Data-driven Identification of Causal Dependencies in Cyber-Physical Production Systems

Kaja Balzereit[1], Alexander Maier[1], Björn Barig[2], Tino Hutschenreuther[2] and Oliver Niggemann[1,3]
[1]*Fraunhofer IOSB-INA, Fraunhofer Center for Machine Learning, Langenbruch 6, Lemgo, Germany*
[2]*IMMS GmbH, Ehrenbergstraße 27, Ilmenau, Germany*
[3]*Institute Industrial IT, OWL University of Applied Sciences, Lemgo, Germany*

Keywords: Machine Learning, Causal Dependencies, Cyber-Physical Production Systems, Case-based Reasoning, Timed Automaton, Decision Tree Classifier, Principal Component Analysis, Data Science.

Abstract: Cyber-Physical Systems (CPS) are systems that connect physical components with software components. CPS used for production are called Cyber-Physical Production Systems (CPPS). Since the complexity of these systems can be very high, finding the cause of an error takes a lot of effort. In this paper, a data-driven approach to identify causal dependencies in cyber-physical production systems (CPPS) is presented. The approach is based on two different layers of learning algorithms: one low-level layer that processes the direct machine data and a higher-level learning layer that processes the output of the low-level layer. The low-level layer is based on different learning modules that can process differently typed data (continuous, discrete or both). The high-level learning algorithms are based on rule-based and case-based reasoning. Thus, causal dependencies are detected allowing the plant operator to find the error cause quickly.

1 INTRODUCTION

In recent years, both the required product diversity and the flexibility in production facilities have increased significantly. As a result of this development, the complexity of the production facilities used has also increased greatly, which in the long run leads to excessive demands on the plant operators (Sauer, 2014).

In these systems, errors can occur that lead to a bad product quality or plant standstill. The search for the cause of an error has always posed major challenges to plant operators. Even if error messages are presented to the operator (e.g. over a control desk), the connection between these messages and the cause of the errors is not easily drawn. Since the error does not always coincide with the message at the control station, this shift not only leads to misleading error messages, but also triggers subsequent errors (chaining effect) that are not necessarily explained by the original problem.

Usually the search for the cause of an error is carried out manually (by the operator) during a plant standstill. In most cases, however, the operator of the production plant is not the plant manufacturer, as the production facilities are often purchased. Thus, a detailed knowledge of the plant behaviour and the causes of possible errors is missing. For this reason, employees must be specially trained or the relevant experts must arrive. An automatic search for the cause of the error is usually not possible, as this would require the causalities between the cause and the associated effect to be known. The longer a cause of failure needs to be searched, the longer the production plant is not working. Accordingly, the cost of a production loss adds up to the duration of the outage. These costs could be avoided or at least significantly reduced by an automatic analysis of the cause of an error.

As part of Germany's platform industry 4.0, the topic of intelligent assistance systems is being researched intensively (Federal Ministry for Economic Affairs and Energy, 2016). This includes diagnosis, planning, and physical assistants. In particular, diagnostic assistants contribute to this, but there is still a need for research for diagnostic assistants who can learn the causalities between an event and the corresponding effect in the system. Here, it is important to learn causality in locally limited modules, but also in distributed systems.

In order to solve the problem, we propose a solution which addresses the topics of

Balzereit, K., Maier, A., Barig, B., Hutschenreuther, T. and Niggemann, O.
Data-driven Identification of Causal Dependencies in Cyber-Physical Production Systems.
DOI: 10.5220/0007362005920601
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 592-601
ISBN: 978-989-758-350-6

1. data-driven detection of causality to help finding the technical cause of a problem and
2. implementation of local diagnostic assistants.

The assistance system includes the learning algorithms, the anomaly detection as well as the recognition of causal relationships for the identification of the causes of the error. When using machine learning algorithms to counteract these problems, one often meets the problem of specification: Each algorithm is well suited for a specific problem making it hard to find a single algorithm well suited for complex problems. The scientific question addressed in this paper is how to efficiently combine different machine learning algorithms so even complex problems can be handled well.

The contribution of this paper is as follows:

1. We present a new algorithm structure to detect causal dependencies in a data-driven way and present them in a human-understandable form. The structure is based on two different layers. One low-level learning layer that directly processes machine data and a high-level learning layer that connects the output of the different low-level modules. Thus, a combination of the different application areas of the algorithms is possible. The output of the high-level learning layer is a human-understandable reasoning.
2. We compare different high-level learning algorithms due to their suitability to draw conclusion about causal dependencies in CPPS.

The paper is structured in four parts: First, the state of the art is described. Following, the algorithmic structure of the diagnosis assistance system is presented. Then, a discussion of the applicability of the algorithms concerning a specific application case is given. Last, the topic is concluded and an outlook to future work is given.

2 STATE OF THE ART

Generally speaking, model-based approaches use a model to compare the system behaviour with the model predictions while the system is running. In parallel, model-based approaches try to find coherences between the symptom and the error cause (Voigt et al., 2015; Van Harmelen et al., 2008).

For diagnosis, mostly models describing the dependency between a root and its consequences are used, for example Fault Tree Analysis (Ahmad and Hasan, 2015; Schilling, 2015) or Event Tree Analysis (Ferdous et al., 2009). Both methods are based on Boolean algebra and probability theory.

To train these models to diagnose failures often a lot of expert knowledge is needed (Niggemann and Lohweg, 2015). Since this knowledge is not available in most cases, data-driven approaches are more applicable. These approaches are based on the data generated by the system (Fullen et al., 2017).

Other methods that can be used to learn the normal behaviour automatically and to diagnose errors are for example self-organizing maps (Frey, 2012), Bayesian networks (Runkler, 2012), neural networks (Jaber and Bicker, 2016), Support Vector Machines (Demetgul, 2013) or fuzzy logic (Wang et al., 2015).

Causality is the relation between root and symptom. Based on an observed error, the goal is to find the root of this error. However, when the causal dependencies are unknown, the interpretation of observations is necessary. Statistical methods can be applied to analyse causal dependencies data-based.

One of the first approaches was developed by Granger (Granger, 1969). This approach compares two autoregressive models of delayed variable values. When the regression gets better due to this delay, the hypothesis is that the variable with the greater delay has an impact on the other.

Another approach is presented by Horch (Horch, 2000): Based on the cross-correlation function, the maximal absolute value of the function and the corresponding time delay are used for the description of the causal dependency.

Schreiber presents a concept for the analysis of causal dependencies that is based on transfer entropy (Schreiber, 2000). Assuming that one variable predicts future values, the reduction of insecurity is measured.

Another possibility is given by the use of Bayesian networks (Eaton and Murphy, 2007). They were extended to also model the dynamic behaviour of real-world systems, which can be used to find causal dependencies.

Bauer proposes various algorithms based on nearest-neighbour methods (Bauer, 2005). The value of a certain variable is predicted by the value of a different variable.

Our approach differs from past research since it is based on a two-level learning algorithm. The low-level learning is based on established algorithms for the detection of coherences in machine data like neural networks (Jaber and Bicker, 2016) and Support Vector Machines (Demetgul, 2013). Additionally, Timed Automata as presented by Maier (Maier, 2014) and a combination of a Principal Component Analysis (PCA) combined with a Nearest Neighbour search (Eickmeyer et al., 2015) are used. This basis is completed by a high-level learning layer, that combines the output of the different low-level modules to draw causal conclusions. Thus, the different application areas of the different algorithms are combined. This makes it possible to handle differently shaped data (periodic, static, continuous, discrete, ...).

3 CAUSALITY ANALYSIS ALGORITHM

The basis for a successful diagnosis in a production plant is the exact knowledge of each individual state. There are a lot of machine learning methods that model plant behaviour. Each of these methods are intended for a specific field of application (e.g. timing machines for the timing of discrete events, or Principal Component Analysis (PCA) for the analysis of a set of continuous signals). In addition, due to combinatorics, it is not possible to learn a single model for a distributed system, since the presence of parallel processes can increase the complexity of the model enormously (Maier et al., 2011). The aim of this approach is a high-level learning layer, (see Figure 1), which is able to infer causal dependencies out of the outputs of different local models.

3.1 General Structure

The purpose of assistance systems is to support the employee in various tasks, such as the fault diagnosis as well as the reduction of the system complexity. Many machine learning methods are unsuitable for the direct use in diagnosis assistance systems because the output cannot be interpreted in an intuitive manner by humans. For example, when using neural networks it is not easy to comprehend how the neural network calculated its output. Since the input is propagated by a lot of hidden layers, a direct coherence cannot be drawn. However, this is a prerequisite for the successful deployment of a diagnosis assistance system. The problem here is that machine learning typically works on a low-level (sub-symbolic level). In order to make the results accessible to humans, however, algorithms must be developed that work on a high-level (symbolic level).

Figure 1 illustrates the structure of the assistance system presented in this approach. The machine data first is given to a number of low-level modules. These modules are represented by different algorithms that have different advantages and disadvantages. Here, only models limited to machine components are learned. These can cover certain aspects of behaviour (discrete, continuous, ...). The high-level earning methods are designed to combine all the underlying techniques. The data is prepared in a human-understandable form, e.g. by integration of semantic knowledge (naming the system states) or an interpretable structure (e.g. "if ... then" - rules).

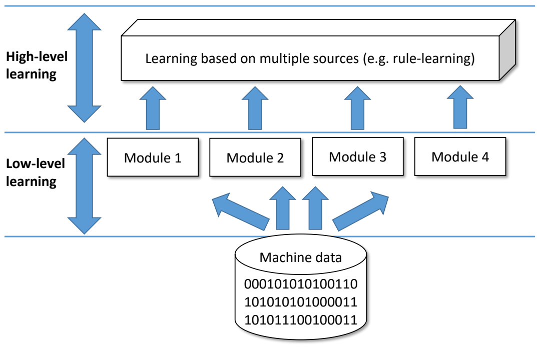

Figure 1: Algorithmic structure of the global diagnosis assistant.

3.2 Low-level Learning

Every module in the low-level is represented by a machine learning algorithm. Thus, for each module one model is trained that is used to give information about the current status of the machine. Each algorithm is specialized on different properties: Some algorithms are able to process continuous data, others work on discrete data. In the presented approach, the machine data is separated based on its type so each algorithm gets that data it can process best.

Many algorithms used for anomaly detection need to be trained with the help of a labelled dataset before they can be used. We call these algorithms *offline* since the model is learned once before the prediction can be done. Algorithms that adapt their model during the prediction are called *online*. These algorithms do not need to be trained before they are used but improve the model during the prediction.

In the following, different algorithms used for the detection of causal coherences in CPPS are presented. An online algorithm based on Timed Automata can process discrete data. An offline working combination of a Principal Component Analysis and a Nearest-Neighbour Classification is used to find patterns in continuous data. To detect outliers in the machine data, an online working algorithm is used.

Online Timed Automaton Learning Algorithm. Timed Automata are often used for programming production systems and are therefore suited for modeling a technical process.

A Timed Automaton is a tuple $A = (S, s_0, F, \Sigma, T, \Delta, c)$, where

- S is a finite set of states, $s_0 \in S$ is the initial state, and $F \subseteq S$ is a set of final states,
- Σ is the alphabet comprising all relevant events,
- $T \subseteq S \times \Sigma \times S$ gives the set of transitions. E.g. for a transition $\langle s,a,s' \rangle$, $s,s' \in S$ are the source and destination states and $a \in \Sigma$ is the trigger event.
- A set of transition timing constraints Δ with $\delta : T \rightarrow I, \delta \in \Delta$, where I is the set of time intervals.
- A single clock c is used to record the time evolution. At each transition, the clock is reset. This allows only for the modelling of relative time steps.

In (Niggemann et al., 2012) and (Verwer, 2010), the authors present an offline algorithm to identify Timed Automata from data. Both approaches are based on the state merging paradigm, both defining individual state compatibility criteria. Another approach can be found in (Maier, 2014), where the method presented identifies Timed Automata in an online manner (Online Time Automaton Learning Algorithm). For this, a different state compatibility criterion is defined which is based on the values of the signal vector.

Usually, Timed Automata are used to describe a discrete behaviour. For the modelling of continuous or hybrid behaviour, further methods like a PCA-based anomaly detection (see next paragraph) are needed.

In the following, the algorithm is referred to as *OTALA*.

PCA-based Anomaly Detection. Eickmeyer et al. presented an approach for anomaly detection that is based on a Principal Component Analysis (Eickmeyer et al., 2015). First, the input data and the training set are transformed to a lower-dimensional euclidean room with the use of the PCA. Then, the nearest neighbour to the input data point is searched for in the transformed training set. With the help of a Marr wavelet function a probability is calculated, that determines how likely the given point is part of the normal space.

This algorithm works on continuous data that satisfies the assumption that most of the information is located in the direction of the maximal variance.

Many algorithms used for the determination of an observation being part of the normal space is done using fixed thresholds. These thresholds are given using expert knowledge. A great advantage of the presented approach is that it needs less expert knowledge: Since a Marr wavelet function that uses probabilities is used, no fixed threshold is needed. The expert knowledge is only needed when generating the training set since in the training set only normal behaviour must be represented.

In the following, the algorithm is referred to as *PCNA*.

Online Outlier Detection. In this paragraph, an algorithm for outlier detection is presented. In general, it is not useful to apply this method to discrete data, therefore, in the presented approach it is only used to process continuous data. The algorithm works completely online and does not need to be trained before its use. Only the mean value and the standard deviation are needed for this algorithm. In the following, formulas to calculate these online are presented.

Based on these values, the data is checked and labelled okay or faulty. Therefore, first the mean value for the $n-th$ data point E^n is calculated recursively for every component j by

$$E_j^n = \frac{n-1}{n} E_j^{n-1} + \frac{1}{n} x_j^n \quad \forall j \in \{1,2,...,M\}, \qquad (1)$$

where E_j^{n-1} describes the mean value for the $n-1^{st}$ data point of component j and x_j^n describes the current observation. Note that $E_j^1 = x_j^1$ and $\dim(x^n) = M$.

Similar, the variance can be calculated recursively by

$$V_j^n = \frac{1}{n}\Big((n-1) \cdot V_j^{n-1} + \frac{n-1}{n}(E_j^{n-1} - x_j^n)^2\Big) \qquad (2)$$

$\forall j \in \{1,2,...,M\}$ and $V^1 = 0$. Based on this, the current standard deviation can be calculated by

$$\sigma^n = \sqrt{V^n}. \qquad (3)$$

With this information an outlier detection can be done by checking if a given data point x^n fulfils the equation

$$|x_j^n - E_j^n| < r \cdot \sigma_j^n \quad \forall j \in \{1,2,...,M\} \qquad (4)$$

with a $r \in \mathbb{R}$. r has to be defined by an expert since it is a measurement how far data points may differ from previous data points. If equation (4) is satisfied for every component of x^n, it is labelled okay, otherwise it is labelled as an outlier.

Other Algorithms. The algorithmic structure (see figure 1) offers a lot of flexibility concerning the low-level modules, therefore many different algorithms may be used. Since self-organizing maps (Frey, 2012), Bayesian networks (Runkler, 2012), neural networks (Jaber and Bicker, 2016), Support Vector Machines (Demetgul, 2013) and fuzzy logic (Wang et al., 2015) have been used for the data-driven diagnosis in real-world applications, their suitability as a low-level module is high. Additionally, established implementations of these algorithms exist, so that the integration effort is low.

3.3 High-level Learning

High-level learning enables the combination of different model formalisms (from low-level-learning) as well as the combination of models from different modules, which in turn allows a global system modelling.

3.3.1 Rule-based Reasoning

In particular, rule-based learning methods are promising at this point because they can determine causal relationships due to their "if ... then ..." - structure. Based on different learning methods of low-level learning, the individual modules are able to broadcast messages about the condition. The goal now is to evaluate these "broadcast messages" for existing causalities. Figure 2 shows a minimal example of how learning high-level rules can be used to combine low-level models to deduce causal relationships. A similar rule might look like this:

If conveyor A is standing, then module B will not get a bottle.

The technical background of this rule would be the following:

If module A is in state 4 and $t > 5s$, then module B is in error state F.

The learned automaton provides further details on the individual states: State 4 of module A corresponds to "conveyor belt", while state F of module B corresponds to the error state "bottle is missing". The underlying state machines can be learned using algorithms like (Hy-)BUTLA (Niggemann et al., 2013).

Here, an approach based on different decision tree methods is used to generate the decision rules. First, generating rules with the use of a single decision tree then the use of random-forest classifiers is presented.

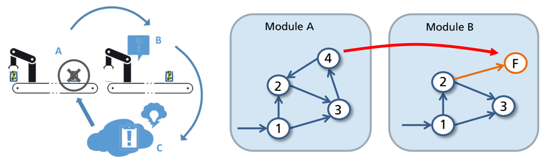

Figure 2: Rule-learning to detect causal dependencies between error cause and anomaly.

Decision Tree Learning. One way to realize rule-based learning is by the usage of decision tree algorithms. A great advantage of decision trees is the high interpretability and transparency (Murphy, 2012). Decisions made with the help of decision trees are easy to understand, since every node represents a single decision rule. When applying a rule the output leads to further nodes with further decision rules. An "if ... then ..." - structure is found easily.

Trees can process continuous as well as discrete input data (Murphy, 2012). Continuous data can be separated using inequations, discrete data can be enumerated. So, the different output of the low-level modules (like a discrete state from (Hy-)BUTLA or a continuous probability from the PCNA) can both be interpreted and processed.

When handling trees often the problem of overfitting is occurring. Trees are very unstable and strongly depending on the training data (Murphy, 2012). Thus, when different changes are applied to the training data, the shape of the decision tree may vary largely. For the application case of anomaly detection in CPPS, this means that the training data should approximately consist of so many data points describing correct machine behaviour as data points describing anomalies.

There are many different methods to find the optimal partition of a dataset (Murphy, 2012). The most popular methods are CART (Breiman et al., 1984), ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993). Here, the implementation of scikit-learn is used which is based on the CART algorithm (Pedregosa et al., 2011).

Random-forest Classifier. Since using a single tree as classifier can lead to a high variance in the result, decision tree methods are expanded to methods handling more than a single tree. These methods are called Forest Classification methods.

Therefore $N \in \mathbb{N}$ differently shaped decision trees are trained. The different shape of the decision trees are achieved by using different training sets. They are randomly chosen from the original training set.

The prediction $f(\boldsymbol{x})$ of the forest for an input vector $\boldsymbol{x}$ is then calculated by

$$f(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{x}) \quad (5)$$

where $f_n(\boldsymbol{x})$ represents the prediction of the n-th tree.

The result calculated by forest classifiers cannot be interpreted directly. Since it is a weighted average of outputs by a set of trees, the interpretation is more complex than the interpretation of the output of a single tree. This makes it more difficult to extract the causal dependencies searched.

3.3.2 Case-based Reasoning

As an alternative to rule learning, case-based reasoning (CBR) (including pattern recognition and clustering algorithms) can be used to detect causal dependencies. CBR methods are based on reasoners, that remember previous situations similar to the current one and propose a solution approach (Kolodner, 1993).

When using CBR methods, one challenge is how to generate the case base. Previous cases need to be indexed and stored in a case memory (Xu, 1994). Another challenge is how to define similarity between cases (Fullen et al., 2017).

To apply CBR to a data set, the data is searched for frequently recurring patterns, which can be used to detect relationships that may be useful for further analysis, e.g.

Every morning, energy consumption increases dramatically.

or

State pairs [Module A / State 4] and [Module B / State F] often appear together.

The second-mentioned pattern corresponds to the rule from the previous section. However, caution is advised not to confuse observed correlations with causality.

For each detected error, a case is stored in the case base. Subsequently, similarity measures are learned which are functions of the low-level output describing a cause. For new cases, similar cases are searched using the similarity measures. It is important that the cases are formalized with features. Furthermore, relationships between these anomalies and causes of errors are stored in a database and thus provide a memory for the diagnostic assistant, which can be used in future diagnoses to identify anomalies even more reliably and assign them to a cause.

The core of every CBR strategy is the case base. It consists of cases that occurred in the past. It is important that the cases are well suited for the prediction about new cases (Kolodner, 1993).

For the high-level learning, the case base consists of output generated by the low-level modules. Therefore, the machine data is propagated by the low-level modules. The output represents one case. Since the output of the low-level modules can be both, continuous (e.g. a probability from the PCNA) and discrete (e.g. a machine status by OTALA or a cluster number by a clustering method), the case is represented by a hybrid data vector. Then the case is labelled with the error cause by the machine operator.

So, a case is described by a vector $\boldsymbol{y}$ and an error cause, coded by an integer $g \in \mathbb{N}$. The case base then can be described as a set of tuples $Y = \{(\boldsymbol{y}_1, g_1), (\boldsymbol{y}_2, g_2), ...\}$.

When using CBR strategies a major challenge is the determination of similarity of data points (Fullen et al., 2017). Since the high-level learning algorithm has to handle hybrid data, a similarity measure that can handle both, continuous and discrete values, is needed. In the following, a hybrid approach based on the euclidean distance and the discrete metric is presented.

Let $\boldsymbol{x} = (\boldsymbol{c}_1, \boldsymbol{d}_1)$ be the current (unlabelled) output of a machine data set propagated by the low-level modules where $\boldsymbol{c}_1 \in \mathbb{R}^k$ stores the continuous values and $\boldsymbol{d}_1 \in \mathbb{R}^l$ stores the discrete values. For this data point, the most similar case in the case base Y is searched. Therefore, a definition of *similarity* has to be given.

Let $(\boldsymbol{y}, g)$ be a tuple from the case base Y which similarity to $\boldsymbol{x}$ shall be determined. $\boldsymbol{y}$ can be described by $\boldsymbol{y} = (\boldsymbol{c}_2, \boldsymbol{d}_2)$ with $\boldsymbol{c}_2 \in \mathbb{R}^k$ representing the continuous values and $\boldsymbol{d}_2 \in \mathbb{R}^l$ representing the discrete values.

For the similarity measure of the continuous values, first, the data is scaled using the current mean value and the current standard deviation. Let E_c be the current mean value of the continuous data (recursively calculated using (1)) and σ_c be the current standard deviation of the continuous values (recursively calculated using (2)). Then, every component of $\boldsymbol{c}_1$ can be scaled by

$$\tilde{\boldsymbol{c}}_{1_i} = \frac{\boldsymbol{c}_{1_i} - E_{c_i}}{\sigma_{c_i}} \quad \forall i \in \{1, ..., k\} \tag{6}$$

(analogue for $\tilde{\boldsymbol{c}}_2$). Based on this, the continuous distance d_c of $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined using the euclidean distance

$$d_c(\boldsymbol{x}, \boldsymbol{y}) := \|\tilde{\boldsymbol{c}}_1 - \tilde{\boldsymbol{c}}_2\|_2 = \sqrt{\sum_{i=1}^{k} (\tilde{\boldsymbol{c}}_{1_i} - \tilde{\boldsymbol{c}}_{2_i})^2}. \tag{7}$$

Calculating the similarity of the discrete values cannot be done as simple as the similarity for the continuous values since discrete values in general are not interval typed, i.e. if a module is stuck in state 4 this is not twice as bad as if the module is stuck in state 2. Therefore, a measurement just scoring if there is a difference between two states is needed. This can be done by using the discrete metric

$$\delta(a, b) := \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{if } a = b. \end{cases} \tag{8}$$

For the definition of the discrete distance d_d of $\boldsymbol{x}$ and $\boldsymbol{y}$ the discrete metric δ is extended to

$$d_d(\boldsymbol{x}, \boldsymbol{y}) := \sum_{i=1}^{l} \delta(\boldsymbol{d}_{1_i}, \boldsymbol{d}_{2_i}). \tag{9}$$

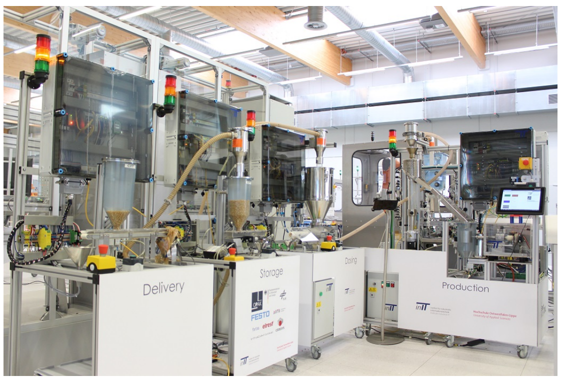

Figure 3: Graphical image of the Versatile Production System of the SmartFactoryOWL. The VPS consists of four different modules (Delivery, Storage, Dosing and Production).

Thus, for two given data points $\boldsymbol{x}$ and $\boldsymbol{y}$ a discrete as well as a continuous distance can be calculated. To determine the *most similar* case in the case base Y, a priority between these two measurements has to be defined.

The most similar data point $\boldsymbol{y}_x \in Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ...\}$ then is defined by

$$\boldsymbol{y}_x = \arg\min_{\boldsymbol{y}\in D_{\boldsymbol{y}}} \{d_c(\boldsymbol{x},\boldsymbol{y})\} \tag{10}$$

with

$$D_{\boldsymbol{y}} = \arg\min_{\boldsymbol{y}\in Y} \{d_d(\boldsymbol{x},\boldsymbol{y})\}. \tag{11}$$

This means, that first the data points in Y are searched, which discrete distance is smallest (note that this can be more than one data point). These are represented by the set $D_{\boldsymbol{y}}$. From these data points, that one is searched, which continuous distance to the given data point $\boldsymbol{x}$ is smallest. That data point is the most similar case $\boldsymbol{y}_x$.

4 DISCUSSION

In this section, a discussion of the different proposed methods based on an industrial use case is given.

4.1 Description of the Industrial Use Case

For the discussion of the presented solution approach a demonstrator of the SmartFactoryOWL is used. It is a versatile production system which represents typical industry processes (see figure 3).

The system consists of four different modules. The first is a delivery module where material is input. From that module, the material is transported to the second module which task is to store and transport material. After that module a dosing module is placed, which fills the material into bottles. Finally, the material is transported to the production module where it is transported through a blow pipe into a heater. The system produces more than 200 signals (Bunte et al., 2016).

These signals are continuous or discrete (boolean or integer). As described above, they are separated so that each low-level module gets the data that the algorithm is best suited for. E.g. the Principal Component Analysis as well as clustering methods get continuous values while the Timed Automata (OTALA) can handle discrete data.

4.2 Application of Presented Methods

The presented high-level learning methods are compared regarding their interpretability, the implementation effort and their suitability for the presented use case.

Training. To apply these methods, first, a training set has to be generated. It needs to contain correct as well as false behaviour of the machine.

Therefore, the machine is run and the machine data is propagated by the low-level modules. The different low-level algorithms calculate predictions based on their individual training. After that, the outputs of the low-level modules are collected and stored. When no error occurred, the data points are labelled with an error code representing *OK*. After that, typical errors (missing bottle, blockage, ...) are inserted into the system. The related datasets are labelled with the specific integer coding the error and its cause. By doing so, a dataset containing the output of low-level data and the related machine behaviour (error and its cause) is created.

Single Decision Tree Classifier. With this dataset the training of the different high-level modules is done. When training a decision tree using the CART algorithm, the whole training set is used. It is important to prevent the tree from overfitting by defining a maximum depth and a minimum split.

Figure 4 shows an example of a decision tree learned with the CART algorithm. The CART algorithm is based on a greedy search to separate the differently labelled data points in the given training set. With this greedy search, the result of the PCNA r_{PCNA} is the most expressive value to separate the given training set. When a condition is satisfied, the left branch is used for further separation. In contrast, when a condition is not satisfied, the right branch is

Table 1: Comparison of single decision tree classifiers (SDTC), random forest classifiers (RFC) and case-based reasoning (CBR) is given using the attributes interpretability, implementation effort and suitability.

method	**interpretability**	**suitability**	**implementation effort**
SDTC	high	medium	low
RFC	low	low	low
CBR	high	high	high

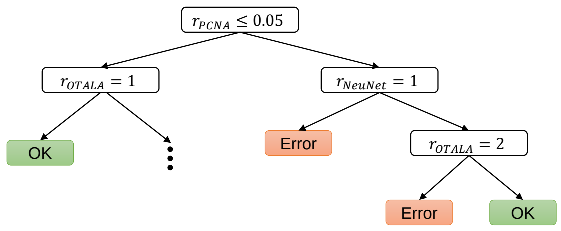

Figure 4: Decision Tree learned with the CART Algorithm and the machining data from the VPS.

used. In the example, if the condition $r_{\text{PCNA}} \leq 0.05$ is satisfied, the result of OTALA r_{OTALA} is checked. Practically, this means the machine is checked for its current status. Based on this status, a decision is made whether an error has occurred or not or if further low level modules have to be checked.

Based on this decision tree, decision rules like

When $r_{\text{PCNA}} \leq 0.05$ and $r_{\text{OTALA}} = 1$ then OK.

can be derived. Practically speaking, this means

When the PCNA module labels the data points okay and the machine is in state 1, no error has occurred.

Since this rule is easy to understand by a human operator, the interpretability of the Decision Tree Classifier is high.

Random Forest Classifier. When using Random Forest Classifiers, multiple (for example ten) decision trees are created using different training data sets. These sets are randomly created from the initial training data set. The trees created this way differ in their structure and therefore even may differ in their predictions. For example, when using ten decision trees for an online prediction while the machine is running, eight may predict *OK* and two may predict *ERROR*. For a classification of a single data point, a number of classification rules is used. This makes the prediction hard to interpret since the rules even may contradict each other. Thus, a unique reasoning for the error cause is impossible.

Case-based Reasoning. Using the training data set described above, the case base for the CBR strategy can be created. Defining the case base as the whole training data set would make the search for the most similar case of a given data point last very long. Thus, a representative subset of the case base is selected. Cases representing normal and false behaviour are chosen.

For example, a case in the case base can be

$$r_{OTALA} = 1, r_{PCNA} = 0.025, r_{NeuNet} = 1, \qquad (12)$$

labelled with *OK*. This case is representing correct behaviour. Another case could be

$$r_{OTALA} = 2, r_{PCNA} = 0.95, r_{NeuNet} = 0, \qquad (13)$$

labelled with a specific code describing the error and its cause.

When running the machine, the machine data is first propagated by the low-level modules. So, a data point, for example

$$r_{OTALA} = 2, r_{PCNA} = 0.85, r_{NeuNet} = 0, \qquad (14)$$

is generated. This data point is compared to the known cases using the similarity measurement presented above. The most similar case - here

$$r_{OTALA} = 2, r_{PCNA} = 0.95, r_{NeuNet} = 0 \qquad (15)$$

- is chosen and its error cause and a solution approach are presented to the machine operator. The machine operator then can check for the proposed error and correct it. If the error cause is as assumed, the machine operator labels the solution approach as useful. The current case is then added to the case base with the related error cause and its solution.

If the solution proposal was not helpful but wrong, the machine operator can re-label the dataset using a proposal of possible error codes. So, the case is added to the case base with a new solution proposal. When, in future, a similar case occurs it is more probable that the correct solution approach is presented since the similarity to the currently added case probably is the highest.

Comparison. In table 1, the three different high-level learning methods are compared concerning their interpretability, the implementation effort and the suitability for the application in an industrial use case.

As mentioned above, a great advantage of single decision trees is their high interpretability since decision rules can easily be drawn from the decision tree. Established algorithms like CART can be used, so the implementation effort is rated low. Since overfitting to the given training set is a big problem of single trees, the suitability for the application is rated medium.

Since the prediction is based on multiple decision trees, the interpretation of the result of a random forest classification is difficult. Contradicting decision rules may be used for the current prediction, causalities cannot be found easily. For the application case, this leads to a low rated suitability. Established algorithms can be used for the implementation, the effort for this is rated low.

Using a case-based reasoning strategy for the learning returns a similar case from the past and the then detected error cause. The result is easy to interpret for the machine operator and the machine operator can check if the past case has occurred again. For the implementation of a case-based reasoning strategy a lot of implementation effort is needed: The case base, consisting of representative cases, has to be generated. Additionally, a similarity measurement has to be defined, which is well suited for the differently typed outputs. When a case has been handled, the case base needs to be updated. Since a relation between past behaviour of the production machine and current behaviour is drawn, the suitability is rated high.

5 CONCLUSION

In this paper, a data-driven approach to identify causal dependencies in CPPS is presented. The structure of the presented analysis tool is based on two layers: one low-level learning layer that directly processes machine data and a high-level learning layer that processes the output of the low-level modules.

The low-level modules are based on established machine learning algorithms like cluster analysis, Timed Automata or Principal Component Analysis. The specific algorithms are given that data, that they are best suited for. For example, cluster analysis works good on continuous data while Timed Automata are used to process discrete data.

The high-level learning algorithms differ in rule-based and case-based algorithms. As rule-based algorithms, decision tree classifiers based on a single tree and on multiple trees (also named forest classifiers) were used. Additionally, the usage of a case-based reasoning strategy is presented. These methods are compared concerning their interpretability, their implementation effort and their suitability for the application case. Even though the implementation effort for a case-based reasoning strategy is high, it outperforms the rule-based strategies.

In future work, different algorithms for the low-level modules will be evaluated. Since the presented concept is very flexible, the applicability of different algorithms can be compared quickly.

Additionally, the case-based reasoning strategy will be improved. Since the generation of the case base requires a lot of effort, methods to automatically generate this are examined. Furthermore, different similarity measurements than the presented one are examined concerning their applicability for the specific use case of high-level learning.

ACKNOWLEDGEMENTS

This work was partially funded by the BMWi (German Federal Ministry of Economics and Technology), funding code IGF 19341 BG, as a part of the project "AgAVE: Assistenzsysteme zur Überwachug von vernetzten Anlagen - Herausforderungen beim Vernetzen sowie beim Erkennen von kausalen Zusammenhngen in Industrie 4.0 Umgebungen" (2017-2019) and partially funded by the Fraunhofer Cluster of Excellence "Cognitive Internet Technologies".

REFERENCES

Ahmad, W. and Hasan, O. (2015). Towards formal fault tree analysis using theorem proving. In *Conferences on Intelligent Computer Mathematics*, pages 39–54. Springer.

Bauer, M. (2005). *Data-driven methods for process analysis*. PhD thesis, University of London.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

Bunte, A., Diedrich, A., and Niggemann, O. (2016). Semantics enable standardized user interfaces for diagnosis in modular production systems. In *International Workshop on the Principles of Diagnosis (DX)*, Denver, CO, USA.

Demetgul, M. (2013). Fault diagnosis on production systems with support vector machine and decision trees algorithms. *The International Journal of Advanced Manufacturing Technology*, 67(9-12):2183–2194.

Eaton, D. and Murphy, K. (2007). Belief net structure learning from uncertain interventions. *Journal of Machine Learning Research*, 1:1–48.

Eickmeyer, J., Li, P., Givehchi, O., Pethig, F., and Niggemann, O. (2015). Data driven modeling for system-level condition monitoring on wind power plants. In *DX@ Safeprocess*, pages 43–50.

Federal Ministry for Economic Affairs and Energy, editor (2016). *Ergebnispapier - Aspekte der Forschungsroadmap in den Anwendungsszenarien*. Berlin.

Ferdous, R., Khan, F., Sadiq, R., Amyotte, P., and Veitch, B. (2009). Handling data uncertainties in event tree analysis. *Process safety and environmental protection*, 87(5):283–292.

Frey, C. W. (2012). Monitoring of complex industrial processes based on self-organizing maps and watershed transformations. In *Industrial Technology (ICIT), 2012 IEEE International Conference on*, pages 1041–1046. IEEE.

Fullen, M., Schüller, P., and Niggemann, O. (2017). Defining and validating similarity measures for industrial alarm flood analysis. In *Industrial Informatics (INDIN), 2017 IEEE 15th International Conference on*, pages 781–786. IEEE.

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438.

Horch, A. (2000). *Condition monitoring of control loops*. PhD thesis, Signaler, sensorer och system.

Jaber, A. A. and Bicker, R. (2016). Industrial robot backlash fault diagnosis based on discrete wavelet transform and artificial neural network. *American Journal of Mechanical Engineering*, 4(1):21–31.

Kolodner, J. (1993). *Case-based Reasoning*. Artificial intelligence. Morgan Kaufmann Publishers.

Maier, A. (2014). Online passive learning of timed automata for cyber-physical production systems. In *The 12th IEEE International Conference on Industrial Informatics (INDIN 2014)*. Porto Alegre, Brazil.

Maier, A., Vodencarevic, A., Niggemann, O., Just, R., and Jaeger, M. (2011). Anomaly detection in production plants using timed automata. In *8th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 363–369.

Murphy, K. P. (2012). *Machine Learning. A Probabilistic Perspective*. Massachusetts Institute of Technology.

Niggemann, O. and Lohweg, V. (2015). On the diagnosis of cyber-physical production systems: state-of-the-art and research agenda. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 4119–4126. AAAI Press.

Niggemann, O., Stein, B., Vodenčarević, A., Maier, A., and Kleine Buning, H. (2012). Learning behavior models for hybrid timed systems. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*.

Niggemann, O., Vodencarevic, A., Maier, A., Windmann, S., and Büning, H. K. (2013). A learning anomaly detection algorithm for hybrid manufacturing systems. In *The 24th International Workshop on Principles of Diagnosis (DX-2013), Jerusalem, Israel*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Runkler, T. A. (2012). *Data Analytics*. Springer.

Sauer, O. (2014). Information technology for the factory of the future–state of the art and need for action. *Procedia CIRP*, 25:293–296.

Schilling, S. J. (2015). Contribution to temporal fault tree analysis without modularization and transformation into the state space. *arXiv preprint arXiv:1505.04511*.

Schreiber, T. (2000). Measuring information transfer. *Physical review letters*, 85(2):461.

Van Harmelen, F., Lifschitz, V., and Porter, B. (2008). *Handbook of knowledge representation*, volume 1. Elsevier.

Verwer, S. (2010). *Efficient Identification of Timed Automata: Theory and Practice*. PhD thesis, Delft University of Technology.

Voigt, T., Flad, S., and Struss, P. (2015). Model-based fault localization in bottling plants. *Advanced Engineering Informatics*, 29(1):101–114.

Wang, T., Zhang, G., Zhao, J., He, Z., Wang, J., and Pérez-Jiménez, M. J. (2015). Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural p systems. *IEEE Transactions on Power Systems*, 30(3):1182–1194.

Xu, L. D. (1994). Case based reasoning. *IEEE potentials*, 13(5):10–13.