Expert Systems With Applications 183 (2021) 115419

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A cellular-based evolutionary approach for the extraction of emerging patterns in massive data streams

Ángel M. García-Vico [a,*], Cristóbal Carmona [b], Pedro González [b], María J. del Jesus [b]

[a] Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, 18071 Granada, Spain
[b] Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Jaén, 23071 Jaén, Spain

ARTICLE INFO

Keywords:
Big data
Data stream mining
Emerging pattern mining
Fuzzy logic
Evolutionary algorithms

ABSTRACT

Today, the number of existing devices generates immense amounts of data on a continuous basis that must be processed by new distributed data stream mining approaches. In this paper we present a new approach for extracting descriptive emerging patterns in massive data streams from different sources through Apache Kafka and Apache Spark Streaming whose objective is to monitor the state of the system with respect to a variable of interest. For this purpose, the proposed algorithm is a cellular-based multi-objective evolutionary fuzzy system that uses an informed strategy for efficient data processing and a re-initialisation and filtering mechanism to eliminate redundant and low-reliable patterns. The experimental study carried out demonstrates an interpretability improvement of 25% in the extraction of high-interest knowledge by the proposed algorithm, which would make it easier for experts to analyse the problem. Finally, the proposed algorithm is up to five times faster than another proposal on the processing of the same amount of data. In this experimental study, up to 750,000 instances have been processed in approximately four seconds.

1. Introduction

The amount of information generated is growing exponentially nowadays. For example, the experiments at the Large Hadron Collider at CERN could generate about 90 petabytes of data per year and process one petabyte of data a day (CERN, 2021). According to a Cisco report it is expected that the amount of information generated worldwide by 2021 will be approximately 850 zettabytes (Cisco, 2021). These data are mainly produced by the high amount of information transmitted between devices and the explosion of the Internet of Things devices (Sezer, Dogdu, & Ozbayoglu, 2017; Nord, Koohang, & Paliszkiewicz, 2019). From this huge amount of data, the information generated but not stored is two orders of magnitude higher than the amount of information finally stored. This means that the majority of the generated data are considered interesting only at their creation. However, they are usually neither stored nor analysed after that. In the worst case, these data are stored, but they are never analysed, creating the so-called data tombs. Nevertheless, these data can contain interesting insights about its application domain. This information could be relevant for companies in order to improve their services and productivity, together with many other applications that requires a quick response, whereas computational resources are efficiently employed.

It is undeniable that we live surrounded by data. These huge amounts of data are commonly known as big data. Big data can be characterised by the 5 V's model (volume, velocity, variety, veracity and value) (Mayer-Schonberger & Cukier, 2013), which describes the massive volume of data, their fast generation, their diverse nature and their usefulness for the experts. In many cases the short life of generated data force us to process them as soon as they arrive into the system for providing a fast, reliable response. These data that continuously arrive into the system at an undetermined speed are known in the literature as a data stream (Gama, 2010). In this scenario, the learning model must be continuously updated and adapted to the incoming data. However, volume, velocity and variety of data could be so huge nowadays that classical approaches are not suitable to handle them. Thus, a distributed data stream processing approach becomes compulsory. The interest in the analysis of this kind of data is evidenced by the number of distributed, large-scale processing frameworks that have been developed up to date for this purpose. In particular, Apache Spark (Zaharia, Chowdhury, Franklin, Shenker, & Stoica, 2010), Kafka (Garg, 2013), Storm (Foundation, 2021), or Flink (Carbone et al., 2015), amongst others, are gaining special attention for both its distributed real-time performance,

* Corresponding author.
E-mail addresses: agvico@decsai.ugr.es (Á.M. García-Vico), ccarmona@ujaen.es (C. Carmona), pglez@ujaen.es (P. González), mjjesus@ujaen.es (M.J. del Jesus).

https://doi.org/10.1016/j.eswa.2021.115419
Received 14 January 2021; Received in revised form 7 May 2021; Accepted 9 June 2021
Available online 23 June 2021

and fault-tolerant processing within iterative methods such as machine learning algorithms.

In the last few years, the literature on data streams have been focused on concept drift analysis (Brzeziński, 2015; Ramírez-Gallego, Krawczyk, García, Wozniak, & Herrera, 2017; Khamassi, Sayed Mouchaweh, Hammami, & Ghédira, 2018). These works, amongst others, have greatly improved the development of the area. However, many other aspects within data stream mining are not completely developed, so real-world applications are still challenging. For example, it is very difficult to find completely labelled data within high-speed, massive data streams. In these cases, it would be more realistic the development of supervised-learning-based, descriptive models with good interpretability for system monitoring. In this way, supervised learning techniques can be employed in order to monitor the behaviour of data with respect to a naturally-generated property of interest, which can be employed as class label.

Emerging Pattern Mining (EPM) (Dong & Li, 1999; García-Vico, Carmona, Martín, García-Borroto, & del Jesus, 2018) is a data mining task within the Supervised Descriptive Rule Discovery (SDRD) framework (Kralj-Novak, Lavrac, & Webb, 2009). The aim of this task is the description of the discriminative relationships in data with respect to a property of interest. In particular, it tries to describe the characteristic differences between the values of a property of interest or the description of emerging behaviour in data. In this way, experts can obtain an easy-to-understand pattern model which describes the underlying phenomena in data. Hence, EPM can be useful within data stream mining as the purpose is to monitor the behaviour of the stream using a simple, readable, reliable model. EPM has been successfully applied in many different fields such as disease management (Piao, Lee, Sohn, Pok, & Ryu, 2009; Park, Lee, & Park, 2010; Tzanis, Kavakiotis, & Vlahavas, 2011; Poezevara et al., 2017), toxicology (Sherhod, Gillet, Judson, & Vessey, 2012; Sherhod, Gillet, Hanser, Judson, & Vessey, 2013), renewable energies (García-Vico, Montes, Aguilera, Carmona, & del Jesus, 2016), management (Li, Law, Vu, Rong, & Zhao, 2015) and social networks (Peng et al., 2018), amongst others. In addition, approaches based on Evolutionary Fuzzy Systems (EFSs) have been recently proposed in García-Vico, Carmona, González, and del Jesus (2018) which surpasses the descriptive capacities of the classical methods. Nevertheless, the development of EPM algorithms within massive data stream mining is still challenging. This is mainly due to the computational complexity of the mining methods (Wang, Zhao, Dong, & Li, 2004) and the difficulties for the development of fast, distributed strategies. This makes unfeasible its application to massive data stream environments, as they require an almost real-time response. In addition, one of the main drawbacks of EPM methods in data stream mining is that they require a finite dataset in order to compute the required quality measures for the extraction of the patterns. A first approach to solve this issue is a multi-objective EFS following a block-based learning approach for data stream mining, proposed in García-Vico, Carmona, González, and del Jesus (2020). Although the quality of the knowledge extracted is good, its learning method is continuously executed to be adapted with respect to the stream. Moreover, it does not provide any distributed mechanism to efficiently scale up the mining process. Therefore, its application within high-speed, massive data stream environments could be a problem, as many unnecessary executions of the learning method are carried out without any data distribution mechanism.

In this paper, a Cellular-based Evolutionary approach for the Extraction of Emerging Patterns in Massive Data Streams (CE3P-MDS) is proposed. The main contributions of this paper are as follows:

1. Learning method inspired on a cellular-based, multi-objective evolutionary algorithm (Nebro, Durillo, Luna, Dorronsoro, & Alba, 2009) which improves the diversity-exploitation trade-off, together with a reinitialisation method based on the odds ratio measure which removes those redundant patterns with the highest complexity.
2. Smart triggering of the learning method, which updates and adapt the current pattern model with respect to the state of the data stream only when it is necessary, according to the user requirements.
3. Scalable approach for the processing of massive, high-speed data streams from several sources thanks to the employment of Apache Kafka and Apache Spark.
4. Comprehensive experimental evaluation of the proposed method.

This paper is organised as follows: firstly, the main concepts related to big data analysis, data stream mining and EPM are presented in Section 2. Next, the main components of CE3P-MDS and its working scheme are shown in Section 3. After that, the experimental study, the results extracted and its discussion are depicted in Section 4. Finally, the conclusions of this work are presented in Section 5.

2. Related work

In this section, the main concepts related to this paper are presented below: big data analysis (Section 2.1), data stream mining (Section 2.2) and and EPM (Section 2.3).

2.1. Big data analysis

One of the most popular paradigms for the processing of big data is MapReduce due to its easy, automatic, transparent deployment through a cluster of computers (Dean & Ghemawat, 2004). The framework is based on a divide-and-conquer strategy where the user is only required to implement two main operators: map and reduce. The map operator is a function $map : \mathbb{Z} \times \mathbb{R}^n \rightarrow \mathbb{Z} \times \mathbb{R}^n$ which transforms key-value pairs that comes from the distributed file system to another key-value pair with intermediate results. In this phase, each node reads and transforms data from different partitions in a distributed fashion. The reduce operator is a function $reduce : (\mathbb{Z} \times \mathbb{R}^n) \times (\mathbb{Z} \times \mathbb{R}^n) \rightarrow \mathbb{Z} \times \mathbb{R}^n$ which aggregates the intermediate results of the map phase by the same key in order to obtain the final result. For further information about MapReduce and other big data processing frameworks, please refer to Fernández et al. (2014) and Ramírez-Gallego, Fernández, García, Chen, and Herrera (2018).

Apache Spark (Zaharia et al., 2010) is one of the most powerful big data engines developed up to date. This is because of the employment of a data structure called Resilient Distributed Dataset (RDD) (Zaharia et al., 2012). RDDs provide the user with a set of primitives with an intensive use of main memory. Spark is able to load data into memory and query them repeatedly without loosing performance, making it suitable for iterative methods such as machine learning algorithms. An example of its popularity can be shown in the MLlib package (Meng et al., 2015), which contains a large set of common machine learning algorithms and statistical utilities for Spark.

For the processing of massive data streams, Spark provides the Apache Spark Streaming library. This allows the processing of streams by means of automatically forming RDDs with the data that arrive into the system within a user-defined time window. This design allows the user to employ the same RDD primitives for distributed batch processing within stream data without significant code modifications.

On the other hand, Apache Kafka (Garg, 2013) is one of the most popular streaming platforms. One of its main characteristics is that it is able to collect data from different sources by means of a publication/subscription scheme. To do this, it is necessary to define identifiers for the buffers where the publications or subscriptions are made, called topics. In this way data are read/written from/to the topic thus the number of subscribers and publisher can be dynamically modified.

2.2. Data stream mining

A data stream is classically defined as an unbounded, ordered sequence of instances that arrive into the processing system throughout time at a variable speed (Gama, 2010). According to this definition,

methods must provide mechanisms to update its structure according to the current state of the stream (Krawczyk, Minku, Gama, Stefanowski, & Woźniak, 2017). Also, they must provide a low update and response time for processing data as soon as they become available to avoid queueing them in order to maintain the system stability. Data stream mining algorithms must assume limited storage and memory as data size is infinity by definition (Ramírez-Gallego et al., 2017). Finally, due to the evolving nature of streams, it is possible that the underlying distributions that produce the data change throughout time. This phenomenon is known as concept drift (Webb, Lee, Goethals, & Petitjean, 2018).

In general, concept drift can be defined as a change in time t with respect to a time $t+\delta$ on the joint probability of a given instance e and its output C, i.e., $P(e,C)^t \neq P(e,C)^{t+\delta}$ (Khamassi et al., 2018). However, considering cause and effect produced on the probability distribution associated to the data, two main types of concept drift are interesting from a descriptive point of view: real drift (Gama, Žliobaitė, Bifet, Pechenizkiy, & Bouchachia, 2014) and virtual drift (Khamassi et al., 2018). In a nutshell, when a real drift occurs, the changes could degrade the performance of the model as it affects the decision boundaries. On the other hand, virtual drifts changes the distribution of input data, which do not necessary affect performance. However, virtual drifts are also interesting for the experts, so it is necessary to report them. Finally, it is important to remark that these changes could happen at different speeds throughout time. These can be categorised as abrupt, where a sudden drift occurs instantly; incremental, where the change is produced gradually over time; gradual, which is an incremental change with variations in the class distribution; and recurrent, where drifts occur in a seasonally fashion (Sayuri-Iwashita & Papa, 2019).

All of these aspects together lead to online adaptive learning whereby data arrive into the system and an update of the current knowledge model is obtained using the new data. Later on the estimation of the quality of this update is analysed where it can be confirmed or discarded in order to correctly adapt to the current state of the stream. In this way, five elements should be defined for the extraction of knowledge from data streams (Gama et al., 2014; Khamassi et al., 2018):

1. Memory. This element is in charge of collecting new information from the stream, it provides them in an appropriate way to the learning method, and it forgets the old one. Two main approaches can be found in the literature: sequential, where data is provided to the method one by one as soon as they arrive; or windowing, where a set of data is collected and stored as a buffer, or a set of buffers, usually with the most recent instances.
2. Learning process. Learning methods should be adapted to work in an online fashion, usually by the incorporation of some forgetting mechanisms that allow them to unlearn obsolete knowledge. A popular approach is the employment of ensembles (Krawczyk et al., 2017) which provides flexibility to properly handle concept drifts.
3. Change monitoring. It is in charge of the early detection of a concept drift. Different methods can be employed, e.g., supervised indicators such as accuracy or sensitivity when feedback is immediately available (Khamassi & Sayed Mouchaweh, 2014; Demšar & Bosnić, 2018); or unsupervised indicators such as time and space similarity, among others (Shaker & Lughofer, 2014; Toubakh & Sayed-Mouchaweh, 2015; de Mello, Vaz, Grossi, & Bifet, 2019).
4. Learning adaptation. This component contains the elements for updating the knowledge model. Two main approaches can be found: incremental, where forgetting mechanisms are applied for discarding old concepts; and evolving, where both the concept drift adaptation mechanisms and the model are updated according to the learning needs (Lughofer, 2016; Škrjanc et al., 2019). The adaptation process can be triggered using different strategies such as blind, where the model is updated at regular intervals; or informed, where the adaptation is triggered according to the change monitoring criteria.
5. Evaluation strategy. One of the most common strategies for the evaluation of the model within data stream mining is the well-known interleaved test-then-train evaluation (Wald, 1973). In addition, other performance factors such as execution time or memory consumption over time should be taken into consideration.

As mentioned previously, in most real-world applications a learning approach that is able to explain or monitor the behaviour of the stream is more interesting than a predictive approach. This is due to the high demand in the interpretability of the patterns discovered, together with the need of discovering them while they are happening in big data scenarios. These requirements can be fulfilled by tasks such as association rule mining, EPM, amongst others. Nevertheless, this application area is still recent and there are not many functional association rule mining algorithms for data streams. In particular, many of the proposed approaches in this area are focused on the extraction of frequent items without the subsequent rule generation process (Rehman, Shahbaz, Shaheen, & Guergachi, 2015; Li, Zhang, Zhu, Wang, & Cao, 2018; Saleti & Subramanyam, 2019; Huynh & Küng, 2020), or this is carried out in an offline fashion (Gama, 2010). Furthermore, most of these algorithms can only deal with categorical features (Li, Shan, & Lee, 2008; Cheng, Ke, & Ng, 2008; Wang & Chen, 2011; Farzanyar, Kangavari, & Cercone, 2012). Three main approaches have been employed to properly deal with numeric features: to discretize the features (Srikant & Agrawal, 1996; Miller & Yang, 1997; Wang, Tay, & Liu, 1998), to use an interval-based representation (Mata, Alvarez, & Riquelme, 2002; Orriols-Puig, Casillas, & Bernadó-Mansilla, 2008), and the employment of fuzzy logic in order to avoid the hard boundaries produced by the interval-based representations.

To the best of our knowledge, the Fuzzy-CSar algorithm (Ruiz & Casillas, 2018) is a method that partially solves all the mentioned problems, so it is an approach that can be applied in real problems. It is able to extract fuzzy association rules in both categorical and numeric data in an incremental fashion. Its learning method is based on a steady-state EFS which is only applied at a fixed interval on the set of rules that matches the incoming instance. Therefore, the method can be efficiently applied within data stream mining. However, its runtime is very high to be applied in high-speed, massive data streams.

2.3. Emerging pattern mining

EPM (Dong & Li, 1999; García-Vico et al., 2018) is a data mining task that belongs to the SDRD framework (Kralj-Novak et al., 2009) whose main aim is the extraction of those patterns that contains a significant change in its support from one dataset $\mathscr{D}_1$ to another $\mathscr{D}_2$. In particular, the set of EPs is defined as $EP = \{\mathscr{P} | GR(\mathscr{P}) > \rho\}$, where $\mathscr{P}$ is a pattern, GR is the growth rate measure and $\rho > 1$ is a threshold value. The GR is defined as in Eq.1.

$$GR\left(\mathscr{P}\right) = \begin{cases} 0, & IF\ Sup_1(\mathscr{P}) = Sup_2(\mathscr{P}) = 0, \\ \infty, & IF\ Sup_1(\mathscr{P}) \neq 0 \wedge Sup_2(\mathscr{P}) = 0, \\ \dfrac{Sup_1(\mathscr{P})}{Sup_2(\mathscr{P})}, & otherwise \end{cases} \tag{1}$$

where $Sup_i(\mathscr{P})$ is the support of the pattern $\mathscr{P}$ on dataset $\mathscr{D}_i$. The main objectives of this task are the description of the discriminative characteristics between classes and the description of emerging tendencies in data. The patterns extracted are usually represented by means of conjunctions of attribute-value pairs or in disjunctive normal form (DNF), which represents the described discriminative characteristics, together with a label for the identification of the class they are describing. In this work, the main objective is to continuously describe the discriminative characteristics between the different classes of the data stream. A detailed review where the most relevant algorithms for EPM are clas-

sified according to the approach they employ to mine EPs is presented in García-Vico et al. (2018).

In EPM a good descriptive capacity is key for the extraction of an easy-to-understand model that facilitates the analysis. This descriptive capacity is characterised by a trade-off between three main aspects:

1. Reliability of the patterns, i.e., how accurate these descriptions are.
2. Generality, in terms of number of instances affected by the description made by the pattern.
3. Novelty or interest that produce the knowledge extracted by the expert.

This trade-off becomes even more relevant within the data stream mining context as a fast decision-making process must be performed within an evolving environment. Therefore, the interpretability of the results is a fourth aspect that must be considered for facilitating the analysis.

As can be observed, the EPM problem is clearly multi-objective. Several quality measures have been defined for the determination of these characteristics within the SDRD framework (Kralj-Novak et al., 2009; Herrera, Carmona, González, & del Jesus, 2011; García-Vico et al., 2018). A quality measure for EPM can be defined as a function $\phi : \mathbb{N}^4 \rightarrow \mathbb{R}$, which takes as input the values of a two-class contingency table where the number of correctly/incorrectly covered/uncovered instances are calculated for each pattern, as shown in Table 1. Similarly, the computation of this contingency table is defined with another function $\tau : EP \times \mathscr{D} \rightarrow \mathbb{N}^4$, which takes as input an EP and a dataset. The most widely used quality measures in EPM are outlined in Table 2 (García-Vico et al., 2018).

One of the main drawbacks of EPM in data streams is its necessity to require a finite dataset in order to calculate these quality measures. In addition, the majority of the developed approaches up to date are not suitable for massive data stream mining due to its computational complexity and its difficulties for the creation of distributed strategies (Wang et al., 2004). Thus, the improvement of EPM algorithms focused on data stream mining is still challenging. In fact, to the best of our knowledge, the problems presented previously for association rule mining are also related to EPM. A recently-proposed alternative that fulfils these requirements is the FEPDS algorithm (García-Vico et al., 2020). This approach is based on a multi-objective EFS, together with a sliding window that allows an easy computation of the required quality measures using the data within such window. FEPDS uses a blind strategy able to continuously update the pattern model, biasing the search towards the most recent data. Therefore, the model is able to quickly adapt to concept drift. However, it cannot be applied to massive data streams due to (1) the employment of a blind strategy produces unnecessary computations, specially when data do not significantly change, and (2) it does not present any distributed computation approach able to efficiently scale up with respect to the data size.

3. CE3P-MDS: cellular-based evolutionary approach for the extraction of emerging patterns in massive data streams

Fuzzy Rule-Based Systems (FRBSs) (Mamdani & Assilian, 1975) are knowledge systems composed by a set of IF-THEN rules where both antecedent and consequent can contain fuzzy sets. There are two main components within FRBSs: the knowledge base (KB), which contains the fuzzy rules, and the data base (DB) that contains the fuzzy sets definitions. Throughout the literature, EFSs have been widely used for learning the KB, the DB, or both from scratch; or for tuning its elements as a posteriori procedure (Fernandez, Lopez, del Jesus, & Herrera, 2015; Fernández, Herrera, Cordón, del Jesus, & Marcelloni, 2019). In data stream mining, the construction of both the KB and DB from scratch is prohibited in this context as the search space and complexity is huge. The construction of the DB is also a complex process, as many aspects such as the granularity degree or the shape of the membership functions, amongst others must be optimised for each variable in a continuous search space. In addition, a predefined KB must be provided, which is not flexible enough to monitor the state of the stream. On the other hand, learning the KB is an interesting approach as it allows flexibility to monitor the state of the stream and its search space is discrete, as numeric values are firstly fuzzyfied, thus it is simpler than the other ones. However, a predefined DB must be provided by the expert beforehand.

Table 1
Contingency table of an emerging pattern.

	Class	No-Class
Covered	tp	fp
Not-Covered	fn	tn

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

Name	Abbreviation	Formula
Confidence (Fayyad et al., 1996)	Conf	$\frac{tp}{tp+fp}$
Unusualness (Carmona et al., 2018)	WRAcc	$\frac{tp+fp}{tp+fp+tn+fn}\left(\frac{tp}{tp+fp}-\frac{tp+fn}{tp+fp+tn+fn}\right)$
Growth Rate (Dong & Li, 1999)	GR	$\frac{tp(fp+tn)}{fp(tp+fn)}$
Support Difference (Carmona et al., 2018)	SuppDiff	$\frac{tp}{tp+fn}-\frac{fp}{fp+tn}$
True Positive Rate (Kloesgen, 1996)	TPR	$\frac{tp}{tp+fn}$
False Positive Rate (Gamberger & Lavrac, 2002)	FPR	$\frac{fp}{fp+tn}$

According to the previous statements, the proposed algorithm, called CE3P-MDS, is an EFSs which learns/updates its KB whenever necessary by means of the extraction of fuzzy EPs using the arriving data from several data stream sources. CE3P-MDS is able to fulfil these requirements thanks to:

1. A procedure for a smart triggering of the learning method for saving computational resources. In this way, the learning method is only executed when necessary according to the requirements of the expert.
2. A learning method based on a multi-objective cellular evolutionary algorithm which contains specific operators towards the extraction of descriptive EPs.
3. An approach based on Apache Kafka and Apache Spark Streaming that provides an efficient, scalable, fault-tolerant data ingestion from multiple sources together with a distributed computing environment based on batch processing of streaming data.

As mentioned in Section 2.3, the extraction of descriptive extraction of descriptive EPs can be considered a multi-objective problem. For this reason, the proposed method is an EFS based on a multi-objective cellular evolutionary algorithm (MOCell) (Nebro et al., 2009). The employment of this kind of method is due to its great diversity-exploitation trade-off that allows us to find a diverse set of patterns that maximises data coverage. Besides, the employment of fuzzy logic provides both an improvement in the readability of the results and in the robustness against slight changes, thus making it more suitable for heterogeneous data stream mining environments. It also employs specific operators towards the extraction of EPs. In particular, a novel coverage-ratio-based procedure is proposed as a mechanism to reduce

the redundancy of the patterns extracted. Finally, the evaluation strategy follows a test-then-train approach for the continuous appraisal of the model with respect to the incoming data. This is efficiently carried out by means of the Bit-LUT procedure (García-Vico, Charte, González, Elizondo, & Carmona, 2020) which allows us to perform a scalable, distributed, exact computation of the quality of the model regardless the number of computation nodes employed.

The main elements of the proposed method are detailed below. Firstly, the memory component is presented in Section 3.1. After that, the learning adaptation process is shown in Section 3.2. Next, the evolutionary learning method and its main components are summarised in Section 3.3. Finally, all these elements are linked in the operational scheme presented in Section 3.4.

3.1. Memory component

It is assumed that data is ingested by the proposed algorithm by means of batches of an undefined size. This is because the Apache Spark Streaming engine works by collecting all the arriving data within a time window. CE3P-MDS is also able to consume data from multiple sources $S = \{s_1, \ldots, s_n\}$ by using Apache Kafka for this purpose. To do that, data sources must publish their data into a specific Kafka topic. These data are joined within a Kafka consumer subscribed to such topic for the creation of an instance e, as shown in Algorithm 1. After that, it is send to Spark Streaming where the data block $\mathscr{D} = \{e_1, \ldots, e_n\}$ is created in order to be processed by CE3P-MDS after that. This process is illustrated in Fig. 1.

Algorithm 1 Kafka consumer of CE3P-MDS.

Input:
$S = \{s_1, \ldots, s_n\}$: a set of data generated from different sources subscribed to the topic.
Output:
e: an instance of the problem.
1: $e \leftarrow \emptyset$
2: **for all** $s \in S_i$ **do**
3: $e \leftarrow e \cup s$
4: **end for**
5: **return** e

3.2. Learning adaptation process

The adaptation of the model to the current state of the stream is managed by a change monitoring system, thus following an informed strategy. This component will monitor both real and virtual drifts. In this way, the model is only updated when its performance decreases significantly. Using this strategy, the model is more flexible to the requirements of the experts, as they can find the best trade-off between quality and performance according to their needs.

The proposed change monitoring system is based on the idea that an EP is a description that summarises the current state of the stream. Thus, it is a representative object of the data. In this way, if data change, the quality of the description made by the EP also changes. Therefore, supervised indicators can be employed for detecting drifts. According to Khamassi et al. (2018), a real concept drift is characterised by a change in $P(C|\mathscr{P})$, whereas a virtual drift is defined as a change in $P(\mathscr{P}|C)$. These probabilities correspond to the confidence and the recall of the EPs, i.e, $P\left(C\middle|\mathscr{P}\right) = \frac{tp}{tp+fp}$ and $P\left(\mathscr{P}\middle|C\right) = \frac{tp}{tp+fp+tn+fn}$ (García-Borroto, Loyola-González, Martínez-Trinidad, & Carrasco-Ochoa, 2017). Therefore, the change monitoring system is in charge of the detection of variations in the confidence and recall of the current pattern model.

Fig. 1. CE3P-MDS data collection process based on Apache Kafka and Apache Spark Streaming.

Once the new data batch arrives, the current pattern model $\mathscr{M}$ is firstly evaluated using these data for the determination of its quality. Then, the change monitoring component will check whether the average confidence or recall of all the previously extracted EPs is below a given threshold. Note that these measures are related to the new data batch, so if one of these conditions is true, it is considered that a concept drift (real or virtual) is detected. Then the system triggers the learning process in order to update the current model using this data batch. Algorithm 2 presents the pseudo-code of this procedure.

Algorithm 2 Change monitoring process of CE3P-MDS.

Input:
$\mathscr{M} = \{\mathscr{P}_1, \ldots, \mathscr{P}_n\}$: The set of patterns previously extracted.
ρ_1, ρ_2: Confidence and support thresholds, respectively.
Output:
true if the learning method must be triggered, **false** otherwise.
1: $c \leftarrow \emptyset$
2: $s \leftarrow \emptyset$
3: evaluate M against the arriving data batch
4: **for all** $\mathscr{P}_i \in \mathscr{M}$ **do**
5: calculate $Conf(\mathscr{P}_i)$ and add it to c
6: calculate $Support(\mathscr{P}_i)$ and add it to s
7: **end for**
8: **if** $\bar{c} < \rho_1$ **or** $\bar{s} < \rho_2$ **then**
9: **return true**
10: **end if**
11: **return false**

3.3. Multi-objective evolutionary learning process

In EPM, patterns extracted should be able to accurately describe as many instances as possible. These patterns should also have a high generalisation capacity. In this way, the patterns extracted in one batch can continue to be valid in subsequent batches, improving the efficiency of the algorithm. In addition, it is also required that the method converges to a solution quickly due to the time constraints imposed by data stream mining. Therefore, a good diversity and a fast convergence in the search process are required for the extraction of EPs in data streams. According to these constraints, the proposed learning method is a MOCell-based algorithm (Nebro et al., 2009) which contains specific adaptations to the EPM problem.

The selection of this kind of algorithm is motivated by several reasons. Firstly, the exploration capability can be tuned by means of the grid-to-neighbourhood ratio (Alba & Dorronsoro, 2005), where a smaller ratio improves exploration. Secondly, a better exploration capacity has been shown with respect to other state-of-the-art methods such as NSGA-II (Deb, Pratap, Agrawal, & Meyarivan, 2002) or SPEA2 (Zitzler, Laumanns, & Thiele, 2002) in Nebro et al. (2009). Finally, in Salto and Alba (2019) a faster convergence to the optimal solution of cellular-based methods than traditional evolutionary ones has been recently demonstrated. The good features of MOCell have led to its successful application to many real-world problems in recent works. For example, in energy (Guzek, Pecero, Dorronsoro, & Bouvry, 2014; Luna, Luque-Baena, Martinez, Valenzuela-Valdés, & Padilla, 2018), delivery routing (Osaba et al., 2018), community detection (Pedemonte, Panizo-LLedot, Bello-Orgaz, & Camacho, 2020), amongst others (Kar, Kar, Guo, Li, & Majumder, 2019; García-Hernández et al., 2019; Talaslioglu, 2021). In these works, MOCell has been compared against other state-of-the-art algorithms such as NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2002), IBEA (Zitzler & Künzli, 2004), MOEA/D (Li & Zhang, 2009), AbYSS (Nebro et al., 2008), NSGA-III (Deb & Jain, 2014), or MOMBI2 (Hernández & Coello Coello, 2015). The best results have been obtained by MOCell. According to this, this algorithm is a competitive alternative whose features are interesting for addressing the constraints imposed in data stream mining.

MOCell intensively uses the concept of neighbourhood. The algo-

rithm arranges the population in an *nxn* grid configuration where an individual p_{ij} in the grid only interacts with its closest neighbours in the evolutionary process. An example can be observed in Fig. 2, where a one-hop distance neighbourhood, also known as *Compact9* or *C9* (Sarma & De Jong, 1996), is employed. It is important to remark that the concept of distance employed is related to the adjacency of the individuals in the grid. For example, in Fig. 2, the neighbourhood of the central individual ($p_{3,3}$) is highlighted in red. Additionally, individual $p_{2,2}$ and its neighbourhood is remarked in cyan. These individuals will only interact with its neighbourhood. It is important to remark that the overlapping of neighbourhoods allows the transmission of information throughout the population as their neighbourhoods share several individuals.

Algorithm 3 MOCell general procedure.

Input:
p_s: Population size
p_c,p_m: Crossover and mutation probabilities respectively
Output:
A: a set of chromosomes representing the non-dominated solutions 1: $A \leftarrow \emptyset$
{archive to store the Pareto front.}
2: $Pop \leftarrow$ InitialisePopulation(p_s) {Arranges the population in a grid}
3: **while not** StoppingCriteria **do**
4: **for all** $p \in Pop$ **do**
5: list $\leftarrow$ GetNeighbours(p) {See Fig. 2 for an example}
6: parents $\leftarrow$ Selection(list)
7: offspring $\leftarrow$ Crossover(parents, p_c)
8: offspring $\leftarrow$ Mutation(offspring, p_m)
9: $p \leftarrow$ replace(p, offspring) {Keep best according to non-dominance criteria}
10: $A \leftarrow A \cup$ offspring {Non-dominated or crowding-based criteria}
11: **end for**
12: $Pop \leftarrow$ Feedback(Pop,A) {replace random $p \in Pop$ with a random $p \in A$}
13: **end while**
14: **return** A

The general procedure of MOCell is depicted in Algorithm 3. The key components are shown at lines 5–9, where an offspring is created with respect to the current individual being processed and its neighbourhood. This can also be shown in Fig. 2. In this way, diversification is induced throughout the number of different neighbourhoods and how they are defined, whereas exploitation takes place in each one of them. Therefore, the use of this kind of algorithm provides us a decentralised evolutionary model which usually produces a better sampling of solutions across the search space. This will help us to improve the descriptions extracted as the different neighbourhoods could cover all the interesting aspects of data. In the following subsections, the specific operators and components employed for the extraction of high-descriptive EPs are presented.

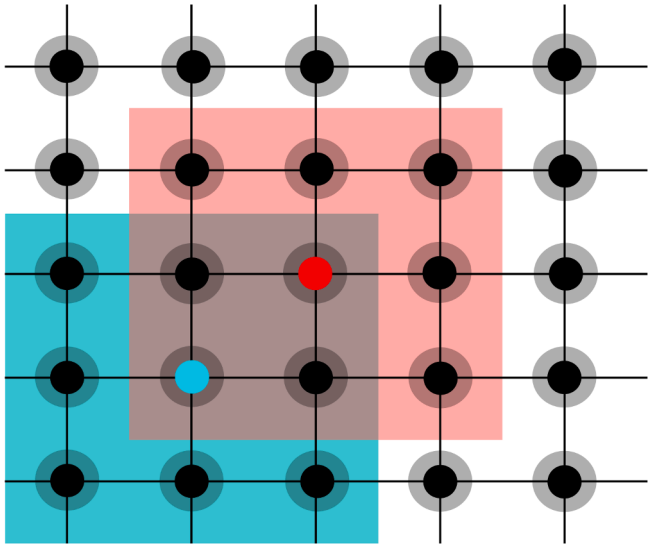

Fig. 2. Example of a population grid of size 5 × 5 in MOCell using a Compact9 neighbourhood.

3.3.1. Chromosome codification

As mentioned previously, the proposed evolutionary process is in charge of learning the KB, which is composed by the EPs that describe the behaviour of the stream. Therefore, a "chromosome = rule" approach (Wong & Leung, 2000) whereby each individual represents a potential EP to be extracted, is a suitable codification in this problem. These EPs follow a DNF representation (García-Vico et al., 2018). They are composed by an antecedent which joins items belonging to the same variable by disjunctions, whereas different variables are grouped by conjunctions. On the other hand, the consequent part represents the class the EP is describing. It is important to remark that an item is a fuzzy set defined in the DB by the expert beforehand. These fuzzy sets can also be defined for categorical variables by means of a step-wise membership function whose value is one if the input value belongs to the category or zero otherwise.

Taking all the previous premises into account, a chromosome p is codified as $p = (\lambda, c)$, where $\lambda \in \{0,1\}^l$ represents the antecedent part of the pattern as a binary vector whose length is equal to l, the number of itemsets of the problem, and $c \in \mathbb{N}$ is the class of the pattern. The number of itemsets, l, is determined by the amount of categories for each nominal variable plus the number of fuzzy sets defined for each numeric one. An example of this codification is shown in Fig. 3.

3.3.2. Genetic operators

In this subsection, the different operators employed within the evolutionary process are detailed. Firstly, the initial population must be created. This process must consider the adaptive nature of the proposed method in order to properly update the current EPs model with respect to the data stream. Therefore, the proposed initialisation firstly introduces the pattern model $\mathcal{M}$ in order to update it. Then, the remaining chromosomes until the population size is reached are randomly initialised.

Next, the evolutionary process begins. In this phase, each chromosome only interacts with its neighbourhood. These are defined following the Compact9 or C9 scheme, as shown in Fig. 2 (Sarma & De Jong, 1996). Firstly, the classic binary tournament selection (Miller & Goldberg, 1995) is applied, where two parents are chosen from the neighbourhood according to a dominance-based criterion. Then, a two-point crossover operator (Holland, 1975) is applied in order to intensify the search process in the area between the two parents. Finally, an EPM-oriented mutation operator (García-Vico et al., 2018) is applied in order to improve generality of the patterns extracted by means of randomly removing a whole variable or flipping a bit of the chromosome, both applied with equal probability.

One of the characteristics of an EP model is that it should be able to describe the maximum instance space avoiding redundancies as they unnecessary increase the complexity of the model. A reinitialisation procedure is applied for achieving this aim. It is triggered when the current population is not able to cover new uncovered examples for at

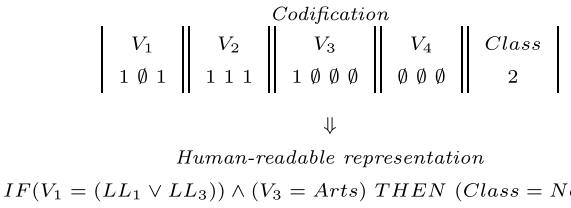

Fig. 3. Representation of a fuzzy DNF pattern in CE3P-MDS within a problem with numeric and categorical variables. V_1, V_2 and V_4 are numeric variables with three fuzzy sets defined for each one, whereas V_3 is a nominal variable with four different categories.

least a 25% of the total evaluations, meaning that the population is stagnated. After that, redundancies are removed from this candidate set of solutions by means of a coverage-ratio-based procedure (Li et al., 2014). This method works as follows: firstly, it computes the set of highly-overlapped chromosomes O_{Pop}, which are candidates to be removed. It is defined as in Eq. 2.

$$O_{Pop} = \left\{ (p_i, p_j) \middle| p_i, p_j \in Pop, i \neq j, \frac{TPs(p_i) \cap TPs(p_j)}{TPs(p_i) \cup TPs(p_j)} \geqslant \alpha \right\} \quad (2)$$

where $TPs(p_i)$ is the set of correctly covered instances by the chromosome p_i and $\alpha \in [0,1]$. In this work, $\alpha = 0.9$ as we are looking for highly-overlapped patterns. Next, for each pair $(p_i, p_j) \in O_{Pop}$, an odds-ratio interval $OddsRatio(p)$ is computed for p_i and p_j which determines its discriminative power. This interval is defined as shown in Eq. 3 (Li et al., 2014):

$$OddsRatio\left(p\right) = \left[\frac{tp * tn}{fp * fn} exp\left(-\omega\right), \frac{tp * tn}{fp * fn} exp\left(\omega\right)\right] \quad (3)$$

where $\omega = Z_{\alpha/2}\sqrt{\frac{1}{tp} + \frac{1}{fp} + \frac{1}{tn} + \frac{1}{fn}}$, and $Z_{\alpha/2} = 1.96$ is the value for a 95% confidence interval. After that, if both intervals overlap, then the chromosome with the lowest number of variables is kept, as the difference regarding to the discriminative power between them are not significant. Otherwise, the one with the highest $OddsRatio(p)$ is kept as its discriminative power is significantly higher than the other one, while the number of correctly covered instances are very similar.

Once the procedure is done, the surviving chromosomes will update the archive A, as shown in Algorithm 3, line 10, by means of a non-dominance criterion. After that, the remaining individuals until the population size is reached are created by means of a coverage-based procedure introduced in a subgroup discovery algorithm (Carmona, González, del Jesus, & Herrera, 2010). The idea is to generate new individuals able to cover instances not previously covered by other chromosomes. In this way, the method is able to explore new areas of the search space while reducing the redundancy of patterns extracted. Finally, it is important to remark that the proposed coverage-ratio-based procedure will also be applied at the end of the evolutionary process.

3.3.3. MapReduce evaluation function

As presented previously, the EPM problem is multi-objective. Therefore, several objectives should be defined for the determination of the quality of the individuals. These objectives are defined as different quality measures ϕ_k for EPM, such as the ones shown in Table 2. Thus the objective values are computed in the proposed method as in Eq. 4.

$$Obj_k(\mathscr{P}, \mathscr{D}) = \phi_k(\tau(\mathscr{P}, \mathscr{D})) \quad (4)$$

where ϕ_k is the value of the quality measure for the objective k in pattern $\mathscr{P}$, with respect to the current batch of data $\mathscr{D}$ and τ is the function that computes the contingency table associated to $\mathscr{P}$.

The main bottleneck in Eq. 4 is the computation of $\tau(\mathscr{P}, \mathscr{D})$. This function requires to traverse the whole batch in order to calculate the contingency table associated to $\mathscr{P}$. This means that whole batch must be traversed each time a chromosome needs to be evaluated. This is unfeasible in massive data stream scenarios. For this purpose, $\tau(\mathscr{P}, \mathscr{D})$ is efficiently calculated by means of the Bit-LUT procedure (García-Vico et al., 2020) in order to perform a distributed, fast computation of each contingency table associated. Basically, the procedure is based on the premise that the definitions of the fuzzy sets in the DB do not change in each chunk of data, as the evolutionary process is in charge of learning the KB. Thus, the membership value $\mu_i(e_i)$ is fixed for each fuzzy set defined in the DB. Bit-LUT takes advantage of this fact by using a function $f : \mathscr{D} \times DB \rightarrow \{0,1\}$ that determines whether the instances of $\mathscr{D}$ are covered or not by the different fuzzy sets defined in the DB. In particular, this is determined as in Eq. 5.

$$f\left(e_i, \mu_i\right) = \begin{cases} 1, & IF \mu_i(e_i) > 0 \\ 0, & otherwise \end{cases} \quad (5)$$

Eq. 5 is computed for all $e_i \in D$ and for all $\mu_i \in DB$. This is computed only once before starting the evolutionary process. Then, posterior evaluations of $\tau(\mathscr{P}, \mathscr{D})$ are performed by means of bitwise operations between the different variables that conforms the pattern. The main advantage of this method is that Eq. 5 can be easily computed in a distributed fashion. Besides, the computation of the contingency table is also parallelised by distributing the results of Eq. 5 across the computation cluster.

3.4. Operational scheme

A graphical representation of the main working scheme of CE3P-MDS is proposed in Fig. 4. Additionally, the pseudo-code of CE3P-MDS is presented in Algorithm 4 in order to ease the understanding of the method.

Algorithm 4 Operational scheme of the CE3P-MDS algorithm.

Input:
$\mathscr{D}$: A batch of data collected from the stream (See efalg:kafka).
$\mathscr{M}_{t-1}$: Previous pattern model.
ρ_1, ρ_2: Confidence and support thresholds, respectively, for the change monitoring system.
ρ_3: Maximum number of generations to run the evolutionary process.
Output:
$\mathscr{M}_t$: A set of patterns for the current batch.

1: *learn* ← false
2: **if** $\mathscr{M}_{t-1} \neq \emptyset$ **then**
3: Test $\mathscr{M}_{t-1}$ using $\mathscr{D}$.
4: *learn*ChangeMonitoring($\mathscr{M}_{t-1}, \rho_1, \rho_2$) {See Algorithm 2.}
5: **else**
6: *learn* ← true
7: **end if**
8: **if** *learn* **is** true **then**
9: g0
10: $A\emptyset$
11: P_gInitialisation($\mathscr{M}_{t-1}$) {Include previous model. See Section 3.3.2}
12: Evaluate(P_g) {Bit-LUT MapReduce evaluation. See Section 3.3.3.}
13: **while** $g < \rho_3$
14: P_{g+1} ← MOCell evolutionary step. See Algorithm 3 for details.
15: F ← DominanceSorting(P_{g+1}) {Get dominance fronts.}
16: **if** F_0 does not evolve **then**
17: P_{g+1} ← Reinitialisation(F_0) {Coverage Ratio procedure. See Section 3.3.2}
18: **end if**
19: $g \leftarrow g + 1$
20: **end while**
21: $\mathscr{M}$ ← CoverageRatioFilter(A)

(*continued on next page*)

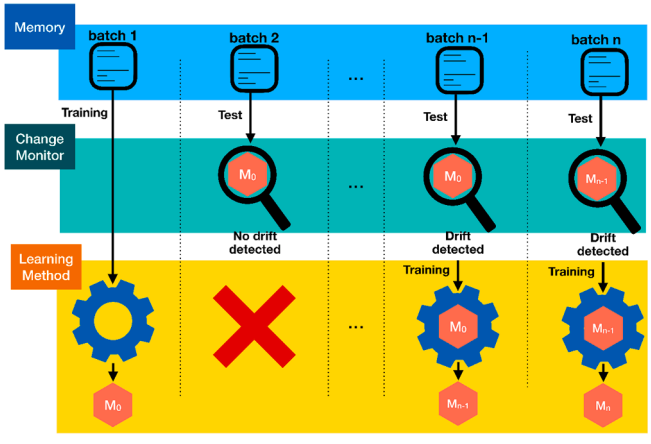

Fig. 4. General working schema of the CE3P-MDS algorithm.

(*continued*)

22: **return** $\mathscr{M}$
23: **else**
24: **return** $\mathscr{M}_{t-1}$ {The previous model is still valid.}
25: **end if**

Once $\mathscr{D}$ is collected from the memory, the method tests the previous model $\mathscr{M}_{t-1}$ using $\mathscr{D}$ following the test-then-train approach (line 3). Next, the change monitoring system will check whether the average confidence or support of the already tested $\mathscr{M}_{t-1}$ is below the given thresholds for the detection of real or virtual drifts, respectively (line 4). If this happens, then a drift is detected and a signal is sent to the learning method. Otherwise, the algorithm returns the current model and it waits for the next data batch (line 24).

If the learning signal is active, the evolutionary learning process starts by including $\mathscr{M}_{t-1}$ in the initial population for its update (line 11). After that, it is evaluated by means of Eq. 4 using the Bit-LUT procedure (Eq. 5). It is important to remark that data is currently distributed throughout the cluster thanks to Apache Spark. Next, the evolutionary process begins until a maximum number of generations is reached (line 13). Then, the population of the next generation P_{g+1} is created by means of applying for each individual the MOCell evolutionary algorithm process (line 14). In this step, it is highlighted that the chromosome evaluation process is also computed in a distributed, exact fashion. Finally, once P_{g+1} is complete, the feedback procedure is triggered in order to replace a random individual in P_{g+1} with a random one of the archive A as an elitism procedure. Next, the reinitialisation criterion is checked and applied if necessary (lines 16–18) where the proposed coverage ratio-based procedure is applied. Finally, this filtering method is always executed at the end of the evolutionary process for removing any redundancy (line 21). This result is returned to the expert and it is considered as the new current model.

4. Experimental study

In this section an experimental study is carried out for the determination of the quality of CE3P-MDS. In Section 4.1 the experimental framework is shown. Next, the study has three main objectives that are analysed: firstly, the quality of the knowledge extracted is shown and compared against other evolutionary approach for the extraction of EPs in data streams, the FEPDS algorithm (García-Vico et al., 2020) in Section 4.2. This comparison is made because, to the best of our knowledge, the FEPDS algorithm is the first approach focused on the extraction of EPs in data streams. After that, the adaptation to the concept drift of the proposed method is analysed in Section 4.3. Finally, the scalability of the approach is depicted in Section 4.4.

4.1. Experimental framework

The details of the experimental study carried out are shown in this section. For the source code of the proposed algorithm and further information for the reproducibility of this study, please refer to our GitHub repository[1]. Both real and artificial datasets are employed. Artificial datasets were generated by means of the MOA software (Bifet, Holmes, Kirkby, & Pfahringer, 2010), whereas real datasets were collected from the UCI repository (Dheeru & Karra Taniskidou, 2017). The advantage of using artificial datasets is that the concept drift can be fully controlled, so it can be easier to determine the adaptation capacity of the algorithms. All the artificial datasets contain ten concept drifts, each every one million instances. Two versions of the artificial datasets are employed: one with gradual drifts, and the other one with abrupt drifts. The characteristics of the datasets employed are depicted in Table 3.

[1] https://github.com/ari-dasci/S-CE3P-MDS

Table 3
Datasets employed in the experimental study.

Dataset	Type	#Inst.	#Attrs.	#Cl.
Aggrawal	Artificial	10,000,000	9	2
Hyperplane	Artificial	10,000,000	10	2
Mixed	Artificial	10,000,000	4	2
RandomRBF	Artificial	10,000,000	10	2
SEA	Artificial	10,000,000	3	2
RandomTree	Artificial	10,000,000	10	2
AWS-Stock-Price	Real	27,500,000	7	3
Hepmass	Real	8,400,000	27	2
Higgs	Real	8,800,000	28	2
Susy	Real	4,000,000	18	2

These static datasets were transformed into a stream of data. They were split into equal-size batches of 25,000, 50,000, 100,000, 150,000 and 200,000 instances, respectively. The idea is to reproduce a real massive data stream environment, where thousands of instances arrive in the system, in order to to determine the descriptive capacities of the proposed method as data grows.

Finally, it is important to remark that the size of the batches in CE3P-MDS could vary due to its time-window-based creation that depends on Apache Spark Streaming. This could negatively affect in the comparison made. Therefore, the execution strategy for the proposed method is as follows: first, the method is launched using a very high data collection time (10 s). After that, a Kafka producer is launched. This producer will only send the required amount of instances every batch, so Spark will only capture the required amount of instances. In this way, although some discrepancies in the number of instances per batch could happen, the gap is minimised.

The configuration of the parameters employed in this study for each algorithm is depicted in Table 4. It is important to remark that parameters of both methods has been set as similar as possible in order to avoid a biased result due to the parameters configuration.

The experiments were performed on a computation cluster composed of four machines with two Intel® Xeon® CPU E5-2603 v4 @1.70 GHz processors with 64 GB of RAM each, connected through an Infiniband network. The software employed were Hadoop 2.8, Spark 2.4.4 and Java version 1.8.131 over CentOS 7. In this study, CE3P-MDS has been compiled using Scala 2.10.6.

4.2. Quality analysis

Table 5 and Table 6 present the average results extracted, and the average execution time, on the datasets analysed for different batch sizes, respectively. In particular,

Table 5 presents relevant quality measures in the field of EPM such as the average number of patterns (#Patt.) and variables (#Vars.) for the determination of the interpretability of the model; WRAcc for the interest; TPR determines the generality; and GR the percentage of EPs extracted. Finally, CONF and FPR measures the reliability with respect to the local context of the pattern and the whole data batch, respectively. The measures FPR, number of patterns and number of variables must be minimised. The best result for each batch size on each column is highlighted in bold.

In data stream mining, a good trade-off between the quality of the

Table 4
Parameters of the algorithms employed in the experimental study.

Algorithm	Parameters
FEPDS	numLLs = 3, nGen = 60, popSize = 50, cProb = 0.6, mProb = 0.1, queueSize = 5
CE3P-MDS	numLLs = 3, nEvals = 3,500, gridSize = 7 × 7, cProb = 0.6, mProb = 0.1, partitions = 2

Table 5
Average results of the analysed methods using different batch sizes. Best result per batch size and quality measure is highlighted in bold.

Batch Size	Algorithm	#Patt.	#Vars.	WRAcc	CONF	GR	TPR	FPR
25000	FEPDS	**2.08**	4.65	**0.64**	**0.66**	**1.00**	**0.56**	**0.27**
	CE3P-MDS	2.54	**1.96**	0.60	0.62	0.91	0.55	0.36
50000	FEPDS	**2.08**	4.59	**0.64**	**0.67**	**0.99**	0.55	**0.27**
	CE3P-MDS	2.66	**1.99**	0.60	0.63	0.90	**0.56**	0.35
100000	FEPDS	**2.08**	4.61	**0.63**	**0.67**	**0.99**	**0.56**	**0.29**
	CE3P-MDS	2.76	**2.03**	0.60	0.63	0.89	0.54	0.34
150000	FEPDS	**2.09**	4.54	**0.63**	**0.67**	**0.99**	0.55	**0.29**
	CE3P-MDS	2.85	**2.02**	0.60	0.63	0.89	**0.57**	0.37
200000	FEPDS	**2.08**	4.56	**0.63**	**0.66**	**0.99**	**0.55**	**0.28**
	CE3P-MDS	2.67	**2.00**	0.60	0.62	0.88	0.51	0.33

Table 6
Average execution time of the analysed methods using different batch sizes. Best result per batch size is highlighted in bold.

Batch Size	Algorithm	Exec. time (s)
25,000	FEPDS	1.042
	CE3P-MDS	**0.251**
50,000	FEPDS	1.579
	CE3P-MDS	**0.340**
100,000	FEPDS	2.693
	CE3P-MDS	**0.519**
150,000	FEPDS	3.955
	CE3P-MDS	**0.985**
200,000	FEPDS	5.517
	CE3P-MDS	**1.055**

knowledge extracted and the execution time of the learning algorithm is key. In addition to these constraints, in EPM the extracted patterns are required to be very easy to read and interpret. This allows us to perform faster decision-making processes which is key in data stream mining. According to these points, several conclusions can be extracted from these experiments:

1. In average, the model extracted in CE3P-MDS is easier to understand than FEPDS as the average number of variables is significantly reduced. In fact, it is reduced up to 56%. Although there is an increase in the number of patterns of up to 25%, the reduction in the number of variables compensates for this increase. This is key in data stream mining as knowledge should be quickly analysed by the experts.
2. The quality of the results obtained by CE3P-MDS is slightly lower than FEPDS. This can be produced due to the employment of different adaptation strategies on both methods. On the one hand, FEPDS uses a blind strategy so it is continuously adapting its knowledge as it prioritises quality over execution time. On the other hand, CE3P-MDS employs an informed strategy based on a concept drift detection mechanism, so the knowledge is adapted only when necessary. This should be considered in this comparison, because the quality of the knowledge in the proposed algorithm could slightly decay due to variations in the stream that do not trigger the learning process. Although these variations could negatively affect the average value, it is highlighted that the difference between both methods is not very high. This means that the proposed algorithm is able to find those EPs whose variables have the most significant discriminative power. This is because the significant reduction in the number of variables does not imply a significant reduction in the quality. This could be produced due to the employment of the new coverage ratio filtering process which removes those high-overlapped patterns with the lowest reliability and a high number of variables.
3. Finally, it is highlighted that the execution time of CE3P-MDS is reduced up to 5 times with respect to FEPDS, as shown in Table 6. This reduction is mainly produced by the proposed change monitoring system, which avoids the execution of the learning process when it is not necessary. In this way, the probability of queueing the arriving data is lower than FEPDS when the batch size is significantly increased. Therefore, the proposed method is more suitable for massive data stream scenarios, as it is able to process a higher amount of data in the same amount of time.

According to these results, the proposed algorithm achieves a higher trade-off between the quality of the knowledge extracted and its execution time. Therefore, CE3P-MDS is a promising alternative that allows researchers and practitioners the extraction of high-descriptive patterns within massive data streams without compromising the stability of the system.

4.3. Concept drift analysis

In data stream mining, the determination of the reaction capacity to concept drift is key. Fig. 5 and Fig. 6 illustrate the average confidence of the patterns extracted on each batch for each method on the analysed datasets. It is important to remark that only artificial datasets are shown as the concept drift in real datasets is unknown. Also, virtual concept drift cannot be controlled in MOA.

Fig. 5 represents abrupt concept drifts, whereas Fig. 6 shows gradual concept drifts. Here, the difference between these executions is only the type of change carried out. The authors remind the reader that in abrupt drifts the new concept suddenly introduces its new instances to the stream. On the other hand, in gradual drifts both concepts gradually interchange their probability of generating instances for a period of time. In this case, this period is 10,000 instances.

From these results, it can be observed that the performance decreases significantly after the first drift in all the analysed datasets. After that, both algorithms are able to quickly recover from the drift. However, it is important to remark that both methods present a very similar behaviour, in spite of the proposed method contains a simpler knowledge. This reinforce the fact that the method is able to capture the most discriminative variables thanks to the coverage ratio filtering process. In the majority of cases, the knowledge extracted by the proposed method remains stable regardless of the concept drift. This means that the knowledge extracted is very robust. This fact can be produced due to the evolutionary learning method employed. In particular, the employment of specific operators within the evolutionary process focused on the extraction of general patterns. Moreover, the use of fuzzy logic provides us an additional layer of robustness as it is able to properly handle the slight differences in data between batches without affecting its quality.

On the other hand, the proposed change monitoring mechanism is able to properly handle the concept drift. This is clearly observed in Fig. 5a, where the method reacts just after the drift happens, increasing its quality. Another aspect that must be taken into consideration is that thanks to the employment of the change monitoring system the method is only executed on those classes that require an improvement due to concept drift. In this way, the execution time of the proposed method is, in average, several times faster than FEPDS, so the computational

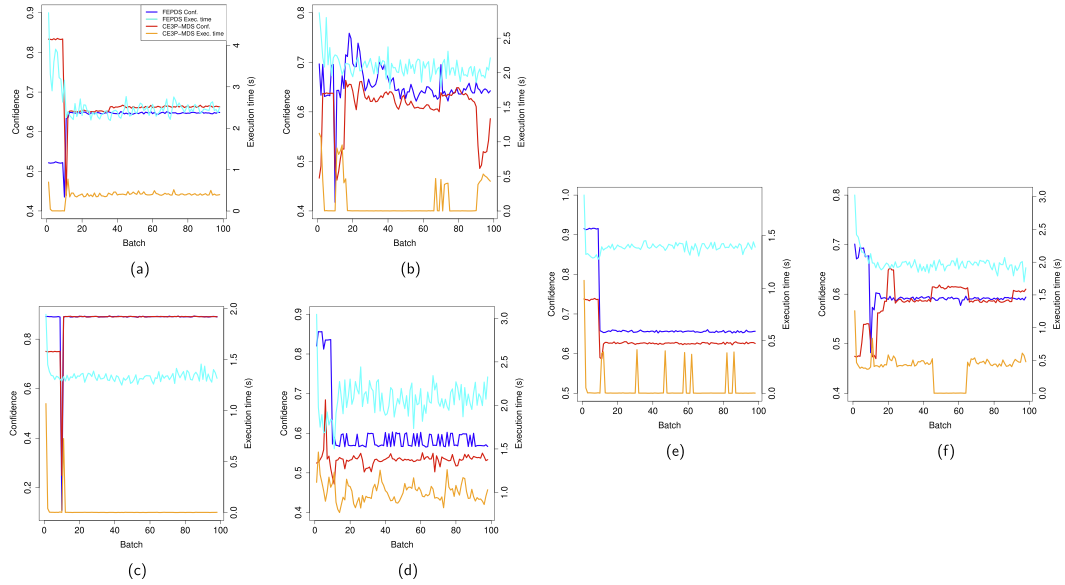

Fig. 5. Average confidence (y-axis, left-hand side) and execution time (y-axis, right-hand side) of the methods analysed on different datasets with concept drift. Batch size = 100,000. Drifts every 10 batches. (a) Aggrawal, (b) Hyperplane, (c) Mixed, (d) RandomRBF. Abrupt Drifts. Average confidence (y-axis, left-hand side) and execution time (y-axis, right-hand side) of the methods analysed on different datasets with concept drift. Batch size = 100,000. Drifts every 10 batches. (e) SEA, (f) Random Tree. Abrupt Drifts.

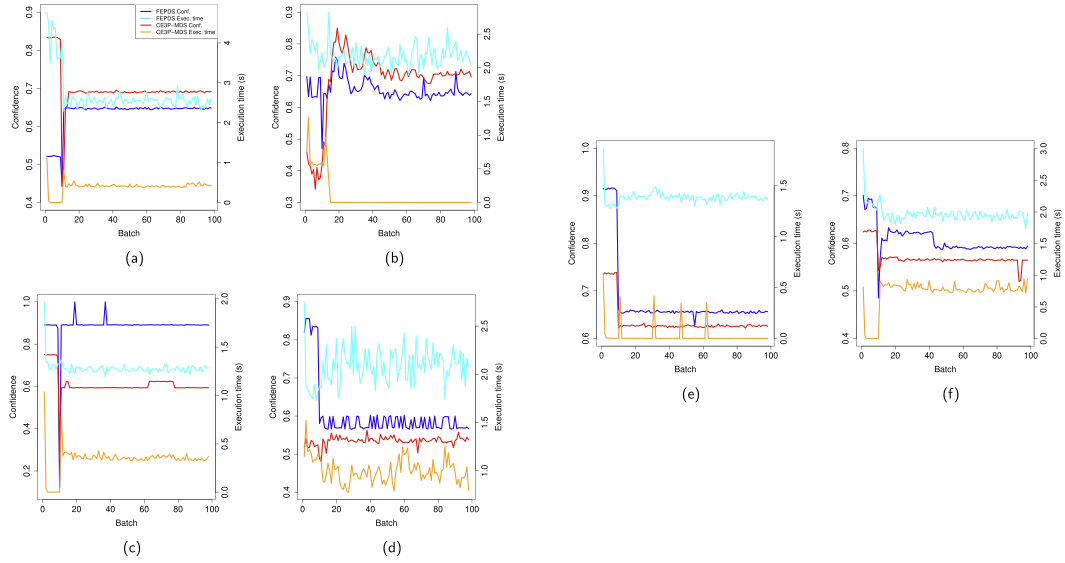

Fig. 6. Average confidence (y-axis, left-hand side) and execution time (y-axis, right-hand side) of the methods analysed on different datasets with concept drift. Batch size = 100,000. Drifts every 10 batches. (a) Aggrawal, (b) Hyperplane, (c) Mixed, (d) RandomRBF. Gradual Drifts. Average confidence (y-axis, left-hand side) and execution time (y-axis, right-hand side) of the methods analysed on different datasets with concept drift. Batch size = 100,000. Drifts every 10 batches. (e) SEA, (f) Random Tree. Gradual Drifts.

resources are better employed as unnecessary computations are avoided. Therefore, it could be stated that the proposed method presents a good trade-off between its adaptation capacity to concept drift and its computational cost.

4.4. Scalability analysis

In data stream mining it is key that the average execution time of the method is below the arriving rate of the instances in order to not compromise the system stability. In this section, the computational complexity of the proposed algorithm is theoretically calculated. After that, it is experimentally determined the maximum instance rate of the proposed method and its scalability with respect to the number of partitions employed in Apache Spark. This is performed by incrementally increasing the batch size with a fixed amount of partitions. On the other hand, the number of partitions are increased while the batch size remains fixed in order to compute the total execution time of the method.

The computational complexity of an algorithm is governed by its most complex component in the worst case scenario. Here, we are going to determine the complexity of the different components of the proposed algorithm:

- The memory component presented in Algorithm 1 has a cost of $O(N)$ where N is the total amount of instances received.

- When a new data batch arrives, the Bit-LUT procedure (García-Vico et al., 2020) is carried out to speed-up the remaining processes. It has a complexity of $O(SN)$ where S is the number of selectors of the problem.
- The proposed change monitoring process in Algorithm 2 has a computational cost of $O(IS)$ using Bit-LUT, where I is the number of patterns in $\mathscr{M}$.
- Regarding the genetic operators presented at Section 3.3.2, the initialisation, crossover and mutation have a complexity of $O(S)$ for each chromosome. The coverage-ratio-based procedure is governed by Eq. 2 which is $O\left(\frac{I(I-1)}{2}\right)$.
- The MOCell step presented in Algorithm 3 has a complexity of $O(IS)$.

According to these elements, the entire evolutionary process has a linear complexity of $O(SN)$ due to the Bit-LUT pre-calculation step. However, it allows the speed-up of posterior processes and it can be easily parallelised by means of MapReduce.

Regarding the experimental study, the dataset employed is the *RandomTree* with real concept drifts every one million instances. The use of this dataset is due to it has the same number of numeric and categorical variables, so the type of variable does not bias the results. In this analysis, a limit of four seconds is established as descriptive applications such as EPM needs to be analysed by humans in real time.

Table 7 presents the average execution time for each batch with respect to different batch sizes. In average, it can be observed that the method can deal with very high rates within a reasonable time limit. This fact is due to the change monitoring system as it is able to avoid unnecessary executions of the learning method. In the worst case scenario, it could be unfeasible the processing of batches with one million instances as its execution time is very high. Therefore, according to the results extracted it can be stated that the proposed method can accept an instance rate up to 750,000 instances per batch without compromising the system stability in the dataset analysed.

Fig. 7 presents the scalability of the approach with respect to the number of partitions employed within Apache Spark. It is important to remark that the total time taken by the algorithm on the processing of 80 data batches of 750,000 instances each is represented in the y-axis. From these results, it can be observed that the total execution time is significantly reduced when the number of partitions increases. However, it is highlighted that the time increases when using the maximum number of partitions as it surpasses the physical resources of our computation cluster. Therefore, the proposed algorithm is able to properly scale up whenever necessary to perform a faster computation.

5. Conclusion

In this paper, an EFS for the extraction of EPs in massive data streams have been presented. To the best of our knowledge, the CE3P-MDS algorithm is the first EPM method focused on the processing of massive, heterogeneous, high-speed data streams. The main aim of the proposed method is to describe or monitor the current state of the data stream with respect to a variable of interest. This is carried out by an informed strategy based on a change monitoring system for finding a good trade-off between efficiency and quality in the knowledge extracted. CE3P-MDS also employs a multi-objective EFS based on a cellular evolutionary algorithm as learning method. The employment of a multi-objective approach allows us the extraction of knowledge with a good trade-off between simplicity, generality and reliability. On the other hand, the use of a cellular evolutionary method provides us a decentralised model that produce a better sample of solutions throughout the search space which improves data coverage. In addition, a coverage-ratio-based procedure is proposed for reducing the redundancy of the patterns extracted. This process is also applied within the evolutionary method as a reinitialisation procedure. Finally, Apache Kafka, together with Apache Spark have been employed in order to ingest data from different sources for processing them in a distributed fashion.

The experimental analysis carried out shows that the proposed algorithm extracts knowledge with a similar quality than other non-distributed EPM approach for data stream mining, whereas the interpretability of the knowledge extracted is significantly improved. This fact is very useful for the experts as the knowledge can be analysed faster without loosing quality in the description performed. Additionally, the proposed informed approach improves the computational cost, so the resources are better employed. This makes the proposed approach more suitable for deployment in lifelong environments as the stability of the system is improved with respect to other approaches. In general, the proposed method is an interesting approach for the extraction of EPs within massive amounts of data that constantly arrives in the form of data streams. This opens an interesting research line in the development of new methodologies for the extraction of EPs in this kind of environments. In this way, different aspects can be improved by means of further research, such as new monitoring systems, new data ingestion policies, or the development of new learning methods based on decentralised methodologies for improving scalability, amongst others.

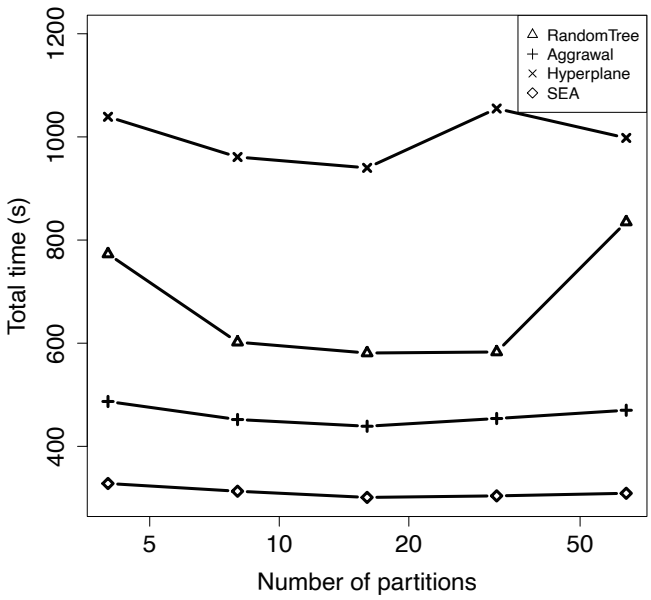

Fig. 7. Scalability analysis for different data streams by increasing the number of partitions (x-axis, log-scale). Total execution time in seconds is represented on the y-axis.

Table 7
Average execution time of the proposed method with different batch sizes on the *RandomTree* data stream. (48 partitions, 80 batches analysed.)

Batch Size	Exec. time (s)
250,000	0.072 ± 0.436
500,000	1.653 ± 1.445
750,000	4.015 ± 0.799
1,000,000	3.581 ± 2.744

CRediT authorship contribution statement

Ángel M. García-Vico: Writing - original draft, Conceptualization, Methodology, Software. **Cristóbal Carmona:** Methodology, Writing - review & editing, Supervision. **Pedro González:** Writing - review & editing, Supervision. **María J. del Jesus:** Writing - review & editing, Supervision, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Spanish Ministry of Economy and Competitiveness under the project PID2019-107793GB-I00 and by the Regional Government of Andalusia, program "Personal Investigador Doctor", reference DOC_00235.

References

Alba, E., & Dorronsoro, B. (2005). The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE transactions on evolutionary computation, 9*, 126–142.

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

Brzeziński, D. (2015). Block-based and online ensembles for concept-drifting data streams (Ph.D. thesis). Poznan University of Technology.

Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 36*.

Carmona, C. J., del Jesus, M. J., & Herrera, F. (2018). A unifying analysis for the supervised descriptive rule discovery via the weighted relative accuracy. *Knowledge-Based Systems, 139*, 89–100.

Carmona, C. J., González, P., del Jesus, M. J., & Herrera, F. (2010). NMEEF-SD: Non-dominated multi-objective evolutionary algorithm for extracting fuzzy rules in subgroup discovery. *IEEE Transactions on Fuzzy Systems, 18*, 958–970.

CERN (2021). Storage at cern. URL: https://home.cern/science/computing/storage. Accessed: 2021-04-15.

Cheng, J., Ke, Y., & Ng, W. (2008). Maintaining frequent closed itemsets over a sliding window. *Journal of Intelligent Information Systems, 31*, 191–215.

Cisco (2021). Cisco annual internet report (2018-2023) white paper. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html. Accessed: 2021-04-15.

Dean, J., & Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *Operating Systems Design and Implementation (OSDI)* (pp. 137–150).

Deb, K., & Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation, 18*, 577–601.

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

de Mello, R. F., Vaz, Y., Grossi, C. H., & Bifet, A. (2019). On learning guarantees to unsupervised concept drift detection on data streams. *Expert Systems with Applications, 117*, 90–102.

Demšar, J., & Bosnić, Z. (2018). Detecting concept drift in data streams using model explanation. *Expert Systems with Applications, 92*, 546–559.

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

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

Farzanyar, Z., Kangavari, M., & Cercone, N. (2012). Max-fism: Mining (recently) maximal frequent itemsets over data streams using the sliding window model. *Computers & Mathematics with Applications, 64*, 1706–1718.

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

Fernández, A., Herrera, F., Cordón, O., del Jesus, M., & Marcelloni, F. (2019). Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to? *IEEE Computational Intelligence Magazine, 14*, 69–81.

Fernandez, A., Lopez, V., del Jesus, M. J., & Herrera, F. (2015). Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges. *Knowledge-Based Systems, 80*, 109–121.

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

Foundation, A. S. (2021). Apache storm. URL: https://storm.apache.org/. Accessed: 2021-04-15.

Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*, 44:1–44:37.

Gamberger, D., & Lavrac, N. (2002). Expert-guided subgroup discovery: Methodology and application. *Journal Artificial Intelligence Research, 17*, 501–527.

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

García-Hernández, L. E., Tchernykh, A., Miranda-López, V., Babenko, M., Avetisyan, A., Rivera-Rodriguez, R., Radchenko, G., Barrios-Hernandez, C. J., Castro, H., & Drozdov, A. Y. (2019). Multi-objective configuration of a secured distributed cloud data storage. In *Latin American High Performance Computing Conference* (pp. 78–93).

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

García-Vico, A., Carmona, C. J., González, P., & del Jesus, M. J. (2020). Fepds: A proposal for the extraction of fuzzy emerging patterns in data streams. *IEEE Transactions on Fuzzy Systems, 28*, 3193–3203.

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

García-Vico, A., Charte, F., González, P., Elizondo, D., & Carmona, C. J. (2020). E2pamea: A fast evolutionary algorithm for extracting fuzzy emerging patterns in big data environments. *Neurocomputing, 415*, 60–73.

García-Vico, A. M., Montes, J., Aguilera, J., Carmona, C. J., & del Jesus, M. J. (2016). 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* (pp. 1–8). San Sebastián, Spain: Springer.

Garg, N. (2013). *Apache Kafka*. Packt Publishing Ltd.

Guzek, M., Pecero, J. E., Dorronsoro, B., & Bouvry, P. (2014). Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems. *Applied Soft Computing, 24*, 432–446.

Hernández Gómez, R., & Coello Coello, C. A. (2015). Improved metaheuristic based on the r2 indicator for many-objective optimization. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation* (pp. 679–686).

Herrera, F., Carmona, C. J., González, P., & del Jesus, M. J. (2011). An overview on Subgroup Discovery: Foundations and Applications. *Knowledge and Information Systems, 29*, 495–525.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems* (2nd ed.). University of Michigan Press.

Huynh, V. Q. P., & Küng, J. (2020). Fpo tree and dp3 algorithm for distributed parallel frequent itemsets mining. *Expert Systems with Applications, 140*, Article 112874.

Kar, M. B., Kar, S., Guo, S., Li, X., & Majumder, S. (2019). A new bi-objective fuzzy portfolio selection model and its solution through evolutionary algorithms. *Soft Computing, 23*, 4367–4381.

Khamassi, I., & Sayed Mouchaweh, M. (2014). Drift detection and monitoring in non-stationary environments. In *2014 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS 2014 June 2–4, 2014 Linz, Austria* (pp. 1–6).

Khamassi, I., Sayed Mouchaweh, M., Hammami, M., & Ghédira, K. (2018). Discussion and review on evolving data streams and concept drift adapting. *Evolving Systems, 9*, 1–23.

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

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

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion, 37*, 132–156.

Li, G., Law, R., Vu, H. Q., Rong, J., & Zhao, X. R. (2015). Identifying emerging hotel preferences using emerging pattern mining technique. *Tourism management, 46*, 311–321.

Li, J., Liu, J., Toivonene, H., Satou, K., Sun, Y., & Sun, B. (2014). Discovering statistically non-redundant subgroups. *Knowledge-Based Systems, 67*, 315–327.

Li, H.-F., Shan, M.-K., & Lee, S.-Y. (2008). Dsm-fi: an efficient algorithm for mining frequent itemsets in data streams. *Knowledge and Information Systems, 17*, 79–97.

Li, H., & Zhang, Q. (2009). Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on evolutionary computation, 13*, 284–302.

Li, H., Zhang, N., Zhu, J., Wang, Y., & Cao, H. (2018). Probabilistic frequent itemset mining over uncertain data streams. *Expert Systems with Applications, 112*, 274–287.

Lughofer, E. (2016). Evolving fuzzy systems fundamentals, reliability, interpretability, useability, applications. In Handbook on Computational Intelligence: Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems (pp. 67–135). World Scientific.

Luna, F., Luque-Baena, R. M., Martinez, J., Valenzuela-Valdés, J. F., & Padilla, P. (2018). Addressing the 5g cell switch-off problem with a multi-objective cellular genetic algorithm. In *2018 IEEE 5G World Forum (5GWF)* (pp. 422–426).

Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies, 7*, 1–13.

Mata, J., Alvarez, J.-L., & Riquelme, J.-C. (2002). Discovering numeric association rules via evolutionary algorithm. In *Advances in Knowledge Discovery and Data Mining* (pp. 40–51).

Mayer-Schonberger, V., & Cukier, K. (2013). *Big data: the essential guide to work, life and learning in the age of insight*. Hachette UK.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., et al. (2015). Mllib: Machine learning in apache spark. arXiv:1505.06807.

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

Miller, R. J., & Yang, Y. (1997). Association rules over interval data. *ACM SIGMOD Record, 26*, 452–461.

Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., & Alba, E. (2009). Mocell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems, 24*, 726–746.

Nebro, A. J., Luna, F., Alba, E., Dorronsoro, B., Durillo, J. J., & Beham, A. (2008). Abyss: Adapting scatter search to multiobjective optimization. *IEEE Transactions on Evolutionary Computation, 12*, 439–457.

Nord, J. H., Koohang, A., & Paliszkiewicz, J. (2019). The internet of things: Review and theoretical framework. *Expert Systems with Applications, 133*, 97–108.

Orriols-Puig, A., Casillas, J., & Bernadó-Mansilla, E. (2008). First approach toward on-line evolution of association rules with learning classifier systems. In *Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation* (pp. 2031–2038).

Osaba, E., Del Ser, J., Nebro, A. J., Laña, I., Bilbao, M. N., & Sanchez-Medina, J. J. (2018). Multi-objective optimization of bike routes for last-mile package delivery with drop-offs. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (pp. 865–870).

Park, J., Lee, H., & Park, J. (2010). Real-time Diagnosis System Using Incremental Emerging Pattern Mining. In *Proc. of the 5th International Conference on Ubiquitous Information Technologies and Applications* (pp. 1–5).

Pedemonte, M., Panizo-LLedot, Á., Bello-Orgaz, G., & Camacho, D. (2020). Exploring multi-objective cellular genetic algorithms in community detection problems. In *International Conference on Intelligent Data Engineering and Automated Learning* (pp. 223–235).

Peng, M., Ouyang, S., Zhu, J., Huang, J., Wang, H., & Yong, J. (2018). Emerging topic detection from microblog streams based on emerging pattern mining. In *Proc. of the IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 259–264). China: Nanjing.

Piao, M., Lee, H. G., Sohn, G. Y., Pok, G., & Ryu, K. H. (2009). Emerging patterns based methodology for prediction of patients with myocardial ischemia. In *Proc. of the 6th International Conference on Fuzzy Systems and Knowledge Discovery* (pp. 174–178). IEEE.

Poezevara, G., Lozano, S., Cuissart, B., Bureau, R., Bureau, P., Croixmarie, V., Vayer, P., & Lepailleur, A. (2017). A computational selection of metabolite biomarkers using emerging pattern mining: A case study in human hepatocellular carcinoma. *Journal of proteome research, 16*, 2240–2249.

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

Ramírez-Gallego, S., Krawczyk, B., García, S., Wozniak, M., & Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing, 239*, 39–57.

Rehman, Z., Shahbaz, M., Shaheen, M., & Guergachi, A. (2015). Fps-tree algorithm to find top-k closed itemsets in data streams. Arabian Journal for Science and Engineering, 40, 3507–3521.

Ruiz, E., & Casillas, J. (2018). Adaptive fuzzy partitions for evolving association rules in big data stream. *International Journal of Approximate Reasoning, 93*, 463–486.

Saleti, S., & Subramanyam, R. B. V. (2019). A mapreduce solution for incremental mining of sequential patterns from big data. Expert Systems with Applications 133, 109 – 125.

Salto, C., & Alba, E. (2019). Cellular genetic algorithms: Understanding the behavior of using neighborhoods. *Applied Artificial Intelligence, 33*, 863–880.

Sarma, J., & De Jong, K. (1996). An analysis of the effects of neighborhood size and shape on local selection algorithms. In *International Conference on Parallel Problem Solving From Nature* (pp. 236–244). Springer.

Sayuri-Iwashita, A., & Papa, J. (2019). An overview on concept dripft learning. *IEEE Access, 7*, 1532–1547.

Sezer, O. B., Dogdu, E., & Ozbayoglu, A. M. (2017). Context-aware computing, learning, and big data in internet of things: A survey. *IEEE Internet of Things Journal, 5*, 1–27.

Shaker, A., & Lughofer, E. (2014). Self-adaptive and local strategies for a smooth treatment of drifts in data streams. *Evolving Systems, 5*, 239–257.

Sherhod, R., Gillet, V. J., Hanser, T., Judson, P. N., & Vessey, J. D. (2013). Toxicological knowledge discovery by mining emerging patterns from toxicity data. *Journal of Chemical Information and Modeling, 5*, 9.

Sherhod, R., Gillet, V. J., Judson, P. N., & Vessey, J. D. (2012). Automating knowledge discovery for toxicity prediction using jumping emerging pattern mining. *Journal of Chemical Information and Modeling, 52*, 3074–3087.

Škrjanc, I., Iglesias, J. A., Sanchis, A., Leite, D., Lughofer, E., & Gomide, F. (2019). Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey. *Information Sciences, 490*, 344–368.

Srikant, R., & Agrawal, R. (1996). Mining quantitative association rules in large relational tables. *SIGMOD Record, 25*, 1–12.

Talaslioglu, T. (2021). A comparative study of multi-objective evolutionary metaheuristics for lattice girder design optimization. *Structural Engineering and Mechanics, 77*, 417–439.

Toubakh, H., & Sayed-Mouchaweh, M. (2015). Hybrid dynamic data-driven approach for drift-like fault detection in wind turbines. *Evolving Systems, 6*, 115–129.

Tzanis, G., Kavakiotis, I., & Vlahavas, I. P. (2011). Polya-iep: A data mining method for the effective prediction of polyadenylation sites. *Expert Systems with Applications, 38*, 12398–12408.

Wald, A. (1973). Sequential analysis. *Courier Corporation*.

Wang, E. T., & Chen, A. L. (2011). Mining frequent itemsets over distributed data streams by continuously maintaining a global synopsis. *Data Mining and Knowledge Discovery, 23*, 252–299.

Wang, L., Zhao, H., Dong, G., & Li, J. (2004). On the complexity of finding emerging patterns. In Proc. of the 28th Annual International Computer Software and Applications Conference (pp. 126–129). Vol. 2.

Wang, K., Tay, S. H. W., & Liu, B. (1998). Interestingness-based interval merger for numeric association rules. *KDD, 98*, 121–128.

Webb, G. I., Lee, L. K., Goethals, B., & Petitjean, F. (2018). Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery, 32*, 1179–1199.

Wong, M. L., & Leung, K. S. (2000). *Data Mining using Grammar Based Genetic Programming and Applications*. Norwell, MA, USA: Kluwer Academics Publishers.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M., Shenker, S., & Stoica, I. (2012). 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* (p. 2).

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing* (p. 10).

Zitzler, E., & Künzli, S. (2004). Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature* (pp. 832–842). Springer.

Zitzler, E., Laumanns, M., & Thiele, L. (2002). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *International Congress on Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems* (pp. 95–100).