IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 19, 2022, accepted April 28, 2022, date of publication May 9, 2022, date of current version May 16, 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3173376*

# Real-Time Event-Driven Learning in Highly Volatile Systems: A Case for Embedded Machine Learning for SCADA Systems

**MANUEL GONÇALVES[1], PEDRO SOUSA[2], JÉRÔME MENDES[1], (Member, IEEE), MORAD DANISHVAR[3], AND ALIREZA MOUSAVI[3], (Senior Member, IEEE)**

[1]Department of Electrical and Computer Engineering, Institute of Systems and Robotics, University of Coimbra, Pólo II, 3030-290 Coimbra, Portugal
[2]Oncontrol Technologies, 3000-174 Coimbra, Portugal
[3]Department of Mechanical and Aerospace Engineering, College of Engineering, Design and Physical Sciences, Brunel University London, London, Uxbridge UB8 3PH, U.K.

Corresponding author: Morad Danishvar (morad.danishvar@brunel.ac.uk)

This work was supported by the project "Eco-Healing-Intelligent Eco-Controller with Self-Healing Capability" through OE-national funds of FCT/MCTES (PIDDAC) under Project UIDB/00048/2020.

**ABSTRACT** Extracting key system parameters and their impact on state transition is a necessity for knowledge and data engineering. In Decision Support Systems, the quest for yet more efficient and faster methods of sensitivity analysis (SA) and feature extraction in complex and volatile systems persists. A new improved event tracking methodology, the fastTracker, for real-time SA in large scale complex systems is proposed in this paper. The main feature of fastTracker is its high-frequency analytics using meager computational cost. It is suitable for data processing and prioritization in embedded systems, Internet of Things (IoT), distributed computing (e.g. Edge computing) applications. The presented algorithm's underpinning rationale is event driven; its objective is to correctly and succinctly quantify the sensitivity of observable changes in the system (output) with respect to the input variables. To demonstrate the performance of the proposed fastTracker methodology, fastTracker was deployed in the Supervisory control and data acquisition (SCADA) system from real cement industry. fastTracker has been verified by system experts in real industrial application. Its performance was compared with other real-time event-based SA techniques. The comparison revealed savings of 98.8% in processing time per sensitivity index and 20% in memory usage when compared with EventTracker, its closest rival. The proposed methodology is more accurate and 80.9% faster than an entropy-based method. Its application is recommended for reinforced learning and/or formulating system key performance indicators from raw data.

**INDEX TERMS** Event Tracking, sensitivity analysis (SA), discrete event systems, input variable selection, real-time systems, distributed computing.

## NOMENCLATURE

| | |
|---|---|
| *AS* | Analysis Span. |
| *B* | Batch. |
| *CT* | Cut off threshold. |
| *ET* | Event Threshold. |
| *nSI* | normalized Sensitivity Index. |
| *SI* | Sensitivity Index. |
| *TT* | Trigger Threshold. |
| DAS | Dynamic Analysis Span. |
| ED | Event Data. |
| EMA | Exponential Moving Average. |
| FAS | Fixed Analysis Span. |
| GSA | Global Sensitivity Analysis. |
| IoT | Internet of Things. |
| LSA | Local Sensitivity Analysis. |
| MI | Mutual Information. |
| NMIFS | Normalized Mutual Information Feature Selection. |
| SA | Sensitivity Analysis. |
| SCADA | Supervisory Control and Data Acquisition. |
| TD | Trigger Data. |

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

## I. INTRODUCTION

The principal challenge of maintaining and continuously improving the efficiency and productivity of industrial processes is assembling the right knowledge as the material and machine evolve during the production life cycle. By establishing the appropriate methods and techniques of data acquisition, time factors and interrelationship between observable system parameters, the emergent intelligence helps to minimize material and energy consumption. The reinforced intelligence allows for increased quality and productivity of industrial products and processes (optimization problem). One big obstacle to optimization problems is the complexity of the majority of the systems or in other words capturing and interpreting the covariations between system parameters [1], [2]. A high amount of input variables (dimensionality) or high sampling rates need appropriate algorithms requiring a lot of computational resources within short periods of time (responsive systems).

In order to monitor, control or optimize complex systems more effectively, it is necessary to minimize the heavy data processing by reducing the number of variables and selecting the most relevant ones to the system's output. Insight about the input variables and how they impact the general performance of the process requires domain knowledge of experts with its own practical and economical limitations. However, to accurately determine the relationships between variables in real world applications is not an easy task. Sensitivity Analysis (SA) allows technicians to focus on the most significant variables, or in other words, variables that held the most information regarding the behavior of the system's output [3]–[5]. Regardless of the goal, SA takes advantage of the heuristic known as the ''sparsity of factors'', which states that only a small fraction of components in a system has a meaningful impact on a specific system output. In these conditions, industrial application of highly fractionated designs becomes a cost-effective option [6].

SA-based methodologies address a number of broad system analysis and modelling goals [5]: 1) causalities and the effects of various processes, theories, factors, scales, and their combinations and interactions on a system; 2) dimensionality reduction; 3) data worth evaluating to determine which processes, parameters, and scales dominantly influence a system; and 4) assess the sensitivity index of an expected outcome to decision support. A common and settled separation of SA methods arises from the local or global analysis of the problem space [7]. Local Sensitivity Analysis (LSA) is evaluated around a nominal point in the problem space. LSA is simple and intuitive but its use has been critiqued for offering merely a confined picture of the problem space [5]. In the modern era of SA, the focus has been on a concept known as Global Sensitivity Analysis (GSA) [8], as it tries to give a ''global'' picture of how the various components interact throughout the entire problem space to influence some function of the system output.

Some of the shortcomings in existing SA methods are related to lack of robustness when facing non-linear or heteroscedastic data, dependence on systems equations, reliance on historical data and complexity in algorithm's implementation. The computational burden of modern SA algorithms has been a key impediment to their applicability to real-world issues. Because they demand a high sample size, many of the available SA approaches have remained underutilized in many disciplines [5]. Through the study of complex industries, like manufacturing, automotive and aerospace industries, it is clear that the process of validation of SA models in real-time is expensive and a lot time-consuming [9]. Therefore, the industry needs an automated method working in real time that suits the data acquisition and simultaneously the data analysis in a raw state. Unaware casual relationship modelling methods, like EventTracker [10] and EventiC [9], attempt to address this problem by simplifying the SA process in a way that is both accurate and efficient, using an event-based sensitivity approach. The ease and speed with which EventTracker and EventiC take all accessible data from the system domain, convert and process the necessary information in near real time, distinguishes them from other automated sensitivity analyses. Besides, these strategies are unbiased because they don't rely on any prejudgment of data relevance, which is typically a characteristic of expert influence. However, both algorithms have a drawback in the normalization process of the sensitivity indices that can be misleading when ranking the relevance (by the sensitivity index) of the input variables. Their analysis span also limits the algorithms' output (sensitivity index) in terms of fast responsiveness and efficiency.

This paper proposes a new real-time sensitivity analysis methodology for complex industrial systems, the fastTracker. The main goal of fastTracker is to be an efficient tool that follows the process behavior's changes and correctly quantifies the sensitivity indices between input variables and the target of the system. fastTracker is based on EventTracker [10], where the following contributions were performed in order to be faster and more resource efficient compared to EventTracker:

- A new normalization process to provide a faster and more efficient normalized sensitivity index that better represents the cause-effect relationships between each input variable and the system target. The normalization process presented in EventTracker [10] doesn't properly isolate the input-target correlation at each moment of the system analysis, requiring the determination of all input-target correlations before being executed (inefficient use of memory).
- A real time analysis span is proposed in order to have a SA index following each data acquisition. EventTracker [10] considers a fixed analysis span, which means that the machine/expert has to wait for the entire analysis span in order to have a SA index (close to real-time rather than real-time). fastTracker proposes a dynamic analysis span that allows a responsive

SA procedure, generating outputs (index) when new data arrives.
- A faster and computationally cheaper algorithm is presented, deployed and tested in live industrial Supervisory control and data acquisition (SCADA) and IoT platforms. Although EventTracker is already fast and cheap when compared to other SA-based algorithms, fastTracker is faster 98.8% and uses less than 20% of memory when compared with EventTracker, and 80.9% faster than the competitor Mutual Information method.

To demonstrate the performance of the proposed fastTracker methodology, real data from cement industry is used for benchmarking purposes. Cement factories are complex systems with multiple objective functions, multivariable, and non-linear processes. The production has several sub-operations with a huge impact on the environment, energy consumption, and raw material usage. fastTracker was deployed in the SCADA system of the plants, its performance was then compared with EventTracker [10] and an entropy-based algorithm [11], same data same flow rate.

The remainder of this paper is organized as follows. In Section II existing SA approaches are discussed, followed by Section III that describes the proposed methodology and its implementation. In Section IV the performance of the proposed methodology is analyzed vis-à-vis an actual industrial case study. The algorithms performance is then compared with an entropy-based algorithm in the same section. The final section contains the conclusions and future work.

## II. RELATED WORK

To build an overview of the existing relevant body of knowledge in the subject area, the authors suggest and appraise a number of SA methods such as regression-based, derivative-based, distribution-based, variogram-based, heuristic-based, and unaware causal relationship modelling methods.

### A. REGRESSION-BASED APPROACH

Traditionally, regression-based methods infer sensitivity information from coefficients of a generally linear regression model fitted to a sample of model response surface points [5]. Each term in the regression model might have a separate linear or nonlinear basis function. From an LSA point of view, these methods have proven beneficial for dimensionality reduction via orthogonal decompositions from parameter samples [12] or locally approximated sensitivity matrices [13]. From a GSA perspective, these methods have been chastised for their excessive reliance on previous assumptions about model response form and the fact that if the fit quality is inadequate, the sensitivity estimates are unreliable. Regression analysis has several potential downsides, including a lack of robustness if crucial regression assumptions are not met, the necessity to assume a functional form for the relationship between output and selected inputs, and potential ambiguities in interpretation [14].

### B. DERIVATIVE-BASED APPROACH

Derivative-based approaches are a straightforward extension of LSA, in which local sensitivity measurements are performed by computing derivatives analytically or numerically at many different base points over factor space and then averaged to offer a global sensitivity assessment [5]. Methods such as Neumann expansion [15] and perturbation [16] methods can help to extract these coefficients by approximating differential equations. Green's function is another example, which minimizes the number of differential equations for solving the SA and replaces them with integrals that can be easily calculated [17]. However, complex and nonlinear relationships between system variables cannot always be guaranteed to exist [9]. Besides, when using analytical methods that require access to the governing model equations, it may be computational impractical and sometimes impossible [17].

### C. DISTRIBUTION-BASED APPROACH

Distribution-based approaches follow a different concept, focusing on the output's distributional characteristics [5]. The assessment of output variance is the most popular distribution-based method. The variance-based SA was initially conceptualized in terms of non-linear dependence [18], then in terms of Fourier analysis [19], and Ilya Sobol set out the entire variance-based SA framework [20]. These methods are applicable independently of the characteristics of the input-output response function (e.g. linear or non-linear); they can be used for both input ranking and screening; and they are easy to implement and interpret. One of the most significant drawbacks of variance-based sensitivity indices is that they implicitly presume that output variance is a realistic measure of output uncertainty, which is not necessarily the case. Using variance as a proxy for uncertainty may produce contradicting conclusions, for example, if the output distribution is multi-modal or strongly skewed [21].

Other distribution-based approaches are ''moment-independent'', in that they quantify the difference between the unconditional and conditional distributions of the output while one or more inputs are fixed [5]. These methods are also referred to as ''density-based'' methods based on the Probability Density Function (PDF). For example, the method of Borgonovo [22] looks at the influence of input uncertainty on the entire output distribution without reference to a specific moment of the output. Entropy-based SA also belongs to the category of density-based SA, where uncertainty is characterized by examining the entire distribution of model outputs, not just its variance. The use of entropy instead of variance is usually justified by the need to analyze the output random variable with heavy-tail or outliers [23]. However, density-based indices are more difficult to implement than variance-based ones, owing to the fact that their computation demands the understanding of a large number of conditional PDFs [21].

### D. VARIOGRAM-BASED APPROACH

Variogram-based methods are based on the notion of variograms and have recently arisen, bridging the derivative and distribution-based methods [24]. This approach is based on the fact that model outputs are not necessarily randomly distributed, and they, like their partial derivatives, have a spatially-ordered (co-variance) structure in the input space. Anisotropic variograms can characterize this structure by quantifying the variance of change in the output as a function of perturbation size in individual inputs.

The Variogram Analysis of Response Surfaces (VARS) [25] deploys a multi-method approach that enables simultaneous generation of a range of sensitivity indices, including ones based on derivative, variance, and variogram concepts, from a single sample. However, this is still a very recent method and there is little evidence of its deployment and usefulness in practice [26].

### E. HEURISTIC-BASED APPROACH

Heuristic techniques are capable of data-filtering and SA through measuring the true impact of each system input on each output. Heuristic methods are typically used when actual knowledge of the system is not based on empirical studies protocols and normally rely on system experts engineering or modelling, knowledge, experience, and intervention [9]. Fuzzy inference models (FIM) are an example of heuristic-based methods. However, expert knowledge has a direct impact on determining the fuzzy inference rules that maximize the manufacturing process's key performance indicators [27]. In order to reduce the dependency on domain expert intervention, automatic artificial intelligence (AI) based learning methods have also been used to infer FIM [28], [29]. The relevant AI approaches analyze the pattern of acquired data and produce the information required for measurement or optimization.

The effectiveness of heuristic methods in tackling complicated data modelling and control systems has been thoroughly documented in both industry and literature. Nonetheless, heuristic methods are strongly problem-dependent, which means they depend on objective function, equality and inequality constraints considered in the problem [30].

### F. UNAWARE CAUSAL RELATIONSHIP MODELLING METHODS

Event-based sensitivity analysis approach [31] is another method for input variable selection for time-constrained applications. The purpose is to identify the inputs that have a true impact on a given output by mapping the cause-effect relationship between system parameters (input-output). Two methods capable of delineating those relationships are EventTracker [10], and EventiC [9]. Both techniques are based on the principle that any observable input has an effect on the system output unless proven otherwise.

In [10] a fast algorithm is proposed that is able to give reliable linear and non-linear local cause-effect relationships between multiple input variables and one output variable, the EventTracker algorithm. EventTracker is proposed to be used to select a set of variables of interest in problems where there are abrupt, unexpected, and difficult to predict changes. For the purposes of sensitivity analysis, this method can create a discrete event framework in which events are loosely associated with their triggers (i.e. cause). An improvement of EventTracker is proposed in [9], the EventiC, that reveals the correlations between multiple input and multiple outputs, allowing data analysts to see the connections between groupings of input and output right away. The key benefit of adopting this unaware causal relationship modelling approach over other SA methods is that it intends to grasp and interpret system state change in the shortest amount of time with the least amount of computational overhead [31], making it affordable and suitable for industrial monitoring and control applications.

Through the study of complex industries, like manufacturing, automotive and aerospace industries, it is clear that the process of validation of SA models in real-time is expensive, time-consuming and challengable [9]. The primary motivation of this research study is to propose an improved Event Modelling technique (i.e. EventTracker [10] and EventiC [9]) to overcome the challenges and improve the shortcoming of the existing SA models. The industry needs an automated method working in real-time that suits the data acquisition and analysis in a raw state. Unaware casual relationship modelling methods, like EventTracker and EventiC, attempt to address this problem by simplifying the SA process in a way that is both accurate and efficient, using an event-based sensitivity approach, fastTracker is an improvement on the two methods.

## III. PROPOSED EVENT TRACKING METHODOLOGY

In order to reach a fast and reliable methodology to identify the local cause-effect relationships between multiple input variables and a target variable/indicator for complex systems characterized by a high amount of input variables (the curse of dimensionality) or very high sampling rates, this section presents the proposed event tracking methodology whose main feature is being very low computational cost and with fast event tracking within short periods of time.

The proposed event tracking methodology is an improvement of the methodologies EventTracker [10] and EventiC [9]. Although EventTracker and EventiC are fastest and less computationally heavy than similar sensitivity analysis methods (like Entropy-based Sensitivity Analysis) [10], the algorithms present limitations to work in real-time and online executions within short periods of time, mainly by their dependence on the fixed analysis span to give a final value of normalized Sensitivity Index ($nSI$). The biggest problem of a fixed analysis period is the lack of prompt answers to new trigger/event values. With the formulation of the algorithms present in [9], [10], the expert (human or machine) has to

wait until the end of each analysis span in order to have one *nSI* value per input variable, which means there is no way of interpreting the system within each analysis span. Besides this disadvantage, the normalization process of the algorithm is not well suited for a cause-effect relationship between input and output variables, once the *nSI* value is affected by other variables' sensitivity indices during the normalization method. Taking into account the above mentioned limitations with the EventTracker and EventiC formulations, this section presents the proposed solution to adapt these algorithms to an online, fast and efficient sensitivity analysis methodology, capable of providing real-time normalized Sensitivity Index values to the system analyst.

A brief formulation of the original EventTracker algorithm is presented in Section III-A. The normalization problem is assessed in Section III-B, where the new formulation ensures that *nSI* values of one variable are not influenced by other variables' sensitivity indices. A real time analysis span is proposed in Section III-C, using a pop and push formulation to determine Sensitivity Index, *SI*, values. Lastly, the algorithm's efficiency is improved using a proposed exponential moving average window, as shown in Section III-D.

### A. EVENTTRACKER FORMULATION

EventTracker [10] employs pair-wise event coincidence analysis. Its approach is based on an event-based SA technique that links input data to target variables in a discrete event system, where the state of the system changes when the input variables and consequently the outputs of the system change. The system's state transitions are referred to as events. Trigger Data (TD) refers to any input variable whose value causes an event to be registered. Event Data (ED) represents the state of a system at a specific point in time.

The EventTracker algorithm is built around four functional parameters that are set by a domain expert, 1) search slot, 2) analysis span, 3) event threshold, and 4) trigger threshold:

- **Search slot** can be described as the scan rate, being a set period of time during which TD and ED batches, *B*, are captured.
- **Analysis span**, *AS*, is the time span within which a period of sensitivity analysis occurs. An analysis span is comprised of a number of consecutive batches of data. At the end of an AS, the several TD and ED batches within that analysis span will be used to determine the normalized sensitivity index, *nSI*, of the cause-effect relationship between the input variable and the output (target) of the system.
- **Event Threshold**, *ET*, and **Trigger Threshold**, *TT*, are two scalar values that determine whether a variable represents a real change in the system state or not, by comparing two consecutive batches for the same variable.

In brief, the methodology follows four steps, depicted in Fig. 1:

**Step 1.** Two consecutive batches, $B_{t-1}$ and $B_t$, of trigger data (inputs) and event data (output) are first analyzed for trigger-event detection, where $t$ is the instant of time.

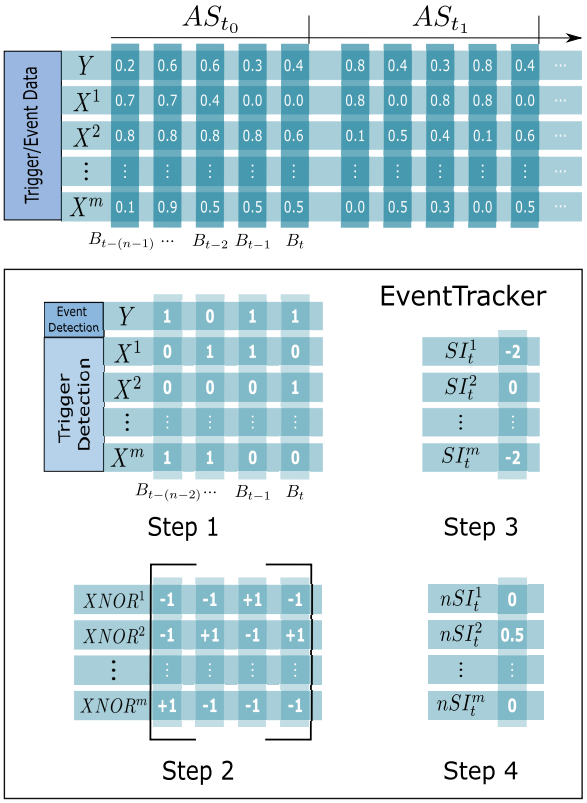



**FIGURE 1.** EventTracker methodology [10]. Step by step implementation for one analysis span. Demonstrative values.

The batch $B_t$ contains one observation per input variable $X^j_{B_t}$, where $j = 1, \ldots, m$ and $X_{B_t} = \left[X^1_{B_t}, \ldots, X^m_{B_t}\right]$, and output (target) variable $Y_{B_t}$ for the instant of time $t$. Trigger and event thresholds are used to spot the presence or absence of triggers and events by:

$$if \left|X^j_{B_t} - X^j_{B_{t-1}}\right| \geq TT^j \rightarrow X^j_{B_t} = Trigger, \quad (1)$$

$$if \left|Y_{B_t} - Y_{B_{t-1}}\right| \geq ET \rightarrow Y_{B_t} = Event, \quad (2)$$

where $X^j_{B_{t-1}}$ and $X^j_{B_t}$ are the values of the input variable $X^j$ in the batches $t-1$ and $t$, respectively, and $Y_{B_t}$ is the value of the output (target) variable in the batch $t$. For a given variable (input or the target), if the values of two consecutive batches are superior to its threshold, it is considered a trigger or an event at that instant of time, respectively.

**Step 2.** A inputs/target coincidence matrix is produced (see Fig. 1) based on the simultaneous existence or nonexistence of a change in each batch. Score $+1$ is given if there are or there aren't coexistent changes in both trigger and event data, i.e.,

$$if \left(X^j_{B_t} = Trigger \wedge Y_{B_t} = Event\right) \vee (X^j_{B_t} \neq Trigger \wedge Y_{B_t} \neq Event) \rightarrow XNOR^j_t = +1, \quad (3)$$

otherwise the score is $-1$,

$$if \left(X^j_{B_t} = Trigger \wedge Y_{B_t} \neq Event\right) \vee (X^j_{B_t} \neq Trigger \wedge Y_{B_t} = Event) \rightarrow XNOR^j_t = -1. \quad (4)$$

This score will be named *XNOR* value on the following subsections, since the operation is similar to a weighted logical Exclusive-NOR.

**Step 3.** The $XNOR^j$ ($j = 1, \ldots, m$) values are summed up within each analysis span (AS) at instant of time $t$, considering all the consecutive $n$ batches in an analysis span. In this way, a sensitivity index, $SI^j$ for the relationship between the $j$-th input variable and the target is obtained at the end of $n$ batches by:

$$SI_t^j = \sum_{l=0}^{n-2} XNOR_{t-l}^j. \tag{5}$$

**Step 4.** The sensitivity index, $SI_t^j$, is then normalized to the unit range, according to

$$nSI_t^j = \frac{SI_t^j - l_b}{u_b - l_b}, \tag{6}$$

where the normalized sensitivity index for each analysis span, $nSI_t^j$, is obtained from the $SI_t^j$ value, considering the lower $l_b = \min_{j=1,\ldots,m} SI_t^j$ and upper $u_b = \max_{j=1,\ldots,m} SI_t^j$ bounds of the set of all sensitivity indices (of all input-target pairs) $SI_t^j$ ($j = 1, \ldots, m$) obtained in the analysis span.

After the period of sensitivity analysis, based on the determined sensitivity indices, the least relevant inputs are filtered out by defining a cut off threshold, $CT$. The results have to be validated and verified utilizing a false-negative testing technique [9].

### B. PROPOSED NORMALIZATION

The normalization process, presented in [9], [10], and described in Section III-A, aims to transform the final value of the sensitivity indices to a value between 0 and 1, where 1 represents perfect correlation and 0 represents no correlation between the trigger data (TD) and event data (ED). The normalization defined in (6) has three main limitations. First, at every batch of data $B_t$, the normalization process will only occur after the maximum and minimum $SI_t$ are found for all inputs variables. In other words, when a new batch of data arrives, first the $SI_t$ values have to be calculated for all input variables $X_{B_t}^j$, next it's determined the maximum and minimum of those values and afterwards it is calculated the $nSI_t$ value for each variable using equation (6). The search of the maximum and minimum $SI$ values slows down the process and will also consume extra memory. Second, this formulation will calculate normalized values of $SI$ which can be misleading and ambiguous, since the same value of $SI$, for a given input-output pair and for two different batches, can have different normalized $SI$, $nSI$, values. This situation can happen because the lower and upper limits, $l_b$ and $u_b$, of $SI$ values can change at every batch, affecting the normalization process. Lastly, as the formulation takes into account the upper and lower limits, $l_b$ and $u_b$, for the set of all sensitivity indices, the $nSI$ value of one input-output pair will depend on the $SI$ values of other inputs-output pairs (dependency among trigger data of different inputs). These bounds seem to weaken the EventTracker and EventiC algorithms as the normalization process suffers from undesirable peaks and troughs, thus disrupting the smooth and uninterrupted SA.

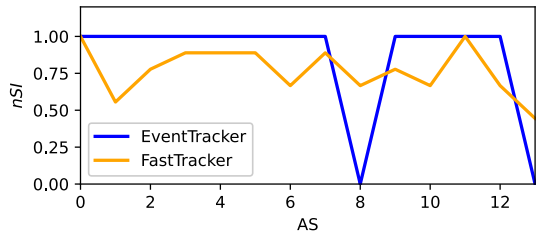



**FIGURE 2.** Normalized sensitivity indices, *nSI*, from EventTracker (6) and from fastTracker (7). Demonstrative values.

Therefore, the EventTracker algorithm suffers from its normalization formula, which should have the ability of being applied at each new batch without any dependence among different variables' sensitivity indices. In order to solve the aforementioned problems, a new formulation is proposed by equation (7) which improves the normalization present in EventTracker, being based only on the size of the analysis span, $n$, and on the $SI^j$ value that is being normalized:

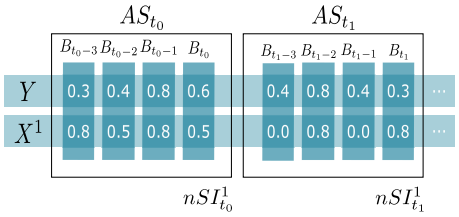
$$nSI_t^j = \frac{SI_t^j + (n-1)}{2 \times (n-1)}, \tag{7}$$

where $nS_t^j$ is the normalized sensitivity index for the relationship between the $j$-th input variable and the target at instant of time $t$, and $n$ is the number of batches in the analysis span (AS). Looking to (7), if there's not a single causal-effect relationship between trigger and event data within the whole analysis span, the value of $nSI_t^j$ will be 0. In this case, all $XNOR$ values are equal to $-1$, so according to (5), $SI_t^j = -(n-1)$. Otherwise, if a perfect correlation between trigger and event data occurs, the value of $nSI_t^j$ will be 1, since every $XNOR$ value will be $+1$, which means $SI_t^j$ is equal to $(n-1)$.

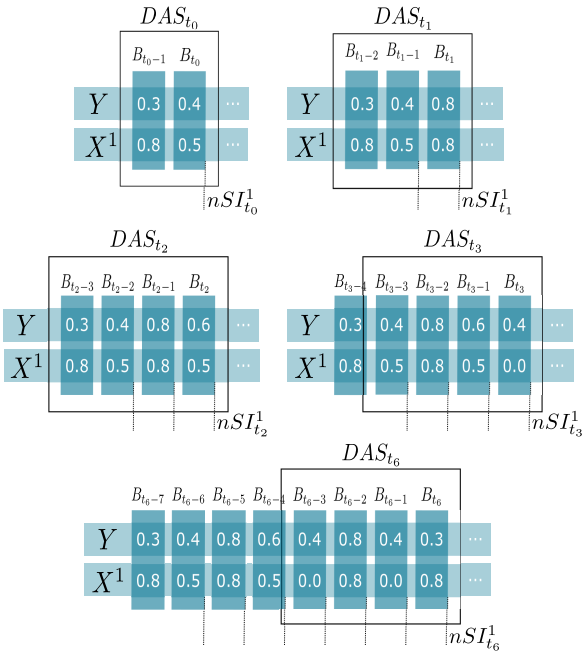
Fig. 2 shows how the new normalization formula (7) leads to a continuous and more informative SA, when faced with the original EventTracker methodology (6), which leads to crests and valleys.

### C. REAL TIME ANALYSIS SPAN

A weakness of EventTracker consists in not being suitable for real time analysis of new batches of data, because the algorithm needs to reach the end of an analysis span in order to give a value of $nSI$ for each input variable. This is not the desired behavior of an algorithm which is intended to work in real-time industry processes. Sensitivity indices should be meaningful for every cause-effect relationship between triggers and events, without the breakages between contiguous analysis spans. Instead of using a fixed analysis span, that translates in a waiting time of the $AS$'s size, i.e., $n$ batches (Fig. 3(a)), the proposed methodology fastTracker uses a dynamic analysis span (DAS), which works as a moving window through the new batches of data that flow into the algorithm (Fig. 3(b)). This way, fastTracker will be able to




(a) Fixed Analysis Span ($n = 4$) from EventTracker.




(b) Dynamic Analysis Span ($n = 4$) from fastTracker.

**FIGURE 3. Analysis span (AS): 3(a) presents a fixed AS (used in EventTracker [9], [10]), and 3(b) the dynamic AS (used in fastTracker). Example for 8 batches of TD/ED.**

give a new value of *nSI* when a new batch of data is fed to the algorithm. Considering the example presented in Fig. 3, EventTracker will only return two values of *nSI* ($nSI_{t_0}$ and $nSI_{t_1}$), and by using a dynamic SA seven values of *nSI* will be obtained ($DAS_{t_0}, \ldots, DAS_{t_6}$). Hence, fastTracker will produce one normalized sensitivity index for each new batch of trigger/event data that pops into EventTracker, instead of having just one value of *nSI* per analysis span ($n$ batches).

In order to have the lowest possible amount of processing, the proposed methodology uses a *pop* and *push* expression to determine the *SI* value when a new batch of data is fed to fastTracker:

$$SI^j_t = SI^j_{t-1} - XNOR^j_{t-w} + XNOR^j_t, \tag{8}$$

where $SI^j_{t-1}$ is the sensitivity index at the previous batch, $w = n - 1$ is the number of *XNOR* values in the DAS, $XNOR^j_{t-w}$ is the *XNOR* value that pops from the array which contains *XNOR* values and $XNOR^j_t$ is the *XNOR* value that is pushed into the respective array.

As the *DAS* is moving through the incoming batches, instead of summing all the *XNOR* values for each batch in the *DAS*, the proposed methodology simply subtracts the oldest value, $XNOR^j_{t-w}$, and sums the new one, $XNOR^j_t$, to the previous *SI* value, $SI^j_{t-1}$.

Using the presented real-time dynamic analysis span, the normalization process of fastTracker will follow the rationale of Section III-B. Once the number of *XNOR* values is $w = n - 1$ in an analysis span, the normalization formula is now given by:

$$nSI^j_t = \frac{SI^j_t + w}{2 \times w}, \tag{9}$$

where $nSI^j_t$ is the value of normalized SI, $SI^j_t$ is the sensitivity index at the new batch $t$, and $w$ is the number of *XNOR* values in the *DAS*.

### D. A MORE EFFICIENT fastTracker

Memory and processing power are two important resources in real-time algorithms. Section III-C presents a formulation to determine sensitivity indices based on a dynamic analysis span (8), in which it isn't necessary to iterate over the array of *XNOR* values to determine the SI, as happened in EventTracker. However, (8) requires the storage of an XNOR array updated at every new causal-effect relationship between trigger and event data, which consequentially consumes extra memory.

With the purpose of improving memory efficiency, fastTracker determines the *SI* values by using an iterative sum algorithm with a low memory profile. In other words, it is intended to get the sensitivity index with the array of *XNOR* values without using extra memory to save that array at every new batch of trigger/event data. Considering an array of *XNOR* values, whose sum is the *SI* value, by definition of mean value it is possible to get the sensitivity index using equation (10):

$$SI^j_t = \overline{XNOR^j_t} \times w, \tag{10}$$

where the sum of the elements of the array, $SI^j_t$, is equal to its mean value, $\overline{XNOR^j_t}$, times the number of elements of the array, $w$. In order to obtain the mean value of the *XNOR*'s array it is used the exponential moving average (EMA). With time series data (in EventTraker's case, with consecutive batches of data), EMA is widely used to smooth out short-term swings and highlight longer-term trends [32]. This method is known to be a very efficient means of forecasting an outcome and it works as a way of obtaining a weighted mean value of an array by giving more relevance to recent values. EMA is simple, quick and resourcefully cheap once it gives the mean value of an array, $\overline{XNOR^j_t}$, based only on the latest mean value, $\overline{XNOR^j_{t-1}}$, and the new value of the variable of interest, $XNOR^j_t$. Equation (11) formulates the application of EMA to fastTracker's algorithm:

$$\overline{XNOR^j_t} = (XNOR^j_t - \overline{XNOR^j_{t-1}}) \times EMA_{factor} + \overline{XNOR^j_{t-1}}, \tag{11}$$

**Algorithm 1** Proposed Real-Time Event Tracking Methodology, the FastTracker

**Inputs** : The input $X = \left[X^1, \ldots, X^m\right]$ and target $Y$ variables; the trigger thresholds $TT^j$ $(j = 1, \ldots, m)$, and event threshold $ET$; and the number of batches per analysis span, $n$.

**Outputs** : Normalized sensitivity indices, $nSI^j$ $(j = 1, \ldots, m)$.

**Procedure**:

**for** $t = 0, 1, 2, \ldots$ *(until sensitive analysis is turned off)* **do**

  **for** ***all*** *inputs variables* $X_t^j$ $(j = 1, \ldots, m)$ **do**

    1. Assess the presence or absence of triggers and events according to (1) and (2).
    2. Determine the *XNOR* (3)–(4) value as explained in Section III-A.
    3. Calculate the sensitive index $SI_t^j$ using (12).
    4. Compute the normalized sensitivity index, $nSI_t^j$, using (9).

where $EMA_{factor}$ is the multiplying smoothing factor defined as $\frac{2}{(w+1)}$. Thus, we'll have an expression for the sensitivity index of the batch $t$, $SI_t^j$, that only uses the last mean value of *XNOR* determined by EMA and the new value of *XNOR* (12):

$$\begin{aligned} SI_t^j &= \overline{XNOR_t^j} \times w \\ &= \left[(XNOR_t^j - \overline{XNOR_{t-1}^j}) \times EMA_{factor} \right. \\ &\quad \left. + \overline{XNOR_{t-1}^j} \right] \times w. \end{aligned} \tag{12}$$

### E. ALGORITHM

A summary of the fastTracker algorithm for an efficient real time causal-effect sensitivity analysis between multiple input variables and a target variable is presented here. Algorithm 1 presents the steps of the proposed methodology fastTracker.

Regarding the time and space complexity, fastTracker algorithm takes in consideration the sensitivity analysis between a pair of variables, taking a long temporal series of values from both variables. Let's denote the pair by $(X_t^j, Y_t)$ and denote by $n$ the number of elements in the temporal series. From the algorithm analysis, over the processing of all temporal series, the memory footprint will be constant with the processing time dependent from the cardinality of the set. This leads to space complexity of $O(1)$ and time complexity of $O(n)$ [33]. fastTracker can expand horizontally by increasing the number of variables $(X_t^1, \ldots, X_t^m)$ correlating with a specified variable $Y_t$. Denote the group of variables by $(X_t^1, \ldots, X_t^m, Y_t)$ and consider a temporal series, with cardinality $n$. It is most common to observe $n \gg m$, leading the complexity analysis to be dependent on $n$. The memory footprint will be constant throughout the temporal series processing, while the processing time will be dependent on the number of pairing groups in the data series. The analysis makes the same conclusion for space complexity to be $O(1)$ and time complexity to be $O(n)$.

## IV. INDUSTRIAL CASE STUDY

fastTracker was applied to the monitoring and control system of cement plant. The data from the suite of sensors, actuation, Human Machine Interface, and control logics were automatically fed into the algorithm. The objective functions determined by plant operations management were continuous product quality stabilization, environmental impact, energy consumption, and raw material usage to be optimized, and they were formulated by inferential models and direct readings from sensors [9]. The cement production process is a multi-variable, complex and nonlinear process, consisting of several sub-operations: quarrying, crushing, raw milling, burning, cooling, and cement grinding [34]. The control of the grinding process is crucial and represents a big challenge because of its multi-factor interdependencies and nonlinearities characteristics.

Improving the cement grinding process efficiency synchronously with minimizing the energy and processing power consumption has a major impact on good overall cement manufacturing process. Energy consumption accounts for about 75% of the cost of cement manufacture, and grinding systems with consumption equal to, or more than $2MW/h$ are typical in the cement industry [35]. Thus, tiny increases in the grinding system's efficiency result in huge energy savings. fastTracker is used to evaluate the relationship between events (target variable) and triggers (input variables), contributing to the selection of the most relevant input variables. Consequently, the efficiency of the process can be improved by integrating techniques to monitor, optimize and control the cement grinding process.

The proposed fastTracker is compared with the EventTracker algorithm, and with a mutual information algorithm [11]. In Section IV-B are shown the results for the new normalization formula (7). The performance of a dynamic analysis span (8) is presented in Section IV-C. The efficiency of the final methodology, using the exponential moving average (12), is given in Section IV-D.

### A. CEMENT GRINDING PROCESS AND DATASET

A simplified scheme of the cement grinding process is presented in Fig. 4. There is a flow of raw material into the mill, which is then grounded by steel balls [9]. Afterwards, the outcome of the mill is transported by a bucket elevator into the separator that will divide that outcome product into a) a flow of fine particles (which will form the cement) and b) a flow of oversized particles (designated coarse return).

The coarse return variable is one of the most important outputs in cement production, namely for the quality of the final product and plays a major role in the optimization and control of the cement manufacturing process. The process can be optimized if the interaction effects of mill and separator factors on coarse return are understood. However, the grinding process is complex and unstable, with nonlinear characteristics caused by natural variation, mill load, and fluctuation of the raw material [36]. As a result, it's critical

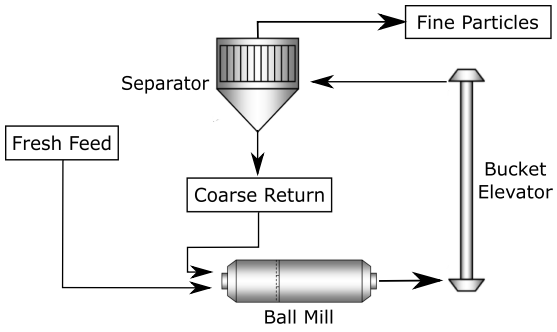



**FIGURE 4.** Scheme of the cement grinding process. Adapted from [37].

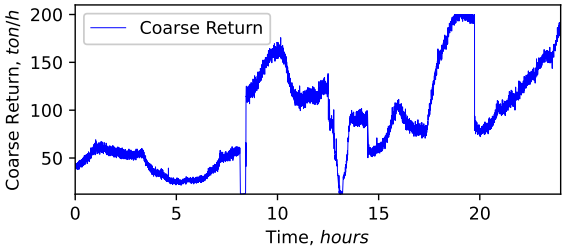



**FIGURE 5.** Target variable (coarse-return) during a day of cement grinding.

to keep track of and rapidly identify which variables are most relevant to the target's variable (coarse return).

The dataset was collected from a real grinding process plant, it is composed by a total of 44 input variables and the target variable, the coarse-return, and 119996 samples that represent approximately 23 days. A month data collection campaign and pre-processing of the data, showed 23 days provided sufficient empirical justification covering the full production life cycle. The number of days of analysis provided a sufficient sample size for statistical confidence (i.e. 95% confidence interval) testing and verification. In Fig. 5, it can be seen the behavior of the output variable during a day of cement grinding. The changes in the coarse return are noticeable and represent the volatility of the process. Following the industry experts' approval, the eight highest-ranked input variables over the output should be: Mill Power Transducer, Fresh Feed, Recycle Elevator Current, Separator Amps, Bucket Elevator Amps, Mill Scan Inlet, Separator Actual Speed and Mill Scan Outlet.

### B. RESULTS

Table 1 presents the *nSI* and *G* criterion for the highest relevant variables, according to EventTracker and fastTracker, and Normalized Mutual Information Feature Selection (NMIFS), respectively. The results of fastTracker and EventTracker were obtained with trigger and event thresholds of 0%, which means every change in the variable's data reflects a trigger/event. This threshold value was chosen so it could be seen the relationship with coarse return at the minimum variation possible. The chosen analysis span was

**TABLE 1.** Highest relevant variables according to normalized Sensitivity Indices, *nSI*, for EventTracker [10] and fastTracker, and to *G* criterion of normalized mutual information feature selection (NMIFS). OR means Order of Relevance by the respective algorithm.

| | EventiC | | Proposed Normalization | | NMIFS | |
|---|---|---|---|---|---|---|
| Variable's Name | nSI | OR* | nSI | OR* | G | OR* |
| Mill Power Transducer | 0.902 | 1 | 0.913 | 1 | - | - |
| Fresh Feed | 0.891 | 2 | 0.901 | 5 | 0.293 | 8 |
| Recycle Elevator Current | 0.883 | 3 | 0.904 | 2 | 0.551 | 5 |
| Separator Amps | 0.879 | 4 | 0.902 | 3 | 0.63 | 4 |
| Bucket Elevator Amps | 0.877 | 5 | 0.902 | 4 | 0.519 | 6 |
| Mill Scan Inlet | 0.841 | 6 | 0.845 | 7 | 0.714 | 3 |
| Separator Actual Speed | 0.831 | 7 | 0.846 | 6 | - | - |
| Mill Scan Outlet | 0.789 | 8 | 0.788 | 8 | 0.325 | 7 |
| Spare Variable 2 | - | - | - | - | 1.962 | 1 |
| SpareVariable1 | - | - | - | - | 0.837 | 2 |

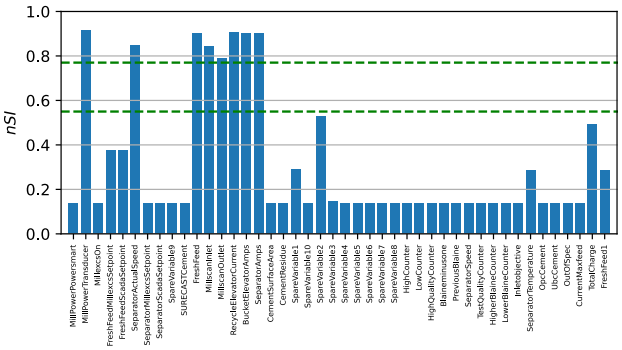
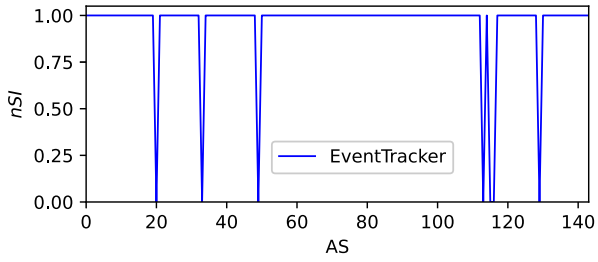


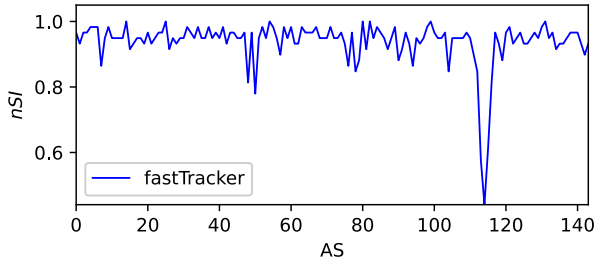

**FIGURE 6.** Normalized Sensitivity Indices, *nSI*, of fastTracker.

$n = 60$ batches of TD/ED, that according with the system experts corresponds to a reasonable interval for detecting changes in the grinding process.

Fig. 6 shows the final normalized sensitivity indices of fastTracker in order to do a false negative test to measure and eliminate false negatives. A false negative happens when the algorithm considers a high correlation variable as a low relevance variable, following the industry experts' approval. The bar plot present in Fig. 6 shows that a cut off threshold, $CT$, between 55% and 77% (green dashed lines) will result in a 0/8 ratio of false negatives (according to the expert human operators). Higher $CT$ values will have higher ratios of false negatives because there will be a higher percentage of filtered input variables, considered as low-relevance variables. Lower $CT$ values will result in a 0/8 ratio of false negatives, but in turn there will be many false positives, i.e., low relevance

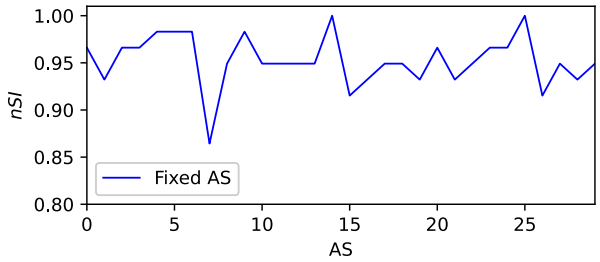



(a) EventTracker normalization.

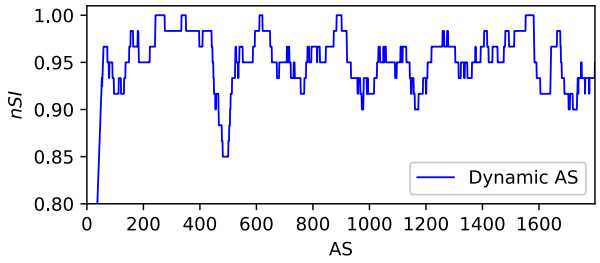



(b) fastTracker normalization.

**FIGURE 7.** Results for the first 144 analysis spans of the Fresh Feed variable (considered by EventTracker and fastTracker as a high relevance variable): (a) EventTraker normalization, eq. (6), and (b) fastTraker normalization, eq. (7).




(a) Fixed Analysis Span.




(b) Dynamic Analysis Span.

**FIGURE 8.** Relationship (*nSI*) between Fresh Feed and the target variable Coarse Return, for a 5 hours cement grinding process. The Fig. presents real-time sensitivity analysis results, using: (a) a fixed *AS* from EventTracker, and (b) a dynamic *AS* from fastTracker.

variables (defined by the experts) will be selected by fastTracker as high relevance variables.

Besides a quantitative validation, Fig. 7 illustrates the need for a new solution in what comes to the normalization process. Fig. 7 shows the relationship between Coarse Return and a high relevance variable, Fresh Feed, during a day of cement grinding process. By analyzing Fig. 7, at each analysis span the fastTracker normalization procedure presents a more balanced and realistic evolution of the respective input-target relationship than the EventTraker algorithm. The gain of information relatively to EventTraker's normalization is evident in the experiments and results.

### C. REAL TIME SENSITIVITY ANALYSIS

In this section, the dynamic analysis span (*DAS*) used on fastTracker is studied on the dataset in a real-time analysis scenario, where the incoming batches of ED/TD were interpreted as soon as they were fed to the algorithm. The results demonstrate the importance of an online sensitivity analysis, instead of using a fixed analysis span as presented in EventTracker. Fig. 8 presents the online evolution of the normalized sensitivity analysis *nSI* between Fresh Feed and Coarse Return variables, for a fixed analysis span (EventTracker) and a dynamic analysis span (fastTracker), during 5 hours of cement grinding process. It was used the same analysis span size in both scenarios, equal to $n = 60$ batches of $ED/TD$. In Fig. 8, it is visible the gain of information when comparing the dynamic *AS* (DAS) and the fixed *AS* (FAS) results, where 1799 *nSI* values are provided by fastTracker (using a DAS) and 30 *nSI* values are provided by EventTracker (using a FAS) for the same number of ED/TD batches. In fastTracker, the number of *nSI* values is increased by a factor equal to the analysis span's size, which will translate in more information, almost instantaneously ready to be understood by production managers. This means that at each new batch of incoming data, a new value of sensitivity index is generated, giving the experts/the machine the ability to interpret the relevance of each input variable, instead of having to wait for the end of an analysis span.

Comparing the initial values of *nSI* for EventTracker and fastTracker methods (Fig. 8), they differ significantly at the beginning of the sensitivity analysis, due to the fastTracker normalization equation (9), which can't give reliable results until the full *AS* fill; in this case, until reaching 60 batches. However, this is not an issue, because this initial peak when using a dynamic *AS* will only happen at the beginning of the sensitivity analysis and will not have an influence on posterior values. Moreover, the duration of this initial peak is just 1 *AS*, like when using a fixed *AS* as done in EventTracker. The dynamic *AS* (12) for the determination of *SI*, which is computationally efficient, allows the algorithm to run smoothly, and at the same time providing the required outcome to evaluate the immediate cause-effect relationships between the trigger data and the coarse return (event data). This trade-off bounded by a simplified algorithm and an immediate gain of information allows fastTracker to do real-time analysis of events, which will consequentially have an impact on the environmentally quality and energy effectiveness of the

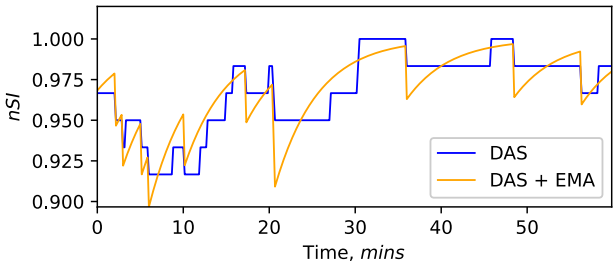



**FIGURE 9.** Sensitivity analysis results using dynamic AS with and without EMA for the relationship (*nSI*) between Fresh Feed and Coarse Return variables, during 1 hour of the cement grinding process. AS size $n = 60$.

**TABLE 2.** Processing time and peak memory usage of EventTracker [10], the proposed improvements on normalization and dynamic analysis span, and of fastTracker.

| Algorithm | Processing Time per nSI (ms) | Peak Memory Usage (kB) |
|---|---|---|
| EventTracker | 6.486 | 5.296 |
| New Normalization | 5.307 | 4.176 |
| Dynamic Analysis Span | 0.088 | 28.844 |
| fastTracker | 0.076 | 4.229 |

cement grinding process (optimizing the control objective functions).

### D. ASSESSMENT OF EFFICIENCY AND VALIDATION OF FASTTRACKER

Any SA computational performance must be evaluated in terms of four factors: efficiency, convergence, reliability, and robustness [5]. The amount of time/number of computations required to conduct SA is referred to as efficiency. Convergence of a SA algorithm depends on the state space definition (i.e., the number of available parameters). Any measure of the correctness of SA results is referred to as reliability, and its accurate assessment necessitates the availability of the ''true'' SA results. When it comes to robustness, a SA method is said to be resilient to sampling variability if its results are almost similar when applied to two different sample sets from the same model.

Fig. 9 presents the relationship between a high correlation variable, Fresh Feed, and the output, Coarse Return, during 1 hour of the cement grinding process. It is shown the smoothing of the edges when using the exponential moving average (EMA), which reflect its typical behavior. This means that the ''optimizer'' (human operator or automatic system) is fed by fastTracker in order to avoid abrupt interruption between each batch of data.

The efficiency of the fastTracker was compared with the processing times and memory usage of EventTracker [10], the implementation of the normalization process presented in Section III-B, and with the use of a dynamic analysis span without using the exponential moving average (as presented in Section III-C). Since the processing time differs accordingly to the device in which the algorithms are processed, the most important feature to analyze the efficiency is the relative time reduction between each algorithm. The results are shown in Table 2, in which it can be seen an improvement from a 6.486ms to a 0.076ms processing time per *nSI* value, which is achieved by using fastTracker. This means a decrease of an astonishing 98.8% in the processing time from the EventTracker algorithm to fastTracker. The implementation of a dynamic analysis span is the main reason for the reduction in processing time per *nSI* (from 5.307ms to 0.088ms). fastTracker can now generate an output (*nSI*) for each new batch of incoming data, avoiding the need to wait until the end of an analysis span. Therefore, for the same period of sensitivity analysis, many more *nSI* values are going to be determined, reducing the processing time per *nSI*.

In terms of memory usage when computing *nSI* values, Table 2 shows a decrease of 20% from EventTracker (5.296kB) to fastTracker (4.229kB). The importance of applying the EMA to the algorithm is evident when analyzing the peak memory usage for the DAS (28.844kB). By deploying the EMA, the array of *XNOR* values used in eq. (8) doesn't need to be stored and updated at every new batch of data, which results in a decrease of 85% of memory usage. The proposed changes to the EventTracker algorithm, therefore, provide a better quality of information regarding the relationships between variables in a reliable and efficient way. When it comes to robustness, the algorithm was run several times and for different subsets of data (i.e., production scenarios), always reproducing the same level of variable's prioritization shown in Table 1.

### E. COMPARISON BETWEEN FASTTRACKER AND MUTUAL INFORMATION ALGORITHM

Apart from the performance assessment among the implementation of the proposed changes to EventTracker, a comparison with an entropy-based algorithm for sensitivity analysis was also conducted. We found entropy-based SA as the closest and most comparable method as it is also designed for real-time systems. The reasoning behind this choice relies on the properties of the entropy concept in information theory, which agree with the intuitive notion of what a measure of information should be [38]. This notion is extended to define mutual information (MI), which is a measure of the amount of information one random variable contains about another.

Mutual information's algorithms have been opportunely adopted in filter feature-selection methods to evaluate both the relevance of a subset of features in predicting the target variable and the redundancy with respect to other variables [39]. There are several approaches on the use of MI as a feature selection algorithm, from which the following ones stand-out: mutual information feature selector (MIFS), MIFS greedy selection method (MIFS-U), min-redundancy max-relevance criterion (mRMR), and normalized mutual

information feature selection (NMIFS). NMIFS outperformed MIFS, MIFS-U, and mRMR on several artificial and benchmark data sets without requiring a user-defined parameter [11]. Thus, NMIFS was the entropy-based method used to select the most relevant features of the cement grinding dataset.

Table 1 presents the results for the NMIFS algorithm using a 10 bin histogram as a probability density function estimator, where it is possible to analyze the $G$ criterion and the order of prioritization for each input variable. The algorithm shows a $2/8$ ratio of false negatives (variables: Mill Power Transducer and Separator Actual Speed) and a $2/8$ ratio of false positives (variables: Auxiliary 2 and Auxiliary 1), concluding that fastTracker is more accurate than the NMFIS method, when used for selecting a group of high-relevance variables from the cement grinding dataset.

In terms of processing time, the entropy-based algorithm took 48.047 seconds to run the entire dataset, contrasting with the 9.159 seconds of fastTracker, being mutual information method 80.9% slower. Relatively to processing power, both fastTracker and NMIFS continuously used on average 20% of the available CPU (Central Processing Unit).

## V. CONCLUSION

fastTracker as a new approach to Sensitivity Analysis was introduced. Its intention was to improve EventTracker algorithm. The core of the methodology remains the same, with no reliance on statistical or model-based equations, simply on the interpretation of system state transitions. In that sense, the technique is still completely ''unaware'', but it provides accurate information about the relationships between triggers and events. Three new formulations were introduced with the aim of developing a faster and more efficient sensitivity analysis method capable of producing real-time normalized sensitivity indices that are suitable for both data acquisition and raw data analysis.

In terms of processing power, the efficiency of fastTracker was assessed and compared with the original EventTracker as well as with an entropy-based SA algorithm. Besides, the results were presented to industry experts. When faced with the original EventTracker, fastTracker can give one $nSI$ value at each 0.076 milliseconds, instead of 6.486 milliseconds. Regarding memory usage, the methodology is 20% more efficient. The proposed algorithm was also shown to be more accurate and 80.9% faster than the normalized mutual information feature selection algorithm.

The proposed improvements have been validated and verified through implementation in the IoT and SCADA system of industrial controllers and deployed in industrial applications. There is no reason not to believe that fastTracker cannot be used in large scale distributed data analysis, such as Edge computing for various industrial, manufacturing, environmental studies and finance (e.g., high-frequency trading) application. As part of future work, one goal is to develop an autonomous method for the provision of thresholds, $ETs$ and $TTs$, in order to optimize the definition of triggers and events. An intelligent forecasting method that finds the optimal dynamic analysis span based on the data collected directly from the system is also desired.

## ACKNOWLEDGMENT

This research was co-financed by the European Regional Development Fund, through Centro Regional Operational Program 2014/2020 (Centro2020), of Portugal 2020. Project iProMo (CENTRO-01-0247-FEDER-069730).

## REFERENCES

[1] R. Devaraj and A. Sarkar, ''Resource-optimal fault-tolerant scheduler design for task graphs using supervisory control,'' *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7325–7337, Nov. 2021.

[2] S. K. Roy, R. Devaraj, A. Sarkar, K. Maji, and S. Sinha, ''Contention-aware optimal scheduling of real-time precedence-constrained task graphs on heterogeneous distributed systems,'' *J. Syst. Archit.*, vol. 105, May 2020, Art. no. 101706.

[3] B. Iooss and P. Lemaître, ''A review on global sensitivity analysis methods,'' in *Uncertainty Management in Simulation-Optimization of Complex Systems*. Boston, MA, USA: Springer, 2015, pp. 101–122.

[4] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Introduction to Sensitivity Analysis*. Hoboken, NJ, USA: Wiley, 2007, ch. 1, pp. 1–51.

[5] S. Razavi *et al.*, ''The future of sensitivity analysis: An essential discipline for systems modeling and policy support,'' *Environ. Model. Softw.*, vol. 137, Mar. 2021, Art. no. 104954.

[6] G. E. P. Box and R. D. Meyer, ''An analysis for unreplicated fractional factorials,'' *Technometrics*, vol. 28, no. 1, pp. 11–18, Feb. 1986.

[7] A. Saltelli, M. Ratto, and T. Andres, *Global Sensitivity Analysis: The Primer*. Hoboken, NJ, USA: Wiley, 2008.

[8] F. Campolongo, A. Saltelli, and S. Tarantola, ''Sensitivity analysis as an ingredient of modeling,'' *Stat. Sci.*, vol. 15, no. 4, pp. 377–395, Nov. 2000.

[9] M. Danishvar, A. Mousavi, and P. Broomhead, ''EventiC: A real-time unbiased event-based learning technique for complex systems,'' *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 5, pp. 1649–1662, May 2020.

[10] S. Tavakoli, A. Mousavi, and P. Broomhead, ''Event tracking for real-time unaware sensitivity analysis (EventTracker),'' *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 2, pp. 348–359, Feb. 2013.

[11] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, ''Normalized mutual information feature selection,'' *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 189–201, Feb. 2009.

[12] N. Kambhatla and T. K. Leen, ''Dimension reduction by local principal component analysis,'' *Neural Comput.*, vol. 9, no. 7, pp. 1493–1516, Jul. 1997.

[13] M. J. Tonkin and J. Doherty, ''A hybrid regularized inversion methodology for highly parameterized environmental models,'' *Water Resour. Res.*, vol. 41, no. 10, Oct. 2005, Art. no. W10412.

[14] H. C. Frey and S. R. Patil, ''Identification and review of sensitivity analysis methods,'' *Risk Anal.*, vol. 22, no. 3, pp. 553–578, Jun. 2002.

[15] B. Lallemand, G. Plessis, T. Tison, and P. Level, ''Neumann expansion for fuzzy finite element analysis,'' *Eng. Comput.*, vol. 16, no. 5, pp. 572–583, Aug. 1999.

[16] A. Buonomo and A. Lo Schiavo, ''Nonlinear distortion analysis via perturbation method,'' *Int. J. Circuit Theory Appl.*, vol. 38, pp. 515–526, Apr. 2010.

[17] S. S. Isukapalli, ''Uncertainty analysis of transport-transformation models,'' Ph.D. dissertation, Dept. Chem. Biochem. Eng., Rutgers Univ., New Brunswick, NJ, USA, Jan. 1999.

[18] K. Pearson, *On the General Theory of Skew Correlation and Non-Linear Regression*. London, U.K.: Dulau, 1905.

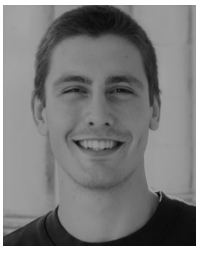
[19] R. I. Cukier, H. B. Levine, and K. E. Shuler, ''Nonlinear sensitivity analysis of multiparameter model systems,'' *J. Comput. Phys.*, vol. 26, no. 1, pp. 1–42, Jan. 1978.

[20] L. M. Sobo, *Sensitivity Analysis for Non-Linear Mathematical Models*, vol. 1. Hoboken, NJ, USA: Wiley, 1993, pp. 407–414.

[21] F. Pianosi and T. Wagener, ''A simple and efficient method for global sensitivity analysis based on cumulative distribution functions,'' *Environ. Model. Softw.*, vol. 67, pp. 1–11, May 2015.

[22] E. Borgonovo, ''A new uncertainty importance measure,'' *Rel. Eng. Syst. Saf.*, vol. 92, no. 6, pp. 771–784, Jun. 2007.

[23] Z. Kala, "Global sensitivity analysis based on entropy: From differential entropy to alternative measures," *Entropy*, vol. 23, no. 6, p. 778, Jun. 2021.
[24] S. Razavi, R. Sheikholeslami, H. V. Gupta, and A. Haghnegahdar, "VARS-TOOL: A toolbox for comprehensive, efficient, and robust sensitivity and uncertainty analysis," *Environ. Model. Softw.*, vol. 112, pp. 95–107, Feb. 2019.
[25] S. Razavi and H. V. Gupta, "A new framework for comprehensive, robust, and efficient global sensitivity analysis: 1. Theory," *Water Resour. Res.*, vol. 52, no. 1, pp. 423–439, Jan. 2016.
[26] A. Puy, S. L. Piano, and A. Saltelli, "Is VARS more intuitive and efficient than Sobol' indices?" *Environ. Model. Softw.*, vol. 137, Mar. 2021, Art. no. 104960.
[27] M. Fallahpour, A. Fatehi, B. N. Araabi, and M. Azizi, "A supervisory fuzzy control of back-end temperature of rotary cement kilns," in *Proc. Int. Conf. Control, Autom. Syst.*, Oct. 2007, pp. 429–434.
[28] J. Mendes, R. Seco, and R. Araujo, "Automatic extraction of the fuzzy control system for industrial processes," in *Proc. 16th IEEE Int. Conf. Emerg. Technol. Factory Autom.*, Sep. 2011, pp. 1–8.
[29] J. Mendes, R. Maia, R. Araújo, and F. A. A. Souza, "Self-evolving fuzzy controller composed of univariate fuzzy control rules," *Appl. Sci.*, vol. 10, no. 17, p. 5836, Aug. 2020.
[30] A. R. Jordehi and J. Jasni, "Heuristic methods for solution of FACTS optimization problem in power systems," in *Proc. IEEE Student Conf. Res. Develop.*, Dec. 2011, pp. 30–35.
[31] S. Tavakoli, A. Mousavi, and S. Poslad, "Input variable selection in time-critical knowledge integration applications: A review, analysis, and recommendation paper," *Adv. Eng. Informat.*, vol. 27, no. 4, pp. 519–536, Oct. 2013.
[32] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Manage. Sci.*, vol. 6, no. 3, pp. 324–342, Apr. 1960.
[33] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[34] L. Kazarinov and D. Khasanov, "Decision making process for operational neurocontrol of mixture grinding in cement production with controversial setting," in *Proc. Int. Russian Autom. Conf. (RusAutoCon)*, Sep. 2019, pp. 1–6.
[35] M. Boulvin, A. Wouwer, R. Lepore, C. Renotte, and M. Remy, "Modeling and control of cement grinding processes," *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 5, pp. 715–725, Sep. 2003.
[36] M. Danishvar, S. Danishvar, F. Souza, P. Sousa, and A. Mousavi, "Coarse return prediction in a cement industry's closed grinding circuit system through a fully connected deep neural network (FCDNN) model," *Appl. Sci.*, vol. 11, no. 4, p. 1361, Feb. 2021.
[37] G. G. Mejeoumov, "Improved cement quality and grinding efficiency by means of closed mill circuit modeling," Ph.D. dissertation, Dept. Civil Eng., Texas A&M Univ., Kingsville, TX, USA, 2007.
[38] T. M. Cover and J. A. Thomas, *Entropy, Relative Entropy and Mutual Information*. Hoboken, NJ, USA: Wiley, 1991, ch. 2, pp. 12–49.
[39] B. Poole, S. Ozair, A. Van Den Oord, A. Alemi, and G. Tucker, "On variational bounds of mutual information," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 5171–5180.



**MANUEL GONÇALVES** is currently pursuing the M.Sc. degree in biomedical engineering with the University of Coimbra. He is also a Research Fellow at the Institute of Systems and Robotics, Coimbra. He believes that empowering communities is the best way to grow. He has participated and coordinated various projects that promote the advancement of technology, such as IEEE SB UC. His research interests include machine learning and biomedical instrumentation.

**PEDRO SOUSA** received the M.Sc. degree in electrical and computer engineering, automation specialization, from the University of Coimbra. He is currently a Co-Manager and the Chief Technology Officer (CTO) at Oncontrol Technologies LDA. He has experience as a control engineer and software developer, strong experience on apply advanced control methodologies in industrial environment. His skilled in research and development (R&D), process optimization, and process control.

**JÉRÔME MENDES** (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Coimbra, in 2014. He is currently a Senior Researcher at the Institute of Systems and Robotics, Coimbra. His research interests include computational intelligence with an emphasis on intelligent control (adaptive, autotuning, predictive, data-driven, evolving, and interpretable); intelligent identification (prediction, soft sensors, evolving design, and interpretable), and intelligent failure detection.

**MORAD DANISHVAR** received the B.Sc. degree in electronic engineering, the M.Sc. degree in engineering management, and the Ph.D. degree from Brunel University London, U.K., in 2015. He is currently a Senior Research Fellow and the Co-Director of the Digital Manufacturing Centre, College of Engineering, Design and Physical Sciences, Brunel University London. His research interests include data analytics, machine learning, AI, and smartification of industrial systems.

**ALIREZA MOUSAVI** (Senior Member, IEEE) received the B.Sc. degree in industrial engineering and the Ph.D. degree in system engineering from Brunel University London. He is currently a Reader at the College of Engineering, Design and Physical Sciences, Brunel University London. His research interests include digital transformation and smartification of industrial systems, especially within the industry 4.0 context covering sensors-actuation, signal processing and feature extraction, machine learning, modeling, control, and optimization.