A First Approach to Handle Fuzzy Emerging Patterns Mining on Big Data Problems: The EvAEFP-Spark Algorithm

A.M. García-Vico, P. González, M.J. del Jesus
Department of Computer Science
University of Jaén
23071, Jaén, Spain
Email: {agvico | pglez | mjjesus}@ujaen.es

C.J. Carmona
Lenguages and Computer Technology Systems.
Department of Civil Engineering
University of Burgos
09006, Burgos (Spain)
Email: cjcarmona@ubu.es

***Abstract*—Internet and the new technologies are generating new scenarios with and a significant increase of data volumes. The treatment of this huge quantity of information is impossible with traditional methodologies and we need to design new approaches towards distributed paradigms such as MapReduce. This situation is widely known in the literature as Big Data.**

This contribution presents a first approach to handle fuzzy emerging patterns in big data environments. This new algorithm is called EvAFP-Spark and is development in Apache Spark based on MapReduce. The use of this paradigm allows us the analysis of huge datasets efficiently. The main idea of EvAEFP-Spark is to modify the methodology of evaluation of the populations in the evolutionary process. In this way, a population is evaluated in the different maps, obtained in the Map phase of the paradigm, and for each one a confusion matrix is obtained. Then, the Reduce function accumulates the confusion matrix for each map in a general matrix in order to evaluate the fitness of the individuals. An experimental study with high dimensional datasets is performed in order to show the advantages of this algorithm in emerging patterns mining.

I. Introduction

Nowadays we live in the information era. An era where several *petabytes* of information are generated each day due to cheaper storage, mobile network access, social networks and other kinds of sensors networks [1]. This produces vast volumes of data, with variety of formats that arrives at high velocity. These characteristics make traditional data mining methods unable to process this "Big Data" [2]. The term "Big Data" is currently a hot topic in industry and academia because of its possibilities [3]. The majority of methods are based on the MapReduce paradigm [4]. MapReduce not only allows to deal with Big Data, it is a paradigm to allow the creation of scalable methods, which is really important in the data sciences to get more accurate knowledge [5].

Emerging Patterns Mining (EPM) is a data mining task that belongs to the supervised descriptive rule discovery (SDRD) framework [6]. This task tries to find out patterns whose supports increases significantly from one class, or dataset, to another [7]. The characteristics of patterns extracted makes EPM methods good classifiers. This methods have been applied with success in several cases in different fields [8]–[10]. However, the majority of works in the literature do not use the descriptive characteristics of the emerging patterns (EPs), losing a good source of knowledge for the experts.

Despite of the possibilities of EPM, the task is extremely complex when the volume of data is huge due to the non-convexity of the problem [11]. For this reasons, experts are looking for subsets of interesting EPs and methods that efficiently mines such subsets. One of the approaches more suitable are the evolutionary fuzzy systems (EFS) [12]. These systems are an hybridisation of evolutionary algorithms (EAs) [13] and fuzzy logic [14]. EAs are a well-known optimisation methods that search for the best solutions on huge search spaces efficiently. Additionally, the use of fuzzy logic allows to obtain a representation of the knowledge which is closer to human reasoning. Moreover, the use of numerical variables without discretisation improves the accuracy and the set of rules can be more precise due to the use of a belonging degree instead of a crisp one.

In this contribution we present a first approach to handle Big Data problems with EPM. The approach is an adaptation of the EvAEFP algorithm [15] to the MapReduce paradigm. The algorithm is an EA that finds EPs with high descriptive characteristics in a faster way than its predecessor on big datasets. To achieve this purpose, this paper is organised as follows: Section II presents a brief background of the main concepts used in this paper. Section III presents the evolutionary proposal for EPM under Big Data environments, the EvAEFP-Spark algorithm. Section IV presents an experimental study that validates the proposed algorithm. Finally, the contribution is concluded with the main findings.

II. Preliminaries

In this section, a brief background of the main concepts employed in the paper are presented. First, a background of EPM is shown in Section II-A. Next, the main properties of EFS are summarised in Section II-B. Finally, a brief review of the MapReduce paradigm can be observed in Section II-C

A. Emerging Patterns Mining

EPM was defined by Dong and Li [7], [16] as the task of finding all patterns with a growth rate (GR) value greater than a threshold $\rho \geq 1$. Given two datasets D_1 and D_2, the GR of a pattern x is defined in Eq. 1.

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

where $Sup_{D_i}(x) = \frac{count_{D_i}(x)}{|D_i|}$ is the support of the pattern in the dataset i.

The objectives of EPM is to find out emerging tendencies in timestamped datasets, discriminative characteristics among classes or serching for differencies among variables.

The main problem of EPM is that it is a non-convex problem [11], i.e., more general patterns can have lower GR values with respect to more specific ones. For this reason, the search space and the number of generated EPs could be huge. Throughout the literature there have been attempts to filter the number of patterns in order to get only high quality EPs with concepts like jumping EPs [17], strong jumping EPs [18], or fuzzy EPs [19], amongst others. This kind of patterns are joined with different search strategies in order to make EPM faster. In the literature, we can find methods following the border-based approach, like the DeEPS method [20], and following a tree-based approach, with methods like SJEP-C [18] and FEPM [19], amongst others.

B. Evolutionary Fuzzy Systems

An EFS is an hybridisation of fuzzy systems augmented by a learning process in an EA [12]. EAs are algorithms based on natural evolution to solve optimisation problems with complex search spaces. They have a good trade-off between the quality of results obtained and the elapsed time to obtain such results and it have been widely used in the literature.

On the other hand, fuzzy systems are based on fuzzy logic [14]. This allows to handle uncertainty and allows the representation of numeric variables in a more human-readable way. This is done by means of linguistic labels (LLs) where a fuzzy set represent a single label on the variable. This labels could be defined by the user or by means of uniform partitions if knowledge is unavailable.

The application of EFSs within the EPM can be observed as a search process in a huge and complex search space, so EFSs are a good solution in order to solve the problem of the EPs extraction. Specifically, the objective is a rule learning process to obtain interpretable knowledge through the use of IF-THEN rules with fuzzy logic statements. This type of systems are known as fuzzy rule-based systems (FRBSs). They are a well-suited method to obtain simple and interpretable knowledge.

C. MapReduce

MapReduce [21], [22] is one of the most popular programming paradigms to deal with Big Data. It is composed in two main parts: the map, and the reduce phases. In a nutshell, the map phase create some intermediate results of the input data and the reduce data joins all those intermediate results of the map phase to produce the final result.

One of the most popular frameworks that implements the MapReduce paradigm is Spark [23], which is a cluster computing framework for large-scale data processing with high efficiency due to an intensive use of main memory. This fact improves the performance on iterative algorithms like EAs in orders of magnitude with respect to other frameworks like Apache Hadoop [24]. The contribution presented in this paper is developed for Spark with the Scala programming language.

III. The EvAEFP-Spark Algorithm

This method is an adaptation of the EvAEFP algorithm [15] for Big Data problems. EvAEFP-Spark extracts an undetermined number of fuzzy EPs in order to describe trends from supervised learning. The fuzzy EPs can be formalised as rules in order to obtain more interpretable results. By this way, a rule is represented as:

$$R : Cond \rightarrow ClassValue$$

where $Cond$ is a conjunction of attribute-value pairs and $ClassValue$ is the value of the target class. It is important to remark that the objective of the method is to obtain discriminative characteristics among classes. In this way, D_1 is a partition of the original dataset where there are only instances of a class, and D_2 is formed with instances belonging to the other class. For multi-class problems, a *One vs All* (OVA) binarisation technique is applied in order to get rules for all possible classes.

A. Main properties

EvAEFP-Spark employs an EA with a "Chromosome = Rule" codification, where each individual in the population represents a potential rule. In this method, only the antecedent part of the rule is represented. This means that, in order to obtain rules for all possible values of the target class, this method must be executed once for each class. In Fig 1 a phenotype and genotype representations of an individual is presented. Note that a value of zero represents the absence of the variable in the rule.

Genotype

x_1	x_2	x_3	x_4
2	0	3	0

$$\Downarrow$$

Phenotype

$$IF\ (x_1 = LL_1^2) \wedge (x_3 = LL_3^1)\ THEN\ (x_{Obj} = Class)$$

Fig. 1. Representation of a rule in the EvAEFP-Spark algorithm.

As we can observe, a numerical variable is represented through LLs in the algorithm. With this representation, an example e is considered as covered by a rule R if its *Antecedent Part Compatibility* (APC) value is greater than zero. This measure is defined as:

$$APC(e, R) = T(\mu_n^1(e_1),\ \dots,\mu_n^i(e_i)) \tag{2}$$

where T is a t-norm function, in this case, the minimum t-norm, and $\mu_n^i(e_i)$ is a function that calculates the belonging degree in the LL n for the variable i. For discrete variables, it checks if the example at variable i has the discrete value n, returning one if matches, or zero elsewhere. This algorithm employs equivalent triangle representations for numerical variables. An example with five uniforms LLs can be observed in Fig. 2.

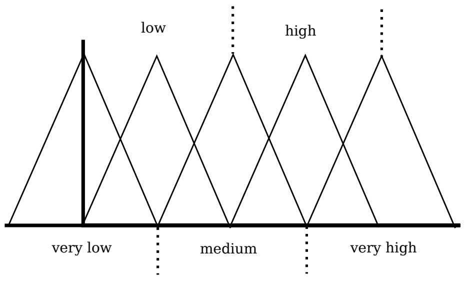


Fig. 2. Represention of a numeric variable with five uniform linguistic labels.

The proposed method is a mono-objective EA with an iterative rule learning (IRL) approach [25]. In this way, the algorithm extracts the best rule of the evolutionary process and iterates until a stopping criteria is reached, i.e., the execution is stopped when the last pattern is not an EP, it does not cover new examples not covered by previous extracted rules, or it has a null support. Additionally, this algorithm is generational with elitism where the best individual is kept in the population of the next generation.

The genetic operators used are a tournament selection [26] with two individuals, a two-point crossover operator [27] and a biased mutation operator presented in [28].

B. Operational scheme

The operational scheme of the proposed method is presented in Fig 3. As can be observed, the algorithm starts reading the data. Then, splits the data in disjoint partitions and send each partition to each computing node in the cluster. Next, the IRL approach is shown where the initial population is generated for the target class by means of a biased procedure. The initial population contains the 50% of individuals generated randomly and the remaining are generated with a maximum of an 80% of its variables initialised. At the end of the evolutionary process, the best rule is extracted and added to the final result set. This process is repeated while the stopping condition is not satisfied.

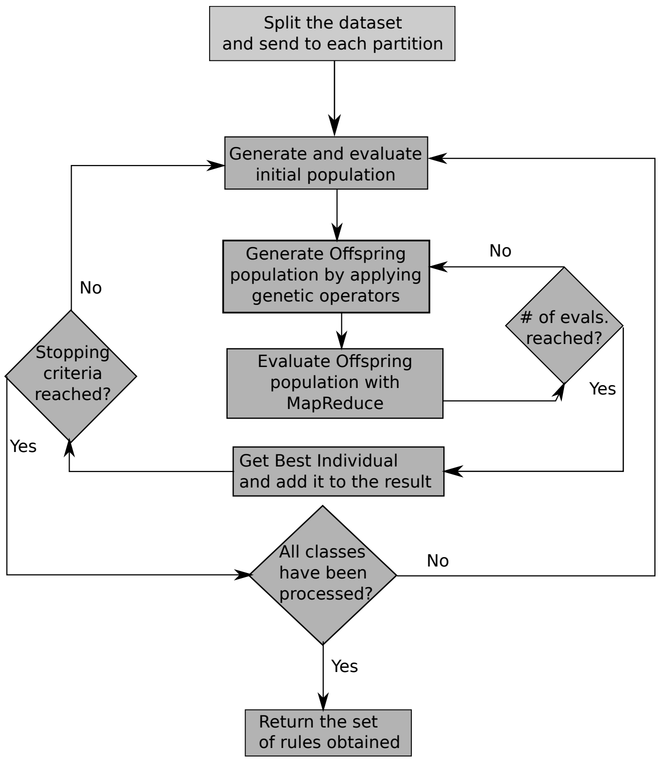

Split the dataset and send to each partition
Generate and evaluate initial population
Generate Offspring population by applying genetic operators
Evaluate Offspring population with MapReduce
# of evals. reached?
No
Yes
Stopping criteria reached?
No
Yes
Get Best Individual and add it to the result
All classes have been processed?
No
Yes
Return the set of rules obtained

Fig. 3. General Operational scheme of the EvAEFP-Spark algorithm.

The key concept of an EA is the fitness function. This is used to evaluate the quality of an individual. In EvAEFP-Spark, the fitness function is defined as:

$$Fitness(R) = \sqrt{TPr * TNr} \tag{3}$$

where a geometric mean between the true positive rate (TPr) and the true negative rate (TNr) is used in order to obtain a balanced way to maximise the precision and generality of a potential rule.

C. MapReduce approach

The fitness function is the most expensive task of an EA. In this way, for each individual is necessary to traverse the complete dataset once for each individual in the population in order to calculate the TPr and TNr. In the analysis of Big Data through EAs this consideration must be taken into account.

For the analysis of the fitness function in Big Data environments within EPM through the use of EAs is necessary the presentation of the results expressed by means of a confusion matrix. Table I represents a confusion matrix for an emerging rule with tp as the number of examples correctly covered, fn such as the number of examples for the class not covered, fp is the number of examples incorrectly covered, and tn is the number of examples non-covered for the not class.

The measures used in the fitness function can be easily calculated from this confusion matrix. In fact, $TPr(R) = \frac{tp}{tp+fn}$ and $TNr = \frac{tn}{tn+fp}$.

TABLE I
CONFUSION MATRIX FOR A RULE

	Predicted	
Real	Positive	Negative
Positive	tp	fn
Negative	fp	tn

In EvAEFP-Spark the fitness function has been developed following the MapReduce paradigm in order to reduce the complexity when evaluating the population. On each generation of the EA, the map phase sends those individuals generated by genetic operators and, thus, not evaluated yet, to the data partitions. Then, on each map, a confusion matrix is calculated for each individual. After that, the reduce phase performs the sum of all the possible confusion matrices, getting only one confusion matrix for each individual with the correct values to calculate the measures back on the driver. A scheme of this MapReduce approach is shown in Figure 4.

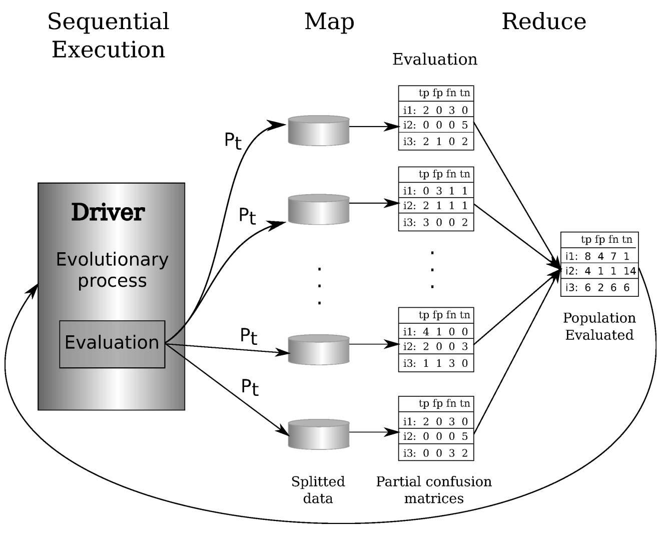


Fig. 4. Scheme of the evaluation procedure of the EvAEFP-Spark algorithm following a MapReduce approach.

IV. EXPERIMENTAL STUDY

An experimental study with some big datasets has been performed in order to validate the proposal presented in this paper. In this study, a comparison of the main descriptive quality measures and execution times with respect to the original EvAEFP algorithm is analysed.

This study has been evaluated through different datasets available in UCI repository [29]. The characteristics of these datasets are summarised in Table II.

TABLE II
CHARACTERISTICS OF THE DATASETS USED IN THE EXPERIMENTAL FRAMEWORK.

Dataset	**# Instances**	**# Instances (R/I/N)**	**# Classes**
census	299284	41 (1/12/28)	3
kddcup	494020	41 (26/0/15)	23
rlcp	5749132	11 (11/0/0)	2
susy	5000000	18 (18/0/0)	2

The study is compared through a 5-fold stratified cross validation which avoids the arbitrariness of the classical k-fold cross validation and the dependence of the results with respect to the partitioning performed in order to get the descriptive quality measures. The parameters used in the execution of the algorithms are summarised in Table III.

TABLE III
PARAMETERS USED FOR THE EVAEFP AND THE EVAEFP-SPARK ALGORITHMS IN THE EXPERIMENTAL FRAMEWORK.

Parameter	**Value**
number of fuzzy labels	3
number of evaluations	10000
population length	100
crossover probability	0.60
mutation probability	0.01

Additionally, EvAEFP-Spark needs to specify the number of partitions used to divide the input data. In this experimental study we execute the algorithm with 4, 8, 16, 32, 64, 128 and 256 partitions. With respect to the quality of the EPs extracted, this is analysed through the following quality measures:

- GR. It is the main quality measure in EPM and it is defined in Eq. 1. However, the range for this quality measure ($[0, \infty]$) implies the necessity to measure the percentage of patterns that are EPs on the test set.
- True Positive Rate (TPr). It measures the ratio of true positive examples covered by the rule, giving a generality measure of the rule obtained.
- False Positive rate (FPr). It measures the ratio of false positive examples covered by the rule, giving a precision measure for the rule obtained.
- Execution time. It measures the time elapsed in seconds for the complete execution process.

Complete results for this experimental study can be observed in Table IV where the quality of the EPs extracted for the new approach is kept. In general, the values of the quality measures analysed are very similar, so the new approach satisfies the objectives of the sequential one. The variations in some values are registered due to the complexity of the patterns extracted for both methods, i.e. the EvAEFP algorithm extracts a number of rules higher than the EvAEFP-Spark and the rules also contain a higher number of variables. Moreover, the sequential algorithm is not able to obtain results in prudential times for RLCP and Susy.

On the other hand, the execution time for the new big data approach are reduced in the process of extraction of EPs. In fact, when the number of maps is increased the time continues reducing. Specially, this assertion can be observed in datasets

TABLE IV
RESULTS OBTAINED BY THE SEQUENTIAL EVAEFP ALGORITHM AND EVAEFP-SPARK WITH DIFFERENT NUMBER OF PARTITIONS

Dataset	Algorithm	GR	TPR	FPR	Time
census	EvAEFP	**1.000**	0.492	**0.395**	10443
	EvAEFP-Sp (4)	0.819	**0.534**	0.430	4915
	EvAEFP-Sp (8)	0.819	**0.534**	0.430	3584
	EvAEFP-Sp (16)	0.819	**0.534**	0.430	2863
	EvAEFP-Sp (32)	0.819	**0.534**	0.430	2522
	EvAEFP-Sp (64)	0.819	**0.534**	0.430	2169
	EvAEFP-Sp (128)	0.819	**0.534**	0.430	**2100**
	EvAEFP-Sp (256)	0.819	**0.534**	0.430	2298
kddcup	EvAEFP	0.509	**0.432**	0.247	57840
	EvAEFP-Sp (4)	**0.602**	0.416	**0.002**	21767
	EvAEFP-Sp (8)	**0.602**	0.416	**0.002**	16709
	EvAEFP-Sp (16)	**0.602**	0.416	**0.002**	13980
	EvAEFP-Sp (32)	**0.602**	0.416	**0.002**	12472
	EvAEFP-Sp (64)	**0.602**	0.416	**0.002**	**11859**
	EvAEFP-Sp (128)	**0.602**	0.416	**0.002**	11960
	EvAEFP-Sp (256)	**0.602**	0.416	**0.002**	12347
rlcp	EvAEFP	-	-	-	-
	EvAEFP-Sp (4)	**1.000**	**0.951**	**0.013**	5628
	EvAEFP-Sp (8)	**1.000**	**0.951**	**0.013**	4138
	EvAEFP-Sp (16)	**1.000**	**0.951**	**0.013**	3660
	EvAEFP-Sp (32)	**1.000**	**0.951**	**0.013**	2969
	EvAEFP-Sp (64)	**1.000**	**0.951**	**0.013**	2746
	EvAEFP-Sp (128)	**1.000**	**0.951**	**0.013**	**2621**
	EvAEFP-Sp (256)	**1.000**	**0.951**	**0.013**	2948
susy	EvAEFP	-	-	-	-
	EvAEFP-Sp (4)	**0.533**	**0.302**	**0.211**	13050
	EvAEFP-Sp (8)	**0.533**	**0.302**	**0.211**	12939
	EvAEFP-Sp (16)	**0.533**	**0.302**	**0.211**	11645
	EvAEFP-Sp (32)	**0.533**	**0.302**	**0.211**	10729
	EvAEFP-Sp (64)	**0.533**	**0.302**	**0.211**	10481
	EvAEFP-Sp (128)	**0.533**	**0.302**	**0.211**	**9929**
	EvAEFP-Sp (256)	**0.533**	**0.302**	**0.211**	10865

with a high dimensionality such as RLCP or Susy where the most reduced execution time are obtained with a high number of maps. A graphical analysis for times and for each dataset are shown in Fig. 5.

For this first approach of Big Data in EPM, it is important to note different questions to take into account into the development of future approaches. The reduction of time has a linear decreasing until the 100 number of maps due to the complexity of the dataset directly, and by the MapReduce overload produced that causes an extra computation time on each generation of the algorithm. The EvAEFR-Spark follows an IRL approach an it is executed as many times as needed while the algorithm discovers new EPs. On the other hand, when the cost of the computation of the confusion matrices is significantly reduced due to the parallel computation (the MapReduce, with a considerable number of maps) overload is not a problem and makes the algorithm faster than the sequential. It can be observed in complex datasets such as kddcup, rlcp or susy. Finally, the study shows the difficulty of determining the optimal number of maps, because this number is dependent on the problem to analyse. For example, the census datasets obtain the best execution time with a number of maps between 30 and 60, whereas datasets complex obtain the best times considering a higher number of maps.

V. CONCLUSIONS

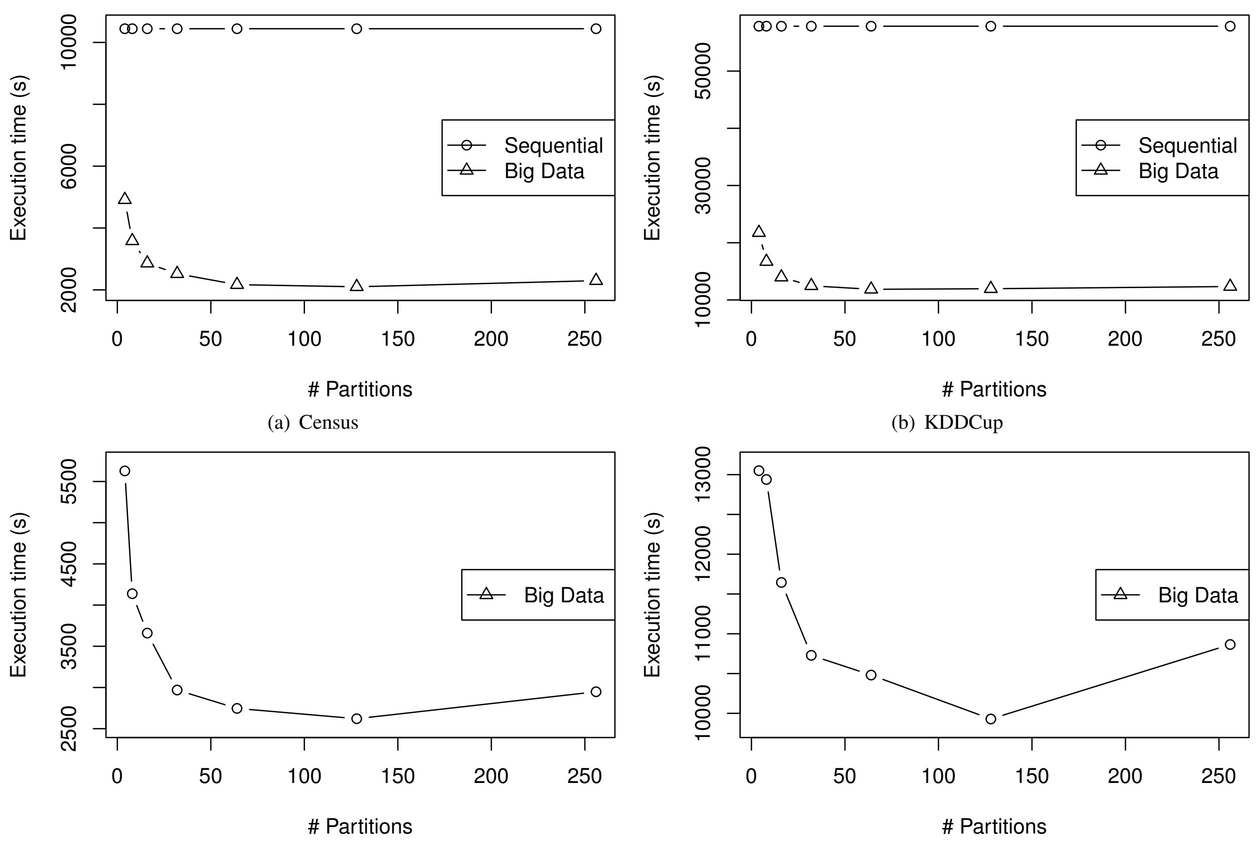
In this paper the EvAEFP-Spark algorithm is presented. This method is an adaptation of EvAEFP for Big Data environments. Additionally, it is the first attempt to handle Big Data problems with EPM which is a SDRD task. The main objective of EPM is to discover patterns with high GR values, but the definition of the problem makes it a hard task when the amount of data is huge. By this way, the evolutionary approach of the EvAEFP algorithm and its lower computational complexity with respect to other classic methods makes it a possible task to handle this kind of problems efficiently. In this way, the proposed MapReduce approach is based on a map phase where the population on a given generation is sent to the partitions and for each individual a partial confusion matrix is calculated. Next, a reduce phase where the confusion matrices of the map phase for an individual are summarised in order to get the final confusion matrix. In this way, the quality measure are calculated efficiently. A comparison between the EvAEFP and the EvAEFP-Spark shows the advantages of this new proposal in big data environments.

ACKNOWLEDGMENT

This work was partially supported by the Spanish Ministry of Economy and Competitiveness under the project TIN2015-68454-R (FEDER Founds)

REFERENCES

[1] T. Kraska, "Finding the Needle in the Big Data Systems Haystack," *IEEE Internet Computing*, vol. 17, no. 1, pp. 84–86, 2013.
[2] S. Madden, "From databases to big data." *IEEE Internet Computing*, vol. 16, no. 3, 2012.
[3] A. Fernandez, S. del Rio, V. Lopez, A. Bawakid, M. J. del Jesus, J. M. Benitez, and F. Herrera, "Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks," *Wiley Interdisciplinary Reviews-Data Mining and Knowledge Discovery*, vol. 4, no. 5, pp. 380–409, 2014.
[4] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[5] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014.
[6] P. Kralj-Novak, N. Lavrac, and G. I. Webb, "Supervised Descriptive Rule Discovery: A Unifying Survey of Constrast Set, Emerging Pateern and Subgroup Mining," *Journal of Machine Learning Research*, vol. 10, pp. 377–403, 2009.
[7] G. Z. Dong and J. Y. Li, "Efficient Mining of Emerging Patterns: Discovering Trends and Differences," in *Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 1999, pp. 43–52.
[8] G. Tzanis, I. Kavakiotis, and I. P. Vlahavas, "Polya-iep: A data mining method for the effective prediction of polyadenylation sites," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12 398–12 408, 2011.
[9] R. Sherhod, V. J. Gillet, T. Hanser, P. N. Judson, and J. D. Vessey, "Toxicological knowledge discovery by mining emerging patterns from toxicity data," *Journal of Chemical Information and Modeling*, vol. 5, no. S-1, p. 9, 2013.
[10] Y. Yu, K. Yan, X. Zhu, and G. Wang, "Detecting of PIU Behaviors Based on Discovered Generators and Emerging Patterns from Computer-Mediated Interaction Events," in *Proc. of the 15th International Conference on Web-Age Information Management*, ser. LNCS, vol. 8485. Elsevier, 2014, pp. 277–293.



(a) Census (b) KDDCup (c) RLCP (d) Susy

Fig. 5. Comparative of execution times of the Big Data algorithm against the sequential one on the analysed datasets.

[11] L. Wang, H. Zhao, G. Dong, and J. Li, "On the complexity of finding emerging patterns," in *Proc. of the 28th Annual International Computer Software and Applications Conference*, vol. 2, 2004, pp. 126–129.

[12] F. Herrera, "Genetic fuzzy systems: taxomony, current research trends and prospects," *Evolutionary Intelligence*, vol. 1, pp. 27–46, 2008.

[13] T. Bäck, D. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*. Oxford University Press, 1997.

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

[15] A. M. García-Vico, J. Montes, J. Aguilera, C. J. Carmona, and M. J. del Jesus, "Analysing Concentrating Photovoltaics Technology through the use of Emerging Pattern Mining," in *Proc. of the 11th International Conference on Soft Computing Models in Industrial and Environmental Applications*. Springer, 2016, pp. 1–8.

[16] G. Z. Dong and J. Y. Li, "Mining border descriptions of emerging patterns from dataset pairs," *Knowledge and Information Systems*, vol. 8, no. 2, pp. 178–202, 2005.

[17] G. Z. Dong, J. Y. Li, and X. Zhang, "Discovering jumping emerging patterns and experiments on real datasets," in *Proc. on International Database Conference Heterogeneous and Internet Databases*, 1999, pp. 155–168.

[18] H. Fan and 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*, vol. 18, no. 6, pp. 721–737, 2006.

[19] M. García-Borroto, J. Martínez, and J. Carrasco-Ochoa, "Fuzzy emerging patterns for classifying hard domains." *Knowledge and Information Systems*, vol. 28, no. 2, pp. 473–489, 2011.

[20] J. Y. Li, G. Z. Dong, K. Ramamohanarao, and L. Wong, "DeEPs: A New Instance-Based Lazy Discovery and Classification System," *Machine Learning*, vol. 54, no. 2, pp. 99–124, 2004.

[21] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[22] ——, "MapReduce: A flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

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

[24] T. White, *Hadoop, The Definitive Guide*. O'Reilly Media, 2012.

[25] G. Venturini, "SIA: A Supervised Inductive Algorithm with Genetic Search for Learning Attributes based Concepts," in *Proc. European Conference on Machine Learning*, ser. LNAI, vol. 667. Springer, 1993, pp. 280–296.

[26] B. L. Miller and D. E. Goldberg, "Genetic Algorithms, Tournament Selection, and the Effects of Noise," *Complex System*, vol. 9, pp. 193–212, 1995.

[27] J. H. Holland, "Adaptation in natural and artificial systems," *University of Michigan Press*, 1975.

[28] M. J. del Jesus, P. González, F. Herrera, and M. Mesonero, "Evolutionary Fuzzy Rule Induction Process for Subgroup Discovery: A case study in marketing," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 4, pp. 578–592, 2007.

[29] A. Asuncion and D. J. Newman, "UCI Machine Learning Repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html