IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 September 2022, accepted 8 November 2022, date of publication 18 November 2022,
date of current version 28 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3223472

RESEARCH ARTICLE

Three-Way Decisions on Streaming Computing Platforms Supporting Decision-Making in Complex Large Real-World Environments

GRAZIANO FUCCIO, VINCENZO LOIA, (Senior Member, IEEE), AND FRANCESCO ORCIUOLI, (Member, IEEE)
Dipartimento di Scienze Aziendali–Management and Innovation Systems, Università degli Studi di Salerno, 84084 Fisciano (SA), Italy
Corresponding author: Francesco Orciuoli (forciuoli@unisa.it)

ABSTRACT In complex environments, decision-making processes are more and more dependent on gathering, processing and analysing huge amounts of data, often produced with different velocities and different formats by distributed sensors (human or automatic). In this contexts, data arrive in streams and often suffer of imprecision. Moreover, Three-way Decision is considered a suitable and convenient approach for imprecise data analysis based on the tri-partitioning of the universe of discourse, i.e., exploiting the notions of acceptance, rejection and non-commitment, as well as the human brain solves numerous problems. On the other hand, Three-Way Decision needs to be implemented according to the stream computing paradigm in order to support the analysis of data streams. With such a paradigm, data arrive, are processed and depart in real-time without needing to be temporarily serialized into a storage system. The paper analyses aspects related to implementation and architectural issues of the Three-Way Decision (based on probability-based Rough Set Theory) approach on a real-time data processing platform supporting streaming computing, i.e., Apache Spark. The paper shows the benefits of deploying Three-Way Decision on Spark and the possibility to apply it to different and heterogeneous scenarios.

INDEX TERMS Three-way decisions, rough set theory, data streaming, distributed streaming computing.

I. INTRODUCTION

Internet of Things (IoT) enables a platform of sensor and actuator devices [1] to communicate seamlessly within real-world environments. Such environments, with the help of IoT, are able to continuously produce data that, once captured, offer a great chance to be processed and analysed in order to support decision-making processes related to the aforementioned environments. Therefore, IoT becomes an important source of contextual data with an enormous volume (size of data), variety (different types of data from several sources) and velocity (data collected in real-time), which represent three of the main fundamental characteristics of Big Data (the other two are veracity and value, respectively related to uncertainty and imprecision of data and benefits provided by data for the organizations using them) [2]. From the application scenario perspective, the fusion of Big Data and IoT technologies has created opportunities for the development of decision-making services for many complex systems like, for instance, Smart Cities [3], Smart Grids, Intelligent Transportation [4], etc. Moreover, from the technological viewpoint, a plethora of distributed computing frameworks, for dealing with Big Data, are available: Apache Hadoop,[1] Apache Spark,[2] Apache Storm,[3] Apache Kafka,[4] etc. In particular, Apache Spark is a data processing framework that can quickly perform processing tasks on very large datasets, and can also distribute data processing tasks across multiple

The associate editor coordinating the review of this manuscript and approving it for publication was Zhe Xiao.

[1] https://hadoop.apache.org/
[2] https://spark.apache.org/
[3] https://storm.apache.org/
[4] https://kafka.apache.org/

machines. Apache Spark allows design, implementation and deploy of a wide range of data engineering and data science custom pipelines.

A. THREE-WAY DECISIONS

Three-Way Decision (3WD) is a theory that models a particular class of human ways of problem-solving and information processing [5]. It is built on solid cognitive foundations and offers cognitive advantages [6] for the human operator when adopted as a formal model behind a decision support system. The basic idea of 3WD is to partition the universe of discourse (available data describing objects that are crucial for making the right decisions in the considered environments) into three pair-wise disjoint regions (tri-partitioning) and act upon each region through a suitable strategy or course of actions. Although the two-way model (i.e., a binary classifier) provides high levels of acceptance and rejection, it does not support deferment. In contrast, the three-way model leads to moderate levels of acceptance, rejection and deferment. The advantage of the three-way model is that when costs of different types of error are considered, it generates less overall cost than the two-way model [7]. Thus, 3WD embraces all aspects of a decision-making process by including tasks such as data and evidence collection and analysis for supporting decision making, reasoning, and computing in order to arrive to a particular decision, and justification and explanation for such a decision [6]. Principles of 3WD are commonly used in our life. For example, *past-present-future* are used when dealing with temporally-described objects or, if the important dimension is the spatial one, people typically adopt the scheme *left-middle-right* or *top-middle-bottom*, etc. Moreover, other plausible schemes are *acceptance-non_commitment-rejection* (judgmentally), *small-medium-large* (volumetrically), and so on.

Three-way data analytics (3WDA) focus on techniques that use the principles of three-way decision. The basic idea of 3WDA is to consider a third option in the conclusions drawn from data and, of course, in the subsequent decision-making. For instance, if the task is to determine if an object x, in a given universe U, belongs to a specific class X, the possible answer of 3WDA can be one of the following three: i) x belongs to X certainly, ii) x does not belong to X certainly, and iii) it is not possible to affirm that x belongs to X certainly nor that x does not belong to X certainly. Therefore, the third option provides reliability in decision-making processes. If the information describing data is not sufficient or reliable enough for saying either yes or no to a decision, one may alternatively choose the third option that may reduce the risk or cost in the decision-making. Furthermore, in a dynamic setting (as in the majority of complex real-world environments) where additional data can be obtained gradually, the third option can be further refined as time goes on. Such refinement may lead to a definite answer of yes or no [8].

B. RESEARCH OBJECTIVES

According to the available scientific literature, the problem of 3WD for streams of data is essentially solved by providing incremental algorithms able to update, as time goes on, structures representing the three regions or, alternatively, lower and upper approximations when Rough Set Theory is adopted [9].

Although existing incremental algorithms offer good solutions in the case of centralized computation, they are not suitable for parallel and distributed computing within contexts in which big data (especially medium-sized or large-sized datasets) must be considered. These algorithms fail when the volume of data contained in a temporal window cannot be processed due to a memory overflow.

The main challenge faced by the paper is overcoming the limitations (e.g., impossibility to scale and distribute the computation, poor support for fault tolerance) of incremental algorithms for 3WD when applied to big data contexts. More concretely, the most important contribution of the present work is the definition of an implementation approach for 3WD-based decision support systems through the adoption of primitives of the state-of-the-art platforms for streaming computing. The approach is defined through: i) the analysis of the dynamic aspects of data within a real-world scenario, and ii) the design of adequate solutions to handle the aforementioned aspects by using streaming platform capabilities.

The objective of the work presented here is also to analyse, considering a real-world scenario including challenges related to big data (velocity, volume, veracity and variety), the implementation of 3WD through the usage of the *map* and *reduce* constructs provided by Apache Spark. Furthermore, such implementation offers several non-functional advantages like, for instance, the possibility to distribute and parallelize the operations and the chance to apply such operations to streaming data. In particular, distributed algorithms are capable of processing data by mapping them into different clusters of computation also providing redundancy mechanisms to prevent data loss without having data inconsistency [10].

C. PAPER ORGANIZATION

The remaining parts of the paper are organized as it follows. Section II describes the available scientific works related to the results produced by this paper and the advancements of the proposed work with respect to its objectives introduced in Section I. Moreover, Section III introduces the adopted computational methods for Three-Way Decision based on probability-based Rough Sets. Section IV offers a motivating scenario and a discussion on the dynamics of decision tables when dealing with data streams. Section V provides design information about the whole architecture in which the proposed approach has been deployed. Section VI describes the novel implementation of Three-Way Decision by using *map* and *reduce* constructs. Such implementation will be called streaming-based Three-Way Decision (S3WD).

Furthermore, Section VII describes the experimentation and evaluation activities. Lastly, Section VIII provides conclusions and introduces some ideas for future works.

II. RELATED WORKS

In this section, according to the context depicted in Section I, the focus is on those works dealing with the streaming computing paradigm, as described in [11] where the authors state that by means of such paradigm it is possible to perform both data storage and computing directly in memory. Data is processed as it arrives within sliding time windows. The difference between streaming computing and the traditional batch computing paradigm is the need, for the latter, for accumulating (on storage systems) abundant data before starting with computation. Note that, at the moment, it is not possible to recognize implementations of Three-Way Decision approach based on probability-based Rough Sets on streaming platforms like Apache Spark. Despite this difficulty that makes impossible fair comparisons with this work, it is important to provide, on one hand, knowledge on alternative solutions based on incremental algorithms for exploring the concepts related to dynamic and temporal aspects of information systems and, on the other hand, MapReduce implementations of Rough Sets in order to provide a fundamental brick for the exploitation of streaming computing platforms, like Apache Spark, which adopt similar programming models based on *map* and *reduce* constructs. Definitely, the advancement of the present work with respect to the state-of-the-art is the combination of solid cognitive support for decision-makers offered by 3WD with high levels of fault tolerance and scalability offered by Apache Spark. In other words, the proposed approach provides the advantages of incremental algorithms for 3WD and overcomes their limitations due to the impossibility to scale, distribute the computation and tolerate faults, which are all fundamental characteristics in big data streaming contexts.

A. APPROACHES BASED ON INCREMENTAL ALGORITHMS

The works based on incremental algorithms exploit approaches in which lower and upper approximations evolve over time [12] as new information arrives or old information disappears or is replaced. In particular, the authors of [13] propose a method to incrementally update approximations when objects change dynamically in the information systems considering the case of Variable Precision Rough Sets [14]. The aforementioned authors explore also the variation of the actual value and value set of an attribute when inserting or deleting an object into the universe of discourse. This paper, in some sense, extends the approach proposed in [15] in which the authors investigate methods for incremental updating approximations in terms of the coarsening and refining of the object set. A similar approach has been described in [16] where the approximations are calculated through the application of Decision-Theoretic Rough Sets, and in [17] where approximations are based on Neighborhood Rough Sets. Moreover, the authors of [18] deal with incremental learning of approximations in the context of incomplete information systems (missing data). Other approaches facing the problem of the evolution of attributes are described in [19], [20], and [21]. Moreover, approaches dealing with the evolution of attribute values are proposed in [22] and [23]. Interesting classifications of dynamic and temporal aspects in information tables are provided by [24] and [25].

B. SUITABLE APPROACHES FOR BIG DATA

First of all, it is important to clarify some aspects related to the term MapReduce. MapReduce (written as one term) is a programming model and an associated implementation for processing large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of *map* and *reduce* constructs (typical of functional programming paradigm), and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks. Hence, MapReduce provides simplicity for parallel programming, while at the same time offering load balancing and fault tolerance [26]. With respect to the scalable implementation of Rough Sets, the work of Zhang et al. [27] provides a parallel method for computing Rough Set approximations based on the MapReduce technique in order to deal with massive data. A further work [28] of the same authors compare the implementation of such approximation operators on three different platforms supporting the MapReduce paradigm (Hadoop, Twister and Phoenix). The results of an extensive experimental evaluation of different large datasets show that the proposed parallel method is effective for data mining. Moreover, the authors of [29] propose a parallel algorithm to compute rough approximations in large-scale datasets. This method is based on MapReduce for efficiently processing large-scale datasets with missing data both in condition and decision attributes. A further work [30] describes a new approach for outlier detection using fuzzy rough set theory. Other works dealing with parallelization of algorithms computing Rough Sets are focused on processing reducts [31], Big Data regression [32], attribute subset selection [33] and Apriori algorithm [34]. Lastly, the authors of [35] propose a rough set fuzzy classification rule generation algorithm (RS-FCRG) for big data through a MapReduce implementation.

Unfortunately, MapReduce (as-is) does not support stream processing, but it can handle streams using a technique known as micro-batching. The idea is to treat the stream as a sequence of small chunks of data. At small time intervals, the incoming stream is packed into a chunk of data and is delivered to the batch system to be processed. Some run-time systems, like Apache Spark, support such kind of streaming computation and offer programming constructs based on *map* and *reduce* for coding efficient streaming algorithms. In this case, the computation is completely in-memory [36]. Therefore, although the scientific literature provides many results on scalable implementations of Rough Set algorithms

through MapReduce [37], there is a lack of works focusing on analysis and definition of such algorithms for streaming computing (over one of the existing Big Data platforms enabling micro-batching). The present work aims at analysing the chance to develop these algorithms through the streaming capabilities of Apache Spark.

III. THREE-WAY DECISIONS AND PROBABILITY-BASED ROUGH SETS

Three-Way Decision (3WD) theory [6] models a particular class of human ways of problem solving and information processing. The basic idea of such theory is to divide a universal set (of objects) into three pair-wise disjoint regions, or more generally a whole into three distinctive parts, to handle complexity, and to act upon each region or part by developing strategies that are appropriate and possibly distinct. In 3WD the Universe of discourse (U) is partitioned into three regions (`RegionI`, `RegionII` and `RegionIII`) respecting the following constraints:

$$\begin{aligned}(RegionI) \cap (RegionII) &= \emptyset,\\ (RegionI) \cap (RegionIII) &= \emptyset,\\ (RegionII) \cap (RegionIII) &= \emptyset,\\ (RegionI) \cup (RegionII) \cup (RegionIII) &= U.\end{aligned} \tag{1}$$

On one side, 3WD is borrowed from the way the human brain tends to represent and solve problems. On the other side, 3WD can be adopted to support human decision-making when the problem is represented by a universe of objects. The aforementioned support is particularly useful for handling complexity due to the huge number of involved data.

Therefore, 3WD provides an overall approach able to sustain human cognitive processes, in general, when dealing with data and, in particular, when data must be processed to represent situations and reason on them in the context of tasks requiring good levels of situation awareness.

Evaluation-based Three-Way Decision [6] and Three-Way Decision based on Rough Sets [9] are two of the main classes of concrete approaches to implement 3WD.

The main limitation of the traditional Rough Set Theory is that it does not allow any uncertainty in the definition of both lower and upper approximations [38]. In order to introduce such tolerance, the probability approximation space was brought into the Rough Set Theory (see [39] and [40]) (i.e., probability-based Rough Sets). A widely adopted variant of probability-based Rough Sets is represented by the Decision-theoretic Rough Set Model [41] based on the notions of rough membership and rough inclusion that can be interpreted in terms of conditional probabilities. In particular, it is considered $P(X|[x])$, i.e., the probability that an object belongs to the concept X given that such object belongs to the equivalence class $[x]$. The conditional probability can be calculated as $P(X|[x]) = \frac{|X \cap [x]|}{|[x]|}$. Thus, the approximation operators, in the Decision-theoretic Rough Set Model, are defined as it follows:

$$\underline{B}(X) = \{x \in U : P(X|[x]_B) \geq \alpha\}, \tag{2}$$

$$\overline{B}(X) = \{x \in U : P(X|[x]_B) > \beta\}, \tag{3}$$

where α and β ($0 \leq \beta < \alpha \leq 1$) are probabilistic thresholds and establish the tolerance degree used to determine both lower and upper approximations. Equations (2) and (3) are defined by considering the Information System $IS = (U, A)$ and the indiscernibility relation I_B, where $B \subseteq A$:

$$I_B = \{(x, y) : x, y \in U \wedge \forall b \in B, b(x) = b(y)\}. \tag{4}$$

Once defined I_B, it is possible to create information granules starting from the available information (or knowledge) about the objects belonging to the universe of discourse. In particular, each granule is constructed as an equivalence class based on I_B:

$$[x]_B = \{y \in U : (x, y) \in I_B\}. \tag{5}$$

Moreover, the equivalence classes $[x]_B$ can be used to compute conditional probabilities in eq. (2) and (3).

Starting from the sets resulting from such equations, the tri-partitioning of the universe U with respect to concept $C \subseteq U$ has been constructed as it follows.

$$\begin{aligned}POS(C) &= \underline{B}(C),\\ BND(C) &= \overline{B}(C) - \underline{B}(C),\\ NEG(C) &= U - \overline{B}(C).\end{aligned} \tag{6}$$

or equivalently:

$$\begin{aligned}POS(C) &= \{x \in U : P(C|[x]_B) \geq \alpha\},\\ BND(C) &= \{x \in U : \beta < P(C|[x]_B) < \alpha\},\\ NEG(C) &= \{x \in U : P(C|[x]_B) \leq \beta\}.\end{aligned} \tag{7}$$

More in detail, the three regions $POS(C)$, $BND(C)$ and $NEG(C)$ respectively represent the sets of i) objects that certainly belong to C, ii) objects for which it is not possible to affirm that they belong to C nor that they do not belong to C, and iii) objects that certainly do not belong to C.

3WD seems to be particularly suited for representing contexts in which observed entities (objects) are classified with respect to their occurring situation (e.g., safe or dangerous in the maritime scenario introduced in Section IV-A).

IV. PRELIMINARY ANALYSIS

A. REAL-WORLD SCENARIO: MARITIME SURVEILLANCE

In the work [42] the authors define a decision support system based on probability-based Rough Sets and Three-Way Decision. The provided case study deals with the maritime domain that is typically represented by complex large environments. In such a domain, it is crucial to understand why some vessel movements take place and when such movements are symptoms of threats. A possible threat could arise when, for instance, the engine is broken or the vessel has been hijacked. A surveillance operator could identify dangerous situations and/or proceed to further investigations by only observing the environment and exploiting her experience. But, in the presence of a huge number of vessels (also of different types) within a large environment, the human operator needs to be

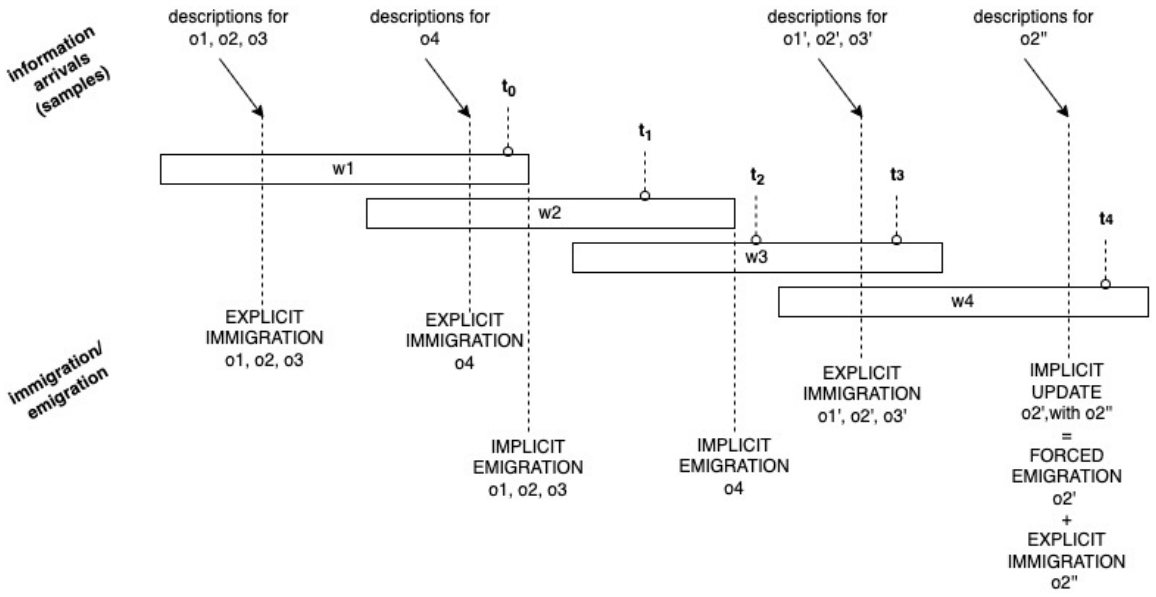

FIGURE 1. Example of object descriptions incoming within sliding windows.

aided by a decision support system that is essentially based on the application of Three-Way Decision for learning the concept of `SAFENESS` that is used to assess the situation in the monitored environment. More in detail, the universe of discourse consists of all monitored vessels that are described by means of a set of condition attributes (for the sake of simplicity, only `drift angle` and `velocity` will be considered) and a decision attribute whose value set is {`safe`, `dangerous`}. A decision table is constructed by observing the monitored vessels (condition attributes) and by applying simple heuristics (a vessel is said to be drifting when it is moving slowly, usually with a velocity between 3 and 5 knots, and its course and orientation have a significant difference, usually an angle of more than 30 degrees) to calculate the values for the decision attribute. Such a decision table is not free from imprecision caused by errors coming from both observations and adopted heuristics.

Therefore, it makes sense the adoption of the approximation operations of probability-based Rough Sets. The result of Three-Way Decision over the above decision table is the tri-partitioning of the whole set of vessels into three regions: i) safe vessels, ii) dangerous vessels, and iii) vessels needing further investigation. The work [42] assumes to start from a complete static decision table and faces the temporal evolution of object descriptions by considering a what-if analysis [43] (simulation) in which evolutionary scenarios are pre-constructed (manually by analysts/operators) and analysed. In the real world, the challenge is to deal with decision tables that evolve over time, i.e., they are constantly up-to-date and a 3WD analysis must take into account such dynamics.

Facing the aforementioned challenge is the aim of the present work that has the ambition of providing generalized solutions, valid in different and heterogeneous domains.

B. DYNAMICS IN DECISION TABLES

In the motivating scenario proposed in Section IV-A, as well as in a plethora of further real-world scenarios, the information related to an object (e.g., a vessel) could change as time goes on, thus old object descriptions are replaced by new descriptions that, of course, could impact upon the results of the analysis. Another important aspect is related to the obsolescence of information. In fact, old descriptions could be dropped in order to reduce the load of data to be analysed and, consequently, the latency. Therefore, it is important to provide an implementation of 3WD that takes into account such dynamics and is able to update the tri-partitioning accordingly. In streaming computing, this kind of dynamics is implemented by using a windowing approach. More in detail, time is divided into sliding windows of fixed length (window length) with a given offset (sliding length) between the starting point of two consecutive windows. Fig. 1 provides an example in which descriptions are updated with new ones, old descriptions become obsolete and new descriptions replace old ones. In terms of incremental algorithms, it is possible to extend the set of basic primitives consisting of *immigration* (a new object enters into the universe) and *emigration* (an existing object exits from the universe) with a third operation namely *update*.

In the proposed model, an *explicit immigration* operation is executed when a new description for an object is pushed into the system. The description of an object is valid along the window in which it is pushed into the system. When an object is no longer valid, an *implicit emigration* operation is executed. In the case that a new description for an object, still having a valid description, is pushed into the system then an *explicit update* operation is executed. Such operation is composed of a sequence of *forced emigration* operations (the description of an object that exits the system

prematurely - before the end of its window) and *explicit immigration* operation. More in detail, we have an overall static universe U including all the monitored objects and an evolving universe W where each element $w \in W$ is a description of an object $x \in U$ produced at a given time t. Additionally, it is needed to consider two functions: $\Delta : W \rightarrow T$ that maps a description $w \in W$ onto the specific timestamp in which such description is produced (T represents the set of timestamps) and $\Omega : W \rightarrow U$ that maps a description $w \in W$ onto the related observed object $x \in U$. At any time instant, the set W contains zero or one description for each object in U. The universe W is continuously modified as immigration, emigration and update operations are executed. The universe W is used to build a decision system (or decision table) $DS = \langle W, A \cup \{d\}, V \cup V_d \rangle$, where A is a set of condition attributes describing objects in U and d is the decision attribute for such objects. Moreover, $V = \bigcup_{a \in A} V_a$, $f : W \times A \rightarrow V$ and $g : W \rightarrow V_d$ are information functions such that $f(w, a) \in V_a$ and $g(w) \in V_d$ for each $w \in W$ and $a \in A$. Being W dynamic also DS is dynamic. Through functions f and g, observations come with all the information related to attributes $A \cup d$ that are needed to construct the decision table to be analysed. The changes to the universe W occur in sliding windows represented by possibly overlapping time intervals $s_0, s_1, \ldots$ belonging to S. Thus, function $\Gamma : W \rightarrow S$ is used to state in which window a given description arrived.

When the computation starts, let say, W^0 contains all the available descriptions of objects belonging to U. If a new description $\bar{u}$, with $\Omega(\bar{w}) = \bar{u}$, immigrates into the system, it is added to the decision table if $\nexists w \in W : \Omega(w) = \bar{u}, \Delta(w) \leq \Delta(\bar{w})$. In this case W evolves into W':

$$W' = W \cup \{\bar{u}\} \tag{8}$$

otherwise, the new observation replaces the older one:

$$W' = (W - u) \cup \{\bar{u}\} \tag{9}$$

Furthermore, assume that the end of the k-th window is reached, W is the universe soon before the end of such window and W' is the universe soon after the end of it, then W' is obtained by dropping from W all the descriptions arrived in the time interval s_k corresponding to the k-th window:

$$W' = W - \{u \in W | \Gamma(u) = s_k\} \tag{10}$$

In other terms, eq. (8) represents an *explicit immigration*, i.e., when a new description is generated and pushed into the system, eq. (10) represents an *implicit emigration*, i.e., when a subset of descriptions become obsolete, and, lastly, eq. (9) represents a *forced update*, i.e., when a recent description replaces an existing one (still valid) and the two are related to the same monitored object.

Let us discuss and better detail the example in Fig. 1. $W(t_0) = \{o_1, o_2, o_3, o_4\}$, $W(t_1) = \{o_4\}$, $W(t_2) = \emptyset$, $W(t_3) = \{o_1', o_2', o_3'\}$, and $W(t_4) = \{o_1', o_2'', o_3'\}$. In the above example, o_2'', o_2' and o_2 are all observations related to the same object. In particular, $\Omega(o_2'') = \Omega(o_2') = \Omega(o_2) = x_{o_2} \in U$ and $\Delta(o_2) \leq \Delta(o_2') \leq \Delta(o_2'')$.

The idea of the evolving universe is that the 3WD analysis can be executed on the current version of the universe that depends on the time instant. Thus executing a 3WD on streaming data means applying such analysis on the most recent available knowledge on the monitored environment.

V. OVERALL ARCHITECTURE

A network of wireless sensors (possibly adopting IoT protocols) is deployed into the monitored environment and produces a set of sensor data streams. For the sake of simplicity, it is assumed that observations into such streams are labelled with the identifier of the observed object belonging to the static universe of discourse (U in Section IV-B). In a given time interval the observations related to the same object arrive at a Kafka process and are integrated in order to generate a timestamped record including values associated with a fixed set of features describing the object. At the end of each interval, all the generated records (possibly related to different objects) are sent to a Spark application that manages them as a DStream and contributes to the evolving universe of discourse (W in Section IV-B), where the Three-Way Decision algorithm is continuously applied producing a time-evolving tri-partitioning. Such results could be sent to a time-aware dashboard to support the work of the decision-makers.

Fig. 2 shows records related to five object descriptions (step 1), produced and sent by Kafka to Spark that realizes the tri-partitioning of the evolving universe (step 2). Then (step 3) a new description for object o_2 arrives in the Spark application that replaces the old description (for the same object) and induces an evolved tri-partitioning (step 4) where o_2 moves from the boundary region to the negative region and, consequently, object o_4 moves from the boundary region to the positive region.

Let us now consider more details about the system architecture that is reported in Fig. 3. Such architecture is Kafka-centred in the sense that the *publish & subscribe* paradigm of Kafka is used to coordinate the work of different components. There are three components respectively based on Faust,[5] Apache Spark and Streamlit[6] deployed as both Kafka consumers and producers. In particular, sensor observations arrive to Kafka and are published under the `Topic A`. A Kafka consumer based on Faust is listening `Topic A` and reads all the incoming observations for processing and transforming them into full object descriptions (see Tab. 1) that are generated and attached to `Topic B` by a Kafka producer based on Faust. A new Kafka consumer based on Spark is listening on `Topic B` and is able to read all the incoming object descriptions. Such consumer can build a decision table with these descriptions. A new Kafka producer based on Spark processes the decision table and computes the three regions of 3WD as results attached to `Topic C`. Lastly, a new Kafka consumer based on Streamlit visualizes

[5]A stream processing library `https://faust.readthedocs.io/en/latest/`

[6]Web-based tool for data (streams) visualization `https://streamlit.io/`

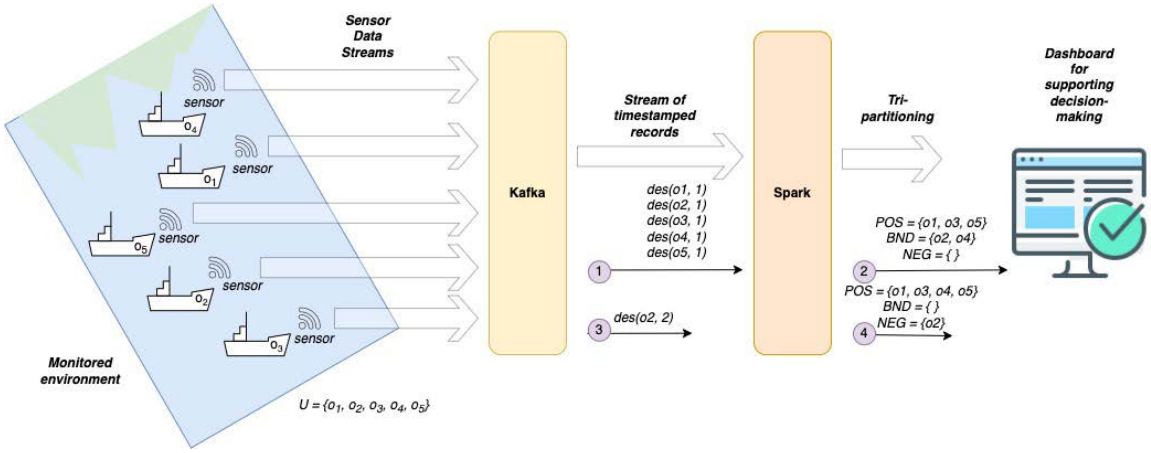

FIGURE 2. Overall view of the system.

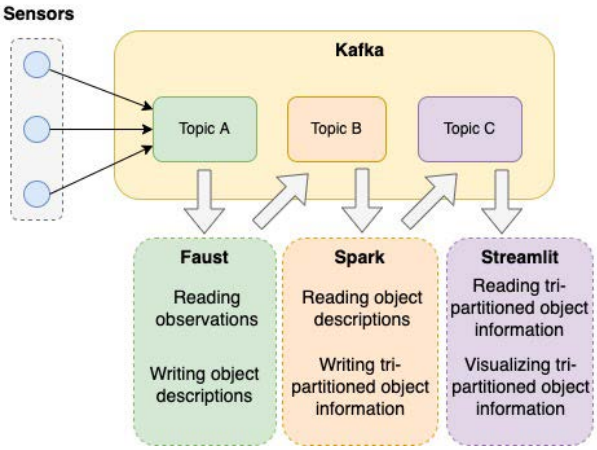

FIGURE 3. Kafka-centred system architecture.

data as generated by Spark. The above process is a continuous process able to wait for new observations and process them.

A. THE STREAMING COMPUTING PLATFORM: APACHE SPARK

Apache Spark is the most used framework for big data analytics thanks to its advanced in-memory programming model and upper-level libraries for scalable machine learning, graph analysis, streaming and structured data processing. The computation, within a Spark application, is managed by the key components provided by Spark in order to transparently distribute its tasks (smallest units of work composed of both data and operations) over cluster nodes that run processes in parallel [44]. In order to write data processing algorithms in a way that distributed and parallel computation is allowed, it is needed to leverage on the Spark data abstractions and the corresponding sets of operations.

1) RESILIENT DISTRIBUTED DATASET

The Spark core data abstraction is the Resilient Distributed Dataset (RDD) [45]. An RDD is a read-only, partitioned collection of records. RDDs provide fault-tolerant, parallel data structures that let users store data explicitly on disk or in memory, control their partitioning and manipulate them using a rich set of operations that are executed in parallel on each partition. Such tasks are executed on the cluster nodes managed by Spark and assigned to a specific Spark application [44].

2) TRANSFORMATIONS AND ACTIONS

There are two types of parallel operations that it is possible to execute on an RDD: transformations and actions [45]. Transformations are deterministic and lazy operations, used to apply a function that returns a new RDD without immediately computing it. With a narrow transformation (e.g., map, filter), each partition of the parent RDD is used by at most one partition of the child RDD. With a wide transformation (e.g., join, groupByKey), multiple child partitions may depend on the same partition of the parent RDD. Moreover, an action (e.g., count, first, take) launches a computation on an RDD and then returns the results to the driver program or writes them to an external storage. When used in pipelines, transformations produce a concrete result only when an action is called. In the presence of such pipelines, Spark can break the computation into tasks, organized into multiple stages, which can be executed in parallel over separate machines. Such stages are separated by distributed shuffle operations for redistributing data and improve the computation performance [44].

3) STREAMING CAPABILITIES

Spark Streaming adopts a micro-batch architecture [46] where a stream is treated as a sequence of small batches of data. In this case, the streaming computation is executed through a sequence of smaller computations of the aforementioned batches. The basic programming abstraction in Spark Streaming is the Discretized Streams (DStreams) [47]. DStream is a high-level abstraction representing a continuous stream of data as a sequence of small batches of data.

Internally, a DStream is a sequence of RDDs that can be processed using normal Spark jobs. As a result, DStreams have the same fault tolerance properties as those of RDDs and streaming data can be processed using Spark core and other upper-level rich libraries. The RDD abstraction itself is a convenient way to design a data computation process as a sequence of small and independent steps [45]. Transformations on DStreams can be stateless, i.e., normal RDD transformations, or stateful, i.e., based on sliding windows and on tracking state across time. Stateful transformations, which are relevant for the present work, use data or intermediate results from previous batches to compute the results of the current batch. In particular, the algorithm, proposed in the present work, is based on windowed transformations combining results from multiple batches and requires two parameters: window duration and sliding duration. The actual execution of DStream transformations is triggered by output operations (similar to RDD actions). Through these operations, it is possible to specify what should be done with the final results of a stream processing [44].

VI. IMPLEMENTING 3WD IN SPARK

3WD allows to define decision support systems in which human operators are supported, in their decision-making processes, by means of the tri-partitioning of the objects included in the universe of discourse. This support is especially needed when the universe is complex (composed of huge volumes of objects changing their characteristics as time goes on). Such systems do not provide automatic reasoning capabilities but offer solid cognitive support that increases the level of situation awareness of the decision-makers.

A. WINDOW-BASED PROCESSING APPROACH IN SPARK

In general, different application programming environments (e.g., Python, MapReduce, Hadoop MapReduce, Apache Spark, etc.) provide different implementations of the programming style based on *map* and *reduce* functions.

Typically, the function *map* transforms sequences of data from one type to another (the input sequence has the same length as the output one). The function *reduce* transforms a sequence of data into a data structure of any shape or size. One of the main advantages is that algorithms, written through custom combinations of *map* and *reduce*, make possible the definition of data transformation pipelines and their distribution over a cluster of nodes. Moreover, such organization enable parallelism of tasks during the computation. Other benefits regard, for instance, the chance to adopt lazy computation allowing better performance also in the case of traditional applications (no distribution, no parallelism). Thus, assume that descriptions of monitored objects arrive in Spark as records stored in RDDs. Spark operations are executed over the RDDs that are valid with respect to the windowing approach and its configuration (window length and sliding length). More in detail, Spark continuously executes, at time instants known as computation times, the processing algorithm on a set of data, whose composition depends on both the arrival time of such data and the windowing configuration. Fig. 4 provides a picture representing an example of how the set of data to be processed is composed by Spark along the timeline, according to the windowing approach and respecting the dynamics illustrated in Section IV-B.

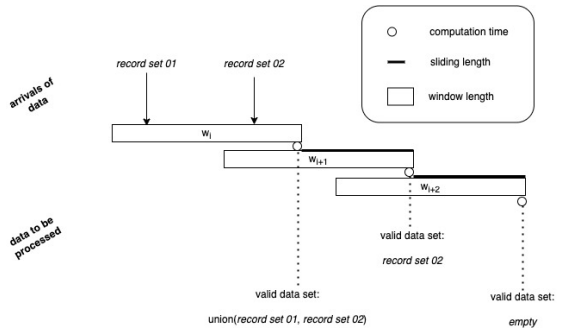

FIGURE 4. Composing the valid set of data to be processed by Spark.

In particular, Fig. 4 represents the cases in which the second burst of data is added to the valid set of data to be processed and persists in such a set also after the first burst is excluded from it. This event happens because the second burst arrives when windows w_i and w_{i+1} overlap. Moreover, after the end of w_{i+1} also the second burst is excluded from the valid set of data that, consequently, becomes empty. In general, Spark computes operations every X seconds (computation times), where X is the sliding length. At each computation point, Sparks applies operations over data arrived up to Y seconds before the computation time, where Y is the window length. Lastly, there is a third time parameter which is the batch time, i.e., the number of seconds between consecutive data gathering operations. Once described how the valid set of data to be processed is composed, let us explain how such a set is represented through the Spark data abstraction supporting streaming computing.

As shown in Fig. 5, the incoming records (e.g., r_1, r_2) are put in different RDDs (according to the arrival time). RDDs are embedded into a DStream. Only a subset of the generated RDDs includes records that are in the valid set of data to be processed. It is important to remember that records, in each RDD, are organized into partitions (e.g., p_1, p_2), hence enabling distribution and parallelism for Spark jobs execution.

B. STRUCTURING RECORDS FOR 3WD IN STREAMING

Conceptually, records are organized as iterable structures of arrays and grouped into RDDs which it is possible to apply Spark transformations and actions to. Each array has length N and is structured as described in Tab. 1.

Therefore, records arrived to Spark are object descriptions in our 3WD problem and contribute to form the evolving universe of discourse W. Hence, the decision table is $(W, A \cup d)$ where $A = \{a_1, a_2, \dots\}$. Record structure is crucial for the algorithm definition. All the records share the same

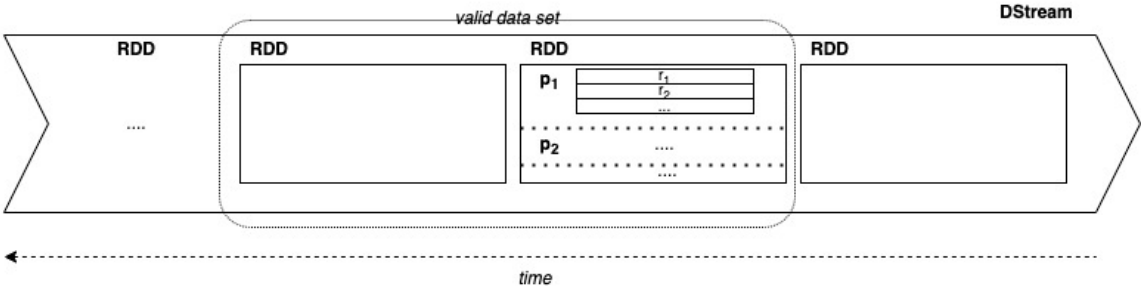

FIGURE 5. Storing records into DStream and RDDs.

TABLE 1. Structure of the input records produced by Kafka.

index	description
0	object identifier
1 to N-3	values for conditional attributes $a_1, a_2, \ldots$
N-2	value for the decision attribute d
N-1	timestamp

structure. Thus, assuming to deal with the scenario introduced in Section IV-A, a sample record could be:

$$(v_1,\ \texttt{LOW},\ \texttt{MID},\ \texttt{FAR},\ \texttt{Ferry},\ S,\ 1)$$

where v_1 is the vessel identifier, *LOW*, *MID*, *FAR* and *Ferry* are respectively the values associated to the condition attributes `Drift Angle`, `Velocity`, `Distance` and `Type`. Moreover, S is the value for the decision attribute `Safe` and, lastly, 1 is the `Timestamp` of the record. Take care that in Spark the record values are serialized as strings.

C. DATA PROCESSING ALGORITHM FOR 3WD

The main data processing algorithm to be deployed as a Spark job is described by pseudocode (1). Such algorithm foresees that data within RDDs (in the DStream) are provided as iterable sequences of strings (to be clear we will assume that iterable sequences are lists or arrays on which it is possible to apply the subscript operator to access individual elements). The algorithm adopts functions based on the `map` and `reduce` programming model. For the sake of clarity, functions used by pseudocode (1) will be explained by using specific examples. Before starting to describe the algorithm, it is needed to formalize the problem. In particular, the valid records (according to the windowing approach along the timeline) represent the universe of discourse, the concept to study is the subset of the universe including records in which the decision attribute assumes a given value d_{value}. In the next lines, we will assume a record structure that is proposed in Tab. 1 and the provided examples (from the maritime surveillance scenario) consider $N = 7$ and $d_{value} = S$.

Let us start by describing all the custom functions. The first one, used in the first operation of the algorithm, is `buildPairByID(e)`, where e is a list containing data coming from one record (a full description of an observed object). This function is used to map all input arrays (records) into key-value pairs. The key will be the element 0 of an input array and the value will be an inner array containing all the remaining elements, from 1 to $N - 1$. If the input array is e (remember that elements in e are organized as in Tab. 1) then the output pair is constructed in the following way: $(k, v) = (e[0], (e[1], e[2], \ldots, e[\text{N-3}], e[\text{N-2}], e[\text{N-1}]))$. For instance, if the input is:

$$(v_2,\ \texttt{LOW},\ \texttt{MID},\ \texttt{NEAR},\ \texttt{Ferry},\ \texttt{D},\ 1)$$

then the output pair will be:

$$(v_2,\ (\texttt{LOW},\ \texttt{MID},\ \texttt{NEAR},\ \texttt{Ferry},\ \texttt{D},\ 1))$$

Such a reshaping is needed in order to support the work of the Spark constructs `reduceByKey`.

The second custom function is `updateInfo(`e_1`,` e_2`)` that is applied to the results of the previous operation by means of a `reduceByKey` construct. The arguments for the function are couples of pairs, with the same key, which will be bound to the parameters e_1 and e_2. The idea is to reduce two pairs into one if such pairs have the same key. The output pair is the input pair with the greater timestamp. The idea is that if two descriptions of the same object (same key) are in the system at the same time, only the most recent one will be considered during the 3WD analysis. If the input pairs are (k_1, v_1) and (k_2, v_2) with k_1 equal to k_2 then the output pair is (k_1, v_1) if $v_1[N-2] > v_2[N-2]$, otherwise the output is (k_2, v_2). For instance, if the result of the previous operation contains:

$$(v_2,\ (\texttt{LOW},\ \texttt{MID},\ \texttt{NEAR},\ \texttt{Ferry},\ \texttt{D},\ 1))$$
$$(v_2,\ (\texttt{LOW},\ \texttt{HIGH},\ \texttt{NEAR},\ \texttt{Ferry},\ \texttt{D},\ 2))$$

Algorithm 1 3WD for Data Streams in Apache Spark

Require: $rdd \leftarrow$ input micro-batch
Require: d_{value}
Require: α, β
$rdd \leftarrow rdd$.**map**(buildPairByID)
$rdd \leftarrow rdd$.**reduceByKey**(updateInfo)
$rdd \leftarrow rdd$.**map**(buildArray)
$rdd \leftarrow rdd$.**map**(buildPairByCondition)
$rdd \leftarrow rdd$.**reduceByKey**(buildEquivalenceClass)
$rdd \leftarrow rdd$.**map**(addConditionalProbability)
$rdd \leftarrow rdd$.**map**(addRegionInfo)
$rdd \leftarrow rdd$.**flatMap**(explodeRegion)

then the output pair will be:

(v2, (LOW, HIGH, NEAR, Ferry, D, 2))

and the first pair has been discarded. Such operation implements the *explicit update* primitive illustrated in Section IV-B. The other two primitives (*explicit immigration* and *implicit emigration*) are directly managed by the spark streaming engine through stateful transformations.

The third and fourth operations map each pair obtained by the previous operation into a new pair. Such transformation is realized through two custom functions applied in the pipeline: `buildArray(`e`)` and `buildPairByCondition(`e`)`. For the sake of simplicity, the above transformation is provided through two steps but it is possible to define only one function offering the whole transformation behaviour. More in detail, the function `buildArray(`e`)` re-transforms an input pair (k, v) (provided by the previous operation) into a flat array $e = (k, v[0], \ldots, v[\text{N-2}])$ exposing the original record structure. Furthermore, the function `buildPairByCondition(`e`)` is applied to the results of the previous function to transform the flat array e into a new key-value pair: $((e[1], \ldots, e[\text{N-3}]), ([e[0]], [e[\text{N-2}]], \gamma))$, where γ is an array containing $e[N-2]$ if such value is equal to d_{value}, otherwise γ is the empty array. For instance, if we start from the pair:

(v3, (LOW, MID, NEAR, Ferry, S, 2))

the first function transforms it into:

(v3, LOW, MID, NEAR, Ferry, S, 2)

and the second function transforms such result into:

((LOW, MID, NEAR, Ferry), ([v3],[S],[S]))

Otherwise, if we start from:

(v3, (LOW, MID, NEAR, Ferry, D, 2))

the first function transforms it into:

(v3, LOW, MID, NEAR, Ferry, D, 2)

and the second function transforms such result into:

((LOW, MID, NEAR, Ferry), ([v3],[D],[]))

The idea of this part of the algorithm is to organize data in a way to support the grouping of records by the condition attributes and obtaining the required equivalence classes (see eq. (4) and eq. (5)).

The fifth operation is the most important one, in fact, it is used to construct the equivalence classes with respect to the condition attributes. Such operation is based on the construct `reduceByKey`, with the custom function `buildEquivalenceClasses(`e_1, e_2`)` that is used to combine couples of pairs (provided by the previous operation) if they have the same key. More formally, if we have two pairs (k_1, v_1) and (k_2, v_2) (with k_1 equal to k_2 equal to k) bound respectively to e_1 and e_2, the result is the pair: $(k, (v_1[0] + v_2[0], v_1[1] + v_2[1], v_1[2] + v_2[2]))$. If applying such reduce operation incrementally, the result will be a number of pairs equal to the number of different equivalence classes based on the condition attributes. For instance, if we have the following starting pairs:

((LOW, MID, NEAR, Ferry), ([v3],[S],[S]))
((LOW, MID, NEAR, Ferry), ([v2],[D],[]))
((LOW, MID, NEAR, Ferry), ([v6],[S],[S]))
((LOW, MID, FAR, Cargo), ([v1],[S],[S]))
((LOW, LOW, NEAR, Research), ([v4],[D],[]))
((LOW, MID, FAR, Cargo), ([v5],[S],[S]))

the resulting pairs are:

((LOW, MID, NEAR, Ferry), ([v3,v2,v6],[S, D,S],[S,S]))
((LOW, MID, FAR, Cargo), ([v1,v5],[S,S],[S,S]))
((LOW, LOW, NEAR, Research), ([v4],[D],[]))

Three equivalence classes have been obtained: $[v_2v_3, v_6]$, $[v_1, v_5]$ and $[v_4]$.

Furthermore sixth and seventh operations map equivalence classes into the traditional 3WD regions, i.e., positive (POS), negative (NEG) and boundary (BND), according to the eq. (7) and the thresholds α and β. We have divided this transformation into two parts. The first part is accomplished by means of the custom function `addConditionalProbability(`e`)` that maps each pair (k, v), obtained by the previous operation, into a pair of the following shape: $(v[0], |v[2]|/|v[1]|)$, where $v[0]$ is the list of IDs of the objects in the same equivalence class (opportunely transformed into an immutable value) and $|.|$ is an operator measuring the array length. In other words, we are calculating the conditional probability because $|v[2]|$ represents the number of objects belonging to the studied concepts and $|v[1]|$ represents the number of objects in the current equivalence class. Let us provide an example. If we have this input pair:

((LOW, MID, NEAR, Ferry), ([v3,v2,v6],[S, D,S],[S,S]))

the output pair is:

((v3,v2,v6), 0.67))

where 0.67 is the rough membership of the equivalence class $[v_2v_3, v_6]$.

Moreover, the seventh operation maps, through the custom function `addRegionInfo(`e`)`, the pairs (k, v) created by the previous operation into pairs (k, R), where R is `POS`, `BND` or `NEG` according to the rules of eq. (7). For instance, assuming $\alpha = 0.65$ and $\beta = 0.25$, if the input pair is:

((v3,v2,v6), 0.67)

then the output pair is:

((v3,v2,v6), POS)

The eighth operation (the last one of the algorithm) is realized by means of the construct `flatMap` that transforms one record into zero or more records. The custom function `explodeRegion(`e`)` is applied by means of the above construct in order to generate arrays for associating each monitored object with the region it belongs to. For instance, if the input pair is:

$$((v_3, v_2, v_6),\ \texttt{POS})$$

the output pairs are:

$$(v_3,\ \texttt{POS})$$
$$(v_2,\ \texttt{POS})$$
$$(v_6,\ \texttt{POS})$$

D. SPARK APPLICATION DEPLOY

The algorithm illustrated by the pseudocode (1) must be deployed within a Spark `StreamingContext` in order to allow Spark to continuously loading the input into the DStream. Such algorithm is executed iteratively at each computation time (as described in Section VI-A) over the up-to-date variable *rdd* (pseudocode (1)) that contains all input records that are valid with respect to the windowing approach and the dynamics illustrated in Section IV-B. The work distribution and the parallelism capabilities are transparent, i.e., it is possible to change the number of worker nodes within the cluster and modify other parameters to optimize the execution. Further optimizations could be realized by adding instructions to the source code. For instance, it is possible to modify window length, sliding length, data gathering frequency (batch time) and also to modify the number of partitions for each RDD. This number is important because the distribution of the work is firstly accomplished (automatically by Spark) according to it.

E. FIRST SAMPLE RUN

Consider a run where the window length is 30, the sliding length is 10 and data are read every second. The first burst of data comes into the system at timestamp `16:26:31` and the first computation time is `16:26:32`. The burst includes the following data:

$(v_1,$ LOW, LOW, FAR, Cargo, S, 1)
$(v_2,$ LOW, MID, NEAR, Ferry, D, 1)
$(v_3,$ MID, LOW, MID, Cargo, S, 1)
$(v_4,$ MID, MID, MID, Research, S, 1)
$(v_5,$ MID, LOW, FAR, Research, S, 1)

thus, such descriptions form the evolving universe of discourse W. Moreover, the result of 3WD is:

$$(v_1,\ \texttt{POS})$$
$$(v_2,\ \texttt{BND})$$
$$(v_3,\ \texttt{POS})$$
$$(v_4,\ \texttt{BND})$$
$$(v_5,\ \texttt{POS})$$

The second computation time happens at timestamp `16:26:42` (remember that the sliding length is 10) and the result (tri-partitioning) is exactly the same as that before because, at this time, the evolving universe W' is equal to its previous version W. The third computation time happens at timestamp `16:26:52`. Soon before, a data gathering operation pushes into the system the following data:

$(v_2,$ LOW, HIGH, NEAR, Ferry, D, 2)

that is an update for the information related to the vessel v_2. Therefore, the third computation time produces a fresh tri-partitioning that takes into account both first and second bursts of data because both arrive into the system within the window length (30 seconds), i.e., the evolving universe $W'' = (W' - \{des(v_2, 1)\}) \cup \{des(v_2, 2)\}$. In this case the result is:

$$(v_1,\ \texttt{POS})$$
$$(v_2,\ \texttt{NEG})$$
$$(v_3,\ \texttt{POS})$$
$$(v_4,\ \texttt{POS})$$
$$(v_5,\ \texttt{POS})$$

It is clear that there is an important update. The situation is more clear now given that the boundary (BND) region disappears. The next computation time occurs at timestamp `16:27:02` and considers data arrived in the interval `16:27:02-16:26:32`. Thus, the first burst is excluded, i.e., $W''' = W'' - \{des(v_1, 1), des(v_3, 1), des(v_4, 1), des(v_5, 1)\}$. In such a case the result (tri-partitioning) is the following one:

$$(v_2,\ \texttt{NEG})$$

The processing continues in this way as the Spark application continues to monitor the environment.

VII. EXPERIMENTATION AND EVALUATION

Experimentation activities were configured in order to emphasize the capability of the defined architecture to early detect dangerous situations in order to allow human operators to plan a suitable course of actions to mitigate the risks in a maritime surveillance scenario. Therefore, the evaluation objective is to show that the 3WD analysis approach deployed into a streaming computing environment: i) allows the real-time detection of potential drifting vessels, ii) anticipates the results of classification rules (based on heuristics and built through the expertise of human experts) that are necessarily affected by imprecision. Such an objective will be achieved by comparing the results of the aforementioned rules to the results obtained by 3WD analysis over a dataset of structured observations describing the evolving status of some vessels along consecutive time windows. As described in the next sections, the streaming-based 3WD provides, in general, a performance comparable to that offered by the considered heuristics. The experimentation also shows that the proposed implementation anticipates, with respect to the

expert-based classification rules, the detection of dangerous situations for specific vessels.

A. DATASET CONSTRUCTION AND STREAM SIMULATION

The dataset has been generated by using Trumania,[7] a scenario-based random dataset generator library in Python 3. The dataset includes 1200 rows. Each row describes an observation of a vessel at a given time with respect to its velocity and drift angle. A row includes also the vessel identifier and the timestamp at which the observation has been generated. The monitored vessels are 20 and the 35% of them have a drifting attitude, i.e., such vessels will drift during the experimentation time. All 20 vessels start the simulation assuming a safe status and could change their values along the monitored period. In order to execute the experimentation, it was needed to simulate the stream of observations and forward them to the streaming-based 3WD algorithm executed within the defined architecture. This operation was accomplished by running a Python script generating a micro-batch of observations every 5 seconds. Each micro-batch has to be processed along all the architecture phases before being analysed by the streaming-based 3WD.

B. DATA WRANGLING

Observations included in the generated dataset are processed and transformed into vessel descriptions (as shown in Tab. 1) by the architecture whose sketch is provided in Fig. 3 and, in particular, by the Faust-based component that executes the following discretization rules applying them as new observations arrive:

- **if** $0 \leq$ velocity ≤ 5 **then** LOW
- **if** $5 <$ velocity ≤ 15 **then** MEDIUM
- **if** velocity > 15 **then** HIGH
- **if** drift angle ≤ 15 **then** LOW
- **if** $15 <$ drift angle ≤ 30 **then** MEDIUM
- **if** drift angle > 30 **then** HIGH

and the following classification rule:

- **if** $3 \leq$ velocity ≤ 5 **and** drift angle > 30 **then** DANGEROUS **else** SAFE

Both types of rules, coming from the works described in [42] and [48], allow to build the evolving decision table that can be processed by the algorithm (1). It is important to note that the classification rule is expert-based and it could produce imprecise results. Thus, the 3WD approach, implemented through probability-based rough sets, well fits to improve the situation assessment and provide interesting results.

C. S3WD SETTINGS

The streaming-based 3WD algorithm is executed over the stream of descriptions provided by Trumania. In particular, window and sliding lengths (Spark Streaming parameters) are respectively 30 and 10, and 3WD thresholds, i.e., β and α, are respectively 0.3 and 0.8. The 3WD target concept is SAFE, hence POSITIVE region will contain all the vessels that are not drifting certainly, NEGATIVE region will include all vessels that are drifting certainly and, lastly, BOUNDARY region will contain all the vessels for which it is not certain that they are drifting nor that they are not drifting. As described in the previous sections, the aforementioned regions evolve as time goes on and, consequently, change their composition in terms of included vessels. Therefore, the results will be provided along the timeline to describe performances at given time instants.

D. RESULTS

Simulation results will be discussed by considering two perspectives. The first perspective is the quantitative one, i.e., we will consider the typical measures, precision and recall, to evaluate the tri-partitioning results. Subsequently, the second perspective will focus on the analysis of the behaviour of specific drifting vessels in order to show the real potential of the proposed approach.

According to the first perspective, Tab. 2 reports precision and recall values of streaming-based 3WD (S3WD) compared to those obtained by applying the classification rules described in Section VII-B. The table includes a row for each application of S3WD. In particular, the first column reports the timestamp associated with an update of the results provided by S3WD, the further two columns report the precision measures calculated by considering that the information provided by the BOUNDARY region consists always of true positives (MPBY), or always of false positives (MPBN). The last two columns report the recall measures calculated by considering that the information provided by the BOUNDARY region consists always of true positives (MRBY) or always of false positives (MRBN). Moreover, Tab. 3 adopts the same scheme of Tab. 2 in order to report precision and recall measures against the attitude of vessels to drift or not along the full monitored period (the attitude informs us if a given vessel will drift or not at any time instant during the monitored period). In order to conclude this first evaluation perspective, it is interesting to observe that the precision values of S3WD are equal to the corresponding values reported by the classification rules (see SectionVII-B). Indeed, the recall values of S3WD tend to become higher, as time goes on, than the corresponding values obtained by applying the classification rules as shown in Fig. 8. In particular, it is clear that S3WD performs better than heuristics when the situation is more tangled and it is needed computational support for increasing the analyst/operator's awareness. Aggregated measures like F1 score[8] and ROC AUC score[9] in the case of comparison with classification rules are respectively reported in Fig. 6 and Fig. 7. Both the measures range from around 0.9 to 1. Therefore, the results, according to the first perspective, show that

[7] http://realimpactanalytics.github.io/trumania/

[8] The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

[9] Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction score.

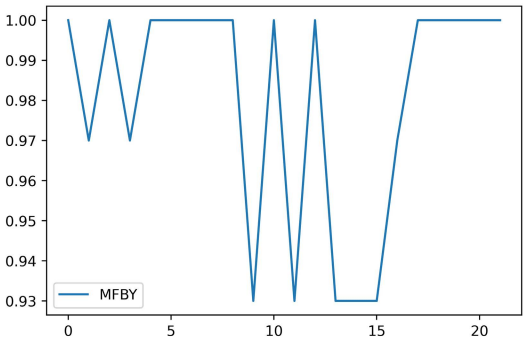

FIGURE 6. F1 score (it is assumed that objects in the boundary region are always true positive) against classification rules.

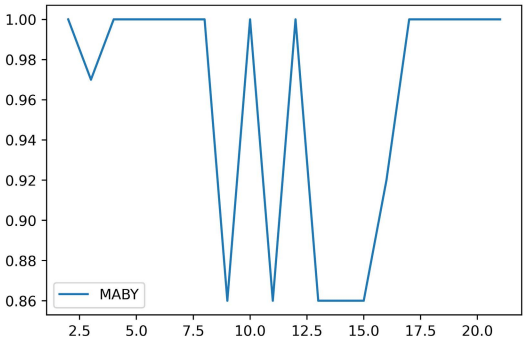

FIGURE 7. ROC AUC score (it is assumed that objects in the boundary region are always true positive) against classification rules.

S3WD is able to maintain good classification performance along all the execution time. Furthermore, according to the second evaluation perspective, it is interesting to analyse the results of S3WD with respect to the vessels having a drifting attitude. Such vessels are those with identifiers 4, 5, 8, 11, 12, 13, and 18. In particular, for vessels 4, 8, 12, and 13 the proposed approach is able to point out the attention on such vessels (putting them into the BOUNDARY region) exactly when the heuristics classify them as DANGEROUS. Moreover, vessel 5 is put into the NEGATIVE region (stating that it is in a DRIFTING status) exactly when the heuristics classify it as DANGEROUS. Thus, for the above vessels, S3WD provides the same performance as the classification rules. The advantages of S3WD are evident when dealing with vessels 11 and 18.

Such cases are reported in Fig. 9 and Fig. 10 and, in some sense, allows to emphasize some qualitative advantages of a three-way decision model when compared to a traditional two-way model. In these figures for the safe-dangerous line (heuristics), value -1 is the DANGEROUS status and value 1 is the SAFE status. Moreover, for the S3WD (that is the proposed approach), value -1 is the NEGATIVE region (DRIFTING behavior), 0 is the BOUNDARY region (uncertain behavior) and 1 is the POSITIVE region (not DRIFTING behavior). In particular, the DRIFTING behaviour of vessel 11 is early detected by means of S3WD that puts it firstly into the BOUNDARY region and secondly into the NEGATIVE region before such vessel is classified as DANGEROUS by the heuristics. The same phenomenon happens for vessel 18.

TABLE 2. Precision and recall values against classification rules.

	MPBY	MPBN	MRBY	MRBN
0	1.0	1.0	1.0	1.0
1	1.0	0.95	0.95	0.95
2	1.0	0.89	1.0	0.89
3	1.0	0.88	0.94	0.83
4	1.0	0.88	1.0	0.83
5	1.0	0.83	1.0	0.88
6	1.0	0.88	1.0	0.83
7	1.0	0.88	1.0	0.78
8	1.0	0.88	1.0	0.78
9	0.87	0.87	1.0	1.0
10	1.0	0.94	1.0	0.84
11	0.87	0.87	1.0	1.0
12	1.0	0.88	1.0	0.78
13	0.87	0.87	1.0	1.0
14	0.87	0.87	1.0	1.0
15	0.87	0.87	1.0	1.0
16	0.93	0.93	1.0	1.0
17	1.0	1.0	1.0	1.0
18	1.0	1.0	1.0	1.0
19	1.0	1.0	1.0	1.0
20	1.0	1.0	1.0	1.0
21	1.0	1.0	1.0	1.0

TABLE 3. Precision and recall values against attitude.

	APBY	APBN	ARBY	ARBN
0	1.0	1.0	0.65	0.65
1	1.0	1.0	0.65	0.68
2	1.0	1.0	0.72	0.72
3	1.0	1.0	0.72	0.72
4	1.0	1.0	0.76	0.72
5	1.0	1.0	0.72	0.76
6	1.0	1.0	0.76	0.72
7	1.0	1.0	0.81	0.72
8	1.0	1.0	0.81	0.72
9	1.0	1.0	1.0	1.0
10	1.0	1.0	0.76	0.68
11	1.0	1.0	1.0	1.0
12	1.0	1.0	0.81	0.72
13	1.0	1.0	1.0	1.0
14	1.0	1.0	1.0	1.0
15	1.0	1.0	1.0	1.0
16	1.0	1.0	0.93	0.93
17	1.0	1.0	1.0	1.0
18	1.0	1.0	1.0	1.0
19	1.0	1.0	1.0	1.0
20	1.0	1.0	0.93	0.93
21	1.0	1.0	0.93	0.93

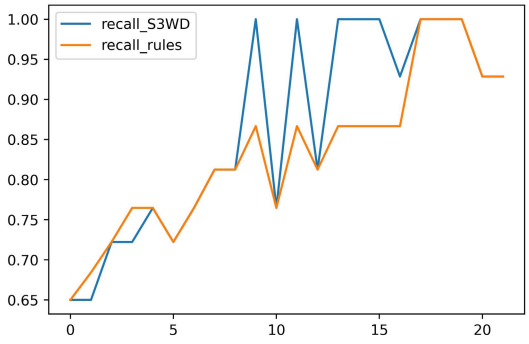

FIGURE 8. Recall values of S3WD compared to those of classification rules.

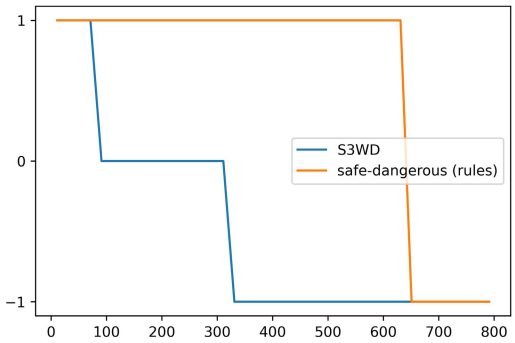

FIGURE 9. Situation assessment for vessel 11.

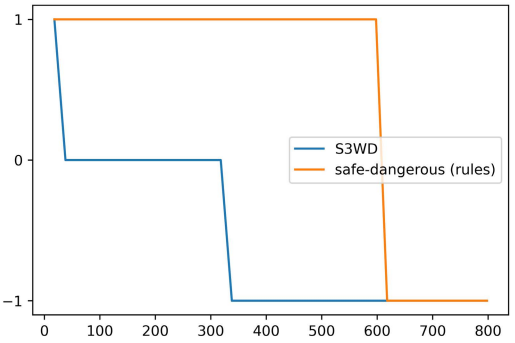

FIGURE 10. Situation assessment for vessel 18.

Lastly, a few additional pieces of information can be provided in order to better clarify processing time. First of all, it is important to underline that we have executed the experiment by using an early prototype in which batches of data (each batch contains 20 records) are sent to the Spark Application through the Google Drive Storage. Thus, the latency includes also the storage time. More in detail, 1200 records (tuples describing vessels) have been sent to the Spark Application (implementing S3WD) during a time window of around 600 seconds. All the S3WD processing tasks have been executed in a time window of 603 seconds. Let us explain. Couples of batches had an overall process time (including Google Drive Storage latency, pre-processing and three-way classification) of around 21 seconds. The throughput, in this case, was 1.9 records/second. During the experimentation activities, a new batch (20 records) is sent to the Spark Application (S3WD) every around 10 seconds. In such a case, the Application was able to process two consecutive batches before the next one arrives.

VIII. CONCLUSION

This paper describes a novel implementation of Three-Way Decisions, based on probability-based rough set theory, over a streaming computing platform, namely Apache Spark. Such implementation is based on *map* and *reduce* programming style and is part of a whole architecture focused on dealing with situation assessment in large and complex environments. The proposed approach provides many advantages being able to support different and heterogeneous real-world scenarios in which data to be analysed come from distributed sources spread over the considered environment and have the characteristics of Big Data (volume, velocity, etc.). The architecture inherits from Apache Spark also additional useful characteristics like, for instance, the fault tolerance one. Furthermore, from the result quality point of view, the experimentation activities show that streaming-based Three-Way Decisions implementation is able to early detect some specific behaviour of the monitored objects as demonstrated by the drifting behaviour of vessels. Lastly, future works have been already planned for executing additional experimentation activities in different domains in order to test the implementation approach considering different contexts and to validate the architecture also for non-functional characteristics.

ACKNOWLEDGMENT

The authors thank the Computational and Data Science (CODAS) Laboratory[10] at the Dipartimento di Scienze Aziendali–Management & Innovation Systems (DISA-MIS) of the Università Degli Studi di Salerno for the technological support.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X13000241

[2] M. Z. Ge, H. Bangui, and B. Buhnova, "Big data for Internet of Things: A survey," *Future Gener. Comput. Syst.*, vol. 87, pp. 601–614, Oct. 2018.

[3] I. A. T. Hashem, V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma, "The role of big data in smart city," *Int. J. Inf. Manag.*, vol. 36, no. 5, pp. 748–758, Oct. 2016.

[4] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 383–398, Jan. 2019.

[5] Y. Yao, "An outline of a theory of three-way decisions," in *Rough Sets and Current Trends in Computing*, J. Yao, Y. Yang, R. Słowiński, S. Greco, H. Li, S. Mitra, and L. Polkowski, Eds. Berlin, Germany: Springer, 2012, pp. 1–17.

[6] Y. Yao, "Three-way decisions and cognitive computing," *Cogn. Comput.*, vol. 8, no. 4, pp. 543–554, 2016.

[7] Y. Yao, "The superiority of three-way decisions in probabilistic rough set models," *Inf. Sci.*, vol. 181, no. 6, pp. 1080–1096, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025510005645

[8] M. Hu, "Three-way data analytics: Preparing and analyzing data in threes," *Inf. Sci.*, vol. 573, pp. 412–432, Sep. 2021.

[9] Y. Yao, "Three-way decision: An interpretation of rules in rough set theory," in *Rough Sets and Knowledge Technology*, P. Wen, Y. Li, L. Polkowski, Y. Yao, S. Tsumoto, and G. Wang, Eds. Berlin, Germany: Springer, 2009, pp. 642–649.

[10] C. Fernandez-Basso, A. J. Francisco-Agra, M. J. Martin-Bautista, and M. D. Ruiz, "Finding tendencies in streaming data using big data frequent itemset mining," *Knowl.-Based Syst.*, vol. 163, pp. 666–674, Jan. 2019.

[11] J. Xu, D. Miao, Y. Zhang, and Z. Zhang, "A three-way decisions model with probabilistic rough sets for stream computing," *Int. J. Approx. Reasoning*, vol. 88, pp. 1–22, Sep. 2017.

[12] W.-C. Bang and Z.-N. Bien, "Incremental inductive learning algorithm in the framework of rough set theory and its application," in *Proc. Korean Inst. Intell. Syst. Conf.* Seoul, South Korea: Korean Institute of Intelligent Systems, 1998, pp. 308–313.

[13] H. Chen, T. Li, D. Ruan, J. Lin, and C. Hu, "A rough-set-based incremental approach for updating approximations under dynamic maintenance environments," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 2, pp. 274–284, Feb. 2011.

[10] https://www.disa.unisa.it/en/department/structures?id=404

[14] W. Ziarko, "Variable precision rough set model," *J. Comput. Syst. Sci.*, vol. 46, no. 1, pp. 39–59, Feb. 1993.
[15] H. Chen, T. Li, C. Hu, and X. Ji, "An incremental updating principle for computing approximations in information systems while the object set varies with time," in *Proc. IEEE Int. Conf. Granular Comput.*, Aug. 2009, pp. 49–52.
[16] H. Chen, T. Li, C. Luo, S.-J. Horng, and G. Wang, "A decision-theoretic rough set approach for dynamic data mining," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 1958–1970, Dec. 2015.
[17] J. Zhang, T. Li, D. Ruan, and D. Liu, "Neighborhood rough sets for dynamic data mining," *Int. J. Intell. Syst.*, vol. 27, no. 4, pp. 317–342, Apr. 2012.
[18] D. Liu, T. Li, and J. Zhang, "A rough set-based incremental approach for learning knowledge in dynamic incomplete information systems," *Int. J. Approx. Reasoning*, vol. 55, no. 8, pp. 1764–1786, Nov. 2014.
[19] Y. Cheng, "The incremental method for fast computing the rough fuzzy approximations," *Data Knowl. Eng.*, vol. 70, no. 1, pp. 84–100, Jan. 2011.
[20] S. Li, T. Li, and D. Liu, "Incremental updating approximations in dominance-based rough sets approach under the variation of the attribute set," *Knowl.-Based Syst.*, vol. 40, pp. 17–26, Mar. 2013.
[21] D. Liu, J. Zhang, and T. Li, "An probabilistic rough set approach for incremental learning knowledge on the change of attributes," in *Computational Intelligence: Foundations and Applications*. Singapore: World Scientific, 2010, pp. 722–727.
[22] D. Liu, T. Li, G. Liu, and P. Hu, "An approach for inducing interesting incremental knowledge based on the change of attribute values," in *Proc. IEEE Int. Conf. Granular Comput.*, Aug. 2009, pp. 415–418.
[23] D. Liu, T. Li, and J. Zhang, "An incremental approach for rule induction under coarsening and refining of attribute values in E-business systems," in *Proc. Int. Conf. E-Business Intell.*, 2010, pp. 541–547.
[24] D. Ciucci, "Classification of dynamics in rough sets," in *Rough Sets and Current Trends in Computing*, M. Szczuka, M. Kryszkiewicz, S. Ramanna, R. Jensen, and Q. Hu, Eds. Berlin, Germany: Springer, 2010, pp. 257–266.
[25] D. Ciucci, "Temporal dynamics in information tables," *Fundamenta Informaticae*, vol. 115, no. 1, pp. 57–74, 2012.
[26] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[27] J. Zhang, T. Li, D. Ruan, Z. Gao, and C. Zhao, "A parallel method for computing rough set approximations," *Inf. Sci.*, vol. 194, pp. 209–223, Jul. 2012.
[28] J. Zhang, T. Li, and H. Chen, "Composite rough sets for dynamic data mining," *Inf. Sci.*, vol. 257, pp. 81–100, Feb. 2014.
[29] T. Cao, K. Yamada, M. Unehara, I. Suzuki, and D. Nguyen, "Parallel computation of rough set approximations in information systems with missing decision data," *Computers*, vol. 7, no. 3, p. 44, Aug. 2018.
[30] E. M. Marouane and Z. Elhoussaine, "A fuzzy neighborhood rough set method for anomaly detection in large scale data," *IAES Int. J. Artif. Intell.*, vol. 9, no. 1, p. 1, Mar. 2020.
[31] P. Sowkuntla and P. S. V. S. S. Prasad, "MapReduce based parallel fuzzy-rough attribute reduction using discernibility matrix," *Int. J. Speech Technol.*, vol. 52, no. 1, pp. 154–173, Jan. 2022.
[32] S. Vluymans, H. Asfoor, Y. Saeys, C. Cornelis, M. Tolentino, A. Teredesai, and M. De Cock, "Distributed fuzzy rough prototype selection for big data regression," in *Proc. Annu. Conf. North Amer. Fuzzy Inf. Process. Soc. (NAFIPS) Held Jointly 5th World Conf. Soft Comput. (WConSC)*, Aug. 2015, pp. 1–6.
[33] E.-S. M. El-Alfy and M. A. Alshammari, "Towards scalable rough set based attribute subset selection for intrusion detection using parallel genetic algorithm in MapReduce," *Simul. Model. Pract. Theory*, vol. 64, pp. 18–29, May 2016.
[34] M. Wu and H. Sakai, "On parallelization of the NIS-apriori algorithm for data mining," *Proc. Comput. Sci.*, vol. 60, pp. 623–631, Jan. 2015.
[35] H. Bhukya and M. Sadanandam, "MapReduce-driven rough set fuzzy classification rule generation for big data processing," in *Intelligent Systems, Technologies and Applications*, M. Paprzycki, S. M. Thampi, S. Mitra, L. Trajkovic, and E.-S. M. El-Alfy, Eds. Singapore: Springer, 2021, pp. 87–102.
[36] S. Shahrivari, "Beyond batch processing: Towards real-time and streaming big data," *Computer*, vol. 3, no. 4, pp. 117–129, 2014.
[37] B. Zhou, H. Cho, and X. Zhang, "Scalable implementations of rough set algorithms: A survey," in *Recent Trends and Future Technology in Applied Intelligence*, M. Mouhoub, S. Sadaoui, O. A. Mohamed, and M. Ali, Eds. Cham, Switzerland: Springer, 2018, pp. 648–660.
[38] G.-Y. Wang, Y.-Y. Yao, and H. Yu, "A survey on rough set theory and applications," *Chin. J. Comput.*, vol. 32, no. 7, pp. 1229–1246, Aug. 2009.
[39] Y. Yao, S. Greco, and R. Słowiński, "Probabilistic rough sets," in *Springer Handbook of Computational Intelligence*, J. Kacprzyk and W. Pedrycz, Eds. Berlin, Germany: Springer, 2015, pp. 387–411, doi: 10.1007/978-3-662-43505-2_24.
[40] S. K. M. Wong and W. Ziarko, "Comparison of the probabilistic approximate classification and the fuzzy set model," *Fuzzy Sets Syst.*, vol. 21, no. 3, pp. 357–362, Mar. 1987.
[41] Y. Yao, "Decision-theoretic rough set models," in *Rough Sets and Knowledge Technology*, J. Yao, P. Lingras, W.-Z. Wu, M. Szczuka, N. J. Cercone, and D. Ślęzak, Eds. Berlin, Germany: Springer, 2007, pp. 1–12.
[42] G. D'Aniello, A. Gaeta, V. Loia, and F. Orciuoli, "A model based on rough sets for situation comprehension and projection," in *Proc. IEEE Conf. Cognit. Comput. Aspects Situation Manag. (CogSIMA)*, Mar. 2017, pp. 1–7.
[43] A. T. Primer, "Structured analytic techniques for improving intelligence analysis," CIA Center study Intell., Charlottesville, VA, USA, Tech. Rep., 2009.
[44] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on Apache spark," *Int. J. Data Sci. Anal.*, vol. 1, no. 3, pp. 145–164, 2016.
[45] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2012, pp. 15–28.
[46] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, *Learning Spark: Lightning-Fast Big Data Analysis*. Sebastopol, CA, USA: O'Reilly Media, 2015.
[47] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized streams: Fault-tolerant streaming computation at scale," in *Proc. 24th ACM Symp. Operating Syst. Princ.*, Nov. 2013, pp. 423–438.
[48] N. Willems, R. Scheepens, H. van de Wetering, and J. J. van Wijk, "Visualization of vessel traffic," in *Situation Awareness With Systems of Systems*, P. van de Laar, J. Tretmans, and M. Borth, Eds. New York, NY, USA: Springer, 2013, pp. 73–87, doi: 10.1007/978-1-4614-6230-9_5.

GRAZIANO FUCCIO received the master's degree in business innovation and informatics from the University of Salerno, Fisciano, Italy, in 2018, where he is currently pursuing the Ph.D. degree. During his university studies, he worked on many research projects related to data science. His research interests include decision support systems, situation awareness, and big data analytics.

VINCENZO LOIA (Senior Member, IEEE) received the master's degree in computer science from the University of Salerno, Fisciano, Italy, in 1985, and the Ph.D. degree in computer science from the University of Paris 6, Paris, France, in 1989. He is currently a Full Professor of computer science at the University of Salerno. He is the Editor-in-Chief of the *Journal of Ambient Intelligence and Humanized Computing* (Springer) and the *Journal of Evolutionary Intelligence* (Springer). He also serves as an associate editor of more than ten international journals.

FRANCESCO ORCIUOLI (Member, IEEE) received the master's degree *(cum laude)* in computer science from the University of Salerno, Fisciano, Italy. He is currently an Associate Professor of computer science at the University of Salerno. He is also focusing his research activities on data science and computational intelligence. He is the coauthor of more than 130 scientific publications indexed by Scopus and a co-founder of a university spin-off involved in several research and development projects related to e-health.