alternative for a fast, scalable evaluation of individuals for evolutionary algorithms, specifically in EPM.

6. Discussion

The EPM method aims at finding those discriminative patterns which are able to describe the highest number of instances of the dataset with the best balance between its simplicity and reliability. The purpose is to provide knowledge about the underlying phenomena in data in order to help decision-making processes. This fact is of relevance in big data analysis as a simple, understandable summary of these vast amounts of data are necessary.

According to the results extracted, the proposed algorithm is able to find a good trade-off between the different objectives of the problem thanks to the use of a multi-objective approach, which is able to find the patterns at the Pareto front. However, one of the main problems of dominance-based multi-objective evolutionary algorithms is their performance degradation when the number of objectives increases [84].

The aim of the adaptive strategy developed for E2PAMEA is twofold: firstly, it removes the need to tune the parameters of the crossover and mutation probabilities; secondly, with the correct combination of operators, the algorithm is able to avoid local optima as there is a trade-off between the exploitation and exploration capacities. In general, this adaptive strategy seems interesting as it promotes a fair application of the genetic operators according to its performance, improving exploitation, while it allows some room for randomness in order to provide a controlled exploration capacity, avoiding premature convergence. Specifically for E2PAMEA, the crossover operators promotes the specificity of the patterns. This means that they look for longer, reliable individuals, from two different perspectives: two-point crossover tries to explode nearby areas of the search space, while HUX crossover explores the areas with the largest Humming distance with respect to their parents. On the other hand, the application of the mutation operators promotes the generality and diversity of the patterns. Therefore, they look for shorter, general ones. In particular, random gene change allows more diversity, while random variable removal allows more generality. This is interesting when the search for reliable patterns gets stagnated. Here, the mutation operators can remove those less relevant variables, improving generality without loosing reliability. In fact, the algorithm obtains similar levels of confidence with respect to BD-EFEP. However, there is a reduction of the error magnitude, as can be observed in FPR, while the generality of the patterns is improved, according to the TPR and GR values. Therefore, there is a great synergy between these operators, as they are able to find reliable patterns, with higher generalisation capacity than the BD-EFEP.

Nevertheless, this adaptive scheme usually converges to one operator, which keeps the applicability of the remaining ones at the minimum level. Therefore, the algorithm can get stagnated in an optimum. In this way, the application of the reinitialisation procedure is necessary in order to collect the best individuals found so far and, then, move to another area of the search space. In the EPM, patterns extracted should avoid overlapping as they provide unnecessary redundancy. For the E2PAMEA, this is done by means of the token-competition-based procedure in the reinitialisation, which keeps those patterns with higher WRAcc, while the

complexity of the patterns set extracted is reduced. Although this is a fast process, it cannot assert the complete removal of redundant patterns due to its niche-based procedure. Therefore, in order to soften this problem, the elite population is only replaced if the average WRAcc is better, as more interesting patterns are extracted. According to the results obtained in terms of WRAcc by E2PAMEA, this strategy seems to be good for finding interesting patterns. Additionally, the reduction of the complexity of the set of patterns shows the suitability of the application of the procedure for this problem.

Distributed approaches are necessary because, by definition, big data problems cannot be handled by a single computer in a reasonable time due to its complexity, volume, and so on. Due to this fact, many researchers deploy their clusters within cloud services [14]. The cost of these services is usually calculated by CPU and memory consumption. Thus, efficiency is key. In the development of evolutionary algorithms for EPM focused on big data, the main bottleneck is its evaluation process. As can be seen from other alternatives such as BD-EFEP, each individual in the population must traverse this huge dataset each time it needs to be evaluated. Thus this a very slow operation. In this paper, the Bit-LUT evaluation process has been developed for a distributed, efficient evaluation, which avoids such dataset traversal within the evolutionary process. This is mainly due to the pre-calculation stage, and the employment of bit sets and bitwise operations. The procedure has several advantages:

- First, the computational complexity of the evaluation within the evolutionary process is reduced in an order of magnitude as checking if an instance is covered by a pattern has a constant time. This is supported by the results extracted as E2PAMEA is approximately fifty times faster than BD-EFEP.
- Secondly, the reduction in memory consumption is also reduced in an order of magnitude as only one bit per instance and selector is stored. This is shown in the experimental study, where E2PAMEA reduces uses approximately fifteen times less memory than BD-EFEP in average.
- Finally, the proposed method linearly scales up in both dimensions of data due to the employment of the Bit-LUT evaluation. This supports the fact that the proposed method is able to analysed bigger datasets than previous approaches presented up to date.

This performance improvement is achieved because the definitions of the different fuzzy sets defined must remain fixed throughout the whole process. Within

the descriptive point of view, the pre-setting of the fuzzy sets employed is a common approach within the EPM algorithms developed so far, so the optimisation of the provided fuzzy sets is usually not allowed. However, it could be interesting to address this point of view in future works in order to improve the descriptive capacities of the methods.

7. Conclusions

In this paper, an EFS based on a MOEA called E2PAMEA has been presented for an efficient extraction of high-quality EPs in big data environments. It is based on an adaptive version of the NSGA-II algorithm in order to provide a better diversification-exploitation trade-off, while the tuning of the crossover and mutation probabilities is removed. The genetic operators employed are the two-point and HUX crossover operators for exploitation or diversity when necessary, together with two mutation operators: the removal of a variable for generalisation of the patterns extracted, or a random change in a gene. Similarly, E2PAMEA uses an elite population where a cooperative-competitive schema together with a token competition-based procedure for the promotion of reliable results.

The efficiency of the proposed method is mainly due to the evaluation procedure presented in this paper, called Bit-LUT. In particular, it is a MapReduce approach based on an exact method. It uses a first pre-evaluation stage by means of a MapReduce job before the beginning of the evolutionary process in order to provide a look-up table for a faster evaluation on the evolutionary process. This table stores a bit set for each selector of the problem which determines which instance of the dataset is covered by the given selector. After that, the proposed look-up table is employed under the chromosome evaluation of the proposed evolutionary process by means of another MapReduce job. In this phase, the map procedure calculates the coverage of those elements that belong to the same variable by means of bitwise OR operations, while in the reduce phase these results are aggregated by means of bitwise AND operations in order to obtain the final coverage of the individual. After that, the objective measures can be calculated using this coverage.

The suitability of E2PAMEA has been proven in this paper with respect to other EPM evolutionary algorithm for big data. As a conclusion, the proposed algorithm outperforms previous approaches in all aspects analysed, so the quality of the results extracted with E2PAMEA is more reliable, more interesting and more general than previous approaches. In addition, the scalability of the Bit-LUT evaluation approach has been analysed. The results show that Bit-LUT is

at least an order of magnitude faster than previous approaches, while the amount of physical memory required is reduced by almost twenty times with respect to previous approaches. In addition, processing time and physical memory are able to properly scale up with respect to the number of instances, together with respect to the number of variables/selectors, which is one of the main drawbacks with EPM algorithms.

Therefore, the E2PAMEA algorithm provides a good trade-off between exploitation and exploration which allows the extraction of high-quality EPs in big data environments, while the Bit-LUT evaluation is a promising alternative for a fast, scalable execution of EAs for big data algorithms, especially for the EPM task.

Acknowledgement

This study was funded by the Spanish Ministry of Economy and Competitiveness under the project TIN2015-68454-R and FPI 2016 Scholarship reference BES-2016-077738 (FEDER Founds). This paper was written at the School of Computer Science and Informatics, De Monfort University, Leicester during the research visit of Á.M. García-Vico. Thanks to Prof. David Elizondo for the invitation.

Conflict of interest

The authors declare no conflict of interest.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] T.-P. Liang, Y.-H. Liu, Research landscape of business intelligence and big data analytics: A bibliometrics study, Expert Systems with Applications 111 (2018) 2–10.

[2] M. A. Beyer, D. Laney, The importance of big data: a definition, Gartner Research Report (2012) 1–9.

[3] A. Gandomi, M. Haider, Beyond the hype: Big data concepts, methods, and analytics, International Journal of Information Management 35 (2) (2015) 137–144.

[4] S. Nativi, P. Mazzetti, M. Santoro, F. Papeschi, M. Craglia, O. Ochiai, Big data challenges in building the global earth observation system of systems, Environmental Modelling & Software 68 (2015) 1–26.

[5] F. W. Glover, G. A. Kochenberger, Handbook of metaheuristics, Vol. 57, Springer Science & Business Media, 2006.

[6] J. L. Olmo, J. R. Romero, S. Ventura, Swarm-based metaheuristics in automatic programming: a survey, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 4 (6) (2014) 445–469.

[7] A. J. García, W. Gómez-Flores, Automatic clustering using nature-inspired metaheuristics: A survey, Applied Soft Computing 41 (2016) 192 – 213.

[8] P. Kralj-Novak, N. Lavrac, G. I. Webb, Supervised Descriptive Rule Discovery: A Unifying Survey of Constrast Set, Emerging Pattern and Subgroup Mining, Journal of Machine Learning Research 10 (2009) 377–403.

[9] D. Martens, B. Baesens, T. Van Gestel, J. Vanthienen, Comprehensible credit scoring models using rule extraction from support vector machines, European journal of operational research 183 (3) (2007) 1466–1476.

[10] D. E. Goldberg, Genetic Algorithms in search, optimization and machine learning, Addison-Wesley Longman Publishing Co., Inc., 1989.

[11] C. J. Carmona, P. González, M. J. del Jesus, F. Herrera, Overview on evolutionary subgroup discovery: analysis of the suitability and potential of the search performed by evolutionary algorithms, WIREs Data Mining and Knowledge Discovery 4 (2) (2014) 87–103.

[12] M. Atzmueller, Subgroup discovery, WIREs Data Mining and Knowledge Discovery 5 (2015) 35–49.

[13] A. M. García-Vico, C. J. Carmona, D. Martín, M. García-Borroto, M. J. del Jesus, An overview of emerging pattern mining in supervised descriptive rule discovery: Taxonomy, empirical study, trends and prospects, WIREs: Data Mining and Knowledge Discovery 8 (1) (2018).

[14] A. Fernández, S. Río, V. López, A. Bawakid, M. del Jesus, J. Benítez, F. Herrera, Big Data with Cloud Computing: An Insight on the Computing Environment, MapReduce and Programming Frameworks, WIREs Data Mining and Knowledge Discovery 5 (4) (2014) 380–409.

[15] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, in: Operating Systems Design and Implementation (OSDI), 2004, pp. 137–150.

[16] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster computing with working sets, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, 2010, pp. 10–10.

[17] Z. Han, J. Wu, C. Huang, Q. Huang, M. Zhao, A review on sentiment discovery and analysis of educational big-data, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (2019) e1328.

[18] K. Soomro, M. N. M. Bhutta, Z. Khan, M. A. Tahir, Smart city big data analytics: An advanced review, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9 (5) (2019) e1319.

[19] Y. Xia, J. Chen, X. Lu, C. Wang, C. Xu, Big traffic data processing framework for intelligent monitoring and recording systems, Neurocomputing 181 (2016) 139 – 146.

[20] M. Pramanik, R. Y. Lau, W. T. Yue, Y. Ye, C. Li, Big data analytics for security and criminal investigations, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 7 (4) (2017) e1208.

[21] N. Bharill, A. Tiwari, A. Malviya, O. P. Patel, A. Gupta, D. Puthal, A. Saxena, M. Prasad, Fuzzy knowledge based performance analysis on big data, Neurocomputing (2019).

[22] M. Makkie, H. Huang, Y. Zhao, A. V. Vasilakos, T. Liu, Fast and scalable distributed deep convolutional autoencoder for fmri big data analytics, Neurocomputing 325 (2019) 20–30.

[23] W. Ding, C.-T. Lin, S. Chen, X. Zhang, B. Hu, Multiagent-consensus-mapreduce-based attribute reduction using co-evolutionary quantum pso for big data applications, Neurocomputing 272 (2018) 136–153.

[24] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[25] J. Dean, S. Ghemawat, MapReduce: A flexible data processing tool, Communications of the ACM 53 (1) (2010) 72–77.

[26] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, F. Herrera, Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce, Information Fusion 42 (2018) 51 – 61.

[27] J. Lin, Mapreduce is good enough? if all you have is a hammer, throw away everything that's not a nail!, Big Data 1 (1) (2013) 28–37.

[28] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, I. Stoica, Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, 2012, pp. 2–2.

[29] H. Kundra, H. Sadawarti, Hybrid algorithm of cuckoo search and particle swarm optimization for natural terrain feature extraction, Research Journal of Information Technology 7 (1) (2015) 58–69.

[30] J. Guo, Z. Sun, H. Tang, X. Jia, S. Wang, X. Yan, G. Ye, G. Wu, Hybrid optimization algorithm of particle swarm optimization and cuckoo search for preventive maintenance period optimization, Discrete Dynamics in Nature and Society (2016) 1516271.

[31] W.-C. Hong, M.-W. Li, J. Geng, Y. Zhang, Novel chaotic bat algorithm for forecasting complex motion of floating platforms, Applied Mathematical Modelling 72 (2019) 425–443.

[32] T. Pant, C. Han, H. Wang, Examination of errors of table integration in flamelet/progress variable modeling of a turbulent non-premixed jet flame, Applied Mathematical Modelling 72 (2019) 369–384.

[33] Z. Zhang, W.-C. Hong, Electric load forecasting by complete ensemble empirical mode decomposition adaptive noise and support vector regression with quantum-based dragonfly algorithm, Nonlinear Dynamics 98 (2) (2019) 1107–1136.

[34] L. Calvet, J. de Armas, D. Masip, A. A. Juan, Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs, Open Mathematics 15 (1) (2017) 261–280.

[35] B. De La Iglesia, Evolutionary computation for feature selection in classification problems, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 3 (6) (2013) 381–407.

[36] W. Liu, J. Wang, A brief survey on nature-inspired metaheuristics for feature selection in classification in this decade, in: 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), 2019, pp. 424–429.

[37] S. B. Sakri, N. B. A. Rashid, Z. M. Zain, Particle swarm optimization feature selection for breast cancer recurrence prediction, IEEE Access 6 (2018) 29637–29647.

[38] A. Nagpal, D. Gaur, Hybrid feature selection approach based on grasp for cancer microarray data, Journal of computing and information technology 25 (2) (2017) 133–148.

[39] L. Shi, Y. Wan, X. Gao, M. Wang, Feature selection for object-based classification of high-resolution remote sensing images based on the combination of a genetic algorithm and tabu search, Computational intelligence and neuroscience 2018 (2018).

[40] F. Han, J. Jiang, Q.-H. Ling, B.-Y. Su, A survey on metaheuristic optimization for random single-hidden layer feedforward neural network, Neurocomputing 335 (2019) 261–273.

[41] C. Qi, X. Tang, Slope stability prediction using integrated metaheuristic and machine learning approaches: a comparative study, Computers & Industrial Engineering 118 (2018) 112–122.

[42] R. Thomschke, S. Voß, S. Lessmann, Metaheuristics and classifier ensembles, in: Business and Consumer Analytics: New Ideas, 2019, pp. 733–779.

[43] T. Inkaya, S. Kayalıgil, N. E. Özdemirel, Swarm intelligence-based clustering algorithms: A survey, in: Unsupervised learning algorithms, 2016, pp. 303–341.

[44] R. A. Mohammed, M. G. Duaimi, Association rules mining using cuckoo search algorithm, International Journal of Data Mining, Modelling and Management 10 (1) (2018) 73–88.

[45] Y. Djenouri, D. Djenouri, A. Belhadi, P. Fournier-Viger, J. C.-W. Lin, A. Bendjoudi, Exploiting gpu parallelism in improving bees swarm optimization for mining big transactional databases, Information Sciences 496 (2019) 326–342.

[46] M. Nandhini, M. Rajalakshmi, S. Sivanandam, Experimental and statistical analysis on the performance of firefly based predictive association rule classifier for health care data diagnosis, Journal of Control Engineering and Applied Informatics 19 (2) (2017) 101–110.

[47] C. J. Carmona, P. González, M. J. del Jesus, F. Herrera, NMEEF-SD: Non-dominated Multi-objective Evolutionary algorithm for Extracting Fuzzy rules in Subgroup Discovery, IEEE Transactions on Fuzzy Systems 18 (5) (2010) 958–970.

[48] J. M. Luna, J. R. Romero, C. Romero, S. Ventura, On the Use of Genetic Programming for Mining Comprehensible Rules in Subgroup Discovery, IEEE Transactions on Cybernetics 44 (12) (2014) 2329–2341.

[49] V. Pachón, J. Mata, J. Domínguez, Searching for the most significant rules: an evolutionary approach for subgroup discovery, Soft Computing 21 (10) (2017) 2609–2618.

[50] A. M. García-Vico, C. J. Carmona, P. González, M. J. del Jesus, MOEA-EFEP: Multi-objective evolutionary algorithm for extracting fuzzy emerging patterns, IEEE Transactions on Fuzzy Systems 26 (5) (2018) 2861 – 2872.

[51] J. A. Sanz, D. Bernardo, F. Herrera, H. Bustince, H. Hagras, A compact evolutionary interval-valued fuzzy rule-based classification system for the modeling and prediction of real-world financial applications with imbalanced data, IEEE Transactions on Fuzzy Systems 23 (4) (2015) 973–990.

[52] D. Peralta, S. Río, S. Ramíez-Gallego, I. Triguero, J. M. Beníez, F. Herrera, Evolutionary Feature Selection for Big Data Classification: A MapReduce Approach, Mathematical Problems in Engineering 2015 (2015) 1–11.

[53] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, MRPR: A MapReduce Solution for Prototype Reduction in Big Data Classification, Neurocomputing 150 (2015) 331–345.

[54] F. Pulgar-Rubio, A. J. Rivera-Rivas, M. D. Pérez-Godoy, P. González, C. J. Carmona, M. J. del Jesus, MEFASD-BD: Multi-Objective Evolutionary Fuzzy Algorithm for Subgroup Discovery in Big Data Environments - A MapReduce Solution, Knowledge-Based Systems 117 (2017) 70–78.

[55] A. Fernández, C. J. Carmona, M. J. del Jesus, F. Herrera, A View on Fuzzy Systems for Big Data: Progress and Opportunities, International Journal of Computational Intelligence Systems 9 (1) (2016) 69–80.

[56] F. Padillo, J. M. Luna, S. Ventura, An evolutionary algorithm for mining rare association rules: A big data approach, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 2007–2014.

[57] M. Barsacchi, A. Bechini, P. Ducange, F. Marcelloni, Optimizing partition granularity, membership function parameters, and rule bases of fuzzy classifiers for big data by a multi-objective evolutionary approach, Cognitive Computation 11 (3) (In press) 367–387.

[58] F. Padillo, J. M. Luna, S. Ventura, A grammar-guided genetic programing algorithm for associative classification in big data, Cognitive Computation 11 (3) (2019) 331–346.

[59] A. M. García-Vico, C. J. Carmona, P. González, M. J. del Jesus, A big data approach for extracting fuzzy emerging patterns, Cognitive Computation 11 (3) (2019) 400 – 417.

[60] G. Dong, J. Li, Efficient mining of emerging patterns: Discovering trends and differences, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 1999, pp. 43–52.

[61] L. A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning. Parts I, II, III, Information Science 8-9 (1975) 199–249,301–357,43–80.

[62] R. S. Michalski, R. Stepp, Revealing conceptual structure in data by inductive inference, Machine Intelligence 10 (1982) 173–196.

[63] M. García-Borroto, O. Loyola-González, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, Evaluation of quality measures for contrast patterns by using unseen objects, Expert Systems with Applications 83 (2017) 104 – 113.

[64] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery: an overview, in: Advances in knowledge discovery and data mining, AAAI/MIT Press, Menlo Park, CA, USA, 1996, pp. 1–34.

[65] C. J. Carmona, M. J. del Jesus, F. Herrera, A Unifying Analysis for the Supervised Descriptive Rule Discovery via the Weighted Relative Accuracy, Knowledge-Based Systems 139 (2018) 89–100.

[66] W. Kloesgen, Explora: A Multipattern and Multistrategy Discovery Assistant, in: Advances in Knowledge Discovery and Data Mining, American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996, pp. 249–271.

[67] D. Gamberger, N. Lavrac, Expert-Guided Subgroup Discovery: Methodology and Application, Journal Artificial Intelligence Research 17 (2002) 501–527.

[68] J. Y. Li, G. Z. Dong, K. Ramamohanarao, L. Wong, DeEPs: A New Instance-Based Lazy Discovery and Classification System, Machine Learning 54 (2) (2004) 99–124.

[69] J. Bailey, T. Manoukian, K. Ramamohanarao, A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns, in: Proc. of the 3th International Conference on Data Mining, IEEE, 2003, pp. 485–488.

[70] H. Fan, K. Ramamohanarao, Fast discovery and the generalization of strong jumping emerging patterns for building compact and accurate classifiers, IEEE Transactions on Knowledge and Data Engineering 18 (6) (2006) 721–737.

[71] Q. Liu, P. Shi, Z. Hu, Y. Zhang, A novel approach of mining strong jumping emerging patterns based on BSC-tree, International Journal of Systems Science 45 (3) (2014) 598–615.

[72] F. Herrera, Genetic fuzzy systems: taxomony, current research trends and prospects, Evolutionary Intelligence 1 (2008) 27–46.

[73] A. M. García-Vico, F. Charte, P. González, C. J. Carmona, M. J. del Jesus, Subgroup discovery with evolutionary fuzzy systems in r: The sdefsr package, The R Journal 8 (2) (2016) 307–323.

[74] J. J. Durillo, A. J. Nebro, F. Luna, E. Alba, On the effect of the steady-state selection scheme in multi-objective genetic algorithms, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2009, pp. 183–197.

[75] A. Fernández, A. Altalhi, S. Alshomrani, F. Herrera, Why linguistic fuzzy rule based classification systems perform well in big data applications?, International Journal of Computational Intelligence Systems 10 (1) (2017) 1211–1225.

[76] E. Hüllermeier, Fuzzy sets in machine learning and data mining, Applied Soft Computing 11 (2) (2011) 1493–1505.

[77] B. L. Miller, D. E. Goldberg, Genetic Algorithms, Tournament Selection, and the Effects of Noise, Complex System 9 (1995) 193–212.

[78] J. H. Holland, Adaptation in Natural and Artificial Systems, 2nd Edition, University of Michigan Press, 1975.

[79] L. J. Eshelman, Foundations of Genetic Algorithms, 1991, Ch. The CHC Adaptive Search Algorithm: How to have Safe Search When Engaging in Nontraditional Genetic Recombination, pp. 265–283.

[80] K. Deb, A. Pratap, S. Agrawal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions Evolutionary Computation 6 (2) (2002) 182–197.

[81] D. Dheeru, E. Karra Taniskidou, Uci machine learning repository (2017).
URL `http://archive.ics.uci.edu/ml`

[82] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: massive online analysis, Journal of Machine Learning Research 11 (2010) 1601–1604.
URL `https://moa.cms.waikato.ac.nz/`

[83] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–83.

[84] H. Ishibuchi, N. Tsukamoto, Y. Hitotsuyanagi, Y. Nojima, Effectiveness of scalability improvement attempts on the performance of nsga-ii for many-objective problems, in: Proc. of the 10th Annual Conference on Genetic and Evolutionary Computation, 2008, pp. 649–656.