UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Máster Universitario en Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería

Trabajo Fin de Máster

Automatic (Data-Based) Learning of Expressive Indicators for Machine Fault Detection: A Proposal

Autor: Adriel Sosa Marco

Tutor: Blas José Galván González

Septiembre de 2015

Acknowledgements

Abandonando toda ortodoxia, me predispongo a agradecer a todas aquellas personas que han contribuido a la consecución de este trabajo. Para ello empleo mi lengua natal ya que, de cualquier otra manera, probablemente no podría llegar a hacerles sentir mi total gratitud.

En primer lugar agradezco a Blas Galván haberme abierto la puerta del grupo de investigación CEANI donde tanto he trabajado y aprendido y donde tan cómodamente lo he hecho. Asimismo, agradezco a Juan Pedro Ramos el haberme cedido una idea original para plantearla y desarrollarla durante este tiempo. Sin sus trabajos previos probablemente este documento versaría sobre otra cosa distinta.

A mi sempiterno Marcos, que nunca duda en privarse de sueño si se trata de apoyarme en cualquiera de las quimeras en las que se me ocurre embarcarme. A Guillermazo, quien me ha ofrecido tanto momentos de desconexión y de risas como contribuido con su genio a que este proyecto vaya cogiendo rumbo. Sin ellos dos este trabajo no podría haberse entregado a tiempo.

A mi pareja Mónica, en primer lugar por ser un apoyo constante durante cada día y, en segundo lugar, por permitirme sin objeciones ni condiciones, dedicar una gran parte de tiempo a lo que me gusta.

Finalmente agradezco a mi padre Carlos, mi madre Montse y mi hermano Ruymán su soporte constante y por haberme transmitido los valores que hoy me definen.

¡Perfecto, seguimos!

Onthology

As existing vocabulary is wide and sometimes lack of standardization it has been found necessary to establish and homogenize the nomenclature for sake of clarity and accuracy.

Task/subtask: tasks have a specific objective and is defined by its inputs and resulted outputs. Required functionalities are described by tasks. Any task could be decomposed in several subtasks.

Method: a method specifies how to perform a specific task. Applied method will set required format of inputs and a data flow. Methods also impose which subtasks have to be considered to perform the intended task.

Inference: It defines a reasoning step related to certain desired objectives and also constitutes the final goal for which a task is built.

Technique: It is any mechanism devoted to the implementation of an inference (e.g. artificial neural networks, expert systems, Markov models, etc.)

Indicator: Every input transformation is considered an indicator. Indicators are meant to enhance input signals providing enough expressiveness to make an inference.

Health indicator: It is an indicator that fulfils a particular condition. This is, its evolution over time have to be a non decreasing functions as a certain failure mode develops. For those failure modes that do not exhibit degradation this indicators have stepped profiles. Health indicators enable prognosis inferences.

False alarm rate: False alarms from the condition monitoring system. When system state is classified as unhealthy while it is healthy instead. It is defined as the ratio between the number of incorrectly classified samples as unhealthy states over the total number of healthy instances.

Missing alarm rate: (or false negative) It occurs when system state is classified as healthy while it is unhealthy instead. Defined as the ratio between the number of incorrectly classified samples as healthy states over the total number of failures.

Fault Diagnosis: Detecting, isolating and identifying an impending or incipient failure condition–the affected component is still working but in a degrade mode.

Failure Diagnosis: Detecting, isolating and identifying a component that has ceased to operate.

Prognostic: It is the ability to predict accurately and precisely the remaining time to failure of a failing component or subsystem.

All contents in this work that will be developed from now on will be referred to this terminology: Hence, author recommend to keep them at hand to avoid misunderstandings trough the exposition.

Table of Contents

List of Figures

List of Tables

Chapter 1
Background, Motivation and Objectives

1. Wind energy technology overview

Of all renewable energy technologies, wind power currently has the strongest market penetration due to its competitiveness to conventional power generation. There are now commercial wind power installations in more than 90 countries with total installed capacity of 318 GW at the end of 2013, providing about 3% of global electricity supply(**Global Wind Energy Council 2014**). Starting from markets such Denmark and Germany, there has been a change in the world market in recent years with strongest growth in China (25% in 2013), India and USA. Besides, attention have to be paid to new emergent regions of growth such as Brazil, Mexico and South Africa.

With respect to Spain, according to 2014 annual report elaborated by the national grid operator (Red Eléctrica de España, REE), installed capacity of wind power covers 22,3% of the global sources of energy technologies, 47,4 % if only renewable sources are taken into account. In 2014, from a total electric energy demand of 243.530 GWh, up to 20,3% came from wind energy conversion, with a peak value of 34,5% registered for an energy demand of 38.666 MWh.

Wind energy development and deployment will keep its pace in future years to meet objectives concerning climate change, greenhouse gas emission reduction and requirements of energy independence mostly demanded by regions without fossil or nuclear reservoirs. In this sense, Global Wind Energy Council (GWEC) provides three possible development scenarios (Figure 1) regarding different management policies about previous objectives. Thus, the most conservative scenario is based on current adopted commitments by governments in such fora as G-8/G-20 but still not enacted into laws. Second proposed scenario assumes that current intentions on wind energy development and carbon dioxide emissions are legislated although in the modest side. Finally, third scenario is the most ambitious and is formulated in relation to the fact that effective and forceful laws are adopted.

Wind energy generation is gradually moving from onshore locations to offshore chiefly because two reasons: quality of resource is better in the sea due to higher and more constant wind velocities, which positively impact both amount of energy that a certain wind turbine is able to produce and production management, since sea winds are more predictable. The second motive accounts for onshore lack of space for constructing large wind farms, especially in small regions such as islands. This panorama has auspiced the development of new designs capable of adapting to these new operational conditions. Thus, leading offshore context to inexorable reliability issues caused by immature technology and lack of experience.

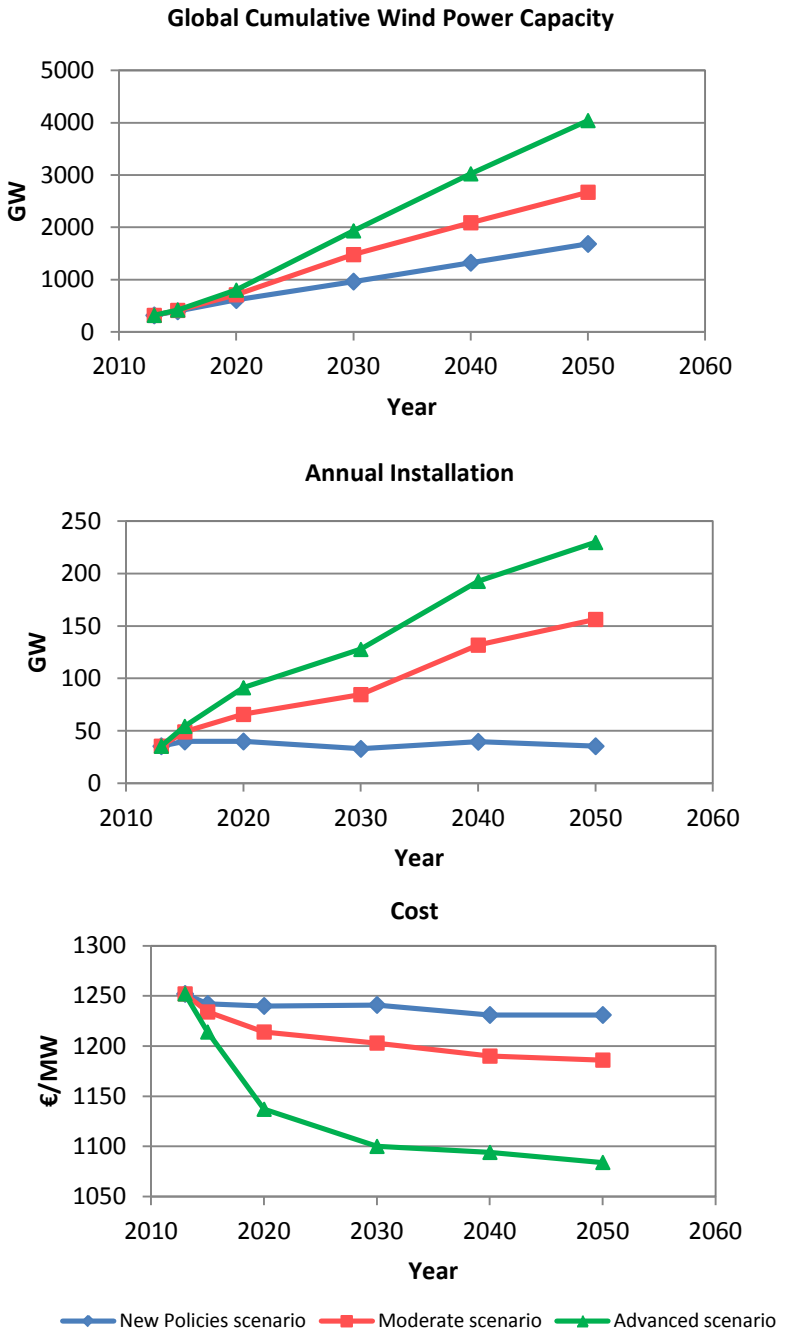

Figure 1: Wind energy forecast based on GWEO report

Levelized Cost of Energy (LCOE) is a measure which attempts to compare different technologies of electricity generation in a comparable basis. It is defined as the net present value of the unit-cost of electricity over the lifetime of a generating asset. Under this frame one may realize that offshore wind energy generation it is still very far from some conventional or even other renewable alternatives (**Kost et al. 2013**). LCOE is significantly affected by maintenance actions, rated by some authors (**Godwin and Matthews 2013**) as 20-25% of total asset cost, of which, up to 75% is due to unscheduled maintenance. This discourage future investment, reducing long term economic viability of wind energy. Thus, failure management is capital in order to make offshore generation competitive, which is not a trivial task. Some reliability studies have shown that operational unavailability of wind turbines reaches 3% of its

lifetime. Blades, control system and electrical system have been identified as the components with highest failure rates, causing several short downtimes per year. By its part, gearbox, generator, drive train and also blades have the highest downtimes per failure. This is also a problem because, besides replacements and maintenance infrastructure costs, loss of production costs are rather severe related LCOE of wind turbines.

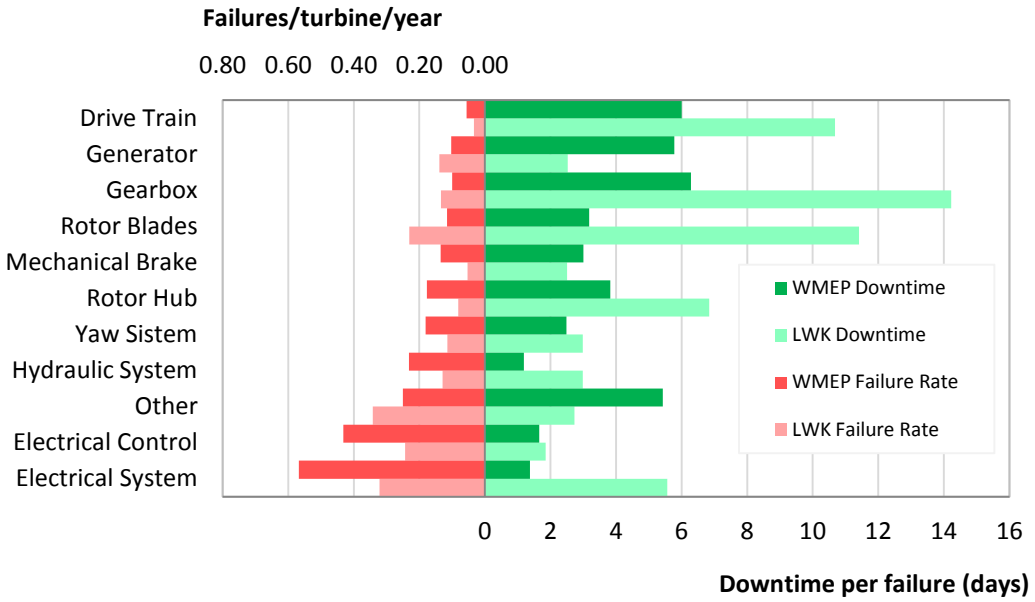

Figure 2: Failure rate and downtime periods for main wind turbine assemblies - Used with permision of (Crabtree et al. 2014)

Addressing maintenance management of such critical components still remain a challenging task and a matter of active research, moreover for offshore facilities. Maintenance carried over offshore wind turbines often involve much infrastructure which also have higher unit cost. As an example, is not unusual to have a fleet of specialized ships to perform different tasks (personal transportation, material transportation, etc.) or even a floating base for operators, since wind farms usually are miles away from the coast.

Given all elements intervening in maintenance, turning to proactive more efficient management policies has become almost mandatory so as to ensure the viability of offshore technology and increment its penetration within the market.

2. Motivation

Condition monitoring for wind farms maintenance management has gained great attention in the last years because its impact in cost of energy and dependability in their operation. Several companies are devoting large efforts investing both monetary and human resources to adapt their maintenance policies to a much more proactive ones. They have realized that new placements of wind farms, which also include offshore locations, will require renewed strategies to maintain efficiency standards in the exploitation of this kind of facilities and keep them profitable.

Moreover, given the consciousness existing around these topics, within the sector several opportunities are being created to obtain financing from both public and private organisms. This framework stimulate companies to be involved in big projects that provide them the opportunity to enhance productivity and setup new operation protocols. Also, this situation encourages research institutions to work into these lines promoting cooperation between industry and research groups, which is highly desirable when it comes to engineering development.

As a proof of previous statement, CEANI R&D group (Computación Evolutiva y Aplicaciones Numéricas en Ingeniería) has been involved within the last six years as part of three consortiums regarding the development of conditioning monitoring systems applied to wind turbines in order to improve the logistic of maintenance of such systems, viz: Ocean Lider (founded by Centro para el Desarrollo Tecnológico e Industrial (CDTI) and Ministerio de Economía y Competitividad (MINECO) from Spanish government), LeanWind (founded by European Commission under the 7th Framework Programme for Research) and a cooperation agreement with a national power company.

The approach presented in this work is the result of the conclusions achieved when working in Ocean Lider project, willing to apply new developments to the latter two projects, which are currently in progress. Two items of improvement were identified back then: on the one hand, the need to establish a systematic procedure in order to extract expressive features for the failure detection task. And, on the other hand, to define a rigorous procedure to obtain a diagnostic about the health of the machine from the symptoms previously isolated. Hence, the main research line in which the group has been interested the last two years has been the obtainment of expressive indicators for specific failure mode characterisation. This work address this line trying to meet some particular goals, described in the following section.

3. Specific objectives

The objectives of this project can be grouped into two different classes: by one side there are those related to academia and, in the other side, those which concerns about the results expected of the proposed methodology. From the point of view of academia three objectives are identifiable:

- To finish and, consequently, to obtain the Master degree on *Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería,* and proceed with Phd studies.
- To get deeper knowledge in the field of industrial reliability, particularly that related to early detection of failures in wind turbines subassemblies. Furthermore, to take further knowledge about data-driven machine learning.
- Reinforce and acquire new skills in research context.

Regarding the expected contributions, the main objectives are the following:

- To develop a methodology for automatic learning of expressive indicators for early failure detection so as to deal with learning tasks in commonly situations encountered when working with industrial datasets.
- Present a vision of condition monitoring technology applied to the field of wind energy. Specially a comprehensive state-of-art on available techniques for failure detection, diagnosis and prognosis will be developed in order to place presented methodology in its context.
- Build up transparent (i.e. interpretable) models to serve as indicators for failure detection.
- Usually, the obtainment of indicators from raw data involves the use of expert knowledge to decide how to transform such space in order to get expressive measures. In this sense, one of the proposed objectives is to change the qualitative role of expert knowledge in the process of identifying potential indicators by limiting its influence to the configuration of the search space of transformations instead of selecting which to deploy.
- Manage databases coming from industry and perform data pretreatment in order to be used for condition monitoring.

3.1. Structure of the document

- Maintenance Paradigms for Wind Turbines: State-of-Art (Chapter 2)

Stages of condition monitoring, different approaches and methods and available techniques involving recent published works are covered. The study is centred in research made in wind turbine technology. Furthermore, current improvable topics and possible contribution opportunities are outlined.

- Proposed Methodology Description (Chapter 3)

In this section we introduce the facts that motivate the work and its expected contributions within the current state of the art. Theoretical notes on the search approach are detailed and also the essentials of used techniques. We devote a subsection to perform some preliminary experiments to test the effect of different parameters on the search process. Besides, some different problems are addressed to test the capabilities of the presented approach.

- Test case: Application on bearing dataset (Chapter 4)

We move from synthetic experiments to deal with real data coming from Run-To-Failure experiments carried over a set of bearings. In this chapter we try to extract some conclusions about technique performance on real data, but obtained under controlled operational conditions.

- Conclusions (Chapter 5)

We review accomplished milestones and propose future research directions to further develop current proposal and apply it to more exigent data.

- References

Consulted sources to develop this work are detailed in this section

- Appendix

Additional plots which complement those already exposed in each chapter are grouped in this section.

Chapter 2

Maintenance Paradigms for Wind Turbines: State-of-Art

I. Maintenance management: generalities

Setting maintenance strategies is not an easy task, it involves numerous variables that have to be considered in order to schedule required actions. Furthermore, each of available approaches needs their own sources of information. In general, maintenance approaches are divided in two big categories: corrective maintenance and preventive maintenance.

Corrective maintenance is performed after a failure has occurred. There are two types: urgent corrective maintenance and programmed corrective maintenance. In the first one, the system remains unavailable in a faulty state, but in the second one, the system is still operating but it cannot be used under its optimal conditions (i.e. performability is downgraded).

By its part, preventive maintenance is intended to prevent the system of undergoing a failure. Employed criteria (trigger events) to anticipate failures actually define the maintenance strategy. Maintenance may be planned using statistical descriptors such as RAMS (Reliability, Availability, Maintainability and Safety) related measures to fix constant intervals at which the system will be maintained. This strategy is called *periodic preventive maintenance*. Other approach consist of monitoring system condition to schedule maintenance activities only when the system is found to be operating degraded. Condition monitoring (CM) could be continuous or discrete depending on whether the monitoring task is done online or on demand. When doing it continuously a remote acquisition system is required to gather data. By contrast, if CM is performed offline then surveillance intervals are planned in order to access the data. These intervals are configured in a similar fashion than periodic preventive maintenance. However, repairs or replacements are only undertaken if detected condition is bad or it will be bad in the near future.

Either be preventive or corrective maintenance, there are three possible actions. First one involves the upgrade of the system in order to fix the failures or detected malfunctions. This activities will restore system performance to Good-As-New (GAN), Bad-As-Old (BAO) or Better than Old but Worse than New (BOWN). BAO reparation is usually known as *palliative maintenance* in which some minor reparations are performed just to keep system available. The second kind of maintenance actions is the replacement. This also comprises some minor reparations which involves the substitution of degraded components. Finally, after some periods it becomes useful (if it is cost effective) to subject the system (a multicomponent system) to a major maintenance action in order to improve significantly its reliability. This is call overhauling actions or simply *overhaul*.

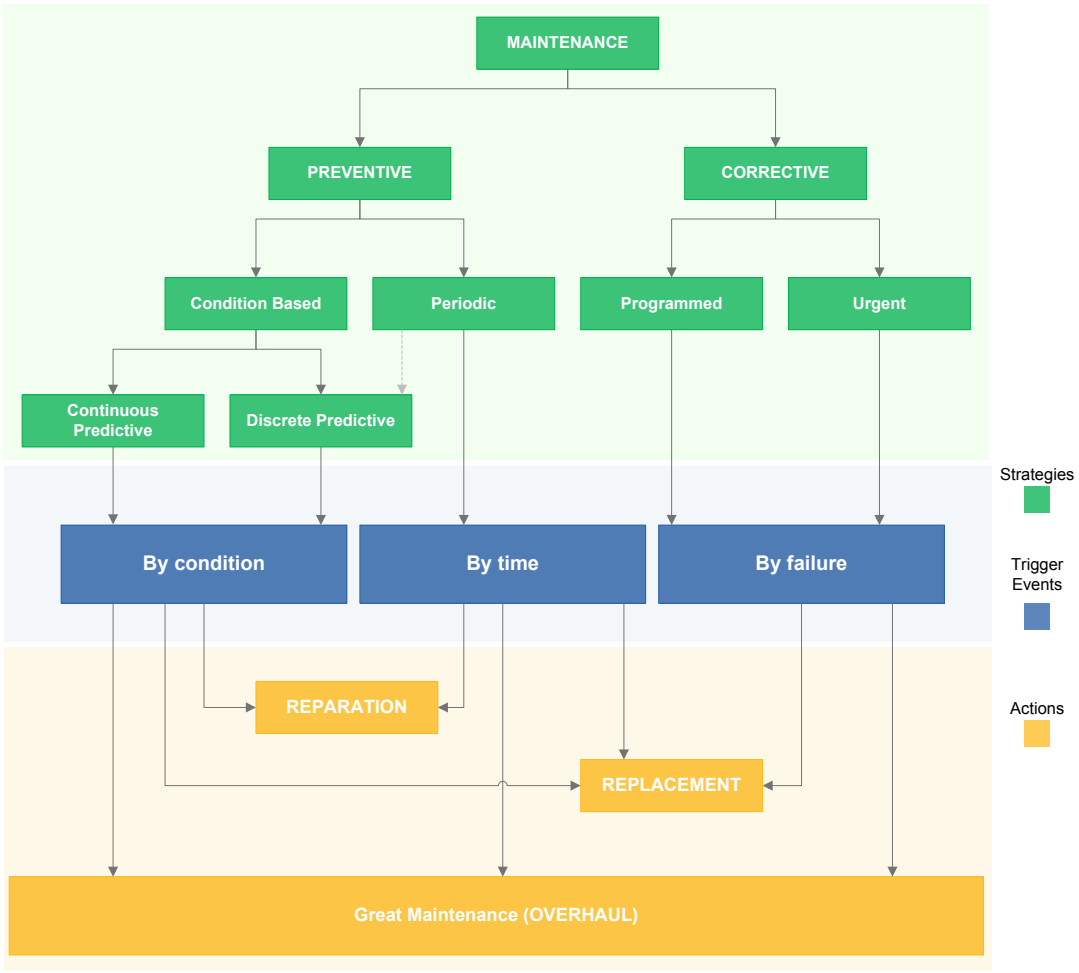

Figure 3: Maintenance strategies and actions

1.1. Integration notes: unifying criteria for different approaches

The maintenance programming looks for the optimal possible maintenance strategy, always trying to improve some criteria involving cost and/or risks. It is not in the scope of this work to go deep into this topic but providing some useful notes for condition monitoring strategy integration within the whole maintenance planning.

The planning task gets harder when all the maintenance possibilities are considered. It have to be noticed that periodic maintenance is a long/medium term strategy, corrective maintenance is reactive and condition based is a short term proactive strategy. This introduces difficulties when trying to set the whole maintenance scheduling. Usually intervals for preventive maintenances are derived at the beginning of the operation and whenever the exploitation goals (availability, risks, costs, etc.) are not fulfilled. However condition monitoring cannot be addressed in the same procedure as it depends in the instantaneous operation of the wind turbine (WT). The way in which both maintenance approaches could be combined is not a trivial matter.

One possible way to overcome this problem is to introduce some indicator variables, whose allowed values are zero (not selected) and 1 (selected), as decision variables in the maintenance optimization task. These variables indicate if a certain component will be managed under the condition monitoring approach or not. It has to be noticed that not every component can be included in the CMS as this means that a high inversion on sensor infrastructure will be required. Furthermore, since sensors are as prone to fail as WT subassemblies, this leads to an ineffective strategy. Accordingly to aforementioned issues it would be beneficial to **derive a cost model** describing

implications of involving a certain wind turbine subassembly or component into the condition monitoring system. Some related facts coming from CEANI R&D previous experience in various projects are detailed below.

- Normally based in midday journeys per maintenance ship. Between 6-7 hours really working on wind turbines.
- With good weather a ship serve up to 20 wind turbines in one journey
- No maintenance activities are carried out during the night, so everybody must be off the wind farm at the end of the day
- Preventive Maintenance only performed between April and September.
- Ships cost is not relevant compared with the cost of wind turbine downtimes.

All described facts but one are referred to the detection horizon, that is the time at which the failure is evidenced. The estimation of remaining time to failure, based on the current condition, has different levels of accuracy and certainty depending on the length of the time interval until failure. The accuracy of the prediction becomes higher when the failure is near to happen and vice versa.

In terms of maintenance planning, useful information that could be provided from condition monitoring analysis is:

Probabilistic detection horizons: this is a curve specifying the probability of early detecting a specific failure mode some certain time in advance.

Rate of false alarms: this account for the proportion of times that an specific failure mode has been incorrectly identified. If this measure is high, then maintenance will be overscheduled and costs will increase dramatically.

Rate of missing alarms: this measure concerns on how many times over the total a specific failure mode is not detected. If this rate remains high it is advisable not to include the component in the condition monitoring system as downtime costs will be also high.

Building a cost measure upon aforementioned metrics would help to fusion both, long term and short term approaches.

2. Maintainable items

The final goal of present chapter is to describe the condition monitoring paradigm for maintenance management in industrial facilities and resume the latest contributions in the field regarding wind farms manangement. For setting up the condition monitoring system, i.e., decide which components will be monitored and which models will be deployed, two sources of information are needed:

- Which are the most critical maintainable items located in the wind turbine and which are their failure modes.
- What is the acquisition system and strategy that will be deployed.

Based on available literature, it is possible to make a first approach to wind turbine critical subassemblies and related failure modes. In a real context, the feedback from industry is crucial to set the requirements of the system depending on client needs.

In the last 20 years, some research groups have been performing qualitative analysis on the wind turbine failures to optimize system performability. Most authors have used FMECA (Failure Modes, Effects and Critically Analysis) technique to address these topics. First of all, decomposition of the entire wind turbine in sublevels is developed.

(Arabian-Hoseynabadi, Oraee, and Tavner 2010) defined 4 levels hierarchy:

- Level 1 for the entire Wind Turbine
- Level 2 for the assemblies (like the generator system).
- Level 3 for the sub-assemblies (like the stator in the generator).
- Level 4 for the parts (like stator coils).

Figure 4: Offshore wind farm (Oostende-Belgium)

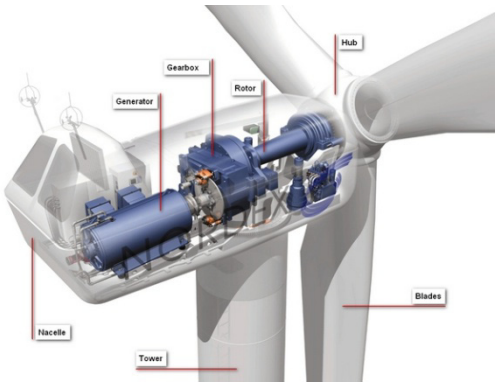

Figure 5: Wind Turbine main assemblies

The next step is the definition of failure modes in all those parts. It's important to associate failure modes to components, but there is important the analysis of causes-consequences. A first cause can be a failure in a part, but this failure could be the cause of a failure in other part. For example, a failure in the pitch control can impact rotor, generator, gearbox, structure, etc. This phenomenon is called chained failure. For this reason, it is importantk that all failure modes are well associated to the real components affected.

The expert criterion over the FMECA is very important: the experience and knowledge of the system is vital for good FMECA analysis. Some of the parameters on the FMECA can be obtained with objective criteria, but majority of the FMECA in bibliography use expert criteria to obtain the most critical assemblies.

Using this methodology, (Arabian-Hoseynabadi et al. 2010) obtained that the most critical assemblies are rotor and blades assembly, generator, electrical controls, hydraulics and gearbox. The top root causes were, by occurrence number, corrosion, mechanical overload, vibration fatigue, presence of debris and overheating. The top failure modes were material failure, fracture, rupture, electrical failure and blockage.

(Das et al. 2011) defined a limit of wind turbine system components to be redesigned on 2MW variable speed wind turbines with doubly fed induction generator configuration and with active pitch control. The conclusion was that crowbar protection and gearbox must be redesigned to those wind turbines. Other critical system components were low speed shaft, pitch controller and current controller.

(Dinmohammadi and Shafiee 2013) applied Fuzzy FMECA to 5 MW Wind Turbine of REpower, model MM92. Using some configurations of Risk Priority Number calculus on the FMECA, the conclusions are that the most critical assembly is the Tower. The following most critical assemblies are rotor blades, gearbox, power converter, transformer and generator.

The most common variation over this procedure is the objective criterion of decision: changing the risk to the cost, the most important assemblies change.

For (Kahrobaee and Asgarpoor 2011), the most critical assemblies using cost criteria are generator, electrical system, blades, converter and hydraulic system. This confronts their risk priority analysis, in which the most critical assemblies are generator, control system, mechanical brake, electrical system and converter.

(Shafiee and Dinmohammadi 2014) used cost priority analysis versus risk priority analysis, and onshore versus offshore. The results of the highest ranks on all the configurations are summarized in Table I.

Table 1: Shafiee and Dinmohammadi Failure ranking

Subassembly	Onshore RPN Rank	Offshore RPN Rank	Onshore CPN Rank	Offshore CPN Rank
Tower	1	1	1	1
Gearbox	2	2	2	3
Rotor blades	3	2	3	2
Generator	4	5	6	6
Power converter	5	4	11	4
Transformer	6	6	4	5
Main frame	10	10	5	7

Attending at these main assemblies, in the literature are covered the most relevant failure modes, which are resumed below:

- Tower
  - Crack
  - Deformation
- Gearbox
  - Gear tooth crack
  - Lubrication oil loss
  - Lubrication oil degradation
  - Gearbox bearing failure
    - Inner race defects
    - Outer race defects
    - Cage defects
    - Roller element defects
- Blades
  - Skin-skin delamination (within the plies of the laminate)
  - Skin-core delamination (between the fibre glass skin and the core)
  - Crack
  - Fatigue
  - Pitch angle implausibility
  - Pitch angle assymetry
  - Rotor overspeed
- Generator
  - Rotor coil shorting
  - Stator coil shorting
  - Brushes wear
  - Generator speed discrepancy
  - Drive and non-drive end bearings
- Power Converter
  - Frequency implausibility
  - Crowbar dysfunction
- Transformer
  - Output voltage implausibility
- Main frame

3. Condition monitoring

3.1. Introduction

Condition monitoring (CM) has four main tasks comprising pretreatment, feature extraction, detection and hypothesis discrimination. The common workflow of a condition monitoring system (CMS) is to first start with some initial observations captured from the technical system (wind turbines) that are fed and processed by the CMS until a final diagnosis (current health state) or prognosis (future health state) are obtained, as shown in Figure 6. **Initial observations** have different nature ranging from numerical data (i.e. measurement of vibrations, temperature, torques, etc.) to nominal data which usually involves status-of-machine data (working, ready, stopped, paused, etc.), triggered alarms or even operational data describing maintenance actions and replacements carried over the system. Last observations are related to supervisory and control systems such as SCADA. As a general rule, numerical data is provided as time series, however its use may not always be linked to time series analysis or prediction, depending on the selected approach for data processing. Initial observation gathering requires acquisition systems, and its selection is not trivial as long as it will condition subassemblies failure modes that can be addressed by the CMS. Different alternatives of acquisition mechanisms will be dealt in subsequent sections.

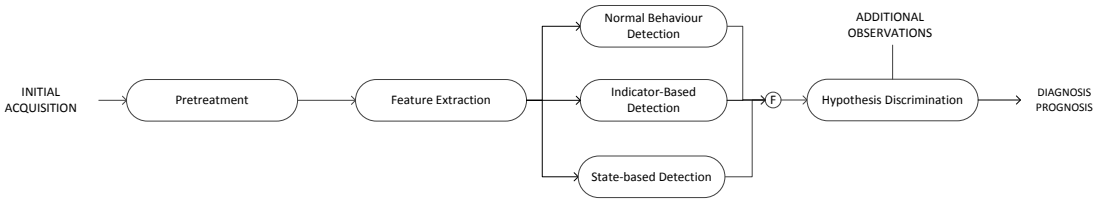

Figure 6: Main stages of CMS

Collected data is subjected to **pretreatment** in order to validate and adequate information for subsequent tasks. Data pretreatment have different goals with dependency on if CMS configuration (training) is being performed or if by contrast, trained CMS is actually running. After this stage, procedures for data transformation are performed. **Feature extraction** aims to enhance input observations so as to facilitate and improve posterior steps in CM through the synthesis of indicators. Once data has been appropriately treated, **detection** task provides hypotheses about possible malfunctions taking place within the machine which may result in a future component failure. Put another way, detection gives unconfirmed inferences about system dysfunctions. Detection task may be addressed using several methods and/or techniques simultaneously that will result in multiple inferences about one specific potential failure. Consequently a **fusion** task can be added to the basic four stage CMS in order to reduce the hypotheses space before further processing. As hypotheses made in detection task could have attached a confidence measure, fusion task also contributes to modify the confidence on each of them. Finally, previous generated inferences are corroborated or disposed in a procedure including reasoning. This is the so called **hypothesis discrimination** task. Additional observations may be requested in this phase specially if higher certainty is needed to validate a specific declared state. Each confirmed state of the machine is considered a diagnostic. By its part, prognosis is made over a health indicator to predict the future state of the asset under study. A common parameter linked to prognosis is the remaining useful life (RUL) which is actually referred to the assessment of the next *time to failure* (TTF). Because a failure does not actually mean the end of useful life for an asset if it is repairable, in this text the latter term is adopted from now on. Existing techniques for TTF calculation often relies on diagnostics to properly modify predictions of the health of assets.

Although previous characteristics of CMS are common regardless the system being considered, there are some particular issues that have to be taken into account when dealing with wind turbines. They are addressed in the following sections.

3.2. Data acquisition

As it has been mentioned above, wind farms are usually located in difficult access areas. Particularly in the framework of Leanwind, in which offshore wind farms are on the focus, there are special difficulties that have to be overcome by deployed acquisition systems. In general, as reflected in the literature, there are two main systems: the here called *standalone* systems and Supervision Control and Data Acquisition (SCADA) systems.

The standalone approach involves the use of ad-hoc systems for sensing expressive variables related to faults. Traditionally, standalone solutions for wind turbine condition monitoring (WTCM) have focused in drive train problems through vibration monitoring at the most extent. However, because of their conception they are not restricted to this particular monitoring technique neither to the drive train subassembly. Deployment of standalone technology has been previously based on mobile devices carried to a desired site to take measurements and make analysis. Given the particularized conditions in wind technology, this configuration is far from being optimal because the so commented access difficulties and also because that will limit the benefits of using condition monitoring technology in the maintenance strategy. For offshore wind turbines there are significant additional costs for vessels and crew, the transportation will take longer and access can be impossible in harsh weather conditions. Access from a boat to the turbine is also potentially dangerous. Then, in the context of WTCM the need of remote sensing equipment is particularly demanding.

Standalone technology is constantly evolving accordingly to the development of new and better sensors, permitting to extend the number of subassemblies embraceable. At present, support exists for blades, main bearing, gearbox internals, generator bearings (Drive end and Non-Drive end) and generator internals. An exhaustive revision on standalone systems for WTCM is developed in (**Crabtree et al. 2014**).

Within the context of wind turbine condition monitoring special attention have been devoted to SCADA systems. They offer a valuable opportunity to save costs of inversion in additional sensors to monitor wind turbine health. Parameters recorded by the SCADA system are broadly categorized (**Kusiak and Verma 2010**) as controllable parameters, environment parameters, performance parameters and status parameters. Controllable parameters are those that can be tuned to adjust instantaneous operation of wind turbine (e.g. pitch angle, yaw angle, generator torque). Environment parameters comprise all those variables that are uncontrollable such as wind velocity, external temperature or air density, among others. Performance parameters comprise active/reactive power production, generator speed or voltage phase. In global, usually more than fifty parameters are recorded by SCADA systems. Data coming from these systems are generally low frequency data provided every ten minutes. Each datum is in fact an average of every sample at the working sampling rate.

By analyzing each gathered signal it is possible to indentify relations between them and discover failure patterns. Because of the nature of this kind of data there are some drawbacks and advantages on its use. On the one hand it is clear that some failure modes, that are commonly detected using techniques that rely in high frequency data, cannot be addressed with the data provided by these systems. This is mainly because of the working characteristics of SCADA and also because they have not installed such specific sensors within their structure. In the pros side, SCADA systems are able to deal with some failure modes that cannot be addressed by conventional standalone systems (**Yang et al. 2013**), even when latter systems are highly customable, and they do that in a more economical way. However, given the properties of SCADA data, which will be treated in following sections, advanced processing techniques have to be implemented

in order to extract expressive information about faults. A survey on currently available commercial tools for SCADA data handling is covered in (Chen et al. 2014).

A lot of work have been made in the field of SCADA data processing as will be detailed in posterior sections, but there are still challenges to be faced as outlined in (Tchakoua et al. 2014). These are:

- Determine the most cost-effective measurement or monitoring strategy.
- Improve the use of SCADA system data to provide a more reliable, flexible, and efficient tool for automatic WT monitoring and control.
- Requirements for remote and e-monitoring.
- Protocols for integration of several data sources

3.3. Pretreatment

In present section we are interested in the analysis and description of data preparation methodologies retrieved from SCADA systems. However, some of the problems suffered by these data are also shared with data coming from other acquisition systems as they are caused by communication failures or also by the sensors themselves.

Data preparation procedures vary depending on whether CMS is being trained or if it has been already deployed. Two additional subtasks need to be considered when dealing with the first case, consisting in a process of understanding the data and properly select it for models training. This phase is not trivial because SCADA recordings are affected by the operational condition of the wind turbines. For example, when a data driven model is trained for describing the power curve, one could observe that power reading is 0 MW while the wind speed is 15 m/s and hastily conclude that a fault is taking place, when actually the cause was that the wind turbine was stopped due to demand management. Different data sources recorded by SCADA systems have to be crossed in order to obtain deeper knowledge about their meaning to afterwards obtain accurate models. Frequently, failure events are not explicitly identified within recorded data and efforts should be devoted to extract information about each sample of the dataset. A referent work about this topic was conducted by (Kusiak and Verma 2010) using pattern machine learning. How data is selected for CMS training is also important as it will condition how the trained models will perform once implemented. Class unbalance is a common problem that must be tackled when setting the CMS. Known effects of this issue are that models are biased to accurate characterise properties largely present in the dataset in detriment of the rest. Some solutions to overcome these problems are to oversample the properties less present in the dataset, use some kind of cost-sensitive training method or appropriately select some key properties that represent the whole set discarding other samples that could be seen as redundant. (Kusiak and Verma 2010) proposed Tomek links to implement the latest approach. SCADA data also reflect problems due to communication interruption and perturbations in form of missing and implausible (e.g. nacelle temperature readings showing 500 ºC) values, respectively.

This is especially critical when data need to be considered as a time series, because some previous entries must be available to make inferences at current time or in the future. Some methods addressing these issues are described below:

- *Method of discarding:* this approach takes out of the data set each missing or implausible value. It clearly limits the later use of time series analysis as temporal discontinuities are incorporated.
- *Method of labelling:* wrong data found is labelled as incorrect and kept until handled in posterior stages. As previous method, when labelling time series analysis is not allowed either, but there is not loss of information.
- *Method of averaging:* every wrong value in the data set is substituted by the mean of the data set. In a non stationary data series a moving average could be a better estimator. Major inconvenient with this method is that if there is a large lack of data within a period, tendency of the true data series will be lost

- *Method of interpolation:* this could be seen as a refinement of the averaging method where each wrong value modified by an interpolation technique. Interpolation is only suited for CMS training as future data is not available at the time.
- *Method of extrapolation:* each unseen datum is obtained with a model which represents the historic extrapolating its results. Extrapolation is suitable for CMS when it is deployed.

Figure 7 reflects the nature of typical real world SCADA dataset in which CEANI R&D group have had experienced. Missing values coming from the acquisition system are usually identified by "NaN" (Not a Number). Potentially implausible values are found by the statistical Box and Whiskers technique and labelled as outliers. Each datum that is nor missing neither outlier has been considered "Normal" (i.e. plausible). A pie chart is used to show the segregation of each data type. Around 24% is missing data, whereas only 3% are potential outliers. Bars chart offers an insight of the missing values giving how many times certain number of variables are simultaneously not available. The most repeated situation is having 34 out of 37 recorded variables simultaneously unavailable. For further details a histogram was constructed in order to show duration distribution of missing value periods. What is evidenced in the graphic is that periods are biased to short values, meaning that in the vast majority of cases, filling methods can be applied without introducing high levels of uncertainty.

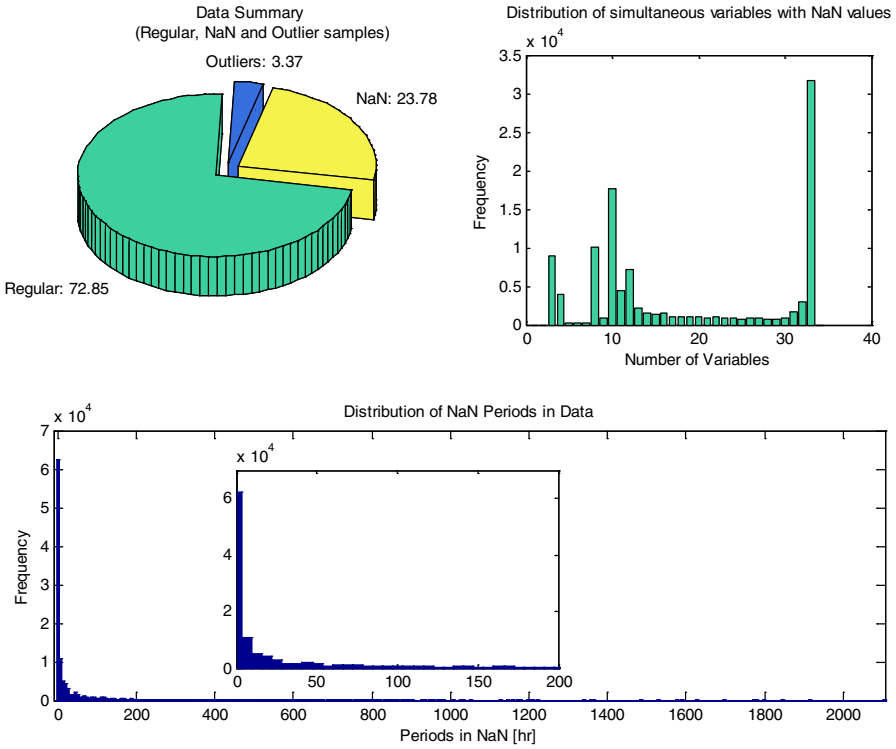

Figure 7: Raw SCADA data quality properties

Other problem concerning inputs is data consistency, which is not always as evident as having a completely out of domain value. Some techniques to face this issue have been extensively treated in (Piero Baraldi et al. 2010; P Baraldi et al. 2010; P Baraldi et al. 2011) using ensemble methods. Preventing faulty sensor data to enter in the detection module it is of crucial importance to reduce false alarms rate and false negative rate.

Every time a value is replaced by any of the methods described above uncertainty is added. This uncertainty has to be appropriately propagated to subsequent stages and taken into account when submitting a final diagnostic through confidence measures.

3.4. Feature extraction

When detection algorithms are applied, if inputs are in a favourable shape the process becomes easier and the performance is substantially improved. Task of adapting data for further processing is called feature extraction. Then, the main objective lays in the generation of expressive indicators for failure detection. The methods for feature extraction vary widely with regard to the particular method used for failure detection and also depending on the data type under consideration (low/high frequency data, …). Now there are exposed some methods and techniques largely applied for wind turbines in the literature.

Input signals can be used as they come and optionally (sometimes highly recommendable) performing some filtering to diminish random noise attached to it. This is the so called **identity and filtering** method. *Identity* is referred to the fact that the signal space is not transformed. These methods are used principally when dealing with high frequency data as vibration and acoustic signals which usually suffer of low signal-to-noise ratio (SNR). Some techniques used in the literature are Kalman filters when the linear and white gaussian noise hypothesis applies; wavelet based filtering is used in (**Yang et al. 2008**) for denoising and smoothing vibration signals for gearbox condition monitoring. The same model is applied in (**Qiu et al. 2006**) for bearing failure detection. Other extended technique for signal denoising is the time synchronous averaging (TSA). It removes noises that are not synchronous with the rotating frequency of a specific component, allowing to isolate its own excitations while diminishing other components contributions. This approach is covered in (**Zhu et al. 2014**) for gearboxes and applied in (**Bechhoefer et al. 2013**) for stationarity preservation through signal resampling. Isolating signals comming from desired component is important because other component exitations always act as random noise which masks important features for diagnostics. An alternative to TSA is proposed in (**Wang et al. 2014**) using independent component analysis (ICA). For moderated frequency data, like when performing temperature analysis, a more rough but still effective mean filtering can be used to smooth raw signals. These are commonly implemented through moving averages.

The vast extent of monitored variables with SCADA systems usually causes the known *dimensionality course*, which negatively affects performance of detection techniques. Hence, **dimensionality reduction** methods are an option to select most relevant inputs expressive of a specific failure mode. Principal component analysis (PCA) is a popular technique to suppress redundant variables because of their correlation. Data is reduced by finding principal directions axis and transforming input data into the PCA space and selecting as many variables as needed to cover a target extent of data variance. This approach has been used in (**Godwin and Matthews 2013**) when studying blade pitch faults. Another alternative is to use the information gain criterion to rank inputs according their statistical dependence with the target variable. Works conducted by professor Kusiak in the field of wind turbine condition monitoring cover several alternative techniques belonging to the data mining discipline. Among his contributions Kusiak applies some not deterministic search algorithms, as genetic search (GS) to explore variable set space, but also uses some embedded algorithms such as decision trees, instance training based methods such as RELIEF and many other techniques. Other possibilities rely on wrapper methods which during selection consider the ulterior applied technique for fault detection. A way to accomplish this is to perform a sensitivity analysis of input variables on output variables. To get a deeper insight of these techniques it is recommended to explore (**Kusiak and Verma 2010**; **Kusiak and Verma 2012**; **Verma and Kusiak 2012**; **Yan et al. 2014**) references.

High frequency data point values are meaningless by themselves for diagnosis and prognosis purposes. They only represent an instantaneous oscillation state. Usually, instances recorded in vibration analysis are in the form of one-second samples (at very high sampling rate) taken in ten minutes intervals. Each sample contains information about health status of the component being monitored. **Statistical moments**

are common extracted features from these data for compression and description purposes. Kurtosis and skewness are two popular metrics to condense information contained in oscillating data. The first measures how peaked or flat a distribution, indicating errors when vibration levels increase due to failures. The second one measures the symmetry of a probability density function of a time series. Kurtosis is sometimes obtained in the frequency domain and then is called *spectral kurtosis*. A survey on available descriptors for vibration analysis is presented in (**Zhu et al. 2014**).

Spectral analysis methods are widely extended for condition monitoring as they can also deal with either high and low frequency data. The most representative techniques within this group of methods are Fourier and Wavelet transforms. Usually they work as data preparation techniques to afterwards apply other methods. In (**Zhu et al. 2014**) there are some descriptors obtained once data is transformed in the frequency domain. By its part, in (**Manohar and Lanza di Scalea 2013**) Fourier transformed data is used to reconstruct thermographic images based on phase information.

Tendencies methods provide information about the rate of change of a certain measure which may be expressive of incipient faults within technical systems. **Delta methods** are employed to generate indicators that reflect intrinsic or extrinsic relationships in analyzed signals. Some examples of intrinsic relationships are peak-to-peak measure, root mean square, delta root mean square or crest factor. To account for extrinsic relationships, discrepancy between two temperatures is a possibility. This approach was found useful when applied on NDE and DE generator bearings to detect faults in the generator.

As a matter of fact, every exposed method for indicator generation can be used solely or combined. Last approach is called **composite methods** and is often put in practice regarding the requirements of each particular application. For example, statistical moments methods are in general applied after filtering methods.

3.5. Evaluation: diagnosis and prognosis

3.5.1. Detection task and techniques

Detection techniques rely on monitoring of interest variables or signals related to wind turbine performance. As experience is gained in the operation of wind farms more failure modes are discovered and become a matter of concern to maintenance managers. Consequently new challenges arise when trying to detect some kind of faults. With the development of sensor technology more specialized monitoring techniques have emerged to support diagnosis and prognosis. Regarding those related to wind turbine technology literature, the most popular approaches for monitoring are:

- Vibration analysis (VA).
- Oil analysis (OA).
- Acoustic analysis (AA).
- Thermographic analysis (TGA).
- Structural analysis (SA).
- Parameter performance analysis (PPA).

According to hierarchy proposed by (**Tchakoua et al. 2014**) all techniques but the latest, belong to intrusive approach, being the last one classified as non-intrusive. The intrusive/non-intrusive nomenclature lies in whether additional sensor infrastructure has to be added to the wind turbine. In other words, parameter performance analysis is linked to SCADA monitored variables while, in general, remaining approaches are related to standalone systems. Table 2 contains techniques covered in the current literature and its applications.

Table 2: Monitoring techniques for fault detection

Technique	Variables monitored	Subassembly	Failure modes
Vibration Analysis	Acceleration	Main bearing Gearbox bearings DE bearing NDE bearing Gearbox internals	Inner race defect Outer race defect Cage defect Roller elements defect Tooth crack
Oil Analysis	Particles (Fe, Pb,…) Viscosity Dielectric constant	Lubricantion systems	Oil degradation Gearbox gears wear
Acoustic Analysis	Acoustic signals	Blades Gearbox internals Generator Refrigeration system Lubrication system	Parts wear Leakage
Structural analysis	Strain Stiffness High resolution images	Tower Blades Main Frame	Cracks Incipient deformation Creep defects
Termographic Analysis	Thermal images	Blades Generator Gearbox	Blade delamination Coils wear Loss of lube
Parameter performance analysis	SCADA performance parameters (Temperature, torque, rotational speeds, voltage, power,…)	All wind turbine main assemblies (blades, main bearing, gearbox, generator, power converter,…)	Bearing defects, Oil degradation, Generator coil defects Blade pitch defects Gearbox defects due to overheating

The main objective of detection task is to generate a set of fault hypothesis that will be validated or discarded afterwards. This particular task has received great attention by many researchers in the literature. In fact, even when speaking of diagnostic many papers actually perform detection. In this document three methods have been identified that enable detection task, these are: normal behaviour modelling methods, indicator-based methods and state classification methods. Differences among them are clear from the preparation techniques point of view and also due to the information that is possible to extract.

Normal behaviour modelling (NBM) have large acceptation within condition monitoring practitioners community mainly because its closeness to human perception of anomalies. Broadly speaking this approach is founded in the idea that anything differing from what is expected is an anomaly. Therefore, outlier analysis techniques are suitable to generate fault hypothesis. Some precautions have to be considered when using this approach, for example definition of normal models requires locating failure free and stable regions within training dataset. This is not an easy task because wind turbines operates in a large extent in transitory state due to changing environmental conditions electric power demand. This means that just selecting a certain interval for normal model construction is not enough, even when it is failure free. As have been said normality means that no failure is taking place but does not implies that normality is a homogeneous class or set. A typical problem present when working with normal approach is that if normal class domain is not enough covered by selected instances to train the models, then false alarms will become an issue when the detections system is deployed. Hence, it is recommendable to use segregation techniques prior training normal models. Afterwards, one can construct several models specialized in one or few similar working conditions or sampling each isolated region and create a single general model. Clustering techniques are suited to achieve data segmentation. Among them, K-means heuristic is one of the most simple and popular algorithms. Self organizing maps (SOM) have been applied in some works as in (Garcia et al. 2006). Particularly last one have demonstrated its ability to handle highly non linear boundary surfaces nor circular class shape neither convex. There are many alternatives for clustering techniques in the literature whose revision is out of scope of this document.

Another issue has to be accounted for normal behaviour modelling such as subassemblies failure dependencies. Large residuals from normal models often means that might be an abnormality, but this is not certainly true. For example, if only one model has been constructed to monitor performance of generator DE bearing, when a significant residual arises then there will be three possible interpretations: the problem is related to one of the input variables, the problem resides in some output variable, or relation between input and output variables that are not consistent. Under this framework, additional models will be needed to certificate that the large residual is caused by the presence of some kind of defect in the bearing and not because a failure occurred in other part of the wind turbine and it has been propagated until the bearing.

Many authors have addressed normal behaviour modelling, and techniques proposed vary considerably depending on each particular application. In (Yang et al. 2013) normal behaviour is represented by means of a least square procedure to fit a quartic polynomial function in single input-single output (SISO) relation. As data coming from SCADA is noisy, a statistical smoothing procedure is used to enhance the relation of modelled variables. The technique is run in a batch fashion, so every time a certain amount of data is received a polynomial fit is made. A failure criterion is set representing the mean absolute error between predictions made by normal model and those made by the latter. The magnitude of the criterion is related to the development of the failure (i.e. incipient, degraded or critical). This procedure is applied to a generator bearing and pitch blade. Power curve analysis is a subset of performance parameter monitoring techniques in which fault hypotheses are made when discrepancies between predicted and actual power appears at a given wind velocity value. Power curve is modelled using a probability model based on copulas in (Gill et al. 2012). Power and wind speed are transformed into the copula space {[0 1]x[0 1]} and compared with the reference line $u = v$, where u and v are the power and wind speed in the copula space. Some distance metrics are presented for fault hypothesis submission regarding blade (pitch angle) and orientation system (yaw angle). Other technique for power curve analysis was proposed by (Osadciw et al. 2010) constructing a model based on gaussian distribution functions (single or mixture models). First, wind speed and output power are normalized in the range 0-1 and then gaussian

parameters are optimized through particle swarm optimization (PSO). The linearized version of predicted power and wind speed are used to get insights about wind turbine performance.

Another subset of PPA is the temperature analysis. When faults are evident through temperature analysis, degradation often is in a advanced state. Nevertheless, many times, prediction horizon is enough to schedule a maintenance task. As there are several temperature signals measured by the SCADA system, this approach has gained some popularity. Furthermore, temperature trends are linked to a wide variety of failure events such as loss of coolant, lubricant degradation, bearings wear or generator coils deterioration, among others. In (**Yan et al. 2014**) normal behaviour bearing temperature model is constructed in through artificial neural networks (ANN). Analysis of residuals is done to generate hypothesis of failure in the bearing using root mean square error metric (RMSE). When this error exceeds a preset threshold a warning of failure is triggered. In (**Kusiak and Verma 2012**) a similar procedure is put in practice, with the major difference that model inputs are systematically selected using three data mining algorithms comprising a wrapper method based on genetic search, best first search algorithm and boosting trees. One drawback of this study is that detection horizons are considerably short (less than an hour) giving a poor margin to carry a maintenance before the failure occurs. Having this narrow operation windows, chances for an effective actuation requires, for example, to embed the CMS within the control system as proposed in (**Frost et al. 2013**) in other context. One interesting approach was proposed by (**Guo et al. 2012**) using nonlinear state estimate technique (NSET), which actually is a similarity approach where normality is characterized by historical observation matrix. Each new observation is estimated regarding its closeness to normal prototypes by a weighted sum of every historical instance. Then, residual analysis is made for failure detection. What is really attractive from this technique is that online training implementation is quite simple. Once performance of the technique (false alarm rate, false negative rate, etc.) decreases one can update historical observation matrix adding new instances describing normal behaviour, letting the method be scalable.

A pioneering work in the field of normal behaviour modelling was proposed by (**Garcia et al. 2006**). They created an integrated framework (SIMAP) for diagnosis and predictive maintenance scheduling through SCADA data analysis. The proposal may be enclosed in the PPA approach which was accomplished by using neural networks as detection technique and treating residuals to infer faults. Later in the process, a fuzzy expert system deals with generated hypothesis to submit a diagnostic. Once a diagnostic has been confirmed SIMAP enables the estimation of remaining time to failure with a similarity comparison between current residual evolution with historical evolution of residuals for a given failure mode.

A more recent work conducted by (**Schlechtingen et al. 2013**) keeps the same philosophy of SIMAP. The main novelty within their work is the use of artificial neural fuzzy inference system (ANFIS) to construct normal behaviour models, and the use of a systematic variable selection procedure through a combination of expert knowledge and genetic search. Any diagnostic system is a decision support tool for maintenance operators and is beneficial to be transparent (i.e. interpretable) for humans. ANFIS is an approach that, as authors defend, allows backtracking of their inferences procedures suited for human understanding. Beyond the use of either ANFIS or ANN, what it is really interesting of both, García *et al.* and Schechtingen *et al.* proposals is that they have created a holistic framework in which any kind of model, either normal behaviour ones or the later covered in this section may be combined altogether.

Instead of trying to model normal behaviour and analyze predictions deviations, an alternative would be to represent a specific failure mode evolution. This is the **indicator approach** method for failure detection. Actually, health indicators are the matter of interest within this approach. These indicators must characterize accumulated damage and then purely grow until the system failure or until the system

undergoes an eventual repair action. In some cases, since there are some failure modes that occur abruptly without exhibiting any degradation, health indicators show a stepped profile. In general, although they can be used solely starting from processed initial observations, it is also possible to combine this method with normal behaviour modelling. Since residuals obtained when performing NBM are noisy, some techniques such as moving window averaging are used to smooth and reveal failure mode increasing tendency over time, becoming a health indicator. In Figure 8: Case uses of health indicator approach are schematized the two ways of implementing indicator method. Additionally, it has to be noticed that when using health indicators it is desirable that constructed metric shows specificity towards as few failure modes as possible. This means that its response should be as insensitive as it can be when any other failure mode occurs within the system.

Health indicators have found applications for bearing failures particularly when using vibration analysis techniques. As has been commented previously in the document, bearings suffer four main kind of failures: inner race, outer race, roller elements and cage faults. Four indicators, one for each failure mode, has been proposed in (**Zhu et al. 2014**) which account for inner race, outer race, ball and cage energy. These indicators represent the energy of the bearing vibration signal around each subcomponent characteristic failure frequency. As failure mode develops indicators start to rise until failure finally occurs. Another application of indicator based detection was used when analyzing coil shorting and unbalanced rotor failure modes (**Yang et al. 2008**). Indicator was delta type and was obtained as the ratio of torque and angular speed of the generator.

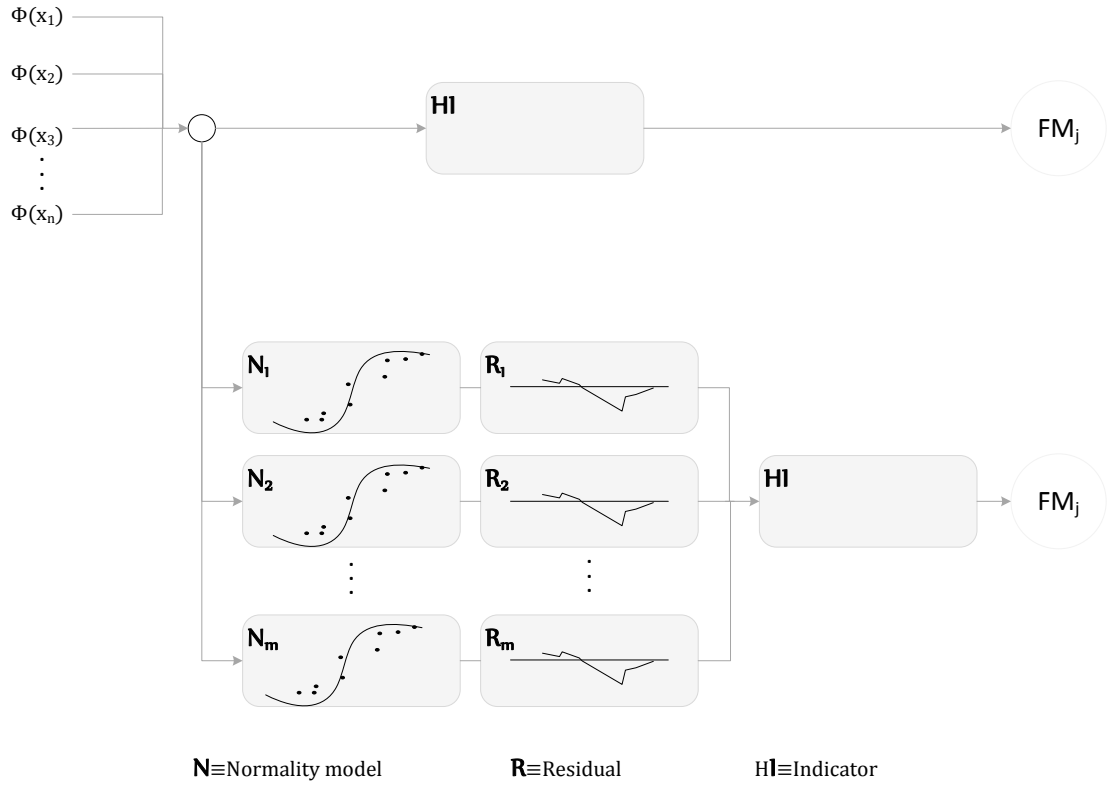

Figure 8: Case uses of health indicator approach

The third approach for failure detection is the **state-based method**. This approach differs from the others in the fact that detection task is based on the functioning state of the system. Some groups or classes are previously isolated, which represent the health of the system ranging from normal to total malfunction. The information contained in monitored signals is somehow abstracted and converted into a discrete state space. Two subtasks can be identified in the state-based methods: the first one is

the *membership qualification subtask* which faces the problem of assigning each entry (record) to an specific class or state. The second subtask performs selection on the actual class or state of each new entry, then submitting a fault hypothesis. First subtask may be accomplished using model-based classification method, in which neural networks, symbolic regression, decision trees or similarity-based modelling are some of the available techniques. Additionally, clustering methods are also suitable where k-means, Gustaffson-Kessel algorithm or self organizing maps are some representative techniques. Several of the proposed techniques for the first subtask have not crisp outputs, which justifies the second subtask. For final selection of current actual state, historical patterns (i.e. state sequences) can be used by means of Markov chains or more general bayesian networks. Nevertheless, there are other more simple criteria such as the selection of the best candidate for hypothesis making. Best candidate will be the state with the highest membership score among all possible classes. Failures in both blades and generator brushes were treated in (**Kusiak and Verma 2010**; **Verma and Kusiak 2012**) respectively, under the state-based detection perspective. The proposed procedure in the two works makes use of supervised learning once data is appropriately labelled as faulty or healthy. Data is identified as faulty accordingly a criteria founded in the use of a preset sized moving window. If a faulty status is found within the window all data is labelled as fault, otherwise is labelled as healthy. For the blade case study, they analyzed *pitch angle implausibility* and *pitch angle asymmetry* failure modes using genetic programming to construct a decision tree. The "one-class" classification approach is used in the analysis. Each constructed model is only capable to distinguish one failure mode at a time. This procedure is prone to suffer specificity issues as one particular entry could be recognized by both models as a faulty state. Disambiguation has to be made in following steps. By its part, generator brushes are subject to continuous friction, eventually exhibiting wear failure modes. To detect wear problems in brushes, (**Verma and Kusiak 2012**) make a comparative study on different classification techniques such as k-nearest neighbours (K-NN), ANN, support vector machines (SVM) and boosting trees.

As it has been mentioned in the text, human-intended interpretability of inference steps is a matter of great concern for maintenance operators. In this sense, (**Godwin and Matthews 2013**) proposed decision trees to detect blades failure modes addressed by Kusiak, arguing that generated rules have improved interpretability when compared with other classification techniques. In this work, three states are taken into account instead the two identified in (**Kusiak and Verma 2010**), incorporating intermediate degree of damage. Finally an expert system is considered so as to submit the detected state to an operator, which may be replaced by an automatic hypothesis discrimination module in other situations.

In (**Manohar and Lanza di Scalea 2013**) Lock-In thermography exploration of wind turbine blades was performed to detect skin-skin and skin-core delamination failure modes. A periodic (pure frequency) thermal wave is applied into the surface and when the heat wave encounters a defect, part of it returns reflected with a phase shift respect to the input wave. Reflected image is processed by Fourier transform to create a "phase" image. To avoid the effect of blind frequencies, all those tested were aggregated to construct a descriptor. Mahalanobis distance is used to reconstruct input images based on the distance of the current descriptor to the mean value of the descriptor when the blade is healthy. Finally, the resulting image contrast is enhanced using a threshold method which is set using receiver operating characteristic (ROC) curves.

In **¡Error! No se encuentra el origen de la referencia.** it is contemplated the fusion phase coming after the detection task. As has been described previously, all available methods for detection may be used simultaneously, but also even when using a single approach several techniques could be deployed, then having multiple contributions for one single studied characteristic. Maybe, several models for power curve modelling have been deployed, so there are chances of having different conclusions from them. Their conclusions may be sent to hypothesis discrimination task separately or they may

be combined in order to obtain one single conclusion. There are studies supporting the theory that it is better to have many weak models finally aggregating their responses than having only one strong learner: this is the *ensemble theory*. As it has been mentioned, various models can be created, each of them specialized in bounded regions of the whole domain. Then, when a new data coming from the acquisition system undergoes detection task multiple hypothesis will be made about detection. This paradigm is particularly beneficial when modelling highly nonlinear systems. Nonlinearities are difficult to track by humans and too complex models would lack of interpretability. Instead of using one holistic model, partitioning the problem into several easier ones will provide more clear outputs jointly potentially more accurate outcomes.

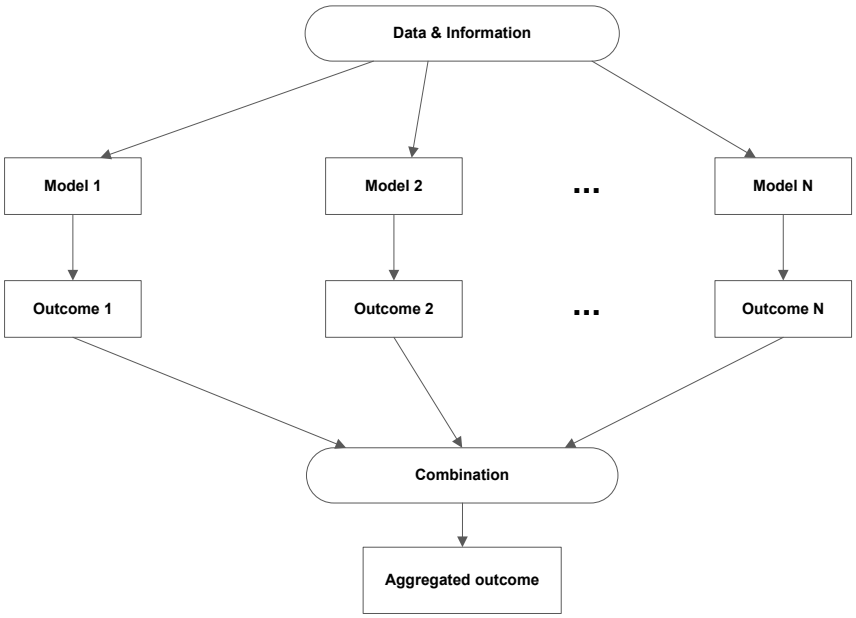

Figure 9: Ensemble approach

Moreover, ensemble methods offer a valuable opportunity to update models when operational conditions change and false alarm or missing alarm rates start to increase. Under the ensemble method approach, instead of retrain an existing model using current and past data to amend its performance, new data could be used to create another model while preserving the previous one. Notice that the latest impose much less data storing requirements and also speed up model updating process.

3.5.2. Hypothesis making and discrimination

This task is intended to evaluate current state of the wind turbine (or any specific system) making use of the symptomatic and/or state information identified in the previous task. This first conclusion is called a diagnostic, which is one of the outputs of the condition monitoring system. Once a diagnostic has been proposed, we can project into the future health status of the machine to obtain information about the remaining time to failure in order to schedule maintenance actions. Then, we speak of prognosis which stands for the second output of a CMS. Both diagnosis and prognosis should provide an uncertainty measure of their inferences in order to qualify them.

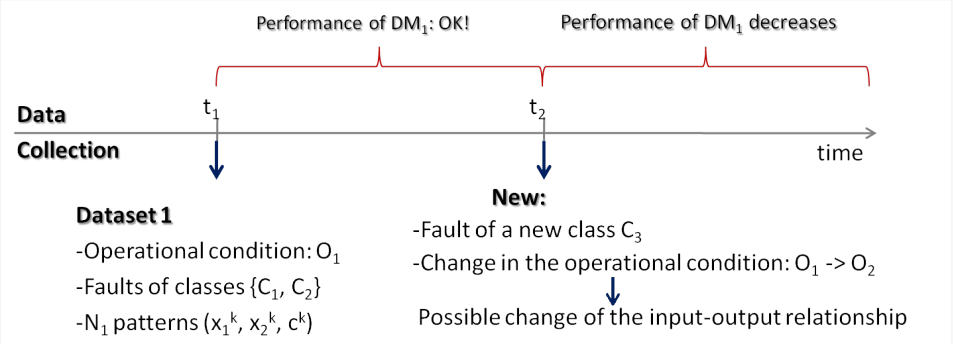

Figure 10: Ensemble incremental learning

There exist different techniques to submit a diagnostic, several of them already exposed in this text. Expert Systems (also known as knowledge systems (KS)) are an option whose attraction lies in its closeness of human thinking. In fact, its structure is a direct translation of experts reasoning procedures. Expert systems exhibit the property of being scalable when more experience is acquired of wind turbines failure modes. One example of KS operation is exposed below.

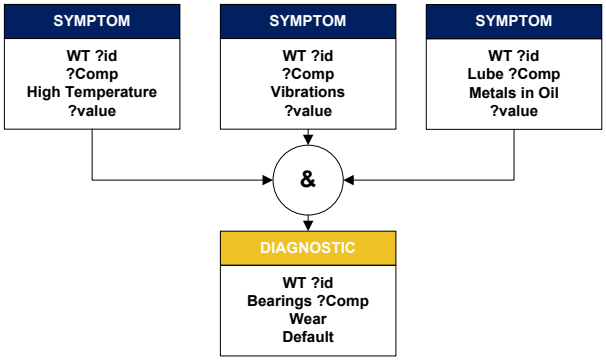

Figure 11: Example of fuzzy rule for failure diagnostic extracted from a set of rules

Fuzzy versions of KS have gained acceptation from diagnosis practitioners as (**Garcia et al. 2006**; **Schlechtingen et al. 2013**). On the one hand, the fuzzy engine allows to abstract information from detection task in a reduced and more understandable manner. On the other hand, it permits the handling of uncertainties in a natural way. Probabilistic reasoning is also suitable to handle uncertainties and bayesian networks have been proved to be an effective technique for hypothesis qualification (**Plumley et al. 2012**). In bayesian networks, prior knowledge (i.e. hypothesis of that a specific symptom actually means a malfunction) is corrected by the likelihood of the symptom (i.e. probability of observing the symptom given the diagnostic), to obtain the posterior probability of a diagnostic.

Decision trees are also an appropriate tool for diagnosis, whose main advantage is that its results are somehow expressed as understandable rules.

It is possible that identified diagnosis do not match certainty standards and further validation information is needed. Also if there are several diagnostics that are equally evidenced it is necessary to make discrimination between them. For diagnostic discrimination, firstly a hypothesis should be selected. There are two general options to select a diagnostic for further evaluation: a diagnostic could be chosen randomly, disregarding the cost of hypothesis evaluation and the criticality of each failure mode. It is strongly discouraged to employ this approach unless cost and criticality of each hypothesis is similar. The other way of selection is to base the choice in a preference metric which considers the cost of evaluation, the critically and the certainty degree associated to each hypothesis.

Once a potential diagnostic has been chosen, additional data will be required to accept or discard the hypothesis. Some methods for new data gathering are described below:

- *Replacement method:* we replace the component associated to selected diagnostic to check if they are the actual source of the problem. This procedure has to be deployed carefully because it may involve high validation costs.
- *Additional indicator method*: new observations are requested through inspection (in site or remotely) or more metrics are computed to ratify current diagnostic. It is possible that some indicators are executed on demand because their computational costs or costs derived from acquiring required data to be performed.
- *Tuning method:* There are circumstances in which some operational parameters of the wind turbine could be tuned so as to observe if the response of the system match the expected behaviour under the failure hypothesis made. If this is the case, then hypothesis is confirmed; otherwise it is abandoned.

3.5.3. Prognosis

Subsequent steps, once any diagnostic is confirmed, are intended to address the prediction of the evolution of system health. This has been a matter of great concern for practitioners because this stage is the real responsible of providing proactive nature to maintenance. Without this information it is not possible to adjust the scheduling regarding system health.

There are several algorithms suitable to obtain TTF estimations, from the pure statistical based ones (weibull analysis), to those based on models. Data-driven approaches have also had a relevant role for the prediction task, learning system dynamics from raw data. Temporal series forecasting algorithms are an option when estimating TTF. Autorregresive (AR) models or more general Autorregressive Moving Average (ARMA) models work reasonably fine when dealing with statistically stationary and linear systems. Latest constrain may be removed using Autorregresive Integral Moving Averages (ARIMA), which under some special conditions may become a random walk. A stochastic process somehow related to random walks is the Wiener Process, which was proposed by (**Valis et al. 2014**) as mean to obtain TTF distribution of lube, i.e. time when the lube losses its properties. Parameters for the Wiener process were estimated through linear regression of iron (Fe) and lead (Pb) particles concentration varying with time. Linear assumptions are too tight for several dynamical processes and estimations based on them are poor. Dynamic neural network NARX (nonlinear autorregressive model with exogenous inputs) and ANFIS are used to account for nonlinear systems in (**Hussain and Gabbar 2013**) for gearbox TTF calculation based on vibration data.

Sometimes there are some variables which are taken as health indices, such as crack length for blades or gearbox gears, or contamination for lubrication system. However they are not directly measurable and need to be modelled in terms of other observations. In (**Zhu et al. 2013**) the lube degradation is addressed considering the water contamination of lube as the health index. Water contamination is measured indirectly through dielectric constant and kinematic viscosity monitoring. Then the particle filter algorithm is used to predict remaining time to failure.

Aforementioned water contamination level is considered as a continuous state-space model. There are other approaches that consider discrete state-space characteristic of the degradation paths. Markov models, particularly its hidden versions, have gained much attention to deal with these situations. Markov models restrictions are convenient for many situations and are also simpler from the mathematical/numerical point of view. Within hidden approaches the semi-markov one is focusing several of the recent work, since the exponential constraint (representative of Homogeneous Poisson Processes) is eliminated in the aftermath of a more general formulation. Hidden Semi-markov models may operate under any arbitrary transition probabilities. In (**Su and Shen 2013**) Multi-Hidden Semi-Markov Model are used to predict TTF of a vehicle engine due to cylinder wear. In this work Weibull distribution is selected as both emission and transition probabilities.

Chapter 3

Proposed Methodology Description

1. Overview

Generally, data coming from industry does not include an identifier informing about if a certain sample corresponds to a fault or a normal operation regime. In its place, what it is often available is a set containing monitored operation variables and, on the other hand, the set of registered alarms and work orders related to the monitored system.

For this reason, normal behaviour modelling has gained great popularity and it has become the main approach in the vast majority of the works gathered in the review performed in the previous section. Under this approach is reasonable and quite intuitive that, among the whole set of samples, the ones that are used to create the model are those which are located in a free of alarm region or in periods where any maintenance or replacement activity has been performed. Samples which satisfies previous conditions are said to be representative of the normal behaviour of analysed system. Besides, one of the major advantages of normality approach over failure centred approach is the number of samples often available to create the models. Is expected that the machine typically works in normal condition so dataset size of normal samples is potentially bigger than those related to a specific failure. However, within normality set may be several substates. Also the machine may be functioning normally at different operations regimes. If all these particular cases are not considered when the model is created it would potentially lead to wrongly identify a normal sample as symptomatic.

Moreover, failure detection is not the final goal of the condition monitoring task, and encountered symptoms will need to be assigned to a particular failure mode. Under the normality modelling approach it is mandatory to cross validate the generated symptoms in order to conclude to reason of the detection.

In this work the purpose lies in finding models expressive of some certain failure mode. Consequently, any time that a significative response of the model arises will mean that a potential anomaly is being detected. Besides since the model is obtained regarding a specific failure mode related to a certain component, indicator response will inform about a plausible root cause of the anomaly. Regarding the nomenclature presented in the Ontology section of this work, any model which satisfies aforementioned property is denominated *indicator*.

The training sets for indicator learning are linked to the time instant in which a certain failure mode is evidenced. This instant serves as starting point to set an interval in which is probable that the failure is being revealed in the monitored signals, i.e the degradation period. During this interval it is expected that the indicator response begin to be enough significant.

An indicator, as intended in this particular work, may be understood as a transformation of a n-variable space to one dimensional space, whose response verifies that previous a specified time remains null or at least as low as possible. After that time its response should be considerably higher.

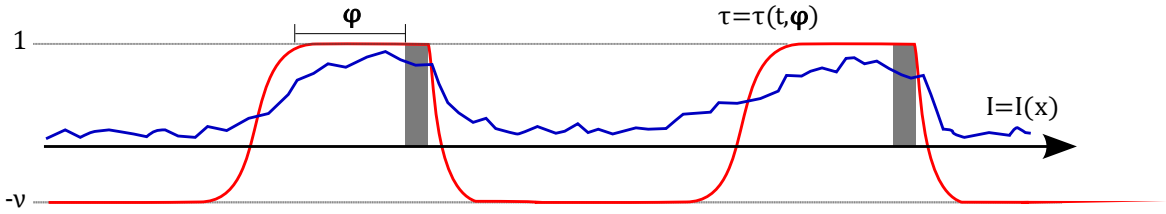

Figure 12: Main components of proposed approach - Grey bars are failure events; Red curve: Test or guide function; Blue curve: Obtained indicator

Concerning the temporal evolution of the transform, two classes of indicators are distinguished:

- First generation indicator (lst-gen): This class of indicators verify sensitivity to the studied failure mode, although its evolution is not constrained to any particular profile.
- Second order indicator (2nd-gen): Second generation indicators are alike health indicators. Their evolution is not decreasing in time and in the case that the analyzed failure mode does not exhibit degradation patterns they present a stepped profile.

In accordance to the described classes, proposed methodology do not ensure the obtainment of second generation indicators. This may occur only if the search space contains the appropriate functions that yield this kind of evolution.

In this work we use genetic programming as a tool for indicator synthesis. Each individual managed by the genetic algorithm is an indicator proposal for the addressed problem. Using genetic programming as a search tool accounts for three main criteria:

- Given the inherent characteristics of the evolutionary search process, relevant variables are automatically selected for being part of the model, while the rest are discarded. Nevertheless, in practice one should be careful because evolved expressions tend to bloat as evolution progress.
- Appropriate transformation of variable space are automatically obtained in virtue of some defined quality metric. Then expert knowledge is not considered when deciding how to transform input space but contributing to set the functional search space for the genetic programming algorithm.
- Resulting models are interpretable using semantic analysis of achieved expressions, or even carrying some sensitive analysis. This fact is highly desirable in terms of acceptance by the maintenance crew of the provided tool. They could trace which elements has been involved in a certain response of the indicator which allows them to have a deepest insight in what is occurring.

Even though the failure detection approach through indicators is no exclusive of this work, as it has been previously pointed out, presented proposal actually present some novel aspects. Among reviewed literature, this work share some common topics with an already research performed by (**Kusiak and Verma 2010**). In their work the authors propose the use of genetic programming to set one-class classifier for wind turbine pitch failures. They start establishing faulty condition by analyzing the status data of the machine. Once these are identified, different prediction horizons are explored using a time window in which all samples are labelled as normal if the instant of the failure is not within it, or faulty otherwise.

Main differences with Kusiak approach lies in the way in which search process is driven. In kusiak's approach there is not consideration about the growing pattern of the failure. When the prediction window is set, all samples within it are considered the same (i.e. they share class). In practice the line between normal samples and failure ones is too blurred. In fact, the machine often experiments different degradation stages before the faulty state is reached. If all samples after some particular instant (especially if the failure is desired to be foreseen with enough margin) the labelling process for further classification will assign the same label to dissimilar samples, and what it is more problematic, it will assign failure labels to samples that are more close to healthy

ones. On the contrary, the proposed methodology searches for models that change their behaviour after some defined time stamp. Usually one knows or at least expect that a certain failure could be early identified before it causes more severe consequences. In the presented approach a function is employed to guide search of indicator models which satisfies imposed requirements for the failure detection task. However, the way in which the indicator grows is not preconditioned.

2. Theoretical notes on the methodology elements

2.1. Genetic programming (GP)

Evolutionary algorithms (AE) belong to the set of global optimisation methods called metaheuristics[1]. Specifically they are part of a class of stochastic learning procedures that *evolve* towards an improved knowledge state regarding to some criterion of interest through the iterative application of learning rules (heuristics) inspired in some natural mechanism, especially those based on the dynamics of living beings. Consequently, this learning ability may be exploited for sake of solving optimization or search problems.

Albeit iterative methods are not a property exclusively linked to evolutionary algorithms, they are applied in a substantially different context. Whereas classic optimisation techniques implement iterative processes in order to take advantage from some mathematical property related to the objective function, evolutionary approach combine potential optimal solutions to a given problem using some defined rules so as to guarantee an overall improvement of all candidates through successive iterations. This property is usually known as *implicit parallelism*.

Evolutionary algorithms emulate natural evolution processes in which individuals from a population pursue them fitness to the environment (adaptation) to ensure their survival. The better the fitness, the higher is the probability of that individual characteristics transcend into future generations (natural selection) via genetic recombination mechanisms (reproduction). Table 3 resumes analogies between natural and artificial evolution processes.

Table 3: Analogies between natural evolution and Evolutionary Algorithms

	Nature	Evolutionary Algorithms
Natural Selection	Survival	Find optimal solution
Individual	Living being	Decision variables vector
Code	DNA	Variable coding (binary, Gray, real, tree,...)
Recombination	Crossover, mutation and reproduction	Simulated crossover, mutation and reproduction
Generation	Life cycle	Iteration

Some of the most common methauristics inspired in Darwinian evolution theory are Genetic algorithms, Evolutionary Strategies, Differential Evolution and, that one which is now centring the focus, Genetic Programming. All these techniques share a common scheme which is reflected in Table 4.

Genetic programming, proposed firstly by Koza (**Koza 1992**), defines a metaheuristic in which a set of programs are evolved through different generations in order to perform a specific task. Basically the process works like any other evolutionary algorithm but with one major distinction: genetic programming aims to seek a solution that is capable of solving a specific user-defined task, hence instead of having a search space comprised by all feasible values for a set of variables, it works with computer programs or algorithms. The way of coding each potential solution or individual is usually done by using tree structure which define the syntax of the program. Other, less common, alternative representation involves the *linear genetic representation* (LGP) in where

[1] Methaheuristics define a specific way in which a heuristic must be implemented.

the program is defined as a set of sequential instructions. The use of one or other codification depends upon the specific application or task being addressed. However, in most of the engineering problems in which GP has been used, the codification was tree-based.

Table 4: Evolutionary algorithm general scheme

Start
$t \leftarrow 0$
Random initialisation of population P^t
Evaluation of P^t
while(stop condition not satisfied) do
Recombination of P^t *to yield offspring* C^t
Evaluation of C^t
Individual selection between P^t *and* C^t *to yield* P^{t+1}
$t \leftarrow t+1$
end
end

Genetic programming, proposed firstly by Koza (**Koza 1992**), defines a metaheuristic in which a set of programs are evolved through different generations in order to perform a specific task. Basically the process works like any other evolutionary algorithm but with one major distinction: genetic programming aims to seek a solution that is capable of solving a specific user-defined task, hence instead of having a search space comprised by all feasible values for a set of variables, it works with computer programs or algorithms. The way of coding each potential solution or individual is usually done by using tree structure which define the syntax of the program. Other, less common, alternative representation involves the *linear genetic representation* (LGP) in where the program is defined as a set of sequential instructions. The use of one or other codification depends upon the specific application or task being addressed. However, in most of the engineering problems in which GP has been used, the codification was tree-based.

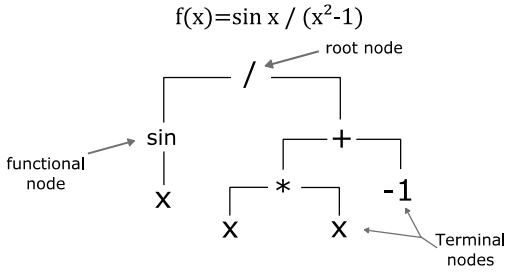

Figure 13: Tree codification of symbolic expressions

In the tree representation there are three basic constructive elements, viz: leaves, internal nodes and root (**Poli et al. 2008**). Leaves, also referred as terminal nodes, are the deepest nodes in the tree which stand for program variables and constants. Internal nodes always include deeper nodes which act like inputs to it. These nodes are called functional nodes. Finally, the root is the shallowest node within the tree structure and is the final output from the program. Actually it is a functional node but receives a special denomination because its importance within the program.

Program is used as a general term which could represent either a mathematical expression describing a physical process or any arbitrary complex algorithm capable of solving a particular task. Here, we are more interested in the use of genetic programming to build an empirical mathematical model of data acquired from the process or system. This procedure is usually known as symbolic data mining (SDM). SMD is an umbrella term embracing a variety of related tasks including generating symbolic equations predicting a continuous valued response variable using

input/predictor variables (symbolic regression); predicting the discrete category of a response variable using input variables (symbolic classification) and generating equations that optimise some other criterion (symbolic optimisation). The latter is the case of current work.

Symbolic optimisation via genetic programming features that have provided attraction to researchers are, among others, the possibility to examine the evolved expressions which can often lead to human insight of the underlying dynamics; the ability to produce compact models (parsimony principle), which is sign of generality and robustness; the limited number of *a priori* assumptions, since the model is not preset or the natural way in which relevant variables are selected to be part of the model. As counterpart, symbolic optimisation take relatively long time to discover acceptable models when comparing it with other nonlinear modelling techniques. Furthermore is difficult to set criteria in deciding if a solution is good in terms of the trade-off between model performance and complexity.

2.1.1. Main elements of genetic programming

Useful terms

- Tree size: Number of nodes that compose a tree
- Node depth: Is the number of edges that have to be travelled from the top node to reach a certain node.
- Tree depth: Is the depth of its deepest node
- Functional node arity: It is the number of inputs used by the function implemented in a node.
- Tree complexity: There is not an unique definition for this term, but in general is related to the number of nodes, tree depth and nonlinearity of the functional nodes of the tree.

Search space

For genetic programming algorithms be able to evolve structures, two main search spaces need to be specified. On the one hand, it is required to setup the function space, indicating the functions that the algorithm will use as building blocks for the structure. The second space may be partitioned into two additional subspaces, one for input variables and other for ephemeral constants. Whenever the algorithm performs an operation over the population it will try any of the selectable specified functions, variables or constants.

Two general properties need two be fulfilled by the functional space in order to GP work effectively, these are *closure* and *sufficiency*. Closure property comprises the *type consistency, i.e.* all functions in the search space accept as inputs the output of any other function in the search space; and *evaluation safety,* which prevent the evolved expressions to lead to singularities or indeterminations, such as dividing by 0.

Closure conditions may be mathematically expressed in the following way:

$$f: \mathfrak{D} \to \mathfrak{D} \quad \forall f \in \Omega \quad (type\ consistency)$$

$$\exists\, f(x) \quad \forall f \in \Omega \ \wedge x \in \mathfrak{D} \quad (evaluation\ safety)$$

, where Ω is the functional search space and $\mathfrak{D}$ is the input space.

By its part, sufficiency condition alludes to the fact that functional space contain necessary and sufficient functions in order to build the desired program or solution. In other words, it means that with available functions it is possible to exactly reproduce the underlying process governing the problem. As an example, if the underlying model of observed data from a specific process is polynomial, then functional space must at least contain the set of operator: $\{+, -, \times\}$. Unfortunately, sufficiency can be guaranteed only for those problems where theory, or experience with other methods, tells us that a solution can be obtained by combining the elements of the primitive set

Initialization

Initial population, as in remaining evolutionary algorithms, is almost always randomly generated. However, instead of generate a random number within a feasible interval, in this case random structures have to be created. Among the several proposed methods to undertake tree generation task, the three more basic and more widely extended procedures are the following:

- Full method
- Grow method
- Ramped half-and-half

Full and grow methods are controlled by the selected tree depth as stop criterion. The first one yields trees in which all terminal nodes have the same depth. Trees are sequentially developed until all terminals have the same length. Grow methods, on the contrary, allows for the creation of trees of more varied sizes and shapes. Nodes are selected from the whole search space (terminal and functional sets) until depth limit is reached. Once depth limit is attained only terminals may be chosen.

Because neither the grow or full method provide a very wide array of sizes and shapes on their own, Koza proposed *ramped half-and-half* method, which is in fact one of the most popular initialisation method. In his approach initial population is set up by using full and grow method for both halves, respectively. This is done using a range of depth limits (hence the term "ramped") to help ensure that we generate trees having a variety of sizes and shapes.

Selection

During generations best fitted individuals have better chances to share its genotypic material to rise subsequent populations. Preserving worthier features along the search is what allows the process to converge to optimal solutions. Although there are several methods to perform selection, the most extended one is the *tournament selection*.

In tournament selection, two or more individuals randomly picked from the population compete among them so as to be part of breeding process. The individual with highest fitness is chosen to be a parent. It is clear that deciding how fitted is a particular individual requires a measure of goodness (i.e. fitness function).

Fitness function is fully problem dependant and specifies the degree of fulfilment of a desired objective. Fitness function may be also devoted to handle exceptions occurred when evaluating each individual such as non-desired infinite values and indeterminations.

Usually selection is solely based upon greedy approaches which take only into account the fitness values of the individuals. This might lead to complex final solutions which are capturing noise from the process instead of remain general, this is called *model bloat*. Several authors also consider model complexity so as to drive the selection process and avoid bloating. The way of introducing complexity has been diverse, ranging from a modified version of fitness function to account for complexity through a weighted combination of both fitness and complexity, to the employment of model complexity to solve the tie whenever several competing solutions performs equal. Last method was proposed in (**Luke and Panait 2002**) and is called *lexicographic parsimony pressure*.

Assigning complexity is not a trivial task as depending on the used metric the heuristic will provide different preference values to each solution. Among existing alternatives (**Smits and Kotanchek 2004**) summarize the following (applied to symbolic optimisation):

- Tree depth
- Tree nodes
- Component function nonlinearity (e.g. "+" is less non linear than exponentiation or sine functions)

- Number of variables: either total number of variables in the model or unique variables.
- Expressional complexity: is based on the recursive aggregation of all nodes tree depths. This sort of metric has the advantage of favouring flatter structures.

Genetic operators

These operators are in charge of establishing the way in which evolution is carried out. In fact they are the actual cause of overall population improvement, allowing the exploration of the whole search space. Genetic operators are intended to keep a trade-off between speed of convergence and diversity through generations so as to avoid local optima stagnation. Classic genetic operators comprise reproduction (a.k.a replication), crossover and mutation.

Reproduction is an asexual method where a randomly selected individual copies itself into the new population without suffering any kind of modification (Walker 2001). This is a way to allocate some anchors in the search space that in some cases act preserving good current solutions but in other cases keeping bad individuals which helps to maintain diversity when population evolves towards local optima. When the evaluation of fitness function is deterministic, reproduction has been found to speed up search since a fraction of the population has not need to be revaluated.

Crossover requires two parents which yields two offspring in the new population. Traditional crossover operator select to random nodes in both parent trees and pending subtrees are exchanged between them to create two new individuals, *standard crossover*. This is shown in Figure 14.

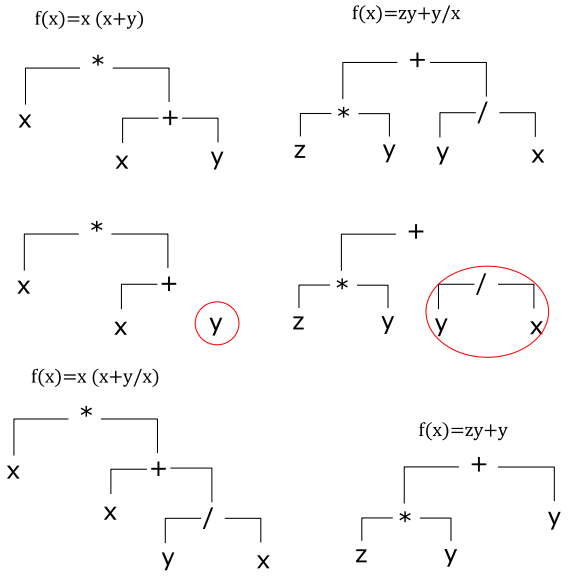

Figure 14: Standard crossover

Crossover have a prominent influence in evolution performance, so it is desirable to know what to expect in average when two parents are combined. Sometimes, crossover yields offspring that barely seems progenitors which would make evolution to pass slowly due to lack of direction in the search. Otherwise, if parents dont share sufficiently different genetic material evolution will suffer of premature convergence. A work that shed light around this topic is (Poli and Langdon 1995) where standard crossover is compared with both, *one-point and uniform crossover* operators in terms of them searching properties. Other authors has addressed this question proposing what they denominate *semantic similarity crossover* (Uy et al. 2010). In their approach, only when subtrees selected for swapping between parents are semantically similar the crossover is performed. This similarity is defined in terms of an arbitrary metric of distance of parents outputs.

Finally, the most commonly used form of mutation in GP (which is usually called *subtree mutation*) randomly selects a mutation point in a tree and substitutes the subtree rooted there with a randomly generated subtree. Subtree mutation is sometimes implemented as crossover between a program and a newly generated random program; this operation is also known as "headless chicken" crossover (**Poli et al. 2008**).

Unlike classic genetic algorithms, in GP exist a vast number of mutation operators that may be applied, which often are used simultaneously providing some benefits. Among them, besides subtree mutation, there are:

- Terminal Mutation: Switch an input terminal to another randomly selected input terminal.
- Random constant mutation
  - Perturbation: mutated constants by adding random noise from a Gaussian distribution. Each change to a constant was considered a separate mutation.
  - Zero mutation: Set a randomly selected constant to zero.
  - Substitution: Substitute a randomly selected constant with a randomly generated constant.
  - One mutation: Set a randomly selected constant to l.

All described mutation operators are included in the genetic programming implementation tool used in this work.

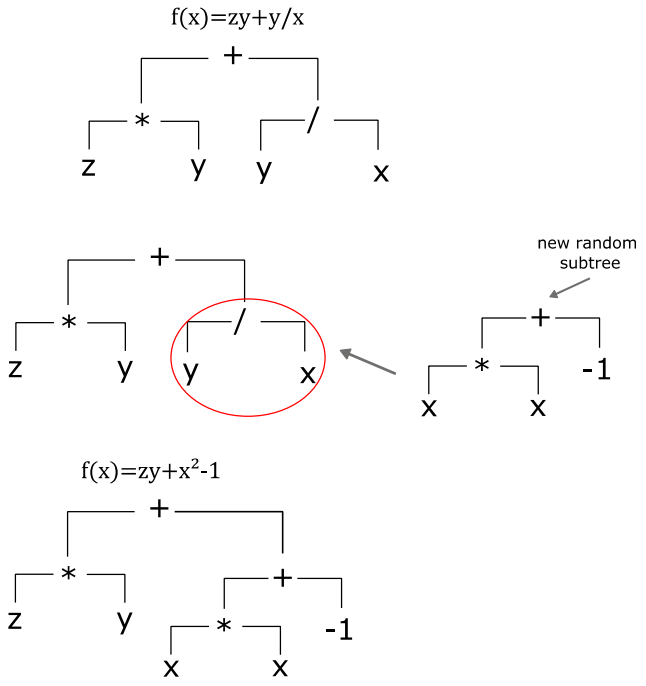

Figure 15: Subtree mutation

2.2. Sampling procedures

This section is devoted to give some brief notes on sampling procedures based on Monte Carlo approaches, as a worthy alternative to take into account when evaluating fitness function under the presented proposal. One of the major drawbacks of evolutionary algorithms, and so it's the case of genetic programming, is that they keep a large number of potential solutions through a usually high number of iterations (generations). Whereas tournament selection and genetic operators does not involve too much computational cost, this is not the case when it comes to the fitness function. Assigning fitness is, in most of the cases, the critical step when optimising by means of evolutionary methods as they usually involves much more evaluation of the targeted objective. Thus, depending on the difficulty of fitness evaluation the process could

become prohibitively long. Hence, it shall be necessary to use some efficient methods such as surrogate methods or another more suitable ones.

In this work, as it will be presented in section 3.l, fitness function is a score function (ς) expressed as the summation of pointwise obtained partial scores (ς_n). Even when, the evaluation does not include difficult operations to compute, the addressed problem suffers from huge amount of data issues, which means that a large number of simple operations have to be performed. However, given the formulation of faced problem, many of these evaluations may be trivial regarding to the actual value of the score of each potential solution, but still they are going to take not negligible computing time.

One way to overcome these limitations is to use an approach based on theory of Bootstrap filters (BF) (**Doucet et al. 200l**). Bootstrap filter is based on the concept of particle filters in which particles are created sequentially over the sampling process. A particle is defined as a pair consisting of the value of the sample and a importance measure, $\{\boldsymbol{x}^{(i)}, w^{(i)}\}$. The key idea is to gradually eliminate particles having low importance weights while multiplying particles which have high importance weights. Particle weights are based on the partial contribution of a sample to the global score.

3. Describing learning approach

This methodology for automatic learning of failure indicators emerge from the actual form of available data coming from industry to generate detection models. As outlined previously in this chapter, the only identified event contained in stored information of systems operation is the time at which the failure arises. In general there is not any information about failure mode evolution until the component or system fails. Therefore learning process have to be carried out in a semisupervised way. Going backwards from the time to failure, a region in which the failure could be expected to appear in monitored signals is a priori established. Then the algorithm must yield an expressive and robust model characteristic of the underlying failure mode.

To train an indicator on a certain failure mode, a set comprising one or (preferably) more historical monitored signals of the operation of the system until the failure event is provided. Each of these subsets will be referred as *Run-To-Failure* case (RTF).

In subsequent sections are described the functions which drives the search process to obtain failure indicators. Afterwards a test bed is presented to test the learning methodology proposed in this work.

3.l. The objective function

To drive the searching process, an objective function is developed, that lead a set of solutions that verify the properties previously defined. Accordingly to proposed approach, a score function is used, whose value is maximized during the searching process. In this work, the score of a potential function is solved using a test function or guide function. This function delimits the time regions where the indicator must modify its response and start to indicate the failure event. The shape of the score function is expressed in the Equation (l)

$$\varsigma = \sum_{i=1}^{M} \sum_{j=1}^{N} I(\boldsymbol{X}_{ij}, t_j) \otimes \tau(\boldsymbol{\theta}, t_j) \qquad \text{(l)}$$

, where $I(\boldsymbol{X}_{ij}, t_j)$ is the indicator value for every n-tuple input data ($\boldsymbol{X}$) in the $j\text{-}th$ instant of time (t_j) located in the $i\text{-}th$ of the RTF that conform the training set. $\tau(\boldsymbol{\theta}, t_j)$ is the value of the test function evaluated in the $j\text{-}th$ instant of time. The partial score assignation of the evaluation of each n-tuple is defined using the operator $\otimes$, in this work defined as:

$$\otimes\,(a, b) \stackrel{\text{def}}{=} \frac{|a|}{P_\alpha(\boldsymbol{A})} \cdot b\,; \quad a \in \boldsymbol{A} \qquad (2)$$

In this case, $P_\alpha(\cdot)$ indicates the alpha percentile of the set of values that adopts the indicator when it is evaluated in every instant of time over the RTF set (X_i). In this work, a value of de α equal to l00% is used, i.e. the maximum value.

Expressing the operator using the indicator, the result is:

$$I(\boldsymbol{X}_{ij}, t_j) \otimes \tau(\boldsymbol{\theta}, t_j) = \frac{|I(\boldsymbol{X}_{ij}, t_j)|}{P_\alpha(I(\boldsymbol{X}_i))} \cdot \tau(\boldsymbol{\theta}, t_j)$$

In general, the result of the expression found by the algorithm is not constrained to positive values, and its value will depend of the transformation found. However, the test function has an image contained in the interval $(-v, 1)$, as it is detailed in section 0. Attending to this, if the response obtained by the indicator were negative in the degradation area, the score metric will be penalized incorrectly. To avoid this issue, the indicator is evaluated in absolute value.

Table 5: Objective function implementation

```
Start
t ← 0
Parse expression to function
For i=1 until number of Run-To-Failure cases
  for j=1 until size of dataset
    I(i,j) ← Evaluate indicator on j-th sample of i-th case
    Take absolute value over obtained output
  end
end

Obtain minimum value from indicator outputs
I ← Substract minimum value to each indicator output
P_a ← Get selected quantile to scale indicator outputs values

I ← Divide each output by calculated quantile

for i=1 until number of Run-To-Failure cases
  for j=1 until size of dataset
    If I ≥ 1 then
      I ← 1
    end
    ς ← I(i,j) · τ(θ,t)
  end
end
return(ς)
end
```

In addition, concerning the score function implementation, each individual value is imposed an upper bound, using the following criterion:

In addition, related to the score function implementation, the individual value is limited in the upper part, using the following criterion:

$$\hat{I}(\boldsymbol{X}_{ij}, t_j) = \begin{cases} 1 & \frac{|I(\boldsymbol{X}_{ij}, t_j)|}{P_\alpha(I(\boldsymbol{X}_i))} > 1 \\ \frac{|I(\boldsymbol{X}_{ij}, t_j)|}{P_\alpha(I(\boldsymbol{X}_i))} & \text{otherwise} \end{cases} \quad (3)$$

Attending to this, Equation (l) can be expressed in the following way:

$$\varsigma = \sum_{i=1}^{M} \sum_{j=1}^{N} \hat{I}\left(\boldsymbol{X}_{ij}, t_j\right) \cdot \tau(\boldsymbol{\theta}, t_j) \quad (4)$$

This modification is proposed after the result analysis obtained in one of the preliminary test cases described in section 4.l of this chapter.

In the Table 5 the algorithm that implements the objective function is exposed.

3.2. The test function

Partial score of each indicator is governed by the test function. This function delimits two different regions in each Run-To-Failure experiment. On the left of the cutting point of abscissas axis (temporal axis) the system us considered to be normally operating, hence the function penalizes every not null response yielded by the indicator. On the right of aforementioned cutting point, test function rewards any response experienced by the indicator.

As it has been mentioned, the only certain available information concerning the component failure is the time in which the component is declared to be operating under its intended performance. Test function is configured using that event as reference through a set of parameters (θ) defining its shape. These parameters should be determined through trial-and-error approach or by means of a optimisation procedure in the context of wrapper methods.

In this work function with three adjustable parameters and another deterministic one is selected for sake of generality and flexibility. This are described below:

$$\boldsymbol{\theta} = \{\delta, \varphi(p, \delta), \nu, T_f\}$$

- δ – defines the interval, starting from time to failure instant, in which any response experienced by the indicator will be positively scored. It ideally represents the period of time in which the degradation of analyzed component should begin to be noticeable. Thus, $\tau\left(\boldsymbol{\theta}, T_f - \delta\right) = 0$.
- p – fraction of δ parameter establishing the instant posterior to, $T_f - \delta$, in which test function value should be equal to b. At first glance, we could have shaped the test function as a step function rising at δ. However, typically the transition periods between normal and faulty states are fuzzy. We then relax the growth of test function a parameter that is governed by p. In the test function it is included in the following described manner:

  $$\varphi(p, \delta) = \frac{1}{2(1+p)\delta} \cdot ln\left(\frac{1+b}{1-b}\right)$$

  , where b is set to 0.999 in this work. It has actually an arbitrary value but its use it is justified since the limit of test function when $b \to 1$ is infinite.
- ν – This parameter modules penalization degree of each indicator output having non null value previous to the instant $T_f - \delta$. The value of ν is theoretically contained within the interval $[0, +\infty)$. However its actual value is problem dependent. Some criteria to set its value relies in the number of samples in both normal and degradation regions, or in terms of cost based learning, depending on the preferences concerning false alarms versus missing alarms ratio.
- T_f – is the instant in which the failure takes place. It acts as a reference point aiding to adjust the rest of tuneable parameters of the test function.

Test function is defined as a piecewise function based conveniently in the hyperbolic tangent function. Through an appropriate change of variable from t to t′, test function has the following form.

$$t' = t + \delta - T_f$$

$$\tau(\boldsymbol{\theta}, t') \stackrel{\text{def}}{=} \begin{cases} \nu \dfrac{1 - e^{-2\varphi t'}}{1 + e^{-2\varphi t'}} & \text{sii } t' \leq 0 \\ \dfrac{1 - e^{-2\varphi t'}}{1 + e^{-2\varphi t'}} & \text{sii } t' > 0 \end{cases} \tag{5}$$

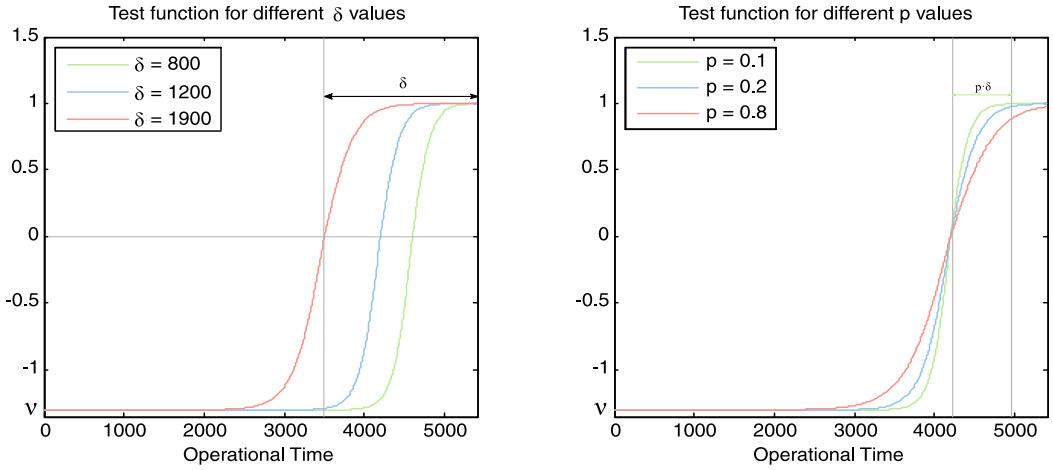

Figure 16: Effect of parameters of the test function on its shape

4. Verification cases

In this section some demonstration cases are presented in order to validate the proposed methodology. Main goals of proposed cases are the study of the effect of test function parameters on indicator synthesis and to explore methodology capabilities in different contexts that might be found actually in machine failure detection tasks.

Test cases will consist of five variables at most, representing monitored signals coming from machines. Hidden between them will be a degradation pattern, expressed as $\omega(t) = 1 - \mathrm{e}^{-(0.0064t+4500)}$, related to one variable or a combination of them. Remaining variables may be either random noise or may have an expressive evolution through time but uncorrelated to the wanted failure mode. Each dataset emulates a Run-To-Failure experiment whose configuration its described in Table 6.

Table 6: Variables considered in the experiments - First row: IC-001; Second row: IC-002; Third row: IC-003; Fourth row: IC-004; Fifth row: IC-005

x_1	x_2	x_3	x_4	x_5	ε
$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$	$\omega(t)$	–	–
$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$	$0.6\mathrm{e}^{-\left(\frac{t-1200}{10^5}\right)^2} + 0.85\mathrm{e}^{-\left(\frac{t-4500}{10^5}\right)^2}$	$\omega(t)$	–	–
$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$	$0.6\mathrm{e}^{-\left(\frac{t-1200}{10^5}\right)^2} + 0.85\mathrm{e}^{-\left(\frac{t-4500}{10^5}\right)^2}$	$\omega(t)$	–	
$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$	$\omega(t) + x_4$	–
$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$	$0.6\mathrm{e}^{-\left(\frac{t-1200}{10^5}\right)^2} + 0.85\mathrm{e}^{-\left(\frac{t-4500}{10^5}\right)^2}$	$\omega(t) + \varepsilon$	–	$\mathcal{N}(0,0.15)$

To perform indicator search a free, open source library coded in MATLAB®[2] named GPTIPS was used (**Searson 2015**). GPTIPS is a symbolic data mining highly customizable tool that allows the user to implement its own problem definitions as also to include new built-in functions to be part of the search space. It also provides functionalities on model simplification and post run population examination in order to evaluate evolution run.

[2] MATLAB is a trademark of The Mathworks Inc.

Is not in the scope of present work to make performance comparisons between multiple algorithm configurations, therefore default evolution parameters with some modifications are proposed for test cases. Selected parameters a summarized in Table 7

Table 7: Configuration parameters of genetic programming algorithm

Parameter	
Type of search	Maximisation
Generations	1000
Population size	100
Initialisation	Ramped half-and-half
Maximum tree depth	8
Selection	Lexicographic Tournament
Crossover operator	Standard Crossover
Mutation Operator	Terminal mutation/Perturbation/Zero mutation
Crossover probability	0.8
Mutation probability	0.1
Mutation operator probability	0.9/0.05/0.05
Maximum depth of mutated subtree	4
Tournament size	2
Constant generation range	[-10 10]
Constant generation probability	0.2
Search space	$\times, -, +, \sum_{i=1}^{3}(\cdot), \prod_{i=1}^{3}(\cdot), \exp(\cdot)$
Stopping criterium	Number of generations

In this section several configurations of the test function are used to evaluate its impact in the search process. As exposed in section 0 test function conditions the shape of potential indicators depending on the parameters selected. For verification cases, this parameters are found in Table 8.

Table 8: Test function parameters for each experiment

Experiment	δ	p	φ	ν
IC-001	1900	0.1	0.020001059	1.87
IC-002	1000	0.1	0.038002012	1.87
IC-003	1000	0.1	0.038002012	3.74
IC-004	1000	0.1	0.038002012	3.74
IC-005	1000	0.1	0.038002012	3.74

4.1. Effect of δ parameter (IC-001)

When setting the test function we may serve from expert knowledge to specify the time instant when the failure will become apparent in the monitored signals. Its important to keep in mind that usually the only known fact is the time of failure. Starting from that point, the objective is to be able to forecast that moment with enough anticipation. This is considered with the δ parameter which controls the raising instant of the test function. Ideally δ is tunned to stay as far as possible from the Time-To-Failure (TTF), which means that the failure will be evidenced in an early stage permitting to perform preventive activities over the asset. However, setting this parameter too far from the TTF will lead to the appearance of false positives incurring in inefficiencies in the asset management and the producing losses due to extra maintenance or surveillance. If viewed under the scope of supervised learning, this corresponds to label incorrectly the samples.

In Figure 17 are represented different runs for the case IC-001 where there is only one expressive variable related to the desired failure mode. If test function raises once the degradation have already started we observe that the search process provides a solution that tries to fit the indicator as much as possible to the test function, delaying the detection of the anomaly (Figure 17(a)). Solutions which actually start to rise at the time of the degradation will be downrated by the test function as they fall outside of the positive region. On the contrary, if the test functions raises, with a reasonably margin, before the degradation process, then the indicator will rise when degradation becomes apparent. As observed in Figure 17(b-c) there are several feasible paths for the indicator to evolve. The last figure contains the results of a run in which is shown that a solution (i.e. an indicator) that better fits the test function is preferred instead other possible indicators. In fact, just selecting an indicator based on x_4 will be enogh to track the degradation process (Figure 17(b)), but there are other solutions that satisfies more search specifications.

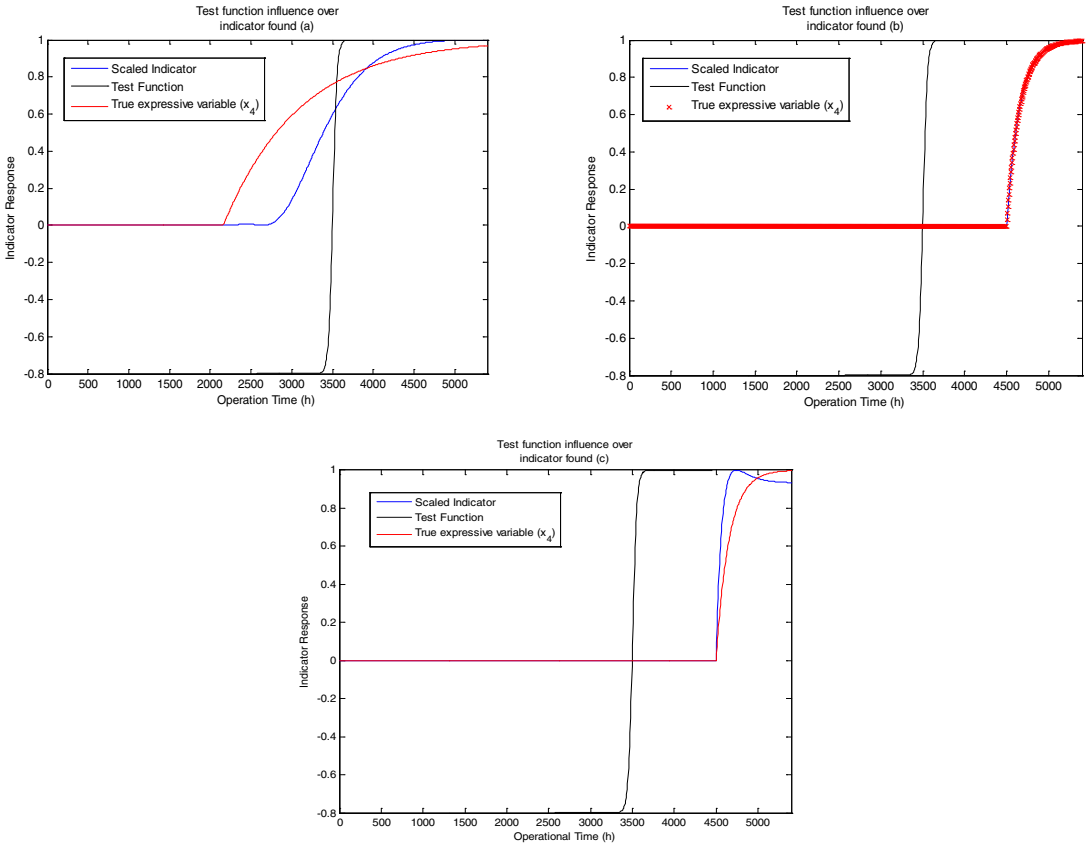

Figure 17: Effect of δ parameter in search process

Beyond the ability to discriminate trivial variables and detect expressive patterns, this experiment has yielded another important conclusion. Proposed methodology allows the option of interactively tune test function parameters based on achieved results. Hence, if indicator raises before the test function one may think that has been conservative for failure anticipation and then shift back in time δ parameter. On the other hand, if indicator raises after test function, it is possible to conclude that first hypothesis on failure detectability has been too optimistic.

Another conclusion extracted from this case was the need of imposing an upper bound to the output of the evolved expression obtained by the genetic algorithm, then yielding expression (3). When the evolved expression was scaled with a percentile of the output instead of the maximum value, the inherent properties of the search process favoured solutions of the form presented in Figure 18. Because the nature of the primitive score function in equation (1), indicators which present very high values in the positive region delimitated by the test function had enormous scores. Therefore, the

methodology was rewarding solutions presenting singularities in the region at the right of the test function, even when they were inexpressive of any failure pattern. Bounding the scaled indicator has been proved to be effective to avoid such solutions and guarantee good performance.

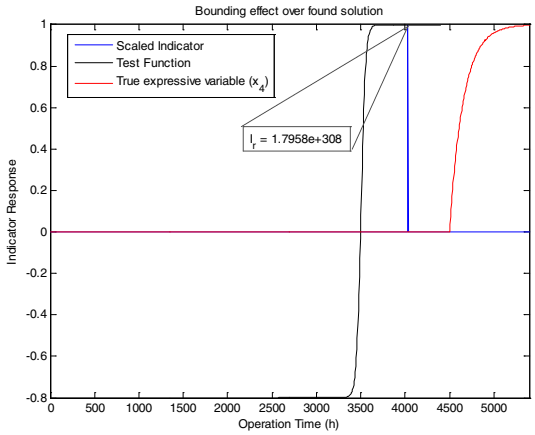

Figure 18: Effect of unbounded indicator

4.2. Testing effect of competing failure modes (IC-002 & IC-003)

In this experiment we are interested in proving whether the methodology works properly when there are more than one expressive variable but only one is related to the desired failure mode. Experiments IC-002 and IC-003 where used to address this question.

Table 9: Summary of genetic programming algorithm runs for IC-002

	Fitness	Complexity	Best Model Included Vars.
lst Run	150	73	x4 x5
2nd Run	150	35	x4 x5
3rd Run	147.65	365	x4 x5
4th Run	124.93	5	x4 x5
5th Run	147.41	417	x4 x5
6th Run	144.95	318	x4 x5
7th Run	136.28	367	x4 x5
8th Run	124.93	107	x4 x5
9th Run	147.53	212	x4 x5
10th Run	150.08	431	xl x2 x3 x4 x5
llth Run	141.03	354	x4 x5
12th Run	125.36	623	xl x2 x3 x4 x5
13th Run	146.68	259	x4 x5
14th Run	141.12	539	x4 x5
15th Run	145.97	943	x4 x5
Resume (μ/σ)			
Fitness	141.59 / 9.35		
Complexity	336.53 / 246.86		

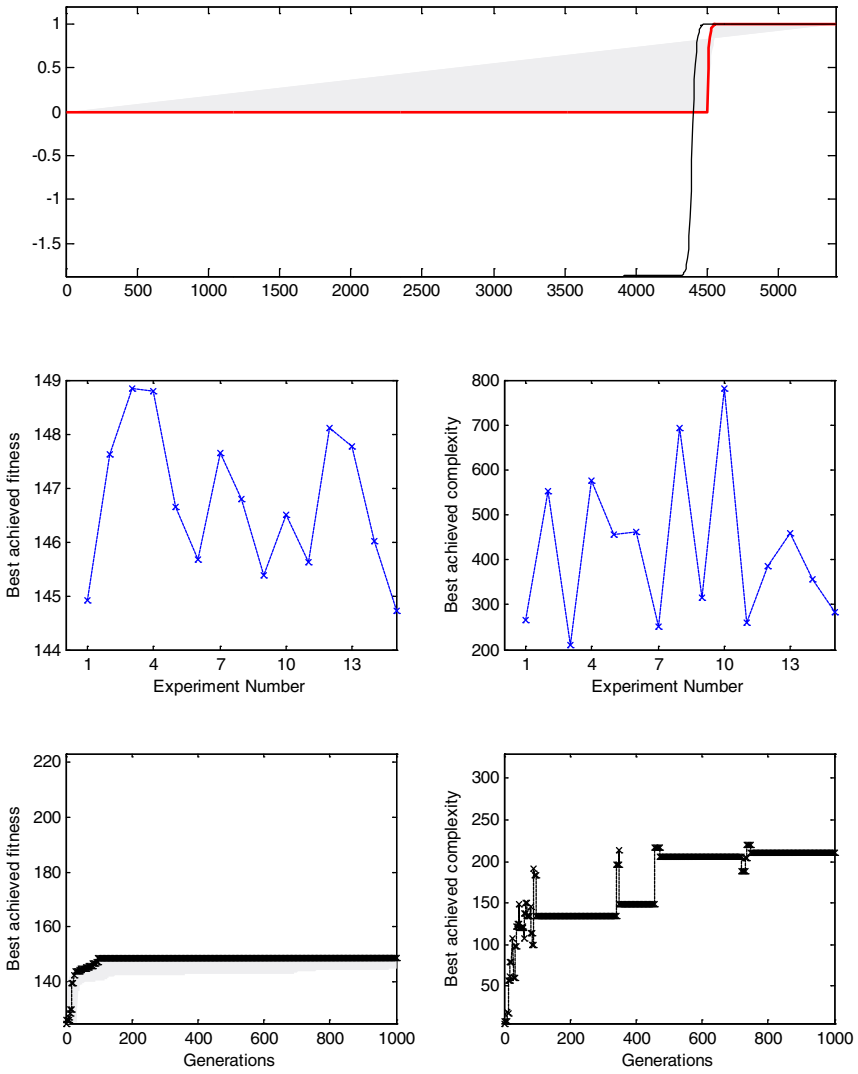

Figure 19: Resuls from expreriment IC-002 results: Top plot - Best achieved indicator (Run3) in red and the interval of the range of instant values of the indicator through different runs. Midle row - Best fitness achieved in each run and the complexity of best fitness solution in each run. Bottom row - Evolution of fitness (left) and complexity (right) over all generations of the best run.

Each experiment was run fifteen times in order to avoid random effects due to the nature of genetic algorithm. In each experiment the test function parameter ν was set to 1.87 and 3.74. The intention was to strongly penalize potential solutions that were sensible to x_3, which is related to a second failure mode that is not under study.

Conducted experiments were not conclusive about this fact. The experiment was arranged for secondary failure mode (x_3) to be partially overlapped with primary failure mode (x_4). But, secondary failure mode also exists in the negative region of the test function. The initial hypothesis was that, if x_3 were included in the indicator model it will yield poor performance in the region before the primary failure mode occurrence. Nonetheless, as almost all constructed expressions are strongly based on x_4 and since this variable remains null in all negative regions, the effect of including x_3 in the model only can improve solutions arbitrarily being combined with the actually expressive variable.

From Figure 19 it is observed that as process progresses complexity becomes dramatically higher while barely improving fitness. This phenomenon occurs in almost all cases, so it is possible to conclude that the genetic algorithm find early well fitted solutions and gradually increase indicator complexity by adding terms unrelated to the failure but eventually having good behaviour within the interest region of the test

function. This is called overfitting. In section 4.4 we try to overcome this issues using two training sets.

4.3. Detecting expressiveness resulting from a combination of two variables (IC-004)

Escalating in complexity, this experiment hides the expressiveness of the desired failure mode as the difference between two of the monitored variables (experiment IC-004). As can be properly seen in Appendix I, Figure 28, variable x_5 is expressive by itself, however it contains noise with the same order of magnitude than its range of variation. In combination with x_4 random noise vanishes and a clear growing failure pattern appears.

Best solutions achieved among different runs have a high average fitness and in almost all runs the variables contained in the best achieved model are only the non trivial ones. Besides, various executions ended with a best model constituted by the exact relation of both variables (i.e. $x_4 - x_5$). Nevertheless, a non negligible amount of runs obtained expressions with stepped growing profiles.

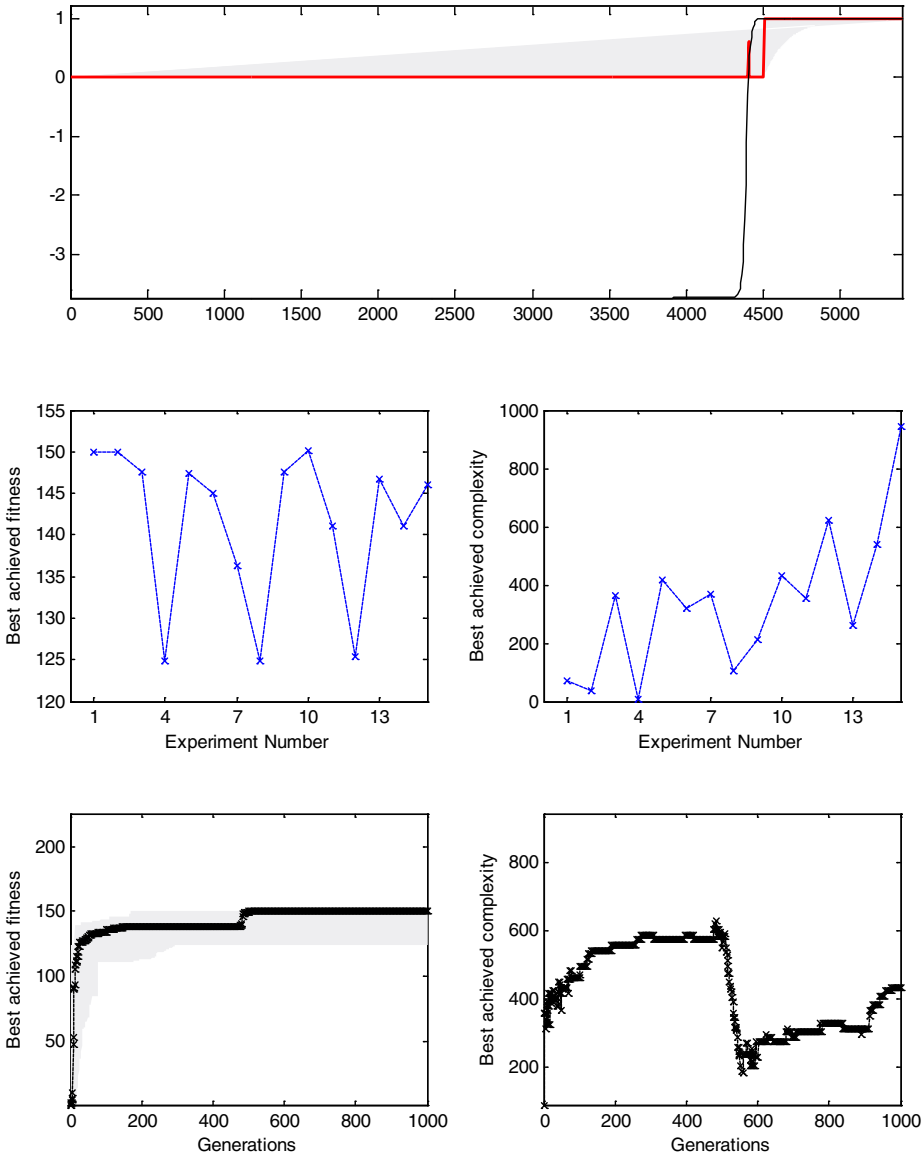

Figure 20: Results from experiment IC-004: Top plot - Best achieved indicator (Run10) in red and the interval of the range of instant values of the indicator through different runs. Midle row - Best fitness achieved in each run and the complexity of best fitness solution in each run. Bottom row - Evolution of fitness (left) and complexity (right) over all generations of the best run.

As it has been commented in previous section, we cannot ensure that this fact takes place because it is actually a better indicator for that kind of degradation mechanism or because overfitting problems. The most obvious example of overfitting is presented in Figure 20 where the achieved indicator exhibits a peak just before starting to growth. This occurs in a similar way than in IC-003, where relevant variables controls indicator evolution and, at some point, noise inputs offer to the process the chance to obtain higher score. Overfitting will cause that obtained models are not general and then will produce many false alarms or missing alarms.

Table 10: Summary of genetic programming algorithm runs for IC-004

	Fitness	Complexity	Best Model Included Vars.
1st Run	144.92	267	x3 x4
2nd Run	147.62	552	x3 x4
3rd Run	148.85	210	x3 x4
4th Run	148.8	577	x1 x2 x3 x4
5th Run	146.64	457	x3 x4
6th Run	145.67	461	x3 x4
7th Run	147.65	251	x3 x4
8th Run	146.79	694	x1 x2 x3 x4
9th Run	145.39	317	x3 x4
10th Run	146.5	781	x3 x4
11th Run	145.63	259	x3 x4
12th Run	148.11	385	x4
13th Run	147.78	460	x3 x4
14th Run	146.01	356	x3 x4
15th Run	144.71	282	x2 x3 x4
Resume (μ/σ)			
Fitness	146.74 / 1.35		
Complexity	420.60 / 170.94		

4.4. Testing effect of training with several datasets (IC-005)

There are many approaches in the existing literature to deal with the problem of overfitting. In search problems using genetic programming, overfitting is closely related to expression complexity (bloating problems).

Table 11: Summary of genetic programming algorithm runs for IC-005

	Fitness	Complexity	Best Model Included Vars.
1st Run	112.97	1070	x1 x3 x4
2nd Run	105.11	1373	x2 x3 x4
3rd Run	122.72	1423	x3 x4
4th Run	113.53	812	x3 x4
5th Run	111.28	1155	x1 x3 x4
6th Run	113.21	653	x3 x4
7th Run	103.84	1217	x2 x3 x4
8th Run	108.54	565	x4
9th Run	127.93	844	x4
10th Run	111.65	1238	x3 x4
11th Run	110.66	626	x4
12th Run	114.07	1299	x1 x2 x3 x4
13th Run	113.13	792	x3 x4
14th Run	112.73	823	x2 x3 x4
15th Run	111.28	454	x3 x4
Resume (μ/σ)			
Fitness	112.84 / 5.96		
Complexity	956.27 / 315.47		

In this experiment (IC-005) we try to tackle overfitting by providing a training set comprised by two RTF cases. One of them has again x_4 as the expressive variable which coexists with the variable x_3 being representative of a competing failure mode that is out of scope of the analysis. Remaining variables are kept as uniform random variables. The second case for model training does not contain the searched failure mode but only the undesired one. Additionally, expressive variable has been perturbed by adding gaussian noise as, in real cases, variables are always subject to some level of disturbance.

Unlike in IC-002/3 cases, synthesized indicator models are based, in their vast majority, in the expressive variable x_4, eventually containing x_3 input and in some remote cases the model also includes noise variables. As it can be seen in Figure 2l average complexity of current experiment models is higher than previous ones. This make sense since to ensure high scores, smoothing operations over the variable of interest have to be applied, which usually involves the use of several function compositions.

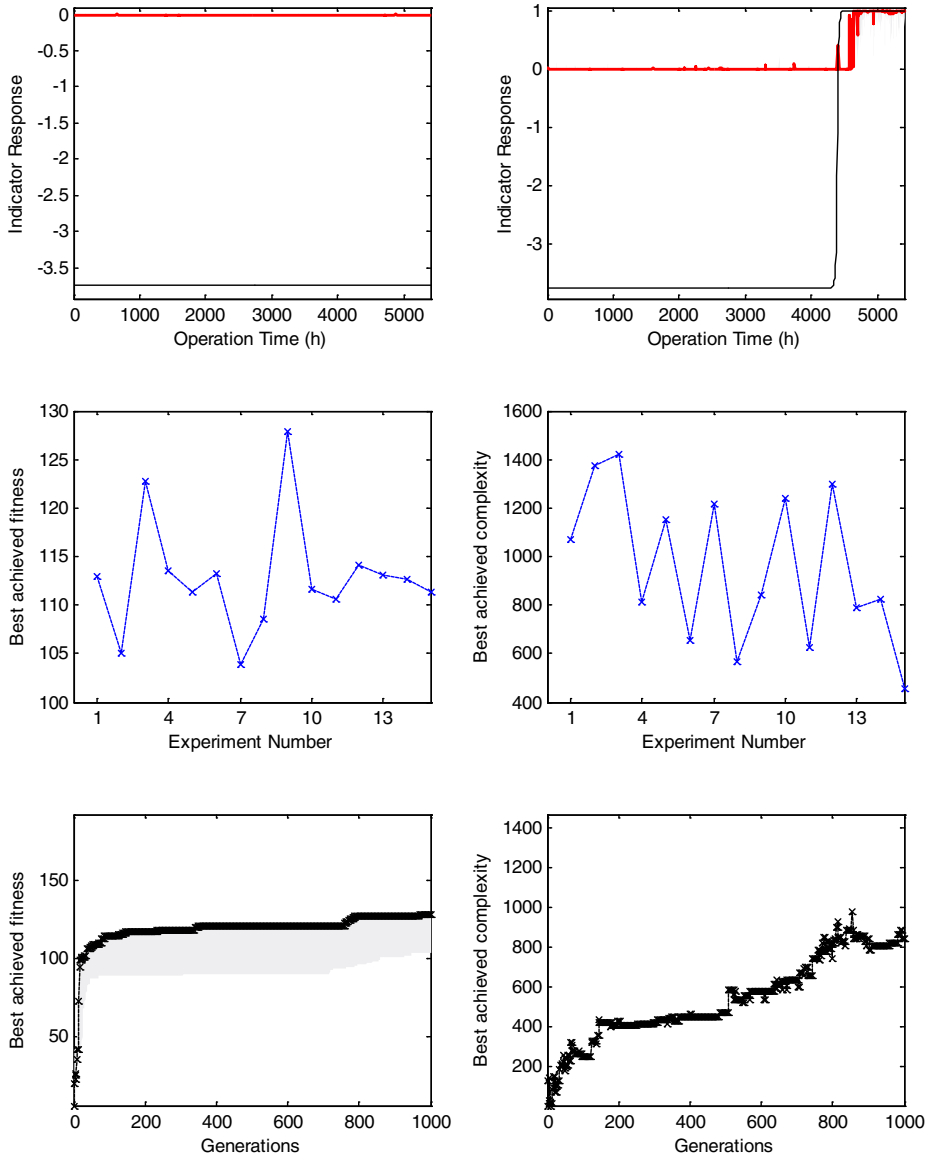

Figure 21: Results from experiment IC-005: Top plot - Best achieved indicator (Run 9) in red and the interval of the range of instant values of the indicator through different runs. Midle row - Best fitness achieved in each run and the complexity of best fitness solution in each run. Bottom row - Evolution of fitness (left) and complexity (right) over all generations of the best run.

As shown in Figure 21, best achieved model in current experiment is able to decrease the noise inherent to x_4 in left region of the test function. In fact, it employs great efforts in doing so, since every non null value is strongly penalized by the test function. However, this yields a very spiky growing of the indicator in the interest region which prevents its potential use as prognosis indicator.

Chapter 4

Test case: Application on bearing dataset

Bearings are mechanical components belonging to wind turbines as members of the drivetrain, amid other subassemblies such as pitch or yaw actioner systems. Drivetrain bearings are specially critical in relation to wind turbine functioning since they are subject to big loads and harsh operational conditions. Furthermore, if a failure arise during their operation, other much more critical subsystems might be affected causing huge losses due to long periods of inactivity and major repair costs. Drivetrain includes main bearing, gearbox bearing, drive-end generator bearing and non drive-end generator bearing. Among them, the last two bearings have the highest working exigencies since they are placed on the high speed shaft (usually rotating at 1800 rpm) which is coupled to the end of gearbox.

Consequently, wind turbine operators are very interested in being able to detect and diagnose generator bearing failures in order to plan with enough margin of time maintenance activities, which presumably will save high life cycle costs of the asset in comparison with corrective actions.

Bearing diagnostic has been traditionally addressed through vibration analysis, yet, in practice, monitoring of wind turbines performance parameters is carried out by means of a Supervisory Control and Data Acquisition (SCADA) system, which often do not comprise required sensors for that kind of analysis. Data coming from SCADA system suffers from dirtiness (outliers, inconsistent data, dataseries lack of completeness, etc.) and often mask uncontrolled events (which many times are not well documented), such as design revisions or modifications or even maintenance actions barely justified. This context prevent us to use this source of data in order to test and validate the proposed methodology for indicator synthesis.

As vibration analysis has been widely used for rotating machinery diagnosis and there is a lot of accumulated knowledge around its application over bearings, it seems fairly suitable to use these datasets to test proposed methodology and compare achieved results with those already existing in the literature.

1. Dataset description

Used dataset (**Lee et al. 2007**) has been contributed by Center for Intelligent Maintenance System, University of Cincinnati, Cincinnati, OH. Data consist of Run-To-Failure experiments taken over Rexnord ZA-2115 double row bearings. These were installed on a shaft as shown in Figure 22. The rotation speed was kept constant at 2000 RPM by an AC motor coupled to the shaft via rub belts. A radial load of 6000 lbs is applied onto the shaft and bearing by a spring mechanism. All bearings are force lubricated.

PCB 353B33 High Sensitivity Quartz ICP accelerometers were installed on the bearing housing (two accelerometers for each bearing [x- and y-axes] for data set 1, one accelerometer for each bearing for data sets 2 and 3). Sensor placement is also shown in Figure 22. All failures occurred after exceeding designed life time of the bearing which is more than 100 million revolutions.

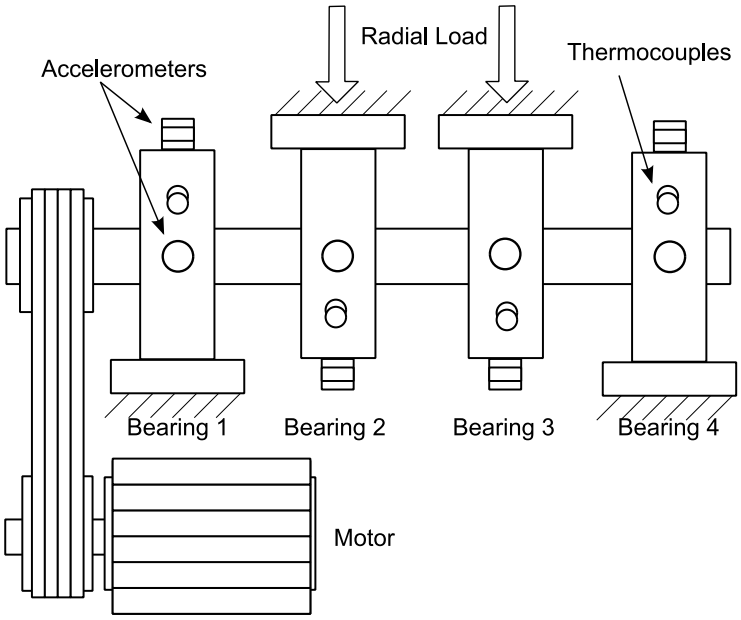

Figure 22: Bearing test rig and sensor placement illustration (Lee et al. 2007)

At the end of the Run-To-Failure experiment the following evidences were found:

- Test 1 (October 22, 2003 12:06:24 to November 25, 2003 23:39:56): Inner race defect occurred in bearing 3 and roller element defect in bearing 4
- Test 2 (February 12, 2004 10:32:39 to February 19, 2004 06:22:39): Outer race failure occurred in bearing 1.
- Test 3 (March 4 2004 09:27:46 to April 4, 2004 19:01:57): outer race failure occurred in bearing 3.

2. Data pretreatment

Vibrations are the result of dynamical response of materials due to external forces. They are measured with vibration sensors located on the monitored component. Depending on the phenomenon causing the vibration response, the measured property of vibration is selected. Thus, for low frequency processes ($\leq 10\,Hz$) displacements are preferred, while for moderate frequencies ($\leq 1000\,Hz$) and high frequencies, velocities and accelerations are the more used magnitudes.

For rotating machinery and particularly for bearing vibration analysis, accelerometers and tachometers (usually optical encoders) are installed on the bearing housing and in the shaft, respectively. Accelerations are commonly expressed as dimensionless quantity given as the fraction of earth gravitational field, commonly denoted by letter G. By its part, rotational speed is expressed in revolutions per minute (RPM).

In order to ensure good signal resolution and avoid aliasing problems, very high sampling rate is normally considered. If component were constantly monitored, this would lead to unmanageable amount of data. Consequently, vibrations are monitored taking signal aliquots of few seconds at specified time intervals, which often range from one second to ten minutes.

Even when having a well defined signal, vibration sensors are subject to collection of several sources of noise such as those induced by adjacent components, among others. This leads to the problem known as low signal-To-noise ratio, which means that responses related to desired component are masked or even distorted, difficulting early detection of component defects. Therefore, a signal enhancing method is needed to provide more evident information for incipient failure detection of rolling element bearings. Within existing literature several methods for signal denoising have been proposed, including Kalman filters, which are useful when the gaussian noise hypothesis applies (**Khanam et al. 2014**); wavelet filters, suitable for cases where the

underlying process is pulsed (Qiu et al. 2006), which is the case for bearings or gear faults or synchronous averaging techniques (Mcfadden and Toozhy 2000; Bechhoefer et al. 2013; Zhu et al. 2014) which relies in the fact that the underlying process is periodic and stationary, and all non-synchronous signal components can be averaged out. Independent Component Analysis (ICA) has been used alternatively in order to isolate vibration signal contributions due to some specific source while discarding the rest (Wang et al. 2014).

2.1. Time Synchronous Averaging (TSA)

Time Synchronous Averaging works under the assumption that analyzed process is periodic and all those non-synchronous signal components behave as random noise. Data values sampled from signal separated by the exact period are then averaged (Braun 2011). Any periodic component, synchronous with this period, is thus unchanged, any other will be attenuated and converge asymptotically towards zero. Formal expression synchronous average is exposed in Equation (6).

$$y(n \cdot \Delta t) = \frac{1}{N} \sum_{r=0}^{N-1} x(n\Delta t - rM\Delta t) \tag{6}$$

, where x is the input signal; y is the averaged signal; N is the number of averages; M is the number of data points inside each period of the whole signal and Δt is the inverse of sampling frequency of the signal.

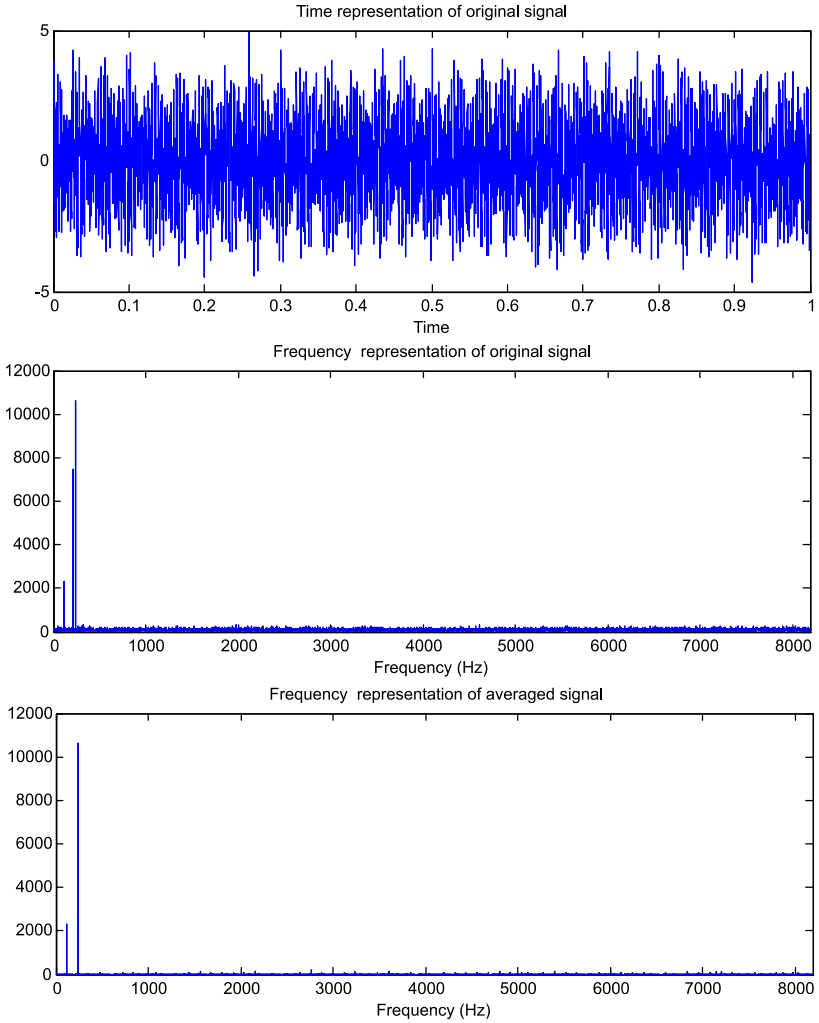

Figure 23: TSA example case over signal $s = \sum_{i=1}^{3} A_i \sin(\omega_i t + \varphi)$

Adopted method for computing TSA is based on (**Mcfadden and Toozhy 2000**) work. The computation of the synchronous average $y(t)$ of a time signal x(t) using a trigger signal having a frequency f_s is equivalent to the convolution

$$y(t) = c(t) * x(t) \tag{7}$$

, where $c(t)$ is a train of N impulses of amplitude $1/N$ (i.e. averaged by N), spaced at intervals $T_s = 1/f_s$, having the following expresion:

$$c(t) = \frac{1}{N}\sum_{n=0}^{N-1} \delta(t + nT_s) \tag{8}$$

In the frequency domain, this is equivalent to the Fourier transform $X(f)$ on the signal times the Fourier transform of train of impulses $C(f)$, i.e.

$$Y(t) = C(f) \cdot X(f) \tag{9}$$

, with $C(f)$ defined as

$$C(f) = \frac{1}{N}\frac{\sin(\pi N T_s f)}{\sin(\pi T_s f)} \tag{10}$$

which is a comb filter. Unlike in Mcfadden's paper, in which theoretical frequencies for inner race defect are exposed, in this work characteristic defect frequencies are obtained from bearing manufacturer's catalogue.

It is worthy to mention some drawbacks on the use of TSA for the bearing case that have been pointed out in (**Bechhoefer et al. 2013**). For synchronous averaging to succeed, signal have to be stationary. However, bearings are quasi-stationary, i.e. there is always some slippage respect to shaft rotation, thus causing that points selected for averaging are not separated by the same period, then reducing dramatically method effectiveness. Another con is the fact that bearings have four different characteristic frequencies: cage, ball, inner and outer race. This would require the TSA to be run four different times for each bearing which may be a limitation concerning online applications.

Nevertheless, bearings might be considered as stationary if glide over shaft remains small (which is often the case). Besides, computational cost derived from number of TSA executions is not a problem for this test case, and then TSA can be used without risk.

To show the effect of TSA algorithm an example using a synthetic signal is presented in the following figure: Original signal is created as the composition of multiple sinusoidal functions with different frequencies and amplitudes. Two of the aggregated sine functions are harmonics so, when looking for that frequency, both components are preserved while the third one is averaged out.

2.2. Vibration signature metrics

Once vibration signal has been denoised, for failure detection purposes, the information contained is compressed in a set of metrics. Pointwise analysis of vibration data is worthless since it does not provide any type of information. Each aliquot is then processed in order to extract features that will be tracked so as to infer failure patterns from them.

In this work, used metrics for vibration signal description are based on those summarized in (**Zhu et al. 2014**).

Root Mean Square (RMS)

RMS describes the energy content of the signal. It is not very sensitive to incipient fault but used to track general fault progression.

$$s_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i)^2} \tag{11}$$

Delta RMS

Delta RMS is the difference between two consequent RMS values.

$$\Delta s_{RMS}(t) = s_{RMS}(t) - s_{RMS}(t-1) \tag{12}$$

If the gear damage occurs, the vibration level will be increased more rapidly than in a normal case without gear damage (Zhu et al. 2014)

Peak value

Peak value is the maximum amplitude of the signals within a certain time interval.

$$s_{peak} = \max s \tag{13}$$

Peak-To-Peak value

Peak to peak value is the distance between the maximum amplitude and the minimum amplitude of the signal. Peak to peak is a measurement of spread in the signal.

$$s_{P2P} = s_{peak} - \min s \tag{14}$$

Signal Kurtosis

Kurtosis indicates how flat or peaked a certain distribution of values is. High kurtosis means that most values are concentrated together while low kurtosis describes a distribution with high dispersed values.

$$s_k = \frac{N \cdot \sum_{i=1}^{N}(s_i - \bar{s})^4}{(\sum_{i=1}^{N}(s_i - \bar{s})^2)^2} \tag{15}$$

Signal Skewness

Skewness is a measure of symmetry of a distribution of values.

$$s_{skew} = \frac{\sqrt{N} \cdot \sum_{i=1}^{N}(s_i - \bar{s})^3}{\left(\sqrt{\sum_{i=1}^{N}(s_i - \bar{s})^2}\right)^3} \tag{16}$$

Crest Factor

Crest factor is the ratio of the single side peak value of the input signal to the RMS level.

$$s_{CF} = \frac{s_{peak}}{s_{RMS}} \tag{17}$$

Crest factor can be used to indicate faults in an early stage. This feature is used to detect changes in the signal pattern due to impulsive vibration sources such as tooth breakage on a gear

3. Problem configuration

The second and third tests contained in the dataset are used to proof the methodology, so, in this section, we are looking for bearings outer race defects. The first bearing from the second test (Bearing 1) and the third one from the third test (Bearing 3) present this anomaly at the end of each Run-To-Failure experiment. Additionally, another bearing is considered to evaluate trained indicator on it in order to analyze its generality. We use second bearing from test 3 (Bearing 2) for this purpose as, after all runs, any fault was detected on it.

This is, in fact, a short set of cases to validate the model, then we propose two arrangements of the bearing data to evaluate the performance:

- **Configuration 1**: Bearing 1 and Bearing 3 are used to train the model as *bad* cases and bearing 2 is left to carry out the evaluation.
- **Configuration 2**: Bearing 3 and Bearing 2 are used to train the model as *bad* and *good* cases while bearing 1 is employed to perform the evaluation.

For second configuration it has to be noticed that test function for Bearing 2 has been set as a constant value equal to the penalization parameter (ν) since this bearing remains healthy along all the time interval. Configuration for the test function is summarized in Table 12.

Table 12: Test function parameters configuration for bearing test cases

Experiment	δ	p	φ	ν
Configuration 1	60	0.1	0.020001059	1.87
Configuration 2	60	0.1	0.038002012	1.87

In order to test how robust behave the proposed methodology in the presence of noise contained in the variables, features described in section 2.2 in current chapter are evaluated, both applying TSA and without the filtering technique. Then they are passed as training variables to the genetic algorithm to search suited indicators.

To perform TSA over vibration signals of selected bearings, we use as a filtering frequency (or reference frequency) the Outer Pass Defect Frequency (RPFO). This has been extracted from Rexnord manufacturer' catalogue, whose value is 236.4 Hz.

In order to evolve indicator expressions the genetic algorithm was set as described in Table 7.

4. Results and discussion

4.1. Raw vibrations

Results from the execution of the indicator search process over the first configured case are shown in Figure 24. It summarizes, in the first row, the time evolution of the best achieved model for bearing 3, bearing 1 and bearing 2, respectively. Below, the evolution of considered variables to create the model, i.e. *delta root mean square* and *peak-to-peak*, by the genetic algorithm are exposed.

At first glance, achieved results are quite far from what it was expected, given the description of the employed dataset. For Bearing 1, indicator exhibit a behaviour which is in accordance with initial hypothesis. It starts with low responses, and two days before the test was stopped and the failure isolated, it begun to raise, although somehow erratically. As it has been mentioned before, this is caused because there are not filtering functions within provided search space to the genetic programming algorithm. Hence, these prevent the algorithm from finding smoother expressions which may cancel or at least diminish noise in a natural way. In fact, as it happened with verification cases, search process involve great efforts in making indicator's output as low as possible in the region on the left of the test function, which at last causes that non zero outputs from the model be volatile.

On the contrary, behaviour of indicator when dealing with bearing 3 monitored variables is far from being understandable. Its evolution over time does not show any noticeable trend even though this bearing has been found to be in faulty state when the experiment ended. After this evidence, taken variables by the indicator model have been plotted versus operational time. Their trends where compared with each other to seek similarities between them. We found that model variables were more similar for bearings 1 and 2 rather than for bearings 1 and 3, as it was awaited.

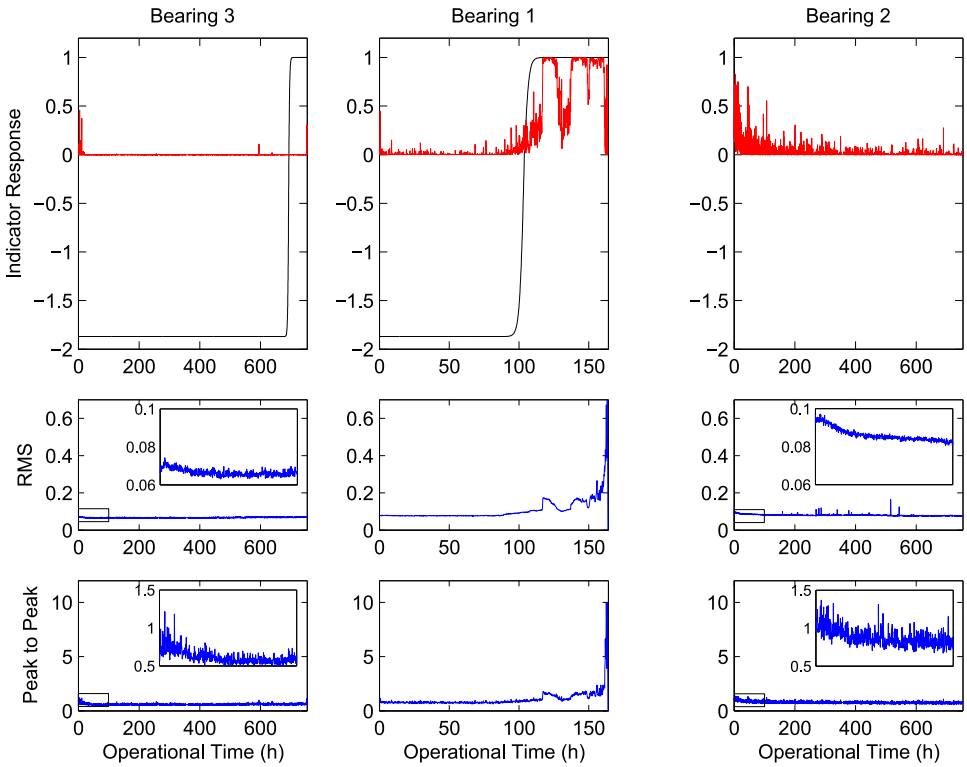

Figure 24: Indicators response and model variables evolution

It can be seen in Figure 24, that bearing 2, which was supposed to be healthy, when its evaluated , the indicator trend seems to that encountered for bearing 3. Indicator evolution for bearing 2 shows a more spiky form than bearing 3. This occurs since bearing 2 was included in the training cases. Thus, algorithm has devoted efforts in keep low its response regarding imposed penalisation conditions, leading to overfitting. As it is represented in Figure 24, indicator response experiment a decreasing trend in its value at the beginning of the monitoring period, corresponding to a similar evolution of model selected variables. After that period of time, evolution of both indicator and variables are flat until the end of the operational time.

Used database was checked in order to discard mislabelling issues when it was created. We did not find any other bearing exhibiting the same variable patterns, so we are prone to think that bearing 3 has been incorrectly considered as failed.

Chapter 5
Conclusions and future work

In this work we have proposed a methodology for automatic learning of indicators for failure detection. Key goals of developed technique address automatic data selection in order to create failure detection models, deploy human interpretable models to favour interaction with their outputs, and allow learning in contexts having partial information of studied event.

The methodology is aimed to summarize health information of a certain component based on monitored signals to allow decision making in relation to maintenance tasks. To accomplish this objective we do not perform any a priori labelling of operational data. Instead we define a reasonable interval, starting backwards from the studied undesired event, in which we expect that degradation process has already began to be noticeable in monitored signals. However, we do not strongly constrain the evolution of the indicator expression to have a specific shape, although those indicators exhibiting saturation trends are rewarded. Under this proposition we are also interested in discriminate other failure modes different to the one being under the scope (i.e. specificity). A comprehensive review on the state of the art has been developed in order to look for different approaches to failure detection task. Among all consulted references, we have not encountered yet a work presenting a method matching the central idea described in this proposal. Hence, we think that this offers an opportunity to address in a different way the failure detection task.

Indicators search is performed by means of genetic programming. Although there are many alternatives to find appropriate measures for degradation quantification, the herein proposed offers the possibility to further process the achieved model since it is actually an manageable expression. Hence, model may be explored by syntactic and/or sensitivity analysis to acquire information of relevant variables and to gain further insights about failure mechanisms. The genetic algorithm is driven through generations by an objective function which actually constitutes the kernel of presented methodology.

Each expression generated by the genetic algorithm is scored regarding its ability to respond against the desired failure mode. The rest of the time, indicator response should be null or remain as low as possible (see Figure 12). The, so called, test function is introduced to drive the search process in order find indicators which meet desired behaviour. In test function, information of different kind is accounted. On the one hand it allows to summarize expert knowledge or desired detection horizon (time of early detection of an emerging failure before it becomes a fault) through δ parameter. After this time, every non null response of the indicator will be rewarded in accordance to its magnitude. Besides, through ν parameter we can control the penalisation over solutions that only partially fulfils imposed requirements. We found that its value should be linked to the relation of number of samples lying in the detection region (i.e. times higher than δ) and the normality region. But also it can be set under the approach of cost-based learning, depending on the relative importance of false

positives (also called false alarms) and false negatives (or missing alarms). The third tuning parameter, $\varphi(p)$, relax the interface between normality operation and detection region. Then, all time samples lying around δ are not completely penalized but neither totally rewarded.

Some test cases have been run to test the proposed methodology. At first we use a synthetic database in which degradation pattern is modelled as the complementary of a negative exponential function. This pattern is sometimes hidden in one specific variable or in a combination of them. These cases were built to learn how test function parameters influence the search process and how methodology copes with problems of different nature. Besides, we tried to approach a real application studying common failure mode suffered by bearings.

Regarding performed experiments we extract the following conclusions:

- Proposed technique is able to learn expressive indicators from data having the form of those encountered in real world applications. It serves from few known facts (e.g. the instant of failure or the substitution of a certain component) to set a guide function which just stipulate how to reward or penalize a potential solution.
- Using genetic programming to evolve analytic expressions to model asset degradation yields human interpretable models. This allows the posterior simplification of obtained expressions and get information of relevant signals informing about malfunctioning.
- Given obtained results presented in Figure 17 or even in Figure 24, it is shown that the search process might be successively improved by iteratively refining test function parameters based on outputs given by obtained indicator.
- As it has been observed when working with bearing dataset, as far as it has been tested, the methodology shows robustness when in the training set is included unreliable information.
- Although its averaged computational cost is higher compared with other learning techniques such as Artificial Neural Networks (which are widely extended in the existing literature to obtain detection models), genetic programming has not exhibited prohibitively training times. It is also fair to mention that training sets were not too big.

Although some interesting and encouraging results has been achieved in performed case studies, the amount of experimentation is still insufficient to guarantee its fine behaviour when dealing with other cases. In fact, we have not been able to submit thoroughly conclusions about method performance on bearing dataset due to issues which emerged when the test was run. Therefore, we now describe some identified future research directions to further develop the presented proposal.

- A factorial experiment for test function parameters has to be planned in order to gain further insights on search properties of this method. Specially for ν parameter, whose specification is critical since if too low values are used it won't be able to isolate wanted failure mode or indicator evolution will be too noisy. On the contrary, if too high values are considered, the obtained indicators will focus in cancelling every evidence of noise in the normal region while making the indicator to rise erratically in the detection region.
- We have not been able to prevent overfitting issues when training the model. Until more failure cases are included within the training dataset we won't be able to conclude if the proposed method is inherently prone to overfitting or, conversely, if it is only a matter of lack of data.
- Regarding the genetic programming algorithm, it is worthy to devote some time to study and implement techniques for bloating control. It has been seen when working with verification cases, as the searching process evolves

there is a marginal improvement on solution fitness whilst complexity keeps growing.

- To avoid erratically behaviour which appeared in the most complex cases, it would be worthy to emphasize in the configuration of the search space of the genetic algorithm. A few alternatives that might be considered are to add mathematical filters to smooth the evolution of the indicator; treat input variables before they are included in the dataset used by the genetic algorithm (this is the approach that was chosen in this work, but obtained results on the provided dataset discouraged new experiments) or add to existing dataset, variables from time $t-1, t-2, \ldots, t-T$ in order to allow tendency or frequency operations. Notice that in this work just a basic set of functions has been used because the current implementation of the genetic programming algorithm requires *closure* constrains to be satisfied (see section 2.l.l, from Chapter 3).

References

Arabian-Hoseynabadi, H., H. Oraee, and P.J. Tavner. 2010. 'Failure Modes and Effects Analysis (FMEA) for Wind Turbines.' *Int. J. Electr. Power Energy Syst.* 32 (7). Elsevier Ltd: 817–24.

Baraldi, P, A Cammi, F Mangili, and E E Zio. 2010. 'Local Fusion of an Ensemble of Models for the Reconstruction of Faulty Signals.' *Nucl. Sci. IEEE Trans.* 57 (2): 793–806.

Baraldi, P, G Gola, E Zio, D Roverso, and M Hoffmann. 2011. 'A Randomized Model Ensemble Approach for Reconstructing Signals from Faulty Sensors.' *Expert Syst. Appl.* 38 (8): 9211–24.

Baraldi, Piero, Antonio Cammi, Francesca Mangili, and Enrico Zio. 2010. 'An Ensemble Approach to Sensor Fault Detection and Signal Reconstruction for Nuclear System Control.' *Ann. Nucl. Energy* 37 (6): 778–90.

Bechhoefer, Eric, Brandon Van Hecke, and David He. 2013. 'Processing for Improved Spectral Analysis.' *Annu. Conf. Progn. Heal. Manag. Soc.*, 1–6.

Braun, S. 2011. 'The Synchronous (time Domain) Average Revisited.' *Mech. Syst. Signal Process.* 25 (4). Elsevier: 1087–1102.

Chen, Bindi, Donatella Zappalá, Christopher J Crabtree, and Peter J Tavner. 2014. *Survey of Commercially Available SCADA Data Analysis Tools for Wind Turbine Health Monitoring*. Vol. 44.

Crabtree, C J, D Zappalá, and P J Tavner. 2014. *Survey of Commercially Available Condition Monitoring Systems for Wind Turbines.* Vol. 44. Durham University School of Engineering and Computing Sciences and the SUPERGEN Wind Energy Technologies Consortium.

Das, M. K., S. C. Panja, S. Chowdhury, S. P. Chowdhury, and a. I. Elombo. 2011. 'Expert-Based FMEA of Wind Turbine System.' *IEEE Int. Conf. Ind. Eng. Eng. Manag.*, 1582–85.

Dinmohammadi, F, and M Shafiee. 2013. 'A Fuzzy-FMEA Risk Assessment Approach for Offshore Wind Turbines.' *Int. J. Progn. Heal. Manag.*, no. Special Issue on Wind Turbines PHM: 1–10.

Doucet, Arnaud, Nando Freitas, and Neil Gordon. 2001. 'An Introduction to Sequential Monte Carlo Methods.' In *Seq. Monte Carlo Methods Pract.*, 3–14. New York, NY: Springer New York.

Frost, Susan A, Kai Goebel, and Léo Obrecht. 2013. 'Integrating Structural Health Management with Contingency Control for Wind Turbines.' *Int. J. Progn. Heal. Manag.*, no. Special Issue on Wind Turbines PHM: 1–10.

Garcia, Mari Cruz, Miguel a. Sanz-Bobi, and Javier del Pico. 2006. 'SIMAP: Intelligent System for Predictive Maintenance.' *Comput. Ind.* 57 (6): 552–68.

Gill, Simon, Bruce Stephen, and Stuart Galloway. 2012. 'Wind Turbine Condition Assessment Through Power Curve Copula Modeling.' *IEEE Trans. Sustain. Energy* 3 (1): 94–101.

Global Wind Energy Council. 2014. *Global Wind Energy Outlook*.

Godwin, Jamie L, and Peter Matthews. 2013. 'Classification and Detection of Wind Turbine Pitch Faults Through SCADA Data Analysis.' *Int. J. Progn. Heal. Manag.*, no. Special Issue on Wind Turbines PHM.

Guo, Peng, David Infield, and Xiyun Yang. 2012. 'Wind Turbine Generator Condition-Monitoring Using Temperature Trend Analysis.' *Sustain. Energy, IEEE Trans.* 3 (1): 124–33.

Hussain, Sajid, and Hossam A Gabbar. 2013. 'Vibration Analysis and Time Series Prediction for Wind Turbine Gearbox Prognostics.' *Int. J. Progn. Heal. Manag.*, no. Special Issue on Wind Turbines PHM: 1–11.

Kahrobaee, Salman, and Sohrab Asgarpoor. 2011. 'Risk-Based Failure Mode and Effect Analysis for Wind Turbines (RB-FMEA).' *2011 North Am. Power Symp.*, 1–7.

Khanam, Sidra, J. K. Dutt, and N. Tandon. 2014. 'Extracting Rolling Element Bearing Faults From Noisy Vibration Signal Using Kalman Filter.' *J. Vib. Acoust.* 136 (3): 031008.

Kost, Christoph, Johannes Mayer, Jessica Thomsen, Niklas Hartmann, Charlotte Senkpiel, Simon Philipps, Sebastian Nold, Simon Lude, Noha Saad, and Thomas Schlegl. 2013. *Levelized Cost of Electricity Renewable Energy Technologies*.

Koza, John R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Mass: MIT Press.

Kusiak, Andrew, and Anoop Verma. 2010. 'A Data-Driven Approach for Monitoring Blade Pitch Faults in Wind Turbines.' *IEEE Trans. Sustain. Energy* 2 (1): 87–96.

Kusiak, Andrew, and Anoop Verma. 2012. 'Analyzing Bearing Faults in Wind Turbines: A Data-Mining Approach.' *Renew. Energy* 48. Elsevier Ltd: 110–16.

Lee, Jay, Hai Qiu, Gang Yu, Jing Lin, and Rexnord Technical Services. 2007. 'Bearing Data Set.' *NASA Ames Progn. Data Repos.* Moffett Field, CA.

Luke, Sean, and Liviu Panait. 2002. 'Lexicographic Parsimony Pressure.' *GECCO 2002 Proc. Genet. Evol. Comput. Conf.*, 829–36.

Manohar, A., and F. Lanza di Scalea. 2013. 'Detection of Defects in Wind Turbine Composite Blades Using Statistically Enhanced Lock-In Thermography.' *Struct. Heal. Monit.* 12 (5–6): 566–74.

Mcfadden, P.D., and M.M. Toozhy. 2000. 'Application of Synchronous Averaging To Vibration Monitoring of Rolling Element Bearings.' *Mech. Syst. Signal Process.* 14: 891–906.

Osadciw, Lisa Ann, Yanjun Yan, Xiang Ye, Glen Benson, and Eric White. 2010. 'Wind Turbine Diagnostics Based on Power Curve Using Particle Swarm Optimization.' In *Wind Power Syst.*, 151–65.

Plumley, Charles Edward, Graeme Wilson, Andrew Kenyon, Francis Quail, and Athena Zitrou. 2012. 'Diagnostics and Prognostics Utilising Dynamic Bayesian Networks Applied to a Wind Turbine Gearbox.' In *Int. Conf. Cond. Monit. Mach. Fail. Prev. Technol.*, 12. London.

Poli, Riccardo, and W B Langdon. 1995. 'On the Search Properties of Different Crossover Operators in Genetic Programming.'

Poli, Riccardo, William B Langdon, and Nicholas F Mcphee. 2008. *A Field Guide to Genetic Programming*.

Qiu, Hai, Jay Lee, Jing Lin, and Gang Yu. 2006. 'Wavelet Filter-Based Weak Signature Detection Method and Its Application on Rolling Element Bearing Prognostics.' *J. Sound Vib.* 289: 1066–90.

Schlechtingen, Meik, Ilmar Ferreira Santos, and Sofiane Achiche. 2013. 'Wind Turbine Condition Monitoring Based on SCADA Data Using Normal Behavior Models. Part 1: System Description.' *Appl. Soft Comput.* 13 (1). Elsevier B.V.: 259–70.

Searson, Dominic P. 2015. 'GPTIPS 2: An Open-Source Software Platform for Symbolic Data Mining.' In *Handb. Genet. Program. Appl.*, edited by Amir H. Gandomi, Amir H. Alavi, and Conor Ryan, 1–23. Springer New York.

Shafiee, Mahmood, and Fateme Dinmohammadi. 2014. 'An FMEA-Based Risk Assessment Approach for Wind Turbine Systems: A Comparative Study of Onshore and Offshore.' *Energies* 7: 619–42.

Smits, Guido, and Mark Kotanchek. 2004. 'Pareto-Front Exploitation in Symbolic Regression.' *Genet. Program. Theory Pract. {II}*, 283–99.

Su, Chun, and Jinyun Shen. 2013. 'A Novel Multi-Hidden Semi-Markov Model for Degradation State Identification and Remaining Useful Life Estimation.' *Qual. Reliab. Eng. Int.* 29 (8): 1181–92.

Tchakoua, Pierre, René Wamkeue, Mohand Ouhrouche, Fouad Slaoui-Hasnaoui, Tommy Tameghe, and Gabriel Ekemb. 2014. 'Wind Turbine Condition Monitoring: State-of-the-Art Review, New Trends, and Future Challenges.' *Energies* 7 (4): 2595–2630.

Uy, Nguyen Quang, Michael O Neill, Nguyen Xuan Hoai, Bob Mckay, and Edgar Galván-López. 2010. 'Semantic Similarity Based Crossover in GP: The Case for Real-Valued Function Regression.' In *Artifical Evol.*, edited by Pierre Collet, Nicolas Monmarché, Pierrick Legrand, Marc Schoenauer, and Evelyne Lutton, 5975:170–81. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg.

Valis, D., L. Zak, and O. Pokora. 2014. 'Contribution to System Failure Occurrence Prediction and to System Remaining Useful Life Estimation Based on Oil Field Data.' *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* 229 (1): 36–45.

Verma, Anoop, and Andrew Kusiak. 2012. 'Fault Monitoring of Wind Turbine Generator Brushes: A Data-Mining Approach.' *J. Sol. Energy Eng.* 134 (2): 1–9.

Walker, Matthew. 2001. 'Introduction to Genetic Programming.'

Wang, Jinjiang, Robert X Gao, and Ruqiang Yan. 2014. 'Integration of EEMD and ICA for Wind Turbine Gearbox Diagnosis.' *Wind Energy* 17 (5): 757–73.

Yan, Yonglong, Jian Li, and David Gao. 2014. 'Condition Parameter Modeling for Anomaly Detection in Wind Turbines.' *Energies* 7 (5): 3104–20.

Yang, Wenxian, Richard Court, and Jiesheng Jiang. 2013. 'Wind Turbine Condition Monitoring by the Approach of SCADA Data Analysis.' *Renew. Energy* 53 (May). Elsevier Ltd: 365–76.

Yang, Wenxian, P. J. Tavner, and M. Wilkinson. 2008. 'Condition Monitoring and Fault Diagnosis of a Wind Turbine with a Synchronous Generator Using Wavelet Transforms.' In *4th Int. Conf. Power Electron. Mach. Drives 2008*, 6–10.

Zhu, Junda, Tom Nostrand, Cody Spiegel, and Brogan Morton. 2014. 'Survey of Condition Indicators for Condition Monitoring Systems.' *Annu. Conf. Progn. Heal. Manag. Soc.* 5: 1–13.

Zhu, Junda, Jae M Yoon, David He, Yongzhi Qu, and Eric Bechhoefer. 2013. 'Lubrication Oil Condition Monitoring and Remaining Useful Life Prediction with Particle Filtering.' *Int. J. Progn. Heal. Manag.*, no. Special Issue on Wind Turbines PHM: 1–15.

Appendix I

This appendix aims to complete section 4 of chapter 3 by presenting additional information of the results obtained for experiments IC-002, IC-004 and IC-005. For each experiment, the evolution with time of the variables is depicted. Then, the population obtained in the last generation and the frequency of appearance of the variables of its 5% best individuals are presented for all of the l5 runs.

Regarding Pareto Fronts exposed in Figure 26, Figure 29 and Figure 32, It can be seen that, generally, increasing complexity of model marginally improves the fitness value. On the other hand, the histograms presented in Figure 27, Figure 30 and Figure 33 reveal the ability of the proposed methodology to identify the significant variables (x_4 or x_4 and x_5) as they are the most frequent variables in the final expressions. Additionally, the reduction in the use of noise terms and non primary failure variables (x_3) when several dataset are used for training is manifested comparing the histograms corresponding to IC-002 and IC-005.

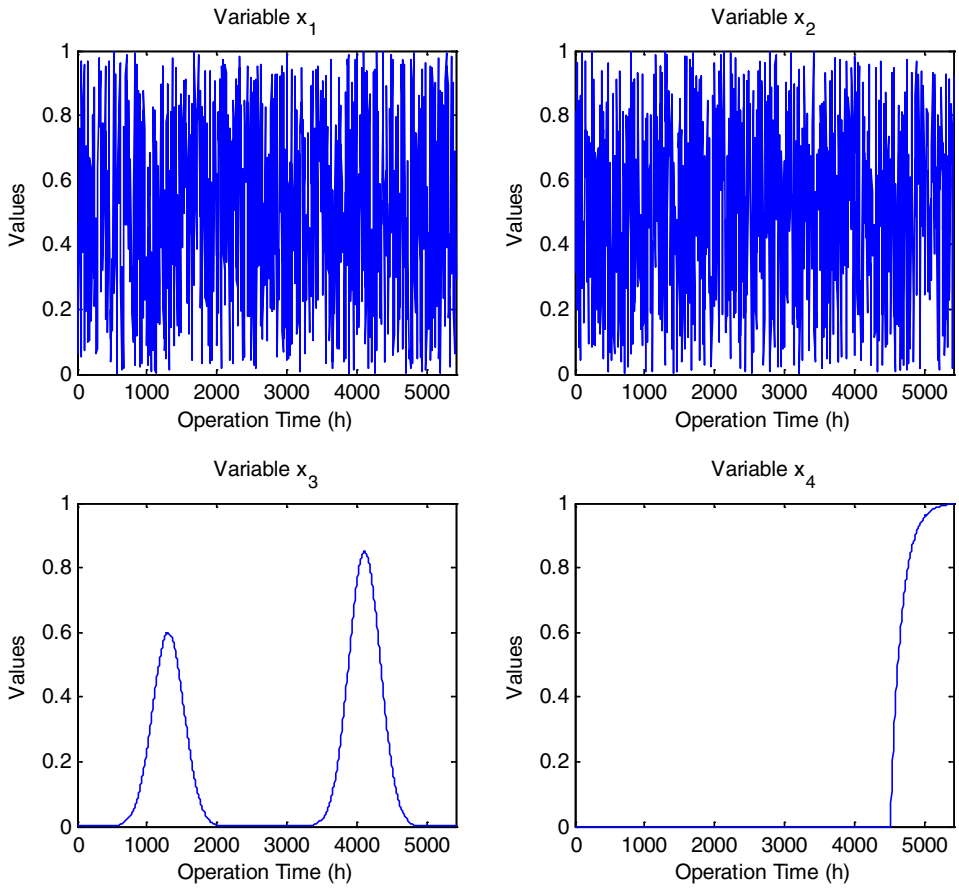

Figure 25: Time evolution of the variables involved in experiment IC-002.

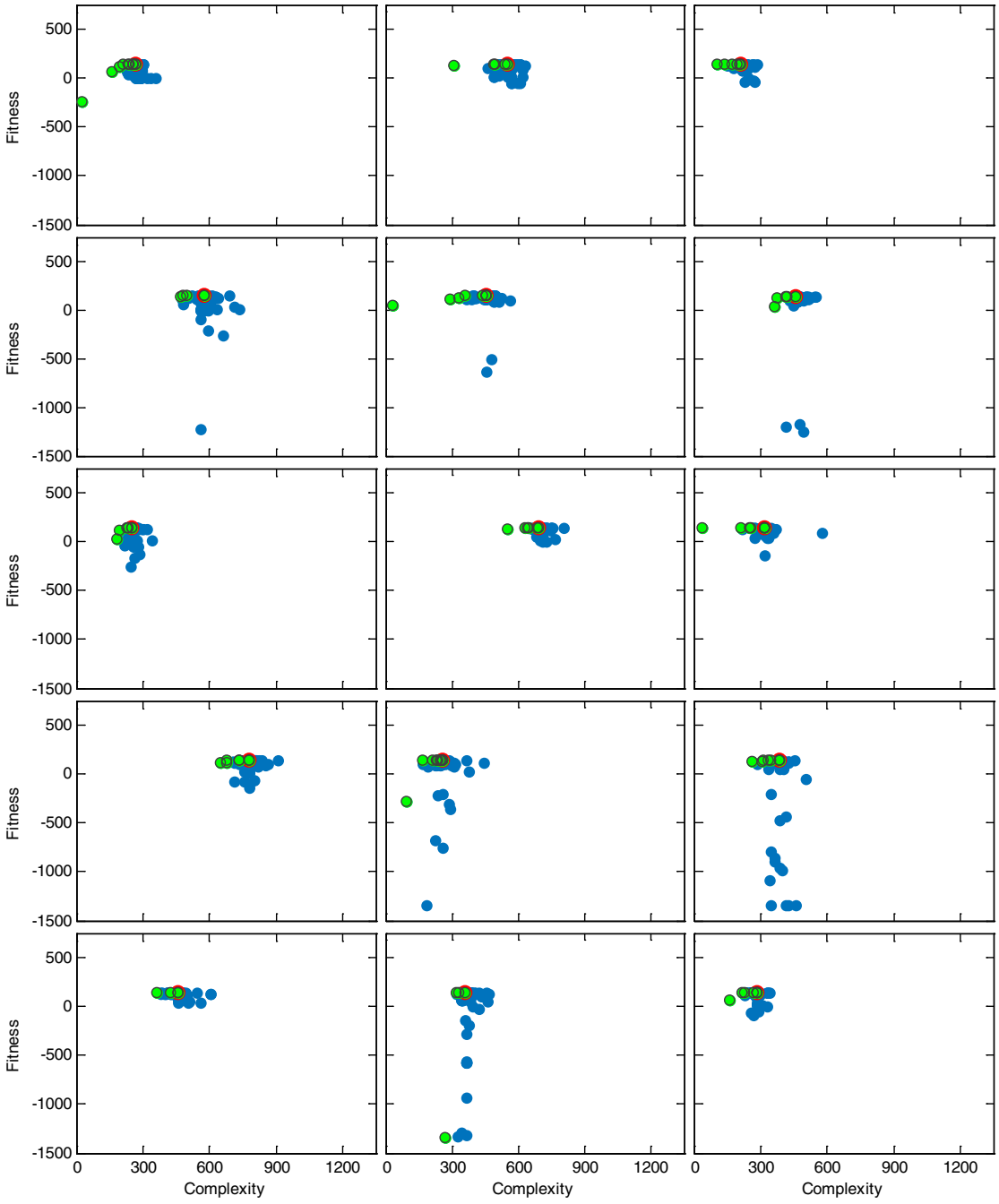

Figure 26: Population of the last generation for each run of experiment IC-002. Red dot: best individual. Green dots: Pareto Front.

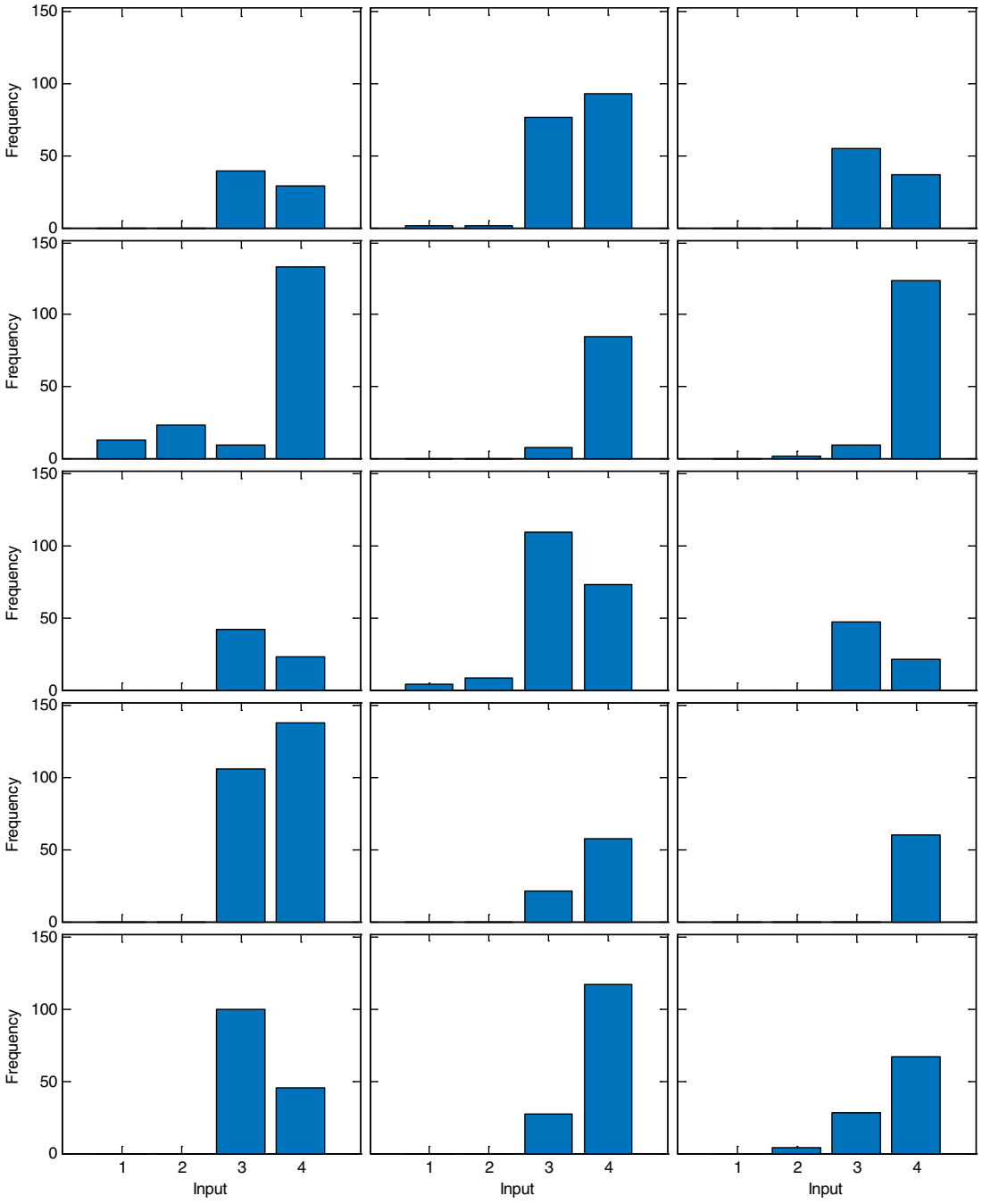

Figure 27: Variable appareance in the 5% best individuals of the last generations for each run of experiment IC-002.

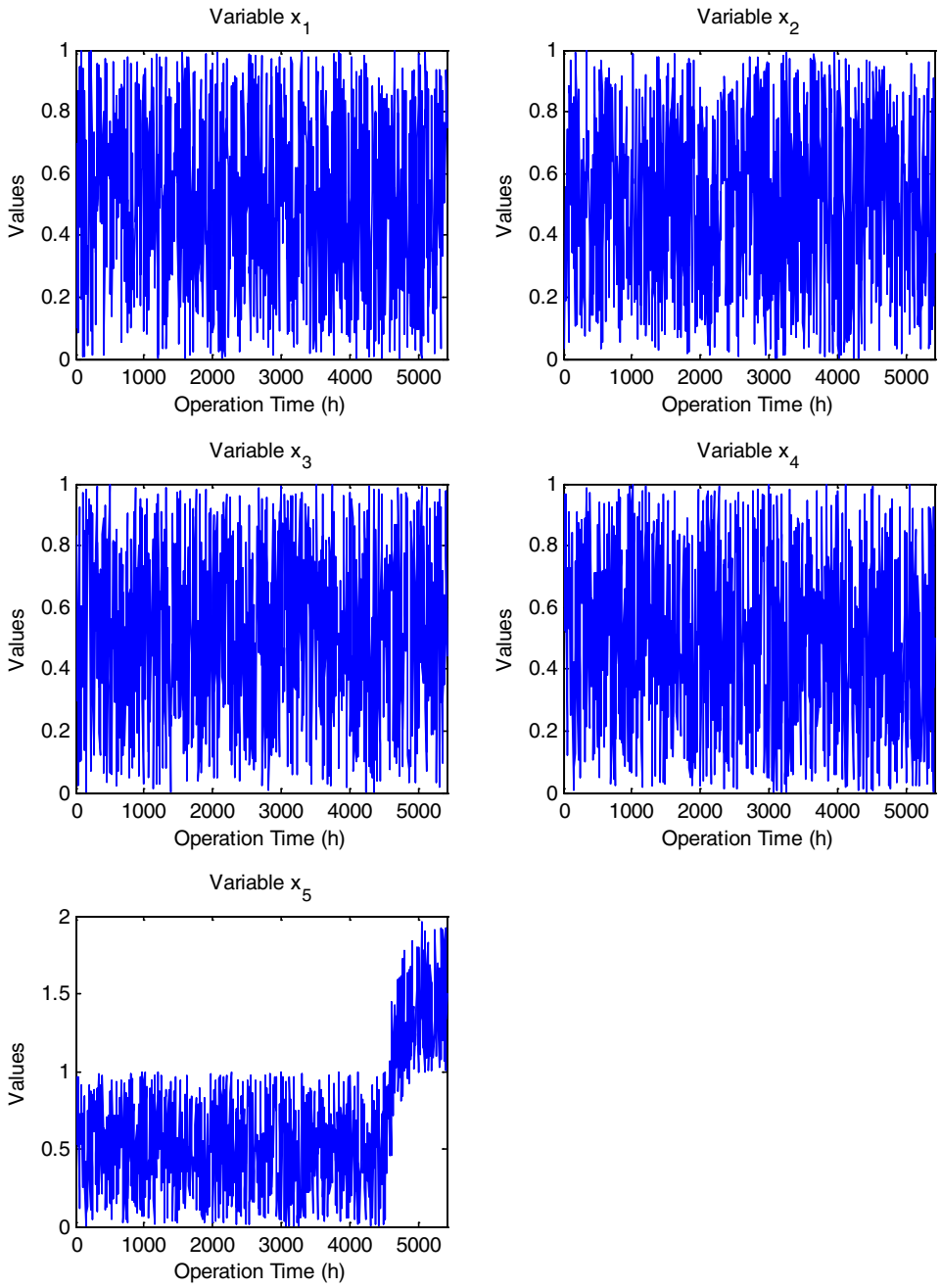

Figure 28: Time evolution of the variables involved in experiment IC-004.

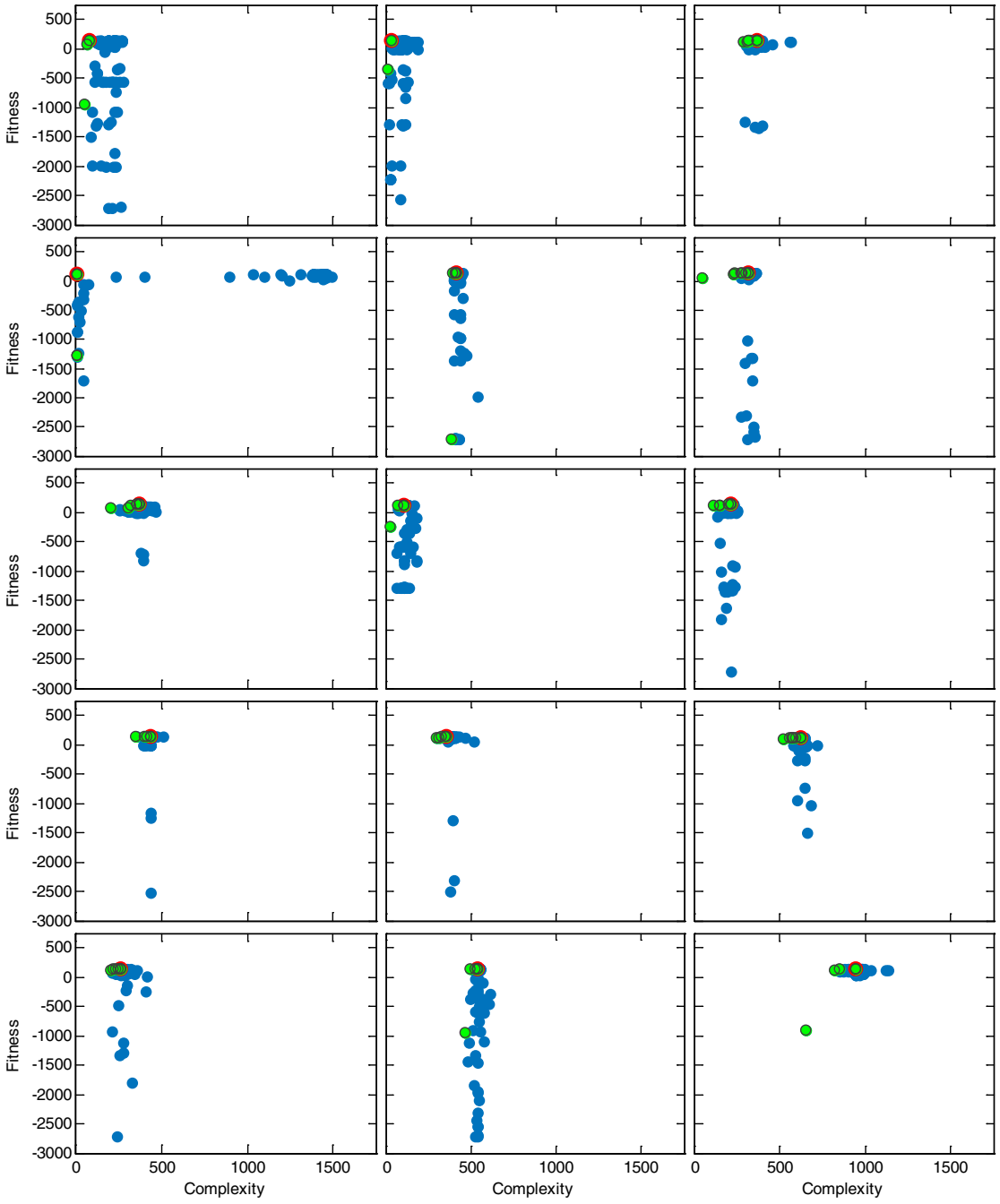

Figure 29: Population of the last generation for each run of experiment IC-004. Red dot: best individual. Green dots: Pareto Front.

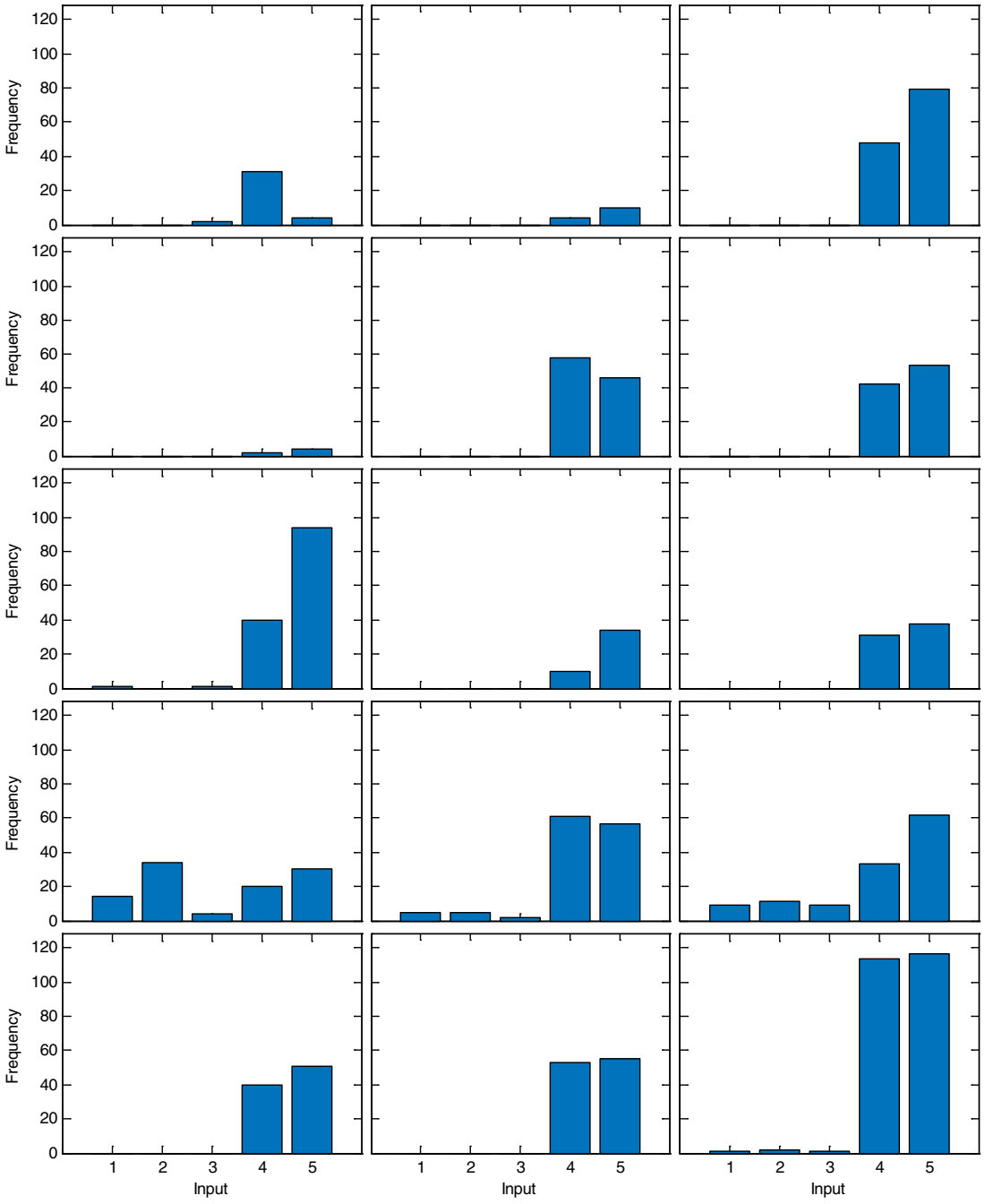

Figure 30: Variable appareance in the 5% best individuals of the last generations for each run of experiment IC-004.

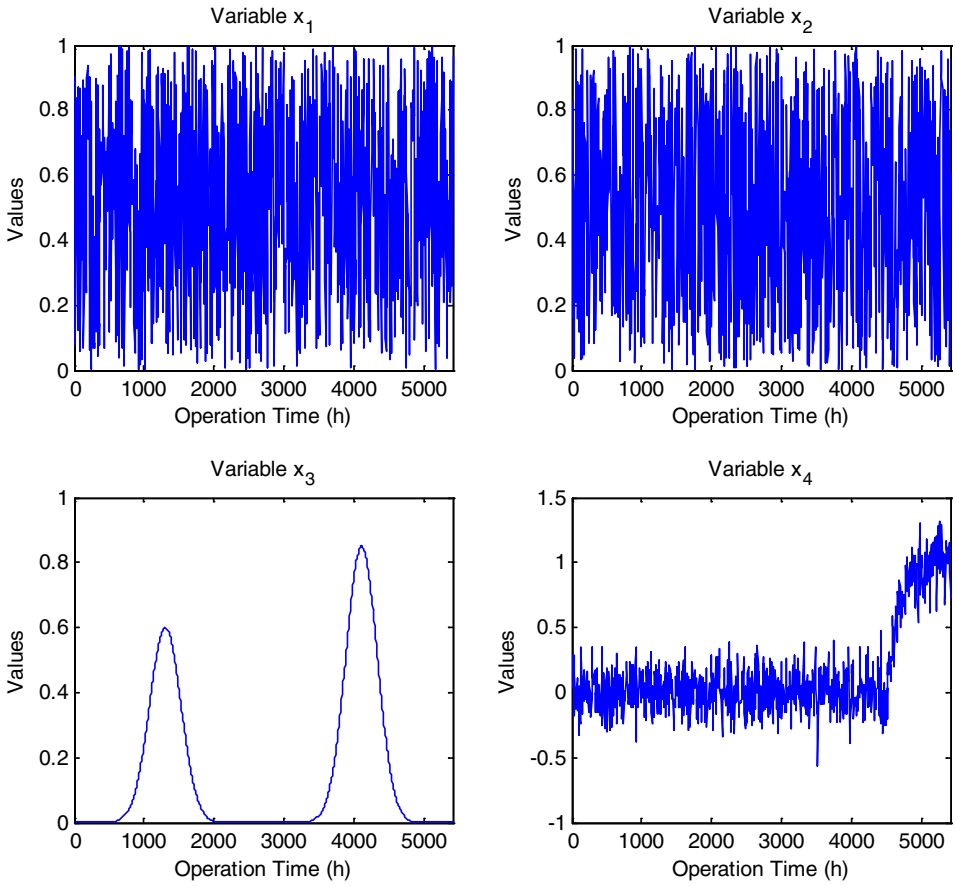

Figure 31: Time evolution of the variables involved in experiment IC-005.

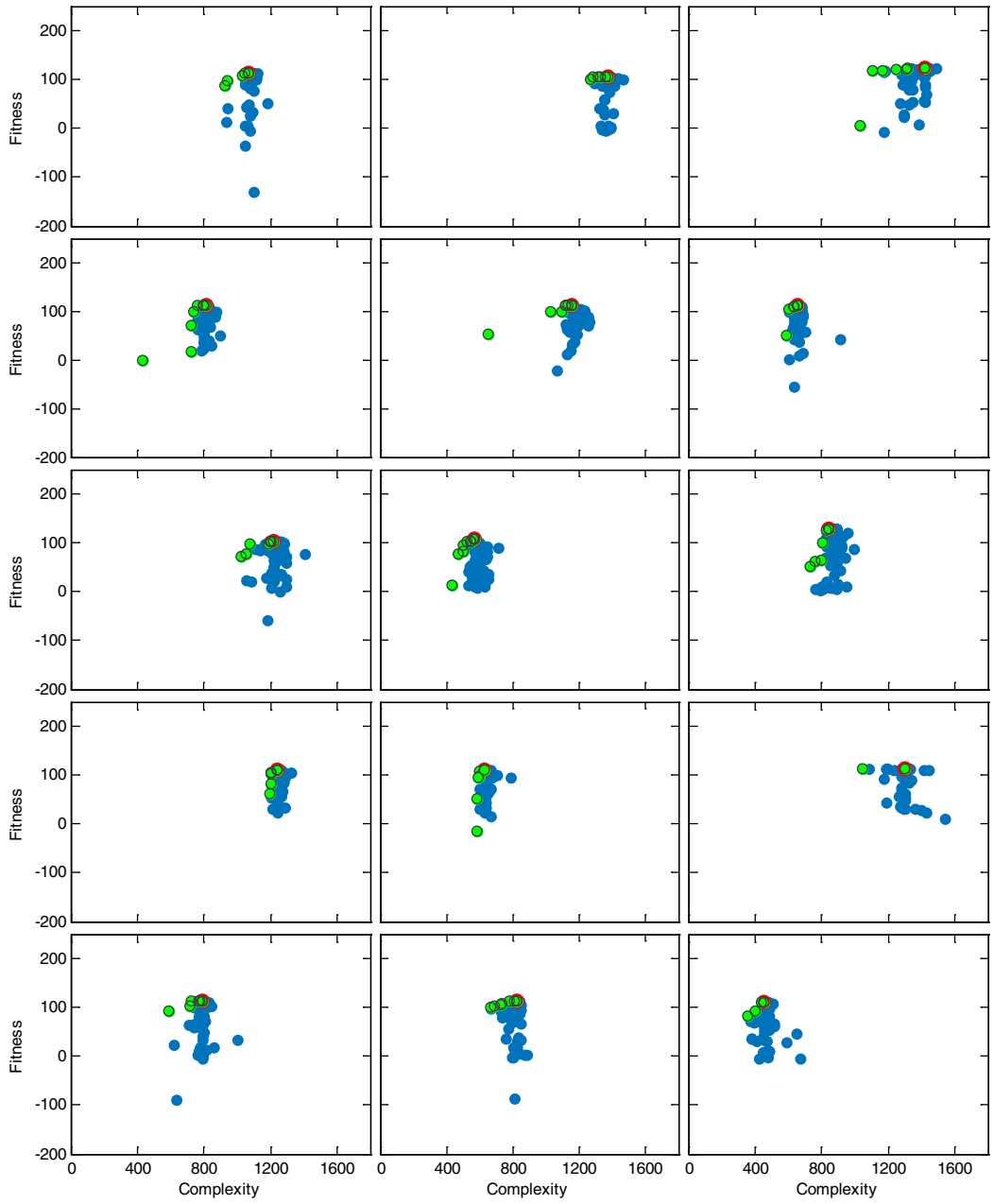

Figure 32: Population of the last generation for each run of experiment IC-005. Red dot: best individual. Green dots: Pareto Front.

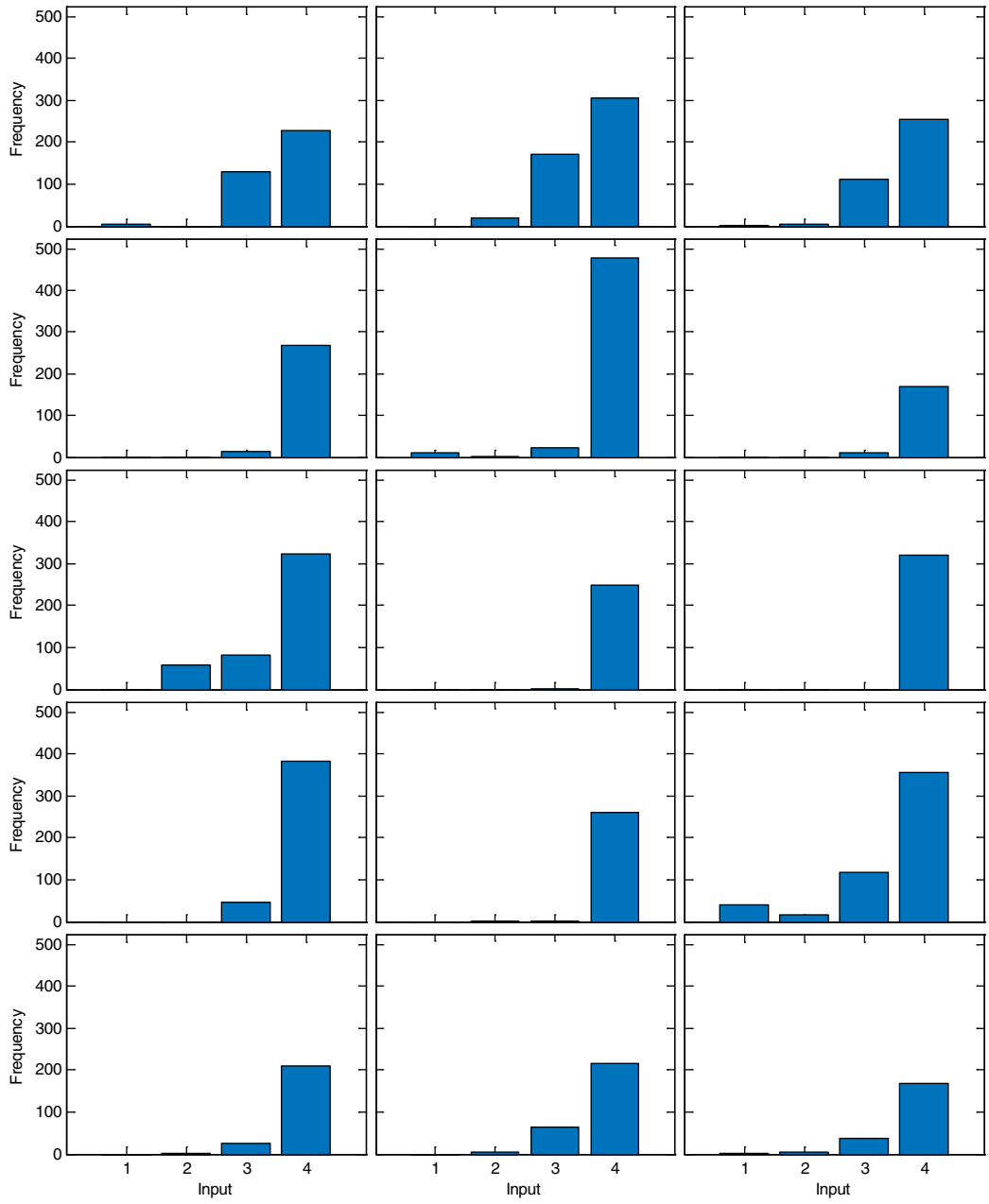

Figure 33: Variable appareance in the 5% best individuals of the last generations for each run of experiment IC-005.

Appendix II

In order to illustrate the results of the proposed methodology, this appendix presents the expressions obtained for two out of the 15 runs for experiments IC-002, IC-004 and IC-005. Both the models which give the highest fitness and the smallest complexity are selected from the best individuals of each final population.

IC-002: Higher fitness expression (fitness=148.85, complexity=210)

$$I(\boldsymbol{x},t) = x_4 \cdot e^{(2\cdot e^{x_3} - x_4 - 2\cdot x_3)} \cdot (x_4 + e^{e^{e^{(x_3 - 2\cdot x_4)}}} + x_3 \cdot e^{(x_4 + 4\cdot x_3^2)})$$

IC-002: Smallest complexity expression (fitness=148.85, complexity=210)

$$I(\boldsymbol{x},t) = x_4 \cdot e^{(2\cdot e^{x_3} - x_4 - 2\cdot x_3)} \cdot (x_4 + e^{e^{e^{(x_3 - 2\cdot x_4)}}} + x_3 \cdot e^{(x_4 + 4\cdot x_3^2)})$$

IC-004: Higher fitness expression (fitness=150.08 ,complexity=431)

$$I(\boldsymbol{x},t) = e^{e^{(e^{\left(3.9596\cdot 10^2 \cdot e^{x_2}\cdot(x_4 - x_5)\right)}\cdot(2.5875\cdot x_2 + e^{x_2} - 7.9452)\cdot(x_1 - 3.5875\cdot x_2 + 98.293\cdot x_4 + 7.8347)\cdot\left(x_1 - x_3 - 2\cdot x_4 + x_5 + e^{-x_1} + 9.7451\cdot 10^2\cdot x_5^2\right)})}$$

IC-004:Smallest complexity expression (fitness= 124.93 , complexity=5)

$$I(\boldsymbol{x},t) = x_4 - x_5$$

IC-005: Higher fitness expression (fitness=127.23, complexity=844)

$$\begin{aligned} I(\boldsymbol{x},t) = {} & 2.8086 \cdot 10^{-14} \cdot x_4^5 \cdot e^{(-6.3893\cdot x_4^2)} \cdot e^{(-6.3454\cdot x_4^2)} \cdot e^{(2\cdot x_4)} \cdot e^{(4\cdot x_4)} \cdot e^{e^{(4\cdot x_4 - 6.3454\cdot x_4^2)}} \cdot e^{(4\cdot x_4^3)} \\ & \cdot e^{(27\cdot x_4^3 \cdot e^{(-6.3454\cdot x_4^2)} \cdot e^{(-6.1709\cdot x_4^2)} \cdot e^{(4\cdot x_4)})} \cdot \left(2 \cdot x_4 + x_4^2 \cdot (2 \cdot x_4 - 6.3893)\right) \\ & \cdot \left(x_4 + 3.6664 \cdot 10^{-6} \cdot e^{x_4} + 4.3654 \cdot 10^{-6} \cdot e^{(3\cdot x_4)} \cdot e^{(3\cdot x_4^3)}\right) \cdot (4 \cdot x_4 + 2 \cdot e^{x_4} \\ & - 6.1709 \cdot x_4^2 - 12.342) \end{aligned}$$

IC-005: Smallest complexity expression (fitness=111.28, complexity=454)

$$\begin{aligned} I(\boldsymbol{x},t) = {} & x_4^3 \cdot e^{(x_4^2\cdot(e^{x_4} - x_4\cdot(x_3 - x_4)^2)\cdot(3\cdot x_4 - 2\cdot x_4^2)\cdot(x_3 - x_4))} \cdot (2 \cdot x_4 + 0.2961) \\ & \cdot (2 \cdot x_4 - x_3 + x_4^2 \cdot (2 * x_3 - x_4)) \cdot \left(2 \cdot x_4 + e^{(x_4 - x_3)} - x_4^2 - x_4^3\right) \end{aligned}$$