A Survey on Diagnosis Methods Combining Dynamic Systems Structural Analysis and Machine Learning

Louis Goupil, Elodie Chanthery, Louise Travé-Massuyès, Sébastien Delautier

A Survey on Diagnosis Methods Combining Dynamic Systems Structural Analysis and Machine Learning

Louis Goupil[1,2], **Elodie Chanthery**[1], **Louise Travé-Massuyès**[1] and **Sébastien Delautier**[2]
[1]LAAS-CNRS, ANITI, Université de Toulouse, CNRS, INSA, Toulouse, France
e-mails: lgoupil@laas.fr, echanthe@laas.fr, louise@laas.fr
[2]Atos, Toulouse, France
e-mail: sebastien.delautier@atos.net

Abstract

This paper reviews diagnosis methods that combine dynamic systems structural analysis and machine learning. A corpus of related articles has been constituted using a thorough research methodology. Three main families of recent research papers have been identified: residual selection methods, residual generation techniques and methods using the structural analysis output to train a machine learning model. A detailed explanation of how each article tackles the diagnosis problem is given. The way these methods make up for structural analysis and machine learning drawbacks by combining them is analyzed.

1 Introduction

Diagnosis methods are often categorized as either model-based or data-driven.

Data-based diagnosis methods, often based on machine learning, stand on algorithms able to learn a diagnosis model without formalized knowledge about the system. They are trained on data recorded from the system — usually with sensors. One of the popular data-driven diagnosis methods is to train a neural network on sensor data paired to the corresponding faults. However, data-driven methods require large amounts of data.

Model-Based Diagnosis (MBD) uses a model of the system elaborated on prior knowledge of the system to estimate how that system should behave. The estimated behavior is then compared with the actual behavior of the system. Nowadays, MBD methods are difficult to use due to the increasing complexity of modern systems. Gathering system knowledge is hard because of this complexity, which leads to some part of these systems being close to impossible to express mathematically. Among MBD techniques, this article focuses on dynamic systems structural analysis (referred to as structural analysis for short in the rest of the paper) [1], which is a tool that allows to compute residual generators (often referred to as just "residuals" with some misnomer) from an abstraction of the model of the system. Residuals are relations linking observable system variables that remain true in nominal conditions. Residuals are designed in such a way that they allow detecting and isolating faults that are diagnosable. Structural analysis specifically computes residuals from analytical redundancy relations identified in sub-parts of the system structural model. It is presented in Section 3. Structural analysis possesses the advantageous particularity that full knowledge of the system is not required to perform diagnosis. This best fits systems which model is not fully known.

Combining data-based and structural analysis approaches is then natural to try to make up for each method's weaknesses. Lately, there has been a surge in synergistic methods combining both model knowledge and the ability to learn from system data.

To the best of our knowledge, there is no survey specifically exploring the combination of dynamic systems structural analysis and machine learning to perform diagnosis, which is the aim of this paper. The goal is to provide a precise understanding of the novel methods in this very specific field. This paper does not address the combination of ML with the full range of model-based diagnostic (MBD) methods. Indeed, there are many approaches that combine ML and MBD, and this would mean a much broader survey.

This paper is organized as follows: Section 2 is about building the corpus of topic-related articles, Section 3 tackles the definition of structural analysis and data-driven methods, while Section 4 describes the methods combining structural analysis and learning from data. Then, Section 5 concludes the paper.

2 Research Methodology

Collecting articles unheard of to broaden the view on a topic is fundamental to aim for the best understanding possible of said topic. This understanding is then key in writing a good and exhaustive review. In order to have an objective corpus of topic-related scientific articles, a three-step systematic search approach was used to complement a premade corpus. This research methodology is inspired by the methodology used in [2; 3].

The first step consists in building a corpus of articles that contains all topic-related articles. For that purpose, a logical search phrase that encapsulates exhaustively all topic-related keywords was defined (see Figure 1).

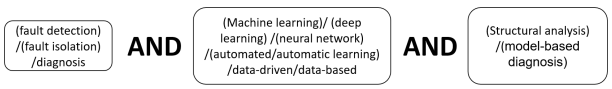

Figure 1: logical search phrase. '/' stands for OR.

Several keywords were obviously given by the survey topic: *structural analysis*, *diagnosis*, *data-driven* for example. Others were taken from articles dealing with the topic

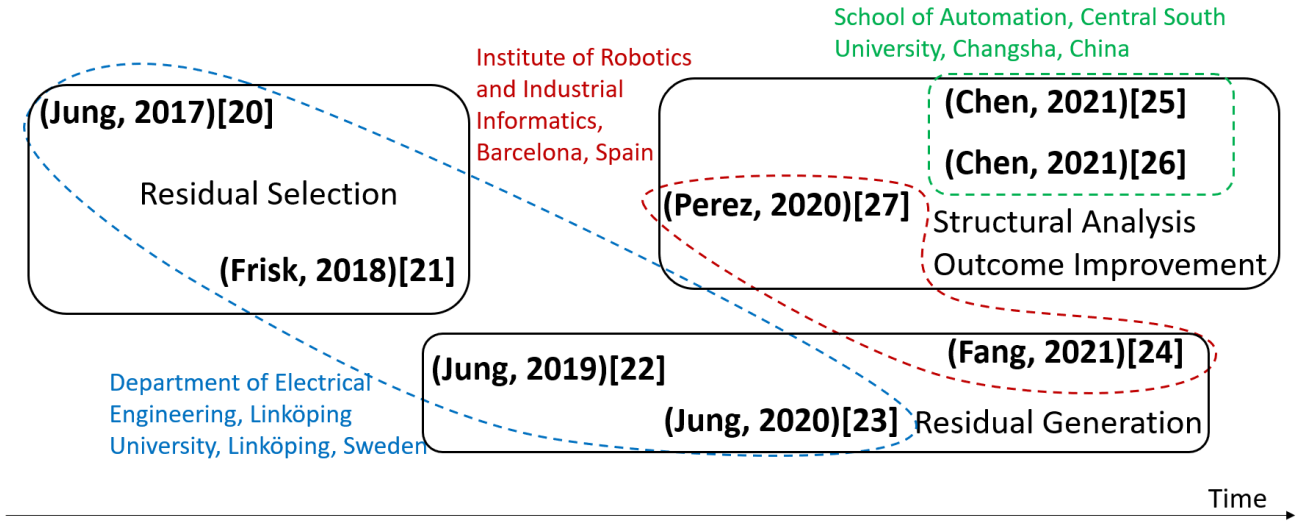

Figure 2: First Analysis of the Corpus

at hand: e.g. *neural network*. Then, this phrase was used to search library databases to constitute a first corpus of scientific articles. The libraries searched were: IEEExplore, ACM, Wiley, Sage, Elsevier (Scopus), PubMed, Web of Science, HAL and Google scholar. The possibility to search in Springer, Crossref and arXiv was explored, but their advanced search option did not allow for a logical phrase to be interpreted. It is notable that, by trying some searches manually on these last websites, all found articles were already in the corpus built from the other websites. For searches in Google Scholar, Scopus, PubMed and Web of Science, a software named *Publish or Perish* [4] was used. It allows searching logical phrases in the databases and returns results ranked by relevance. It advantageously links institutional accounts to search on private databases such as Scopus or Web of Science. A bibtex file can be generated from the results.

The second step aims at trimming this corpus by removing duplicates and documents that are not articles (e.g. presentations, abstracts). The remaining articles are then swiftly read to get rid of those that are not really topic-related. Restricting the search to *fault diagnosis* instead of *diagnosis* was considered, but experience showed it missed some key articles. Indeed, these articles only spoke of *diagnosis* because it was obvious, in the context, that they referred to *fault diagnosis*. For this step, the bibtex display software JabRef was used.

The third step is going in-depth into the remaining articles to make sure they deal precisely with the survey's topic. Table 1 summarizes the number of articles in the set at each step of the process.

At the end of the process, eight articles have been kept. One is in Spanish — it was selected by the systematic search because the title and abstract are in English. The fact that there are only eight articles left shows how niche the topic is. Indeed, the focus is on methods that specifically combine diagnosis using structural analysis and data-driven techniques. The last step, from 62 to 8 articles, consisted in removing all the articles presenting an approach mixing model-based and data-driven diagnosis where the model-based method used was not based on structural analysis.

A first analysis of the articles shows that they are all very recent (oldest is from 2017). They can be classified into three families (see Figure 2). The authors are from three different universities, from China, Spain and Sweden.

After searching in article databases: raw list	898
After removing duplicates (up to four duplicates of a same article)	652
After removing unwanted article types	611
After removing unrelated articles: keeping only model-based and data-driven diagnosis	62
After restricting to only structural analysis and data-driven diagnosis	8

Table 1: Number of articles at each step of the collecting process.

This survey is an attempt to provide a systematic and structured overview of extensive research on diagnosis methods mixing both structural analysis and learning from data using any data-driven method.

3 Background

3.1 General Concepts

Let us define Z, X, and F as the set of known (or measured) system variables, the set of unknown (or unmeasured) system variables, and the set of faults that may impact the system, respectively. Their cardinality is defined as n_z, n_x, and n_f, and z, x, and f are the corresponding variables. The system model $\Sigma(z, x, f)$ — or Σ for short — is a set of differential or algebraic equations $e_k(x, z, f), k \in [1, n_e]$ with n_e the number of equations. Such models can be obtained from physical principles or derived from data using model identification techniques. Identification can be carried out by classical methods [5; 6], but it can also be achieved by methods from artificial intelligence. The model generally represents nominal behavior, thus the violation of one constraint indicates that the system is faulty and points at the responsible component.

Once the model is obtained, the embedded redundancy is studied. The ability to diagnose the system indeed relies on the level of redundancy brought by redundant hardware or

by the sensors. This allows to build residual generators.

Definition 1 (Residual Generator for Σ)**.** A residual generator for Σ is a relation $arr(z', \dot{z}', \ddot{z}', ...) = r$ — with z' a sub-vector of z and r a scalar named residual — such that for all z consistent with $\Sigma(z, x, f)$ it holds that in steady state $r = 0$.

A relation $arr(z', \dot{z}', \ddot{z}', ...) = r$ as defined in Definition 1 is called an *Analytical Redundancy Relation* (ARR). With this definition, an ARR is sensitive to faults in the system. Indeed, deviating from a nominal case will lead to $r \neq 0$. However, not all $f \in F$ necessarily appear in the expression of an ARR. The *fault support* of an ARR is defined as the set of faults that appear in this ARR. Thus, when the residual is non-zero, it means that at least one of the faults of the fault support has occurred.

3.2 Diagnosis Based on Structural Analysis

Structural analysis is a general framework that can be used to analyze large-scale, complex and dynamic systems described by numerous equations, both linear and non-linear. It abstracts equations by only keeping their links with variables. Therefore, it ignores the details of parameter values to base the analysis on the structure of the system by means of efficient graph-based tools [7] and thus a major advantage of structural analysis is that it can be used for systems under uncertainty for which the analytical model is not precisely known [7; 8].

The *structural model* $\Sigma(z, x, f)$ of a system represents this system with its components and the constraints related to these components. It can be obtained by abstracting the functional relations of $\Sigma(z, x, f)$.

The structural model can be represented by a matrix qualified as the *incidence matrix,* which rows are associated to equations and columns to variables. Its elements take the value "1" when the variable is involved in the equation and "0" otherwise.

Equivalently, the structural model can be represented by a *bipartite graph* $G(\Sigma \cup X \cup Z, A)$, where A is a set of edges linking equations of Σ and variables of X and Z. In the context of diagnosis, this graph can be reduced to $G(\Sigma \cup X, \mathcal{A})$, where $\mathcal{A} \subseteq A$ and $\mathcal{A}$ is a set of edges such that $a(i, j) \in \mathcal{A}$ if and only if variable x_i is involved in equation e_j. Hence, each edge links a variable with an equation it belongs to.

Diagnosis via Structural Redundancy

When used for Fault Detection and Isolation (FDI) purposes, structural analysis aims at finding subsets of system equations with redundancy. These can be turned into diagnosis tests, i.e. ARRs or parity relations, which are designed off-line [9]. Diagnosis tests are then checked against observations on-line.

Redundancy in a system of the form $\Sigma(z, x, f)$ can be brought to light by the well-known Dulmage-Mendelsohn (DM) canonical decomposition [9; 10; 11]. It partitions the system into three subsystems:

- Σ^+ has more equations than unknown variables and is named the *structurally overdetermined* (SO) subsystem,
- Σ^0 is the *structurally just determined* subsystem,
- Σ^- has more unknown variables than equations and is named the *structurally underdetermined* subsystem.

If a set of equations Σ is such that $\Sigma = \Sigma^+$ and no proper subset of Σ is overdetermined, this set Σ is qualified as *minimally structurally overdetermined* (MSO) [12]. This means that an MSO set has exactly one more equation than unknown variables, which is a particular case of SO subsystems. Nevertheless, only MSO sets impacted by faults are interesting for diagnosis. This is why the concept of fault support was defined further up.

A *Fault-Driven Minimal Structurally Overdetermined* (FMSO) set is an MSO set which fault support is not empty [13].

Definition 2 (Structural Redundancy)**.** The structural redundancy $\rho_{\Sigma'}$ of a set of equations $\Sigma' \subseteq \Sigma$ is defined as the difference between the number of equations and the number of unknown variables.

If a set of equations is structurally redundant ($\rho_{\Sigma'} > 0$), it means that residuals can be generated using the equations in this set. Once the subsets of equations with redundancy are found using structural analysis, several methods exist to find the analytical expression of the residual generator [14; 15; 16].

An FMSO set φ identifies a just overdetermined subset of $|\varphi|$ equations of the model, among which one is redundant. This means that all the unknown variables can be determined using $|\varphi| - 1$ equations, and that an ARR can be generated by substituting in the $|\varphi|^{th}$ equation. This residual generator can then be used to diagnose faults in its fault support.

Structural Analysis Drawbacks

Diagnosis from structural analysis alone requires deep knowledge of the system. Indeed, the residual generator expression must be known to perform diagnosis and this expression is derived from system equations. Nowadays, the trend is towards increasingly complex systems. This means much harder system equations and thus it is more complex to apply structural analysis to diagnose a system. Also, computational complexity scales with system complexity.

3.3 Data-driven methods

Data-driven methods cover a wide variety of methods ranging from basic machine learning algorithms such as decision tree classifiers to deep multi-layer neural networks such as graph convolutional networks. These methods can have multiple aims such as generation, classification, regression or even encoding. The data-driven methods combined with structural analysis are mainly classification methods.

Those methods can be defined as following [17]: let us consider a classifier Φ from the input space X into the set of class labels C:

$$\Phi \colon X \to C \tag{1}$$

The shape and form of Φ, X and C vary depending on the context and they are specified when required to better understand the method used. The goal of the classifier Φ is to take an unknown $x \in X$ and to predict the right corresponding class $c \in C$ it belongs to. Training Φ corresponds to optimizing its parameters in order to obtain a black box function that predicts correctly — most of the time — the class for the individual of X.

The principle of regression is trying to optimize the parameters of a function with a specific shape to match the input data to the output data. Classification and regression problems are very similar in the sense that both methods

find a Φ that matches inputs with its corresponding outputs. Regression's goal is to find a function that fits corresponding input and output data. A famous data-driven regression method is *Linear Regression* [18].

Data-Driven Methods Drawbacks

When performing diagnosis using data-driven methods, Φ, X and C can take many forms. X can, for instance, be a set of time-series, images, graphs, whereas C can be a multi-class, multi-label space. Φ can also take many forms as presented in Section 3.3. This all adds to the complexity of using data-driven methods since it means requiring different pre-processing and training techniques depending on the case.

Also, diagnosis from data alone requires a gigantic amount of data and often leads to biased diagnosis. What information is learned from the data depends on the quality of the database. Indeed, when a machine learning algorithm learns from data, it can only learn the information contained in this data. There is currently no surefire way to measure the degree of bias in the data. There are some methods to remove some biases, for instance balancing classes, randomizing the order of the training samples, etc. but to be absolutely unbiased would require perfect knowledge of the way data influences training. This often leads to a well-known machine learning problem called over-fitting that consists in having a Φ that is optimal for training data but that does not perform well for unknown data [19]. This explains why there are works directed towards methods complementing data-driven approaches.

4 Combining Machine Learning and Structural Analysis Approaches

This section discusses how structural analysis and machine learning are combined and why this specific combination avoids most drawbacks of a lone use of said methods.

It is interesting to note that combining structural analysis and machine learning for diagnosis is mainly addressed in three laboratories, each having their own way of tackling the problem but with very intertwined topics. Residual selection methods [20; 21] are studied in section 4.1, residual generation methods [22; 23; 24] in section 4.2 and a method used to improve structural analysis results by using a graph convolutional network [25; 26; 27] in section 4.3.

4.1 Residual Selection

Residual selection is an important topic in the field of diagnosis. For instance, if there is an algorithm able to generate a set of candidate residuals, being able to choose, among the candidates, the subset that performs best from the diagnosability point of view, is interesting. Another example of residual selection, in the context of structural analysis, is when choosing which equation of the MSO set is the best support for the ARR in terms of diagnosis performance. The *support equation* is the equation in which unknown variables are substituted so that only known variables remain.

The work presented in [20] tackles this particular case. Residual candidates are computed using different support equations in each MSO. The goal is to find what support equation has the best diagnosis performances. It is assumed that the model of the system is not fully known and that data extracted from the system through sensors is subject to noise. The proposed data-driven method selects the best performing subset of residuals among those generated by structural analysis. The performance criteria are better fault detection and isolation. The generation of candidate residuals is automated using a script that exhaustively computes all possible ARRs. The method flow is described in Figure 3.

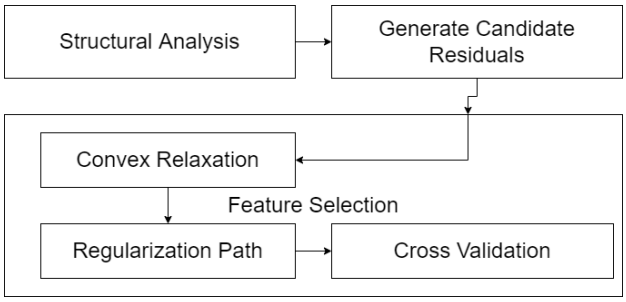

Figure 3: Diagram of method presented in [20]

The *Structural Analysis* step consists in establishing the structural model and identifying the MSO sets. Next step is *Generating* the set of all *Residual Candidates*. Then, a *Feature Selection* algorithm is used to select the most informative residuals among the residual candidates. A feature selection algorithm — usually used in machine learning — describes which feature brings more performance to reach the correct output. A more obvious choice would probably be to go for a multi-class data-driven classifier. However, the author of [20] states that this method is too dependent on the training data and may lead to overfitting. Meanwhile, a feature selection algorithm limits the risk of overfitting and reduces the computational time. The feature selection problem is defined here differently for each fault: the aim is to find a subset of residuals that performs detection and isolation for simple faults, and this is done for all faults. This makes it a binary fault classification problem.

Feature selection is performed as follows: the problem is brought to a convex problem by introducing a logistic regression model. This model is built using the logistic function [28]. The regression is performed on an expression of the form:

$$\lambda + \beta^\top r(t) \tag{2}$$

where $r(t)$ is a vector with a subset of residual candidates evaluated at time t, β and λ are so that $\lambda + \beta^\top r(t) > 0$ means that $r(t)$ belongs to class 0 and $\lambda + \beta^\top r(t) < 0$ means that $r(t)$ belongs to class 1. λ can be interpreted as a threshold for class values. To link this to Equation (1), the regressor is Φ, C is $\{0, 1\}$ and X is the set of all possible subsets of the set of residual candidates. The purpose of convex relaxation is to have a problem where finding a local minimum ensures that it is also a global one. Without going into too much detail, this also allows to prove that a solution $r(t)$ of the convex problem has isolation properties [20].

After *Convex Relaxation* comes a candidate set identification step using *Regularization Paths*. The idea is to find sets that are a solution to the convex problem. In [29], an algorithm is proposed that efficiently finds the regularization path of the β vector for linear models. Regularization paths are a technique that gives all possible residual sets a solution to the convex problem.

Once this list assembled, the last step is to choose among those sets which one has the best performances. This is done by *Cross Validation*. For each candidate set, a new logistic

regression model (see Equation (2)) is trained. Then, the mis-classification rate of each regression model is computed for both training sets and validation sets to be able to select the best candidate residual set.

Once the set is selected, diagnosis can be performed using this set.

Another context where residual selection is useful is when a set of residual generators are at disposal but there are so many of them that only a subset would be enough to reach the same detection and isolation performances.

To answer this case, [21] proposes a systematic method to select, from a set of candidate residuals, a subset with good diagnosis performances. The method proposed is very versatile. Indeed, any model-based method using residuals can be used and any data-driven algorithm on which feature selection can be performed can be used. The proposed method flow is described in Figure 4.

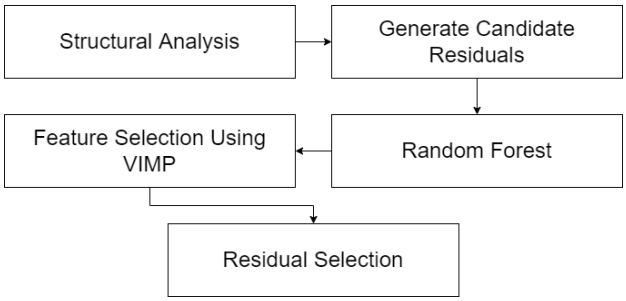

Figure 4: Diagram of method presented in [21]

Just as in [20], the first steps of the method are the *Generation of Residual Candidates* using *Structural Analysis*. In this case, however, only one residual is generated by MSO set. The aim is not to select the best one among those that can be computed in each MSO sets but rather to select a subset of the generated residuals that has the same — or nearly the same — performances as the whole set. Actually, the proposed method strikes the right balance between performance and number of residuals. The main reason for accepting to reduce the performance by reducing the number of residuals is the computational time constraint.

After the generation of candidate residuals comes a data-driven classification algorithm. In this case, a *Random Forest* algorithm [30] is used to predict the fault class according to residual values using all the residual candidates. With reference to Equation (1), Φ is the random forest, X is the set of residual values and C is the set of fault classes. The random forest method takes into account the capacity of each residual to isolate faults. The real aim of this algorithm is not to predict classes but to provide a way to select the features of the algorithm that are the residual candidates.

Indeed, once the random forest is trained, a *Feature Selection* algorithm with the ability to rank features from most relevant to less relevant is used. The metric is called *variable importance*. The feature selection algorithm used is the permuted VIMP [31]. The principle is to measure how the random forest performances would decrease if a particular residual was removed from the inputs. This allows to rank residual candidates by importance.

The last phase is *Residual Selection* where, according to the ranking previously established, a subset of residuals is selected to perform diagnosis. The choice of the number of residuals can be made depending on each specific problem or system constraints. The selected residuals can then be used with, for example, a consistency-based algorithm to perform diagnosis. The input data when training the random forest varies depending on the method used to perform diagnosis in the end. For instance, since consistency-based diagnosis uses binary residuals, the random forest algorithm takes binary residuals as input.

4.2 Residual Generation

This part discusses methods that use only partial knowledge about the system to be able to perform fault detection and isolation through structural analysis by using residual generator built not from the system model but learned using data collected during system operation.

Authors in [22; 23; 24] propose to replace the residual generators obtained through structural analysis based diagnosis with data-driven methods. References [22] and [23] replace the residual generators with *Recurrent Neural Networks* (RNN) although they are designed differently in each article, while [24] uses regression models such as *Robust System Identification* (RSI).

Articles [22] and [23] are written by the same author, Daniel Jung. They are one year apart from each other and use the same method of designing an RNN trainable with system data using the structural model of the system in order to compensate the lack of mathematical knowledge from said system.

An RNN is a type of neural network that is used to model dynamic temporal systems. While artificial neural networks have a fixed number of layers and neurons, an RNN loops on itself by having some neurons outputs used as inputs to other neurons at concurring times. RNNs belong to the class of infinite impulse response networks, since they cannot be unfolded in a straight feed-forward network. Thus, discrete-time non-linear state-space models can be modeled using RNN.

To link this to Equation (1), Φ is the RNN, X is the set of known input variables included in the MSO associated with this RNN and $r \in C$ is the value of the residual.

Figure 5 shows an example of an RNN structure to simulate a residual generator. u is the vector of input variables that intervene in the MSO set, y is the temporal prediction of the residual value. The x are the outputs of each consecutive layer that is fed to the next to retain temporal information. The temporal structure is noticeable since the prediction of time-step $t + 1$ is a combination of the output of time-step t and the input value of time-step t.

Figures 6 and 7 describe the method flow of [22] and [23] respectively. The two first steps are the same: they build the *Structural Model* of the system and then *Identify the MSO Sets*. After that, both articles diverge in their approach.

In [22], Jung uses the MSO sets to write the expression of the ARR by identifying which variables intervene in each ARR and the fault support of those ARRs. An RNN is associated to each residual. An *RNN is Designed* using as input the variables that intervene in the corresponding ARR and trained using data from the nominal mode of the system. The internal structure of the RNN is chosen arbitrarily following general guidelines for neural network design. After training the RNNs, predictions are run on nominal data to estimate threshold output values in which 99% of the data falls. When testing the RNN with unknown data, an output value beyond those thresholds means a faulty situation.

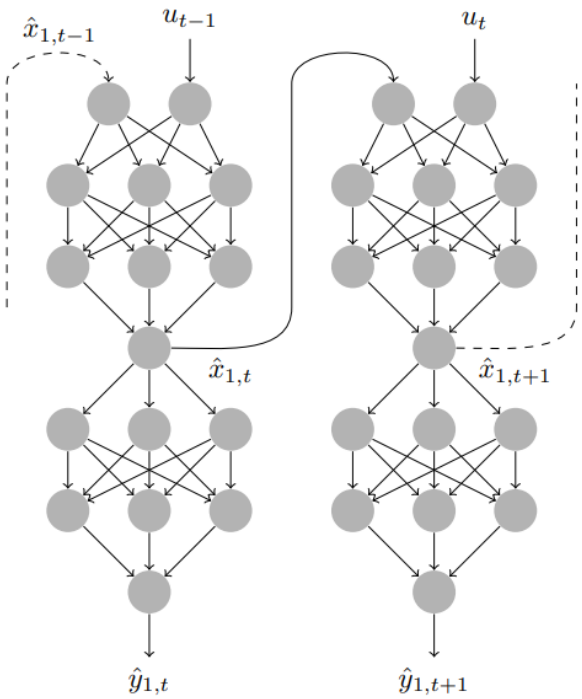

Figure 5: Example of RNN from [22]

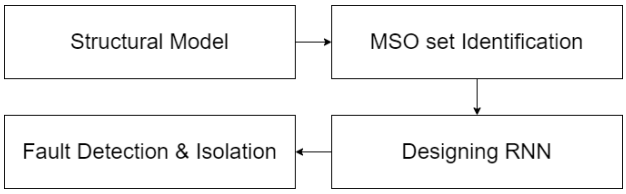

Figure 6: Diagram of method presented in [22]

This accomplishes *Fault Detection*. *Fault Identification* is performed by analyzing the intersection of the model support of the different activated residuals.

In this case, training does not require data from faulty situation, which is very convenient since faults tend to happen less often, making it harder to have a large enough representative dataset of faulty scenarios.

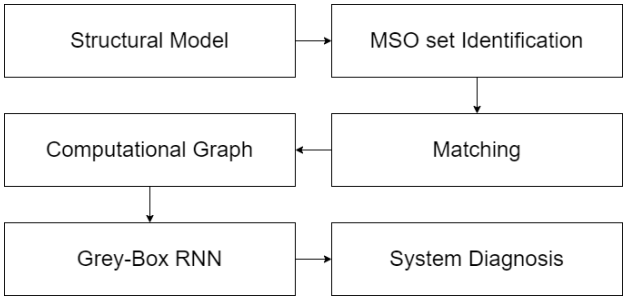

Figure 7: Diagram of method presented in [23]

Meanwhile, after identifying the MSO sets, in [23], Jung performs what is called *Matching*. The matching consists in assigning each unknown variable in each MSO set to an equation of the MSO set. The equation is the one that would allow, with complete knowledge of the system, to substitute this variable with other variables. This allows to identify a remaining redundant equation, theoretically without unknown variables. Next step is designing the *Computational Graph* of each MSO. This is the main difference with the previous article. A computational graph is a directed graph where nodes either denote a variable or a function and edges show how the output of each node are fed as input to other node. Here, the computational graph shows in which order variables are substituted to reach the ARR — meaning the order in which the matching is done. Figure 8 shows an example of such a computational graph. y is sensor data, u is known input, they are both known variables. e_3 is used to get an expression of x_2, then e_5 is used to compute $\dot{x}_2$, e_1 to compute $\dot{x}_1$, e_4 to compute x_1. Combined together, they can be used in e_2 leading to the residual r_2.

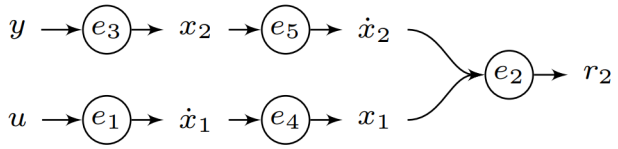

Figure 8: Example of a Computational Graph from [23]

Computational graphs are then used for *Structuring the RNNs* that play the role of residual generators. Those graphs are written in state-space form and the unknown variables included in the equations are computed by backtracking through the computational graph. Once discretized, this formulation can be used as the structure of the RNN model similarly to how it is done in [22]. Here the RNN is called *grey-box* since the inputs and outputs of the RNN are concrete variables of the system. An RNN is usually a black box but this one is partially interpretable through the analysis of input and output values.

After obtaining the RNN, the last step is to train it and use it to *Diagnose the System* in the very same way it is done in [22].

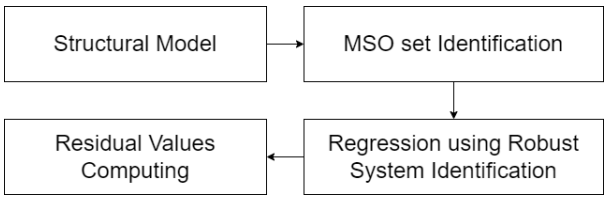

Figure 9: Diagram of method presented in [24]

The work presented in [24] performs structural analysis without concrete mathematical equations of the system. It exploits system knowledge to build the *Structural Model*, *Identifies the MSO sets* and then a residual expression of the form

$$R(z_1, ..., z_n) \tag{3}$$

with $(z_1, ..., z_n)$ being the variables in the MSO set. Which variables are included in each MSO is given by the structural model. R is the residual generator of the MSO set. This is done for each MSO set. [24] then uses *Robust System Identification as a Regression Method* together with labeled data from sensors to evaluate the residual expressions R with actual values. Here the regressor is Φ from Equation (1), the data is X and its labels are the classes C. This method tries

to fit $\Phi(X)$ as good as possible to the labels C so that the resulting residual expression has the best possible accuracy when predicting which faulty scenario is occurring according to input data.

Once a residual expression R is determined for each MSO, they are used to *Compute Residual Values*. These values are converted into Booleans by thresholding. Then, by looking at the *fault signature matrix*, the fault occurring in the system is determined. Figure 9 sums up those steps in a diagram.

Reference [24] has a specific emphasis on the prognosis aspects of the method. While this survey is not about prognosis, it is worth mentioning that the interval model method is used for the purpose of estimating residual thresholds in the case of prognosis.

For all of these residual generation diagnostic methods, it is never necessary to determine the exact expression of the ARR. This means that a complete mathematical knowledge of the system is not a prerequisite to use any of these methods.

4.3 Graph Convolutional Network to Improve Structural Analysis Outcome

The method developed in [25] and [26] both written by Zhiwen Chen consists in performing a full structural analysis to exploit its results through a *Graph Convolutional neural Network* (GCN). It supposes that the results of structural analysis alone are far from perfect and tries to improve it. The method performs fault diagnosis for single-fault scenarios only. The method flow is described in Figure 10.

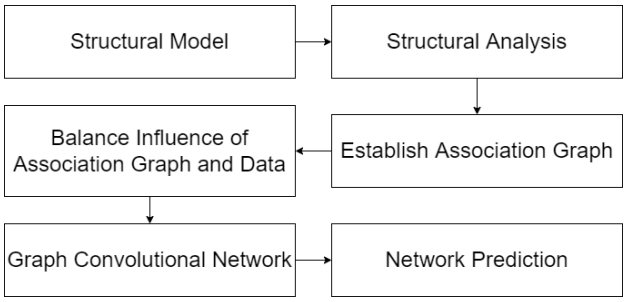

Figure 10: Diagram of method presented in [25] and [26]

Chen first establishes the *Structural Model* of the studied system. Then, all steps of *Structural Analysis* are performed using full knowledge of the system. This means identifying the MSO sets, matching unknown variables and computing the expression of the ARR of each MSO set and then calculating the residual value of each MSO set. A threshold is chosen for each residual and triggered residuals are determined. By considering the intersection of the fault supports of triggered residual, the health state of the system is determined: whether it is nominal or faulty and which fault occurred. This is done for each individual — data point — of the dataset.

To each individual is associated a fault vector f_i of length $|C|$ with C being the set of fault classes (see Equation (1)), including the nominal case. f_i is the null vector except for a 1 at the i^{th} position.

The fault vectors are then used to build an *Association Graph*. The association graph is an undirected graph. Its nodes represent individuals and the value associated to each node is the fault vector of said individual. Individuals that share the same fault vector are linked together by edges. With this definition, the association graph is composed of $|C|$ separate fully connected sub-graphs.

The dataset and the association graph are then fed to a GCN. A GCN is a neural network that can exploit the structure of a graph fed as input. In its first layers, it uses information from neighbor nodes to improve the representation of a node. In [25] and [26], the GCN is based on the spectral domain [32]. In the works of [25] the GCN is made of two graph spectral layers, two convolutional layers and two fully connected layers, in this order. The last layer is a fault vector once again. Figure 11 shows this GCN. The input arrows represent the association graph and the dataset while the output is the fault vector.

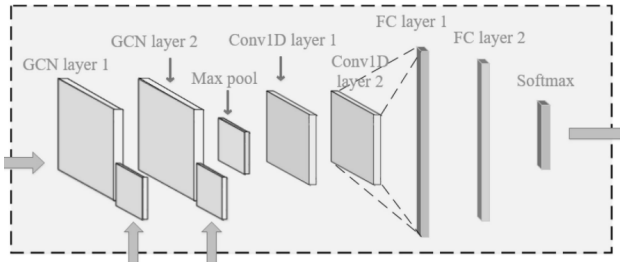

Figure 11: Graph Convolutional Network used in [25] and [26]

Whereas both the association graph and the dataset are input of the GCN, they do not have the same influence. This influence is balanced by a parameter θ following this simplified expression:

$$x = g + \theta a \tag{4}$$

With x being the total influence of what is fed to the graph, g being the influence of the graph and a the influence of the dataset. The value of θ directly impacts what input weights more. To bring back Equation (1), Φ corresponds to the GCN and X is the combination of both the association graph and the dataset weighted by the coefficient θ. x would then be the influence of X. In [25] this θ is chosen arbitrarily.

The main contribution of [26] compared to [25] is the optimization of parameter θ using a particle swarm optimization algorithm [33]. At the start of the algorithm, many possible values of θ are randomly chosen in the solution space. They are the particles. At each iteration, each particle moves towards a local extremum but also moves towards the most extreme local extremum found by the swarm. The authors of [26] use this algorithm to find the best possible θ and thus the best *Balance Between the Influence of the Association Graph and the Influence of the Dataset* on the GCN.

After refining the inputs, the *GCN* needs to be trained. The main benefit of the method happens when training the network. Indeed, only a small part of the dataset needs to be labeled for training. However, the whole dataset is used to build the association graph since it does not need labeling. This allows to extract knowledge by association with unlabeled data when training. Actually, when training the GCN, the whole dataset is passed as input and the loss is the difference between the predicted labels and the true labels of the labeled part only. The rest of the dataset is ignored for the loss calculations.

Online, when the GCN has to predict the class of an unlabeled individual, it first runs it through the structural analysis method to be able to place it inside the association graph. Then it places it as input of the pre-trained GCN. The *GCN Outputs a Fault Vector* that describes the system faulty condition.

Despite having currently no available translation in English, the work proposed in [27] is worth mentioning in this section since it deals with improving the structural analysis results. Just as in [25], a full *Structural Analysis* is performed with full knowledge of the system. The results are then input into an *Artificial Neural Network*. The main difference with [25] is the type of neural network used. The authors of [27] use a fully connected neural network. The framework of the method is summarized in Figure 12.

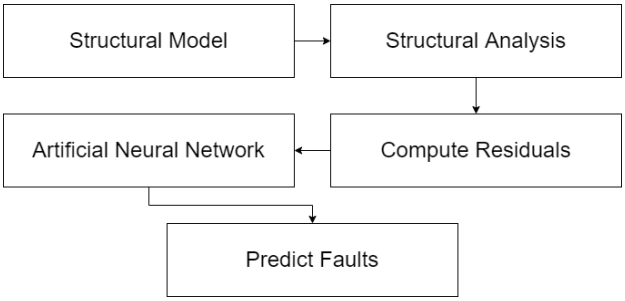

Figure 12: Diagram of method presented in [27]

To reference Equation (1) once again, the dense neural network Φ takes as input the residual values X and outputs the fault class in C. It is trained using residual values obtained through structural analysis and corresponding class labels given by the system actual faults during data collection. The neural network takes as input the actual values of residuals, not thresholded values. The neural network does more than just replace the fault matrix, it extracts information from the actual residual values to help prediction of the fault class.

Once trained, the neural network can be used to *Predict Fault* classes from unknown residual values gathered on the system using structural analysis.

The main drawback of methods presented in this section is that they do not solve the main issue of structural analysis, which is requiring full knowledge of the system.

5 Conclusions

In this paper, we attempt to provide a thorough review of the investigated diagnosis techniques that combine structural analysis as a tool and machine learning. An article research methodology using keyword search in various article databases was used. The 898 collected articles were then reduced to eight that are precisely on the topic at hand. In particular, three main sub-topics were identified: residual selection, residual generation and the use of graph convolutional networks to improve the outcome from the structural analysis method.

- Residual selection: those articles use a machine learning algorithm to determine the best subset of residuals for fault detection and isolation among a set of residual candidates.
- Residual generation: those articles replace the step in structural analysis where knowledge of the system is used to identify mathematical residual expressions with machine learning algorithms that learn these expressions from operating system data.
- Graph convolutional networks to improve the outcome of structural analysis: those articles present a method that performs a full structural analysis in order to diagnose a system, takes the results and feeds them to a graph convolutional network along with data from operating system conditions to output better diagnostics results.

It is interesting to note that some of these methods are not necessarily exclusive and could be used in conjunction to remediate some drawbacks they present.

In the presented papers, machine learning is mostly used around residuals. It might be interesting to explore how machine learning could be used to generate the structural model of a system of which we do not have complete knowledge. Also, many works such as [34] explore residual evaluation with machine learning. This could perhaps be combined with structural analysis.

On a side note, the methods presented in this article often require less training data than fully data-driven methods, but require some knowledge on the system. This is very interesting because it is often easier to get some knowledge of the system (especially from experts) and some data than full knowledge or a lot of data — or data in many different operating conditions of the system. This suggests that these methods might be suitable for deployment in industrial cases.

Acknowledgments

We gratefully acknowledge Atos for providing funds to make this research possible. This project is related to ANITI within the French “Investing for the Future – PIA3” program under the Grant agreement n°ANR-19-PI3A-0004.

References

[1] Marcel Staroswiecki. Structural analysis for fault detection and isolation and for fault tolerant control. *Fault Diagnosis and Fault Tolerant Control*, 2002.

[2] Abdullah and Mohammed Naved Khan. Determining mobile payment adoption: A systematic literature search and bibliometric analysis. *Cogent Business & Management*, 8(1):1893245, 2021.

[3] Guo Haixiang, Yijing Li, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Bing Gong. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73, 12 2016.

[4] A. Harzing. Publish or perish. *Pediatrics*, 89(2):356–356, 2007.

[5] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[6] G. Perez-Zuniga, E. Chanthery, L. Travé-Massuyès, and J. Sotomayor. Near-optimal decentralized diagnosis via structural analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, page 13p., March 2022.

[7] J. Cassar and M. Staroswiecki. A structural approach for the design of failure detection and identification systems. In *IFAC Conference on Control of Industrial Systems, vol. 30(6), pp. 841-846*, 1997.

[8] D. Düştegör, E. Frisk, V. Cocquempot, M. Krysander, and M. Staroswiecki. Structural analysis of fault isolability in the damadics benchmark. *Control Engineering Practice*, 14(6):597–608, 2006.

[9] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer-Verlag Berlin Heidelberg, 2006.

[10] K. Murota. *Matrices and Matroids for Systems Analysis*, volume 20. Springer, 01 2009.

[11] A. L. Dulmage and N. S. Mendelsohn. Coverings of bipartite graphs. *Canadian Journal of Mathematics*, 10:517–534, 1958.

[12] M. Krysander, J. Åslund, and E. Frisk. A structural algorithm for finding testable sub-models and multiple fault isolability analysis. *21st Annual Workshop Proceedings, phm society*, 01 2010.

[13] G. Pérez, E. Chanthery, L. Travé-Massuyès, and J. Sotomayor. Fault-driven structural diagnosis approach in a distributed context. In *20th World Congress of the International Federation of Automatic Control*, pages pp.14819–14824, July 2017.

[14] L. Travé-Massuyès, T. Escobet, and X. Olive. Diagnosability analysis based on component-supported analytical redundancy relations. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 36(6):1146–1160, 2006.

[15] E. Y. Chow and A. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Transactions on Automatic Control*, 29:603–614, 1984.

[16] E. Frisk and M. Nyberg. Brief a minimal polynomial basis solution to residual generation for fault diagnosis in linear systems. *Automatica*, 37(9):1417–1424, sep 2001.

[17] J. Carbonell, R. Michalski, and T. Mitchell. 1 - an overview of machine learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning*, pages 3–23. Morgan Kaufmann, San Francisco (CA), 1983.

[18] Khushbu Kumari and Suniti Yadav. Linear regression analysis study. *Journal of the Practice of Cardiovascular Sciences*, 4:33, 01 2018.

[19] X. Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168:022022, feb 2019.

[20] D. Jung and C. Sundstrom. A combined data-driven and model-based residual selection algorithm for fault detection and isolation. *IEEE Transactions on Control Systems Technology*, 27(2):616–630, 2017.

[21] E. Frisk and M. Krysander. Residual selection for consistency based diagnosis using machine learning models. *IFAC-PapersOnLine*, 2018.

[22] D. Jung. Isolation and localization of unknown faults using neural network-based residuals. *arXiv preprint arXiv:1910.05626*, 2019.

[23] D. Jung. Residual generation using physically-based grey-box recurrent neural networks for engine fault diagnosis. *arXiv preprint arXiv:2008.04644*, 2020.

[24] X. Fang, V. Puig, and S. Zhang. Fault diagnosis and prognosis using a hybrid approach combining structural analysis and data-driven techniques. *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2021.

[25] Z. Chen, J. Xu, T. Peng, and C. Yang. Graph convolutional network-based method for fault diagnosis using a hybrid of measurement and prior knowledge. *IEEE transactions on cybernetics*, 2021.

[26] Z. Chen, J. Xu, H. Ke, X. Fan, and T. Peng. Graph convolution network-based fault diagnosis method for the rectifier of the high-speed train. *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2021.

[27] E. Pérez-Pérez, F. R. López-Estrada, and V. Puig. Diagnosis of faults in a wind turbine using analytical redundancy relations and an artificial neural network. *amca.mx*, 2020.

[28] J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, Mar 2005.

[29] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407 – 499, 2004.

[30] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[31] B. Williamson, P. Gilbert, N. Simon, and M. Carone. A unified approach for inference on algorithm-agnostic variable importance. *arXiv*, 2020.

[32] T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[33] M. R. Bonyadi and Z. Michalewicz. Particle swarm optimization for single objective continuous space problems: A review. *Evolutionary computation*, 25(1):1—54, 2017.

[34] Carl Svärd, Mattias Nyberg, Erik Frisk, and Mattias Krysander. A data-driven and probabilistic approach to residual evaluation for fault diagnosis. *Proceedings of the IEEE Conference on Decision and Control*, pages 95–102, 12 2011.