Hierarchical multi-fault prognostics for complex systems

Pablo del Moral[a,*], Sławomir Nowaczyk[a], Sepideh Pashami[a]

[a] *CAISR, Halmstad University, Kristian IV:s väg 3, 301 18 Halmstad, Sweden*

Abstract

The field of predictive maintenance for complex machinery with multiple possible faults is an important but largely unexplored area. In general, one assumes, often implicitly, the existence of monitoring data specific enough to capture every possible fault independently from all the others.

In this paper, we focus on the problem of predicting time-to-failure, or remaining useful life, in situations where the above assumption does not hold. Specifically, what happens when the data is not good enough to uniquely predict every fault, and, more importantly, what happens when different faults share the same symptoms on the recorded data.

We demonstrate that prognostics approaches learning independent models for each fault are inadequate. In particular, in the presence of faults that produce similar failure patterns, they produce false alarms disproportionately often or miss the majority of failures.

We propose the HMP framework (Hierarchical Multi-fault Prognosis) to solve this problem by extracting a hierarchy of faults based on the similarity of the data they produce. At each node of the hierarchy, we train a regression model to predict the time-to-failure for any of the faults contained in this node. The intuition is that while it might be impossible to predict individual time-to-failure in the presence of similar faults, a model trained on aggregated data can still provide useful information. We demonstrate through experiments the validity of our approach.

*Corresponding author
Email address: pablo.del_moral@hh.se (Pablo del Moral)

Keywords: Predictive Maintenance; Complex System; Multi-fault Prognosis

1. Introduction

In this paper, we are moving towards the ultimate goal of data-driven predictive maintenance of complex systems in real-life applications: given available data, which we know is inadequate, predict all possible failures and isolate the type of fault responsible for them. In particular, we identify, and propose a solution for, the problems arising when different types of faults induce the same symptoms in the data.

This work has been inspired by our collaboration with Getinge AB and Volvo Trucks. Getinge produces and performs maintenance services on sterilizers used in hospitals to sterilize medical equipment, and are critical for their operation. Volvo Trucks produces and carries out service on heavy-duty vehicles used for various purposes. In both cases, we deal with complex machines composed of various components performing heterogeneous tasks. The different components of the machine interact with each other directly through physical influence (contact, friction, heat...). But, they also interact indirectly through the logic of the control systems governing the operation of the machine. For example, in a sterilizer, a leak in a pipe may change the power of the vacuum pump beyond normal limits since the control system requires certain pressure in order to continue the sterilization process.

Based on the most recent surveys dealing with predictive maintenance applications, cf. [1, 2, 3, 4], one can conclude that multi-fault prediction in complex systems is a yet unexplored field. Research focuses on single type of fault problems or multi-fault problems related to a single component. It is not without reason that research has focused on these topics. Under the assumption that we have data good enough to monitor the health of a single component independently of the state of the rest of the system, we can focus on the specific component and direct our efforts to obtain the best possible predictions. In this perfect scenario, a successful monitoring system for the whole system could

be developed simply by combining the models trained for each individual as independent building blocks.

However, this assumption rarely holds in practice when dealing with complex systems. Often, the quality of the data is not good enough to monitor the health of every single component. Also, different components do not act independently; instead, they affect each other, often in unexpected ways. Consequently, different types of faults are related through the interaction of the components they affect. But, more importantly, and this is the focus of our paper, different types of faults are related through the effect they have on the recorded data; the data that we will use to learn to predict future failures and will ultimately define the quality of our final solution.

The most valuable output of a predictive maintenance system is detecting a fault that will develop into a failure, identifying the fault, and predicting the precise when the failure will occur. With this information at hand, precise maintenance operations can be scheduled, reducing downtime, increasing the productivity of the machines, and reducing the cost of maintenance.

In complex systems like a truck or a sterilizer, some faults show distinctive patterns in the data: distinct from the data produced by a healthy system and distinct from data associated with other types of faults. This is the ideal case, implicitly considered "the norm" and treated extensively in the literature; if it was the only case, one could simply focus on each type of fault individually. There are also other faults that do not affect the recorded data, and no approach is capable of predicting them. Finally, however, there are often sets of distinct faults that produce similar patterns in the recorded data. It is this last case that has not been sufficiently addressed in the literature, motivating this paper. The challenges it presents have not been identified, and adequate solutions have not been proposed.

We specifically show, through experiments, the limitations of the approaches that focus on predicting time-to-failure for individual types of faults. We demonstrate that without considering the similarities and dissimilarities between faults, they cannot accurately predict the time-to-failure.

Similarity is not a binary concept. There can be many different levels of similarity between faults in a complex machine with many different components and many different possible faults. We propose a new framework, HMP (Hierarchical Multi-fault Prognosis). HMP consists of first modeling these similarities by extracting a hierarchy of faults. Then, given such a hierarchy, for each internal node, we will train models to predict time-to-failures related to the faults in that node. The output of the node models is the estimated time-to-failure for any of the faults contained in the subtree expanded by the node.

Both trucks and sterilizers exhibit a huge number of possible faults; however, public datasets on predictive maintenance rarely deal with more than a few types of faults. In this paper, we will use the simulated data from the Tennessee Eastman Process that contains a maximum of 28 different faults [5]. Since industrial historical maintenance records are highly sensitive and cannot be shared, we decided to demonstrate the effectiveness of the proposed method using a publicly available dataset; it allows for reproducible results and sets a benchmark for future work.

The contributions of this paper can be summarized as follows:

- We highlight the challenges inherently associated with predicting time-to-failure in multi-fault complex systems, identifying that standard approaches to predict time-to-failure are ineffective, especially in the presence of similar faults.
- We propose the HMP framework, where a hierarchical structure is extracted based on the similarities between faults and subsequently used to predict the time-to-failure for groups of faults.
- We demonstrate through experiments the superiority of our approach compared to the state-of-the-art and identify what the characteristics of the faults that make standard approaches fail are.

The rest of the paper is structured as follows: In Section 2, we formalize the setting and describe the characteristics of real-life problems that motivate

our research; in Section 3, we explain our proposed methodology; in Section 4, we present the experiments and results which demonstrate the validity of our framework and showcase the limitations of the state-of-the-art; in Section 5, we place our contributions in the context of state of the art in the field; and finally, in Section 6, we discuss our findings and present conclusions.

2. Motivation and Problem Formulation

The goal for every company producing machines and carrying out their maintenance is to provide products that are always available when they are needed. One way of achieving this is to have an extensive preventive maintenance system where components are frequently replaced before any risk of breakdown. This is, of course, an expensive solution since it implies replacing components before they are faulty. Another way of achieving this is by having a predictive maintenance system that could tell us when a failure will happen before it actually does. In this way, maintenance operations can be scheduled with minimal disturbance to the operation of the machine, and the cost of replacement of components can be minimized.

For the latter solution, though, we need to record data that can help us monitor the machine's state and its components. Since that also comes with a cost, it is natural that in many practical situations, the data available is of questionable completeness and quality. Therefore, artificial intelligence and machine learning solutions are needed to analyze the data and extract from it the patterns corresponding to different states of the equipment. In the scenarios motivating our research, the data being collected is not recorded to monitor the state of all the individual components, but to control the different processes or validate the processes' quality. For example, sensors are placed in different places of the machine to record pressure and temperature; in addition, control systems provide information about the time it takes for the machine to perform specific tasks. This means that perfect monitoring of the health of every component and the appearance of every fault is not possible.

When all the components of the machine are working as expected, we say that the machine is in a healthy state; throughout this paper, we name the corresponding recorded data as **healthy pattern**. When a fault happens, the machine transitions from a healthy to a faulty state. Eventually, the fault will lead to a discontinuation of the operation, i.e., a **failure** of the machine. We refer to the data recorded between the fault and the failure as the **failure pattern**.

We categorize the different faults into three main categories in terms of their failure patterns.

- We can find types of faults that have no effect on the data collected, i.e., there is no difference between the failure patterns and the healthy data. For example, a fault in the tray that introduces the medical equipment into the sterilizer chamber has no effect on the temperature measured inside the chamber.

- We can find types of faults whose failure patterns have an individual and distinctive effect on the data, i.e., different from the healthy data and different from failure patterns associated with other types of faults. For example, an electronics problem produces a sudden decrease in the "power versus fuel consumption" curve of a truck's engine.

- We can find groups of faults that have very similar effects on the data while being very different from the healthy data. For example, a problem in the vacuum pump in a sterilizer and a leakage in the pipes connecting the pump to the main chamber will have a similar effect on the pressure recorded inside the chamber.

In this paper, we focus on predictive maintenance, more specifically the prediction of time-to-failure, also called remaining useful life, for complex machines. In a real-life application, this includes collecting data in a machine undergoing a fault until the failure happens. Then, we can retroactively label every instance of this data with the time left before the failure. This labeled data is used to

train a regressor which will be used to predict time-to-failure. The output of our predictive system is a list of predicted time-to-failures for each of the possible faults that the machine can undergo.

2.1. Evaluation

In order to evaluate the predictive maintenance solution, we need to understand, and mimic, how this system will be used in practice. For each type of fault, the output of the predictive system is an estimated time-to-failure (sometimes called RUL, Remaining Useful Life). If we depend solely on these outputs to define our maintenance operation, we need to set a reasonable time threshold that gives enough time to schedule a maintenance operation before the failure happens; this is inherently based on the business and application requirements. Once the predicted time-to-failure goes below this threshold, an alarm is issued, meaning that a maintenance operation should be carried out.

The goal of the maintenance operation is to check the prediction and assess the state of the components related to the predicted fault. In practice, several alarms can happen simultaneously if the estimated times-to-failure for different faults all go below the set threshold. Therefore, a technician responsible for the maintenance should check the different alarms until the fault is found.

If the alarm points to a fault that is actually developing in the machine, the maintenance operation should avert the failure. If the alarms do not point at any fault developing at the machine, they will be false alarms. This simplified framework, based on the cost framework presented in [6], gives us a straightforward method to evaluate the quality of our predictive models: we will quantify the percentage of Correctly Predicted Failures (CPF-rate), the percentage of False Alarms when the machine is Healthy (FAH-rate); the percentage of False alarms when a fault is happening, but the corresponding fault is Incorrectly Identified (FII-rate); and the number of tests that the technician needs to perform to isolate the responsible fault (NTest).

In order to ease the evaluation of the models, we will make some idealizations about the performance of the technician and the diagnostics tests that

are carried out. The on-site tests are always correct: when a failure is predicted for a given fault, the diagnostics tests used to evaluate the health of the corresponding components are always correct.

Prediction models are never perfect. Sometimes, a model will continuously keep issuing alarms for the same fault for a long time. For the purpose of the evaluation, if a false alarm is issued and the technician deems it to be not happening, we assume that the fault is not happening and it will not lead to a failure in the near future, so the consequent false alarms will be dismissed.

Our motivating industrial scenarios share common characteristics regarding the costs of unexpected failures and false alarms. An unexpected breakdown is very costly; it means the interruption of service. The machine will not work again until a technician analyzes the machine, diagnoses, and corrects the problem, with all the possible extra delays including lack of necessary material for the repairs, testing the well-functioning of the machine, and so on.

A false alarm is costly since a technician needs to be dispatched to the machine (or the machine needs to be sent to the workshop) and must test the machine to check for the suggested fault. However, the operation of the machine is not affected if the maintenance is scheduled during a natural downtime of the machine. Therefore, the cost of a false alarm is significantly smaller than the cost of an unexpected failure.

In either of the situations mentioned above, more than one alarm can happen simultaneously. In those cases, the technician checks the possible faults iteratively, based on their estimated time-to-failure, until the fault is found. These extra tests on the machine result in an increased cost based on the time and resources needed. We assume that the costs related to checking for each of the simultaneous alarms are small compared to the overall cost of a maintenance intervention.

The specific scenario of the predictive maintenance of sterilizers and trucks is defined by: very complex machines, data that is not good enough to monitor every possible fault accurately, very high cost of unexpected breakdowns compared to false alarms. This scenario is common to many industrial situations.

3. Methods

..

In this section, we describe our framework "Hierarchical Multi-fault Prognosis", HMP. We show how to extract hierarchies of faults, how to train the models at each node and how to use the hierarchy to obtain predictions. But first, we describe the general task of predicting time-to-failure and how to train individual models.

3.1. Time-to-failure estimation for multivariate data

For each failure event, we obtain a dataset $D = (\boldsymbol{x}_t, T, f)$ containing, respectively: $\boldsymbol{x}_t$, the vector of values recorded in the machine describing its operation at time t; the corresponding measured time-to-failure, T; and an indicator of the fault happening in the machine, $f \in \{F_1, \ldots, F_n\}$, where F_i denotes each possible type of fault and n is the number of possible faults.

The general task that we work with in this paper is to train regressors to predict time-to-failure. In our case, the machines undergo only one fault at a time; however, we are interested in predicting, at any given moment, the time-to-failure for many different faults. Therefore, the output of our system is a vector of times-to-failure for the n possible faults:

$$R(\mathbf{x}) = [\hat{o}_1, \hat{o}_2, ...\hat{o}_n], \tag{1}$$

where R is our regressor system, and $\hat{o}_i$ is the estimated time to failure for component i.

3.2. Single-fault regressors to predict time-to-failure

The simplest approach to predict time-to-failure for n types of faults is to train n single-fault regressors. In Figure 1, we can see the workflow to train and use, or exploit, these models. A dataset is collected for each type of fault F_i, and individual regressors R_i are trained for each fault. Then, those models are used to predict time to failure for each type of fault $\hat{o}_i$, the final output presented to the technician.

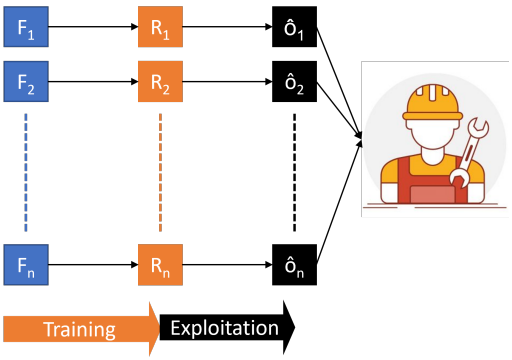

Figure 1: Flowchart for training individual models. For each type of fault F_i a regressor R_i is trained, that outputs an estimated time-to-failure $\hat{o}_i$. The list of the estimated times-to-failure is fed to the technician that will ultimately perform the maintenance if needed.

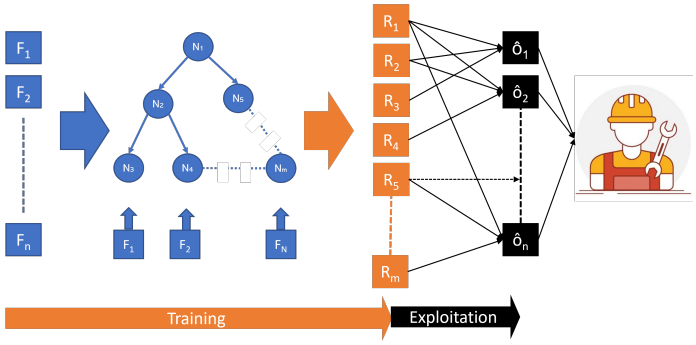

Figure 2: Flowchart for training hierarchical models. A hierarchy of faults is extracted from the data, then for each node N_i, a regressor is trained to predict time-to-failure for any of the faults contained in the node. The output of every regressor will be an estimated time-to-failure for all the faults contained in the node. The list of the estimated times-to-failure is fed to the technician that will ultimately perform the maintenance if needed.

Individual model. The individual model is the most direct adaptation of a single-component model. In this approach, we train an individual model using the failure patterns and healthy patterns associated with the occurrence of faults of the same type. As training set, we only use the failure patterns for the selected fault. $D_{F_i} = D(\boldsymbol{x}_t, T, f = F_i)$. This models can be successful if we assume that there exists a one-to-one relationship between the faults and their effects on the data, i.e., there exists an unambiguous correspondence between each fault and the failure patterns associated with them.

Individual models, crucially, are only train based on the failure and healthy patterns associated with each individual type of fault. If the assumption of one-to-one relationship between the faults and their failure patterns fail, this will lead to many false alarms. For example, if we focus on predicting a fault in a truck related to the tire pressure, we could find a clear pattern in the fuel consumption (the lower the pressure of the tire, the higher the fuel consumption). An individual model could use the fuel consumption signal to estimate the time-to-failure; however, changes in the fuel consumption could be related to many other faults, for example the engine, the hydraulics system or many others. In such cases, the individual model trained to predict a tire-related fault would issue alarms when the fault happening is in fact, caused by another part of the truck.

Contrast model. We can improve individual models by adding information about the rest of the types of faults. In the contrast model approach, we train an individual model for each type of fault, but we also make use of the failure patterns associated with all the other faults, the contrast faults. The goal is to train a regressor that identifies the degree of evolution of the fault, discerning the faulty data from the healthy data, and at the same time can discern between different types of faults.

To achieve this, for the contrast faults' datasets, instead of using the true time-to-failure, we relabel them with a maximum time to failure T_{max}, effectively treating them as if it was data produced by a healthy machine. The

training set we use for this models is therefore, $D_{F_i} = D(\boldsymbol{x}_t, T, f = F_i) \cup D(\boldsymbol{x}_t, T_{max}, f \neq F_i)$

Contrast models, however, also have their drawbacks. For example, assume that two faults are very similar, i.e., they produce the same failure patterns, $\boldsymbol{x}_t$. In this situation, a contrast model would be training a regressor with confusing information: very similar $\boldsymbol{x}_t$ feature vectors with completely opposite time-to-failure labels: first, the correctly decreasing T for the selected fault, and second, constant T_{max} for the contrast fault.

3.3. ***HMP****, Hierarchical Multi-fault Prognosis*

In this subsection, we introduce our proposed framework HMP (Hierarchical Multi-fault Prognosis).

HMP solves the problems related to single-fault regressors by grouping the faults based on their similarity. If two faults are similar, as explained in the previous subsection, the contrast model will fail – but, intuitively, a regressor trained to predict *either* of the two faults would be much more successful. Since there can be many different degrees of similarities, we propose to extract a hierarchy of faults.

In Figure 2, we can see the workflow of the hierarchical approach. Using the data related to each of the faults, we extract a hierarchy of faults based on their similarities; then, at each node of the m modes, one regressor is trained to predict the time-to-failure for any of the faults contained in it. The output of each of the models is the estimated time-to-failure for any of the faults contained in this particular node. It is worth noting that, for a given type of fault i, many different nodes will be independently estimating its time-to-failure, based on different groupings.

3.3.1. Extracting hierarchies of faults

The single-fault contrast models present some advantages over the individual models. However, its performance is harmed in the presence of similar faults. Our goal when extracting hierarchies is to group the most similar faults, while

keeping the rest of the faults as contrast. When extracting the hierarchy, this translates to first grouping faults with a high degree of similarity close to the bottom of the hierarchy, then those that are less similar, and finally those faults that are dissimilar close to the root of the tree.

There is not a single definition of similarity; in our case, we regard as similar those faults that produce similar failure patterns, $\boldsymbol{x}_t$. Using this definition, we characterize each fault by its failures patterns, i.e., the data recorded by the machine from the occurrence of the fault until its failure.

Extracting hierarchies of faults using their failure patterns consist on two basic steps: measuring the dissimilarity between the failure patterns associated with each fault, and clustering them. In this paper, we are going to use the two most popular types of dissimilarity: euclidean based distance and classifier based distance.

Euclidean based distance consists of taking the centroid of each fault's failure patterns and measuring the distances between those centroids in the feature space. Euclidean distance is a very intuitive measure of dissimilarity and very fast to compute; however, it presents certain drawbacks. For example, the centroids of the failure patterns associated with one fault may not be representative or the features that discriminate between fault might be hidden in the presence of multi-dimensional noise.

The idea behind classifier based distance is that if the performance of a classifier trained to separate between two sets failure patterns is good, those sets must be dissimilar, while if the performance is bad, those sets must be similar. To measure the classifier based distance, we are going to train an independent classifier to separate between the failure patterns associated with the different faults.

Once we have evaluated the similarities between faults, we use two methods to cluster them into a hierarchy: agglomerative clustering 1 and divisive clustering 2. To decide the split in the divisive clustering, we will use two methods: k-means, and balanced k-means.

The output of this process is a hierarchy of faults. For our purposes, we

Algorithm 1 Agglomerative clustering

1. Begin with n clusters, each of them corresponding to one fault
2. Compute distance between every pair of clusters.
3. Find the most similar pair of clusters and merge them.
4. Calculate the distance of the newly created cluster to the rest.
5. Repeat steps 3 and 4 until there is just one cluster.

Algorithm 2 Divisive clustering

1. Begin with all faults in one cluster
2. Divide the objects of each cluster into two groups
3. Repeat until all faults are placed in a cluster of its own.

regard the hierarchy as a collection of nodes, each of them containing a set of faults. For example, in Figure 3, there are two inner nodes: S_1, containing faults F_1 and F_2; and S_2, containing F_1, F_2 and F_3.

3.3.2. Measuring classifier based distance

To measure dissimilarity, ideally, we should train a dedicated classifier for every pair of faults. This process is very lengthy, specially as the number of faults increases. Instead, we are going to train a single flat multi-class classifier. During the training phase, we will use a cross-validation scheme to compute the classifier based distance between faults.

For a test set $D = [(\boldsymbol{x}_1, f_1), \ldots, (\boldsymbol{x}_N, f_N)]$ where N is the number of instances in the set, $\mathbf{x}_t$ is the feature vector, $f_t \in \{F_1, \ldots, F_n\}$ is the associated fault; the probabilistic output of the flat classifier is a vector $\boldsymbol{h}(\boldsymbol{x}) = [P(F_1|\boldsymbol{x}), \ldots, P(F_n|\boldsymbol{x})]$. Using these values, we can create a proxy classifier $\boldsymbol{h'}_{j,k}(\boldsymbol{x})$ for any two faults F_j and F_k:

$$\boldsymbol{h'}_{j,k}(\boldsymbol{x}) = \frac{[P(F_j|\boldsymbol{x}), P(F_k|\boldsymbol{x})]}{P(F_j|\boldsymbol{x}) + P(F_k|\boldsymbol{x})} \tag{2}$$

or, in fact, any two sets of faults S_1 and S_2:

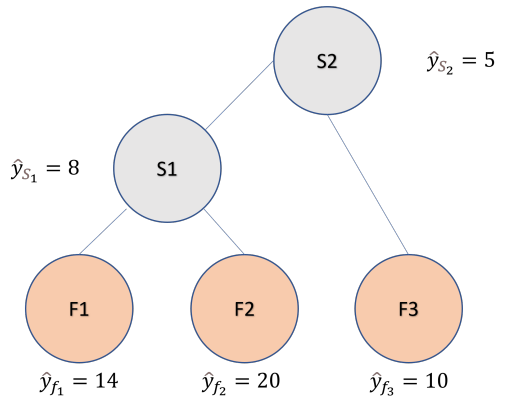

Figure 3: An example of a hierarchy of three faults: F1, F2 and F3; two super-types of faults: S1 and S2; and examples of predicted times to failure.

$$\boldsymbol{h'}_{S_1,S_2}(\boldsymbol{x}) = \frac{[\sum_{i\in S_1} P(F_i|\boldsymbol{x}), \sum_{j\in S_2} P(F_j|\boldsymbol{x})]}{\sum_{i\in S_1\cup S_2} P(F_i|\boldsymbol{x})} \tag{3}$$

Finally, using this proxy classifier, we will evaluate the area under the ROC curve to measure the dissimilarity between two faults, or two sets of faults.

3.3.3. How to train a hierarchical regressor

An example of a hierarchy is shown in Figure 3. The hierarchy consists of a set of nodes: the leaf nodes, each of them corresponding to one of the faults; and the internal nodes, each of them containing a set of faults.

We will use an approach similar to the flat contrast models. At each node, we will train a regressor using the failure patterns of the faults contained in the node labeled with their corresponding time to failure, and the rest of the faults' failure patterns as contrast. If S_n is the set of faults contained in the node and the S_c is the set of contrast faults (i.e., those not contained in the node), the training set will be $D_{node} = D(\boldsymbol{x}_t, T, f \in S_n) \cup D(\boldsymbol{x}_t, T_{max}, f \in S_c)$.

3.3.4. Using HMP to estimate time-to-failure

Single-fault models were implicitly doing fault diagnosis, i.e., they were estimating time-to-failure for a specific fault. The output of a node is an estimated time-to-failure, but the fault responsible is one of the faults contained in the node.

If the estimated time-to-failure falls below a given threshold, an alarm is issued and a maintenance operation needs to be carried out to diagnose the fault and correct it if necessary. An alarm issued by a node regressor is in fact multiple alarms, one for each of the faults contained in the node. In order to reank the faults, and establish an order of preference for the technician, we will make use of the sub-tree expanded by the node.

The output of the prognostic system is a ranked list of estimated time-to-failures for the different faults. We rank the faults by going through the hierarchy starting from the selected node. To select among its children, we will pick the one that predicts the lowest time to failure until we reach one of the leafs.

Once we have found the most likely fault to be responsible for the predicted failure, we repeat the process omitting the selected fault. We repeat this procedure until all faults contained in the node are ranked. In the example in Figure 3, the lowest time-to-failure prediction corresponds to the $S2$ node, meaning that one of the faults contained in that node is predicted to be responsible for a failure in 5 units of time. The order of priority to check the different faults will be $F1, F2, F3$. Note that $F1$ and $F2$ are checked first because the predicted time to failure of $S2$ is smaller than the one of $F3$.

4. Experiments and Results

4.1. The TEP Dataset

The Tennesse Eastman Process (TEP) is a chemical process first introduced by [7]. The TEP consists of a reactor, a product cooler/condenser, a separator and a stripping column with the objective of reacting feed streams and

separating the products. In simple terms, the TEP takes four input streams and converts the contents in the reactor into products. The resulting stream is cooled down and condensed using the product cooler. The resulting liquid stream is then introduced to a stripping column, where dissolved gaseous as well as middle-boiling compounds are removed.

There are different type of signals recorded, such as safety signals: values recorded to monitor some critical values which, if exceeded, would lead to hazardous situations in the plant; process signals, measuring the quantity and the quality of the products of the chemical plant; control variables, the operator or the control system manipulates these variables to control the operation of the plant; and monitor variables, not directly linked to the control or security.

There exist 28 different types of faults. In order to generate datasets, we will use the generator presented in [8].

4.2. Experimental setup

To obtain our dataset, we sample 50 sequences of failure patterns for each type of fault. First, we let the machine run for 1 hour without interference. Then, we introduce the fault, and we assume that the plant goes into failure and stops production after 2 hours of running with a fault. We record data for the 53 features every 3 minutes, i.e., we have sequences of 60 points, and the fault is injected after 20 measurements. We also extract sequences of data without the injection of fault to measure the FAH-rate, i.e., evaluating how often our models predict failure for completely healthy operation. All sequences of data are extracted independently, so we can safely use a 5-cross validation scheme.

Our objective is to learn models that predict the time to failure and identify the fault responsible for it. Without loss of generalization, we set a threshold of 45 minutes as the necessary time to execute maintenance before the failure. If our predicted time to failure goes below this number, an alarm is issued and an intervention by the technician is carried out.

We perform the model selection and final evaluation based on a cost analysis.

	CPF	FAH	FII	# of Tests	Final cost
Empty (baseline)	0	0	0	0	1
Individual	0.78±0.05	1.91 ±0.17	4.6±0.5	1.01 ±0.01	51 ±5
Contrast	0.50±0.03	0±0	0±0	1	0.50 ±0.03
Individual Filtered	0.25±0.06	0 ±0	0.30±0.10	1.01 ±0.01	0.78 ±0.01
Contrast Filtered	0.50±0.03	0±0	0±0	1	0.50 ±0.03

Table 1: Comparison of the individual and the contrast approach on the 28 faults, both before and after filtering the best models.

We assume that the machines are running healthy most of the time, so the frequency of healthy state is 100 times higher than the faulty state. We assume that the cost of an undetected fault is 10 times higher than the cost of a visit of a technician. In addition, we assume that, once the machine is under maintenance, the cost of each test adds 0.1 times more cost to the maintenance operation.

4.3. Predicting all faults

In this subsection, we are going to study the problem of predicting the 28 possible faults using flat approaches (individual and contrast approach), a random hierarchy, and informed hierarchies extracted in different ways.

4.3.1. Flat approaches

In Table 1, we can see the detailed results of applying both flat approaches on the 28 classes. Using all the models, we observe an extremely high cost in the individual approach, due to a large number of false alarms. For every sequence of data being tested, we are getting many alarms from the different models considered. It is worth mentioning that an empty model that would not issue any alarms whatsoever would have a CPF of 0, false alarms rates of 0 and a final cost of 1.

In the case of the contrast approach, we do not observe any false alarms, some models are able to predict the failures they are trained to predict, other models are not successful; but even then, they do not issue any type of false alarm. In total, around 50% of the faults are correctly predicted.

Filtering the per-fault models. Both for the individual and contrast approaches, we are training different models for each fault. Some of those models can be more beneficial, i.e., result in a reduction of costs; but some of them will not be as accurate, or even result in an increased cost due to the large amount of false alarms they produce.

Those per-fault models can be evaluated using the same cost analysis we use for the whole system. There is no need to keep bad models in the final solution, if their contribution to the final cost is negative.

In order to select the per-fault models that result in the best possible cost, we evaluate each model and rank them from most beneficial to the least. Then, we combine them choosing from the most beneficial to the least.

On the left side of Figure 4, we can see the evaluation of the different combinations of models for the individual case for one of the folds. We start from the left side of the x-axis only selecting the best per-fault model, then, iteratively, we add the rest of the per-fault models and evaluate the whole system. The first 3 models added provide obvious benefits, the CPF keeps increasing without any false alarms, neither related with healthy data (FAH) or incorrectly identified faults (IIF). However, with the fourth model, we observe an increased IIF. After the ninth model, the number of false alarms on healthy data (FAH) keeps on increasing. After the seventeenth model, the percent of FAH and IIF exceeds 1, meaning that for each sequence of data, we are obtaining more than one false alarm.

In Table 1, we can observe the evaluation of the combination of models that achieve the best evaluation for the individual approach, only using the best 6 per-fault models. Only 25% of the failures could be averted.

On the right side of Figure 4, we can see the evaluations for the different combinations of models for the contrast approach. 18 of the 28 models are able to predict some failures, the models trained on the other 10 faults are not able to predict any failure. None of the per-fault models produce any false alarm. This the reason why the best combination of per-fault models achieve the same performance as the unfiltered approach, as can be seen in Table 1.

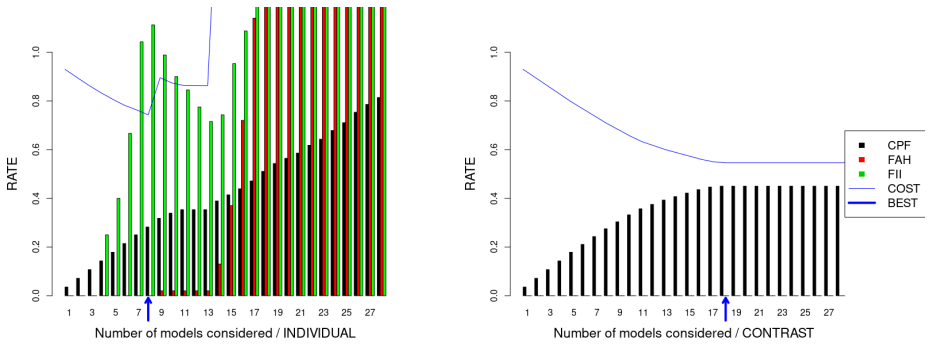

Figure 4: On the left side, evaluation of the different combination of per-fault models for the individual case. On the right side, evaluation of the different combination of per-fault models for the contrast case. For each model, the percent of Correctly Predicted Faults (CPF), False Alarms for Healthy patterns (FAH), and Incorrectly Identified Faults (IIF). In blue, the cost for each combination of models. The best combination of models is highlighted with a blue arrow under the x-axis.

4.3.2. HMP

In this subsection, we are going to evaluate the models created using our framework HMP. We are going to compare different approaches to extract the hierarchies and benchmark it against the single-fault approaches. Our first step is to evaluate our approach on a randomly generated hierarchy. Comparing against a random hierarchy gives us an adequate benchmark to validate the method to extract hierarchy [9]. At each fold, we will draw a new hierarchy, train, evaluate the models for each node, and select the combination of models that provide useful results, similar to how we did in the previous subsection. The results are detailed in the second row of Table 2.

There exists a slight difference in the number of correctly predicted faults between the random hierarchy and the best of the single-fault models, and as a consequence, a difference in the final cost. However, this difference does not prove to be statistically significant after applying the Student's t-test.

In Table 2, we can also observe the detailed performance of the models trained on hierarchies using six different methods. We evaluate hierarchies using the classifier based distance (CBD), using area under the ROC curve as performance, and using the euclidean distance in the feature space between the

	CPF	FAH	FII	# of tests	Final cost
Contrast	0.50±0.03	0±0	0±0	1	0.50 ±0.03
Random	0.52±0.06	0±0	0.003 ±0.003	1.17 ±0.18	0.48 ±0.06
CBD K-means	0.59±0.01	0±0	0.004±0.003	1.24 ±0.01	0.41 ±0.01
CBD Bal K-means	0.58±0.02	0±0	0.003±0.002	1.19 ±0.08	0.42 ±0.02
CBD Agglo	0.55±0.06	0±0	0.002±0.002	1.3 ±0.01	0.45 ±0.05
Centroid k-means	0.55±0.003	0 ±0	0.003±0.004	1.22 ±0.28	0.46 ±0.03
Centroid Bal k-means	0.54±0.006	0±0	0.005±0.002	1.17 ±0.08	0.46 ±0.06
Centroid Agglo	0.53±0.004	0±0	0.003±0.005	1.18 ±0.23	0.47 ±0.04

Table 2: Comparison of HMP with different methods to extract the hierarchies.

centroids of the different types of faults' failure patterns. We also use three different clustering algorithms: agglomerative clustering, divisive k-means, and divisive balanced k-means.

The best performance is achieved by the combination of CBD and K-means. Up to 60% of the failures are correctly predicted, keeping the number of false alarms close to 0. There exists a statistically significant difference between this approach and the flat models with contrast, that in turn were significantly better than the individual models. The improvement over the single-fault model with contrast originates from the models created at the internal nodes of the hierarchy. At least some of them contain faults whose failure patterns are sufficiently similar.

There exists also a statistically significant improvement over HMP using the random hierarchy. This means that in the data, there exist complex and distinctive relationships of similarity between the different failure patterns, complex and precise enough that they can not be found randomly.

On the other hand, hierarchical approaches need a (slightly) higher number of tests to find the responsible fault. This is to be expected. It is our assumption that in the presence of the similar faults, the contrast approach will fail to correctly predict the time-to-failure. A hierarchical approach, able to identify these similar faults and cluster them together, should create a model able to

correctly estimate the time-to-failure to one of the similar faults; but it would be less accurate in selecting exactly which of the faults is the responsible one. In our scheme of costs, where the effort associated with the extra tests needed to isolate the fault is low compared to the cost of false alarms and much lower than the cost of an unpredicted failure, it is clearly acceptable to have this increase in the number of tests in exchange of correctly predicting more faults.

CBD based hierarchies in general outperform hierarchies that used euclidean distance between centroids. Centroid based hierarchies seem to improve the flat model with contrast and the random hierarchy, but the difference is not significant. Euclidean distance can suffer from the curse of dimensionality, specifically, in our case, difference between failure patterns can appear in few specific features, while the rest of the features, which contain a degree of noise in them, remain unaffected by the fault, rendering the calculation of the distance between centroids irrelevant.

Analyzing the results of the different clustering algorithms, we can observe that divisive k-means seems to outperform, marginally, the divisive balanced k-means, and this in turn outperform marginally the agglomerative clustering. None of these differences are significant. We can conclude that, for this specific task and dataset, the biggest source of improvement comes from the method to measure the similarities between types of faults. Once the similarities have been computed, the different clustering algorithms to extract the hierarchies seem to provide marginal gains.

4.3.3. Analysis per class

In Figure 5, we can see depicted the difference in percentage of faults predicted (CFP) by the three different approaches for each type of faults. There are three sets of faults, based on the results of the different approaches. There are faults whose failures seem impossible to predict, those are faults 3, 5, 9, 15, 16, 21, 22 and 28. There are failures whose faults are "easy" to predict, all approaches achieve maximal performance: 1, 2, 4, 6 and 7. Finally, we have the "interesting" faults, those types of faults for which some failures can be

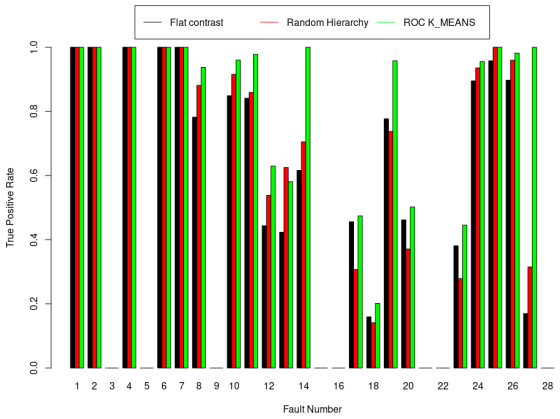

Figure 5: CFP per class evaluated for three different approaches: contrast model, random hierarchy, and the hierarchy obtained combining the CBD dissimilarity and the k-means divisive clustering algorithm.

predicted and some failures can not be predicted depending on the approach taken. Those are faults 8, 10, 11, 12, 15, 17, 18, 19, 20, 23, 24, 25 and 26. If we compare the performances of the flat models with contrast and the model using a random hierarchy, we can see that the random hierarchy tends to outperform the flat models for the "interesting" faults, although there are counterexamples.

The hierarchy using the area under the ROC curve and using a divisive clustering algorithm using k-means always (except one case in fault 13) outperforms the other two approaches. The improvements are not very high for most faults, except for classes 14, 19 and especially 27.

In a complex system, with many types of faults, there are going to be many faults that can be easily predicted, many faults that can not be predicted at all, and some faults where improvements can be achieved by making the right decisions. In our case, evaluating the performance of the whole system, predicting the 28 faults, one can conclude that the hierarchical approach provides a statistically significant but humble improvement over the contrast models. However, if we analyze the results on a per-fault basis, we can see dramatic improvements, for example, for fault 27 we move from less than a 20% CFP to 100%.

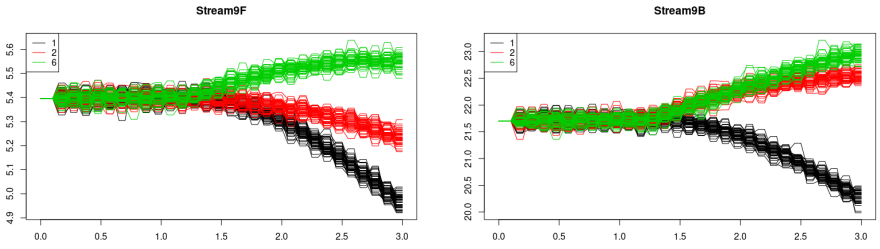

Figure 6: Example of two features corresponding to easy to predict faults. In Stream9F, there is a clear distinction between the 3 types of faults. In Stream9b, however, faults 2 and 6 are harder to distinguish.

4.4. Inherent limitations of the single-fault approaches

In this paper, we are dealing with multi-fault prognosis of complex systems. In the previous subsection, we have demonstrated that hierarchical approaches aiming to capture similarity between faults outperforms single-fault models.

Our hypothesis is that this difference in performance is due to the complexity of the data and the similarity between faults. Different faults can have multiple and varied effects on the data, some of them may be shared by several faults. Models trained only using the failure patterns related to a single fault (Individual models in this paper) risk having a high number of false alarms. On the other hand, if two or more faults produce very similar effects on the data, models trained to estimate time-to-failure and also distinguish between faults (the contrast approach presented in this paper), will fail to predict the time-to-failure since they are presented with confusing information.

4.4.1. Evaluation on easy to predict failures

Looking at the results from the previous subsection, we are selecting the easiest to predict faults. We focus on faults 1, 2 and 6; these faults show a clear trend growing from the injection of the fault until the failure, and these trends are very distinctive for each faults.

In Fig. 6, we can visualize the data corresponding to these three faults. In Table 6, we can see the results of using the flat models with contrast or without contrast, and the best hierarchical approach from last subsection. All

	CPF	FAH	FII	n tests	Final cost
Independent	1 ±0	0 ±0	0.40±0.01	1	0.04 ±0.01
Contrast	1 ±0	0±0	0±0	1	0
CBD K-means	1 ±0	0±0	0±0	1	0

Table 3: Comparison of "individual" and "contrast" for easy to predict faults.

approaches reach a perfect score in the rate of corrected predicted faults (CPF); however we observe false-fault alarms (FII) using the independent models. If we look back at Fig. 6, we could create a model for fault 6, using the *Stream9f* feature, obtaining a model that would tell us when the fault is going to happen with a reasonable accuracy (notice how the data flattens towards the end), and clearly distinguish it from the other faults. However, without using the data from other faults as contrast, it seems better to use the feature *Stream9b*; this feature has a monotonic growth towards the occurrence of the failure. This model will create many false alarms predicting fault 6 for cases where it was actually fault 2 happening.

These results show the importance of not treating different types of faults in a complex system individually. If the same signal can be affected by different types of faults, training individual models can lead to many cross-alarms, predicting the wrong type of fault. Even if the faults are easy to predict, we can have many false alarms, since different types of faults can share similar general symptoms in the recorded data. This situation is alleviated by training individual models for each type of fault, but keeping information from other types of faults as contrast.

4.4.2. Evaluation on hard to predict failures

Based on the results of the previous subsection, we are selecting the hardest to predict faults, the failure patterns related to these faults do not show any particular difference between the data before and after the injection of the fault. These are faults 3, 5, 9, 15, 16, 21, 22 and 28.

In Fig. 7, we have an example of two features related to two of these faults.

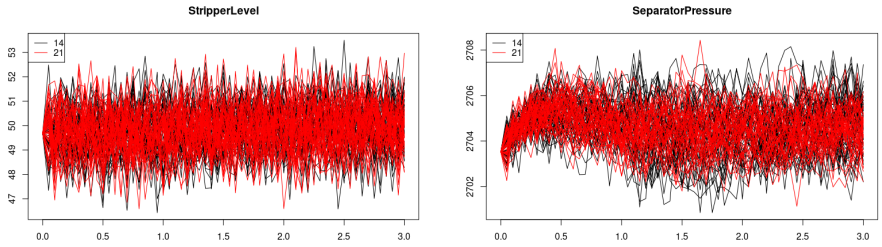

Figure 7: Example of two features corresponding to two hard to predict faults.

	CPF	FAH	FII	n tests	Final cost
Individual	0.61±0.13	2.86±0.34	0.66±0.20	1.63±0.12	30.9±3.6
Contrast	0±0	0±0	0±0	0±0	1±0
CBD K-means	0±0	0±0	0±0	0±0	1±0

Table 4: Comparison of individual flat models, individual models with contrast, and the results of the hierarchical approach with the best extraction method.

It is obvious that it is hard, if not impossible, to predict any failure related to these faults; there is no clear distinction between healthy and faulty data. A side effect of training the individual models on this type of data is the appearance of false alarms when the machine is healthy provoked by random fluctuations in the data. This effect is alleviated in part by training models with contrast, because the models are trained on an unbalanced dataset, since all the contrast faults are labeled with T_{max}, i.e., as if they were healthy.

In Table 4, we can see the results of using the single-fault models trained individually and with contrast, in addition to the hierarchical approach using the CBD dissimilarity and the divisive k-means clustering algorithms. The individual models simply fail, they predict some faults at the expense of creating alarm indiscriminately. As a reference, ignoring the models will "only" have a cost of 1. These types of faults are unpredictable, they do not show any difference between their failure patterns and the healthy data. Both the flat models with contrast and the hierarchical approach fail to predict anything, but they do not incur in false alarms.

	CPF	FAH	FII	Number of tests	Final cost
No contrast	1±0	0 ±0	0.06±0.01	1.5 ±0.1	0.07 ±0.01
Contrast	0±0	0±0	0±0	0	1 ±0.00
CBD K-means	1±0	0±0	0±0	1.5±0.2	0.05±0.02

Table 5: Comparison of individual flat models, individual models with contrast, and the results of the hierarchical approach with the best extraction method with similar faults.

4.4.3. Introducing similar faults

For this experiment, we want to study the effect of similar faults in the models with and without contrast. We will only use faults 1 and 2. To simulate other similar faults, we simply add the same two faults with different label.

In Table 5, we can see the results of using the independent model approach and the models with contrast, and the hierarchy extracted using the CBD distance and the hierarchical divisive clustering algorithm using k-means. The contrast models fails to identify any fault. To train the contrast model, we have provided a dataset with the same failure patterns, but different labels. Both the individual and the hierarchical approaches are able to predict all the faults, at the expense of increasing the number of tests.

There are clear advantages of using the contrast models. It reduces considerably the number of false-fault alarms compared to the individual models, by training the models for each fault using the other faults as contrast. In addition, adding this contrast also reduces the number of healthy false alarms for those types of faults where there is not a clear difference between the healthy and the faulty data. However, they fail to predict failures at all if there are similar ones among the contrast faults.

The hierarchical approaches that are capable of correctly identifying the similarity relationships between faults are able to take the benefits of both approaches: they are able to predict failures in the presence of similar faults, and are able to use the contrast faults to reduce the number of false alarms. They achieve this improvement at the cost of a moderate increase in the number of tests needed to isolate the responsible fault.

5. Literature Review

Estimating time-to-failure for complex systems with multiple possible faults has been pointed out as a challenge as early as a decade ago, in [10]. However, this challenge has not been picked up by the research community. More recently, in [2], this issue is specifically discussed: "with the increase of application level, that is from a component to a system, the complexity prognostics increases... it is suggested to perform prognostics at component or sub-system level. Moreover, in the case of complete system, critical components or sub-system should be monitored or maintained individually rather than prognostics of system".

There are examples in the literature of predicting different types of faults in the same component or subsystem. In [11], the authors introduce seven different types of faults on a rolling bearing-rotor system. Their experiments are carried on in lab conditions, using a test rig by directly introducing faults. In [12], the authors treat a similar problem using an induction motor, instead.

The lack of research on multi-fault systems has been pointed out in other surveys [13], and other research works. In [14], the goal is to predict failures for a multi-fault system in a hydro-electric plant using a Petri net propagation model. The main difference between that research and ours is the data used: while we use signals recorded describing the operation of the machines, they use discrete states obtained using diagnostics tools.

In the field of Condition Based Monitoring, multi-fault prognosis has been extensively studied. In [6], authors propose a method to optimize the maintenance operations on a complex system using as input the multivariate degradation patterns corresponding to different faults. Their work focuses on connecting the failure prognosis and the maintenance optimization. The main difference with our work is that we focus on extracting models that describe the degradation linked to a fault using the data produced by machines, while they take these already found degradation patterns as input.

There has been research focusing on the interaction of faults, more precisely on how some faults can induce other faults in the same system. In [15],

the authors define the different types of interaction between faults. Economic relations describe the effect on the total cost of performing maintenance operations on different components simultaneously or not; stochastic relationships deal with the cases where the operation of one component can affect other components; and structural, where different components form part of the same part or subsystem. In our case, we would add one more category for the relationship between components, components can be related if they affect the recorded data in a similar manner. In [16], the authors do multi-fault prediction focusing on the co-occurrence of faults. The assumptions of their approach is that there is appropriate data to represent all of the failures modes of the system, and that two fault modes must be discernible with the given data. Both assumptions are broken in our scenario; the problems arising from the breaking of these assumptions is the main motivation for our work, and the framework on how to solve them is our key contribution.

6. Conclusions and Future Work

In this paper, we have dealt with the problem of time-to-failure prediction of multiple faults in complex systems given imperfect data, a problem rarely treated in the literature.

We have focused on the challenges that arise when there are similar faults, i.e., faults that provoke the same symptoms on the recorded data. This situation is not ideal for the purpose of predictive maintenance, however, we argue that this scenario happens often in real life industrial applications.

Predicting time-to-failure in the presence of similar faults presents unique challenges. Training individual models for each type of fault risk the presence of a high number of false alarms: the models will pick the most obvious symptoms, that distinguish the faulty data from the healthy data, but will miss the more subtle ones that can discern between different types of faults.

This problem can be partly solved by training individual models for each fault, using the data of the rest of the faults as contrast. Such a model should

be able to distinguish between healthy and faulty data, and between different types of faults. However, this contrast approach can be affected by the presence of faults with similar symptoms. If two faults have the same effect on the data and one of them is labeled as faulty and the other one is used as contrast, the regression model will fail to predict any faults.

We propose a new framework, HMP, that consists on building hierarchies of faults, based on their similarities. We have used such hierarchies to build models that predict the time to failure for groups of faults. The intuition is that, in the presence of two similar faults, models trained individually to predict each one will fail, but a model trained on a combination will succeed. The models trained at the nodes of the hierarchy will output the same estimated time-to-failure for all the faults contained in the node. We further use the structure of the hierarchy to refine this prediction, and rank the faults from the most likely to the less likely to be the responsible one.

We evaluate our approaches in an simplified but realistic scenario using Tennessee Eastman Process data, assuming that maintenance operations will be carried out following the recommendations of HMP predictive models. Our experiments show the superiority of using extracted hierarchies for multi-fault time-to-failure prediction over non-hierarchical ones. Specifically, we show how some types of faults were hardly predictable using single-fault approaches but become predictable with a high degree of certainty by the hierarchical ones.

We have tried different methods to extract hierarchies. We have shown that the key of the improvement in performance resides in how the similarities are computed. In our case of study, classifier based distances seem to outperform hierarchies built using euclidean distance. The clustering method used to extract the hierarchy only gives marginal improvement, and no statistically solid conclusion can be drawn about which method is better.

In real-life applications, the whole predictive system will necessarily be more complex. For example, of different faults will have different costs associated with their maintenance and correction. Similarly, the temporal margin needed to plan, schedule and execute a maintenance operation might be different. An-

other important aspect of many real-life applications is that we do not have information about when the fault starts occurring in a machine, typically we only have information about when a failure happened. It is left for future work to explore how the HMP framework can be used in these circumstances and how to evaluate its corresponding performance.

Acknowledgments

This work was partially supported by "Stiftelsen för kunskaps- och kompetensutveckling" and CHIST-ERA grant CHIST-ERA-19-XAI-012 funded by Swedish Research Council.

References

[1] T. Zonta, C. A. da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, G. P. Li, Predictive maintenance in the industry 4.0: A systematic literature review, Computers & Industrial Engineering 150 (2020) 106889. doi:10.1016/j.cie.2020.106889.

[2] K. Javed, R. Gouriveau, N. Zerhouni, State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels, Mechanical Systems and Signal Processing 94 (2017) 214–236. doi:10.1016/j.ymssp.2017.01.050.

[3] W. Zhang, D. Yang, H. Wang, Data-driven methods for predictive maintenance of industrial equipment: A survey, IEEE Systems Journal 13 (3) (2019) 2213–2227. doi:10.1109/JSYST.2019.2905565.

[4] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, A. K. Nandi, Applications of machine learning to machine fault diagnosis: A review and roadmap, Mechanical Systems and Signal Processing 138 (2020) 106587. doi:10.1016/j.ymssp.2019.106587.

[5] A. Bathelt, N. L. Ricker, M. Jelali, Revision of the tennessee eastman process model, IFAC-PapersOnLine 48 (8) (2015) 309–314. doi:10.1016/j.ifacol.2015.08.199.

[6] K. Verbert, B. De Schutter, R. Babuška, A multiple-model reliability prediction approach for condition-based maintenance, IEEE Transactions on Reliability 67 (3) (2018) 1364–1376. doi:10.1109/TR.2018.2825470.

[7] J. J. Downs, E. F. Vogel, A plant-wide industrial process control problem, Computers & chemical engineering 17 (3) (1993) 245–255. doi:10.1016/0098-1354(93)80018-I.

[8] E. B. Andersen, I. A. Udugama, K. V. Gernaey, C. Bayer, M. Kulahci, Big data generation for time dependent processes: The tennessee eastman process for generating large quantities of process data, in: Computer Aided Chemical Engineering, Vol. 48, Elsevier, 2020, pp. 1309–1314. doi:10.1016/B978-0-12-823377-1.50219-6.

[9] P. del Moral, S. Nowaczyk, A. Sant'Anna, S. Pashami, Pitfalls of assessing extracted hierarchies for multi-class classification, arXiv preprint arXiv:2101.11095 (2021). doi:10.48550/arXiv.2101.11095.

[10] X.-S. Si, W. Wang, C.-H. Hu, D.-H. Zhou, Remaining useful life estimation–a review on the statistical data driven approaches, European journal of operational research 213 (1) (2011) 1–14. doi:10.1016/j.ejor.2010.11.018.

[11] Y. Xue, D. Dou, J. Yang, Multi-fault diagnosis of rotating machinery based on deep convolution neural network and support vector machine, Measurement 156 (2020) 107571. doi:10.1016/j.measurement.2020.107571.

[12] A. Widodo, B.-S. Yang, Wavelet support vector machine for induction machine fault diagnosis based on transient current signal, Expert Systems with Applications 35 (1-2) (2008) 307–316. doi:10.1016/j.eswa.2007.06.018.

[13] J. Z. Sikorska, M. Hodkiewicz, L. Ma, Prognostic modelling options for remaining useful life estimation by industry, Mechanical systems and signal processing 25 (5) (2011) 1803–1836. doi:10.1016/j.ymssp.2010.11.018.

[14] O. Blancke, A. Combette, N. Amyot, D. Komljenovic, M. Lévesque, C. Hudon, N. Zerhouni, A predictive maintenance approach for complex equipment based on petri net failure mechanism propagation model, in: European conference of the prognostics and health management society, 2018. doi:10.36001/phme.2018.v4i1.434.

[15] R. P. Nicolai, R. Dekker, Optimal maintenance of multi-component systems: a review, Complex system maintenance handbook (2008) 263–286doi:10.1007/978-1-84800-011-7_11.

[16] T. Zhang, L. Tao, X. Wang, C. Zhang, S. Li, J. Hao, C. Lu, M. Suo, Hierarchical cognize framework for the multi-fault diagnosis of the interconnected system based on domain knowledge and data fusion, Expert Systems with Applications (2022) 116503doi:10.1016/j.eswa.2022.116503.