6-2012

Intelligent Data Fusion for Applied Decision Support

Xiang Ye
Syracuse University

Abstract

Data fusion technologies are widely applied to support a real-time decision making in complicated and dynamically changing environments. Due to the complexity in the problem domain, artificial intelligent algorithms, such as Bayesian inference and particle swarm optimization, are employed to make the decision support system more adaptive and cognitive. This dissertation proposes a new data fusion model with an intelligent mechanism adding decision feedback to the system in real-time, and implements this intelligent data fusion model in two real-world applications.

The first application is designing a new sensor management system for a real-world and highly dynamic air traffic control problem. The main objective of sensor management is to schedule discrete-time, two-way communications between sensors and transponder-equipped aircraft over a given coverage area. Decisions regarding allocation of sensor resources are made to improve the efficiency of sensors and communications, simultaneously. For the proposed design, its loop nature takes account the effect of the current sensor model into the next scheduling interval, which makes the sensor management system able to respond to the dynamically changing environment in real-time. Moreover, the system design uses a Bayesian network as the mission manager to come up with operating requirements for each region every scheduling interval, so that the system efficiently balances the allocation of sensor resources according to different region priorities. As one of this dissertation's contribution in the area of Bayesian inferencing, the resulting Bayesian mission manager is shown to demonstrate significant performance improvements in resource usage for prioritized regions such as a runway in the air traffic control application for airport surfaces. This research work was supported by Sensis Corporation, conducted in a collaborative fashion (1).

Due to wind's importance as a renewable energy resource, the second application is designing an intelligent data-driven approach to monitor the wind turbine performance in real-time by fusing multiple

types of maintenance tests, and detect the turbine failures by tracking the turbine maintenance statistics. The current focus has been on building wind farms without much effort towards the optimization of wind farm management. Also, under performing or faulty turbines cause huge losses in revenue as the existing wind farms age. Automated monitoring for maintenance and optimizing of wind farm operations will be a key element in the transition of wind power from an alternative energy form to a primary form. Early detection and prediction of catastrophic failures helps prevent major maintenance costs from occurring as well. I develop multiple tests on several important turbine performance variables, such as generated power, rotor speed, pitch angle, and wind speed difference. Wind speed differences are particularly effective in the detection of anemometer failures, which is a very common maintenance issue that greatly impacts power production yet can produce misleading symptoms. To improve the detection accuracy of this wind speed difference test, I discuss a new method to determine the decision boundary between the normal and abnormal states using a particle swarm optimization (PSO) algorithm. All the test results are fused to reach a final conclusion, which describes the turbine working status at the current time. Then, Bayesian inference is applied to identify potential failures with a percentage certainty by monitoring the abnormal status changes. This approach is adaptable to each turbine automatically, and is advantageous in its data-driven nature to monitor a large wind farm. This approach's results have verified the effectiveness of detecting turbine failures early, especially for anemometer failures. This research work was collaborated with AWS Truepower, Inc. (2).

INTELLIGENT DATA FUSION FOR APPLIED DECISION SUPPORT

By

Xiang Ye

M. S. in Electrical Engineering, Syracuse University, 2009
M. S. in Engineering Management, Pittsburg State University, 2006
B. S. in Information Engineering, Wuhan University of Technology, China, 2005

Dissertation

Submitted in partial fulfillment of the requirements for the
degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate School of Syracuse University

June 2012

Dedication

To my Dad, Mom and Sister,
who support me all the time,
for their endless encouragement and love.
To Dr. Lisa Osadciw,
the best mentor and friend in my whole life,
who makes all my research work possible.
To Dr. Can Isik
for his inspirational instruction and guidance.

Contents

CONTENTS

List of Figures

List of Tables

LIST OF TABLES

1

Introduction

1.1 Data Fusion for Decision Support

Data fusion technologies are widely applied to support real-time decision making in complex, dynamic environments, ranging from vehicular traffic control, target classification and tracking to machine condition monitoring and weather forecasting. The main role of such a decision support system is to aid the operators to achieve the appropriate situation awareness state for their tactical decision-making activities, and to support execution of the resulting actions (4). The information provided by the decision support system has an associated level of confidence or uncertainty, through the application of automated algorithms and processes (5). For instance, in a multi-sensor environment, the decision support system analyzes the sensor measurements using expert knowledge to make decisions on target identities, situation assessment, and threat assessment ensuring that the system is operating within defined boundary conditions. These decisions regarding the system conditions are seldom based on a single measurement. More often, decisions are made on the analysis of multiple data sources either from similar types of sensors or from a completely separate and different ones.

As with such complex multi-sensor systems, the communal information needs to be structured and organized in order to be effective. The data fusion process combines data from multiple sensor sources, and gathers that

information in order to achieve inferences, which are more efficient and potentially more accurate than if they were achieved by means of a single source. Various classes of algorithms have been developed to implement data fusion in the multi-sensor systems. The mature techniques include Bayesian inference, fuzzy logic, swarm intelligence, pattern recognition using signal processing algorithms and multi-sensor multi-target tracking.

In addition, several types of data fusion models exist for combining sensor data to support making decisions. Data fusion has its roots in the defence research community of the early 1980's. As a result the first data fusion models were either adapted from existing military process models or designed retaining a distinctly military flavor. More recently the use of data fusion has broadened to include industrial, medical and commercial applications, so that models have acknowledged this migration by reducing the military terminology. In the 1980's, the *JDL model* and *Boyd control loop* were first used for modeling the military command process, but have since been widely used for data fusion systems. The 1990's saw the introduction of the *Waterfall model*, the *Dasarathy model*, and the *Omnibus model*, etc (6). A common theme of these models is the prescription of multiple levels of processing within the data fusion process itself. However, the models are differentiated by the amount of processing applied to the sensor data before transmission to the fusion process, resolution of the data that is combined, and the location of the data fusion process (7). The fusion model selected to combine sensor data depends on the particular application.

With the increasing use of data fusion technologies for industrial and commercial problems, the data fusion model is required to be capable of optimally responding to the dynamically changing system. In this dissertation, I propose a new data fusion model which satisfies this requirement with an intelligent mechanism returning the feedback to the system in real-time. This model is extended from the *Omnibus model* (8), which is shown in Figure 1.1. The new model overcomes some of the main limitations of this previous model while capitalizing on its advantages. Figure 1.2 presents the layout of my data fusion model, the *Intelligent data fusion model*. Compared to Figure 1.1, instead of only processing the context of system to make decisions, a threat assessment

is added into the new model to come up with a set of possible operation as soft decisions based on the current situation, and also to analyze the advantages and disadvantages of taking one course of action over another. This extra process task makes the decision support system more adaptive and reliable by producing many different possible answers rather than only one.

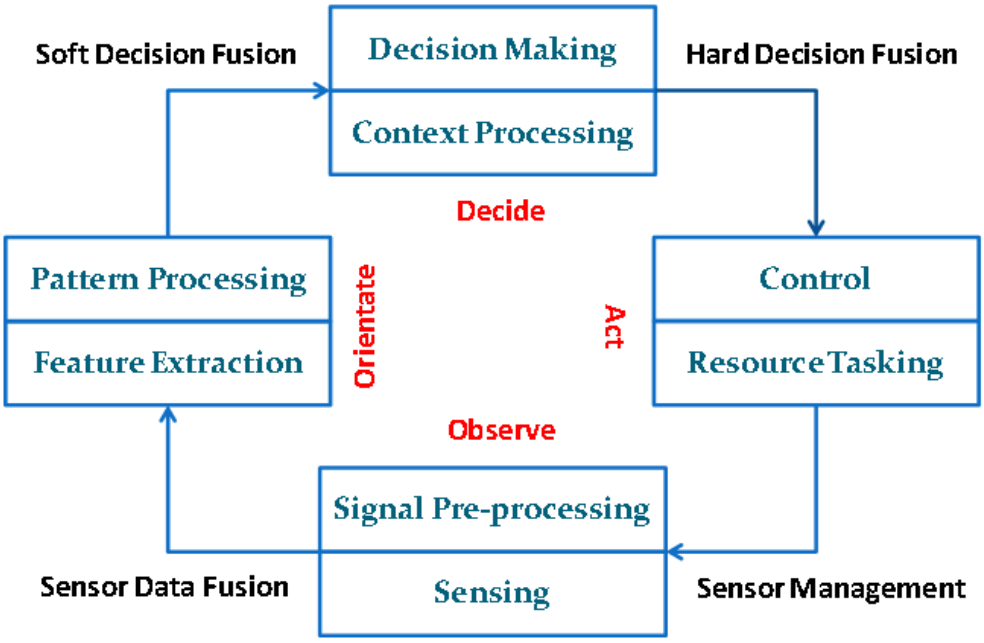


Figure 1.1: The *Omnibus data fusion model*. This is a unified data fusion process model, which consists of four main process tasks: Observe, Orientate, Decide, and Act.

The *Intelligent data fusion model* consists of four main modules, same as the *Omnibus model*. These modules are used to address the process tasks in data fusion and its functional objectives, which are observe, orientate, decide and act. The intelligent model also explicitly closes the loop by taking account of the effect of decisions in the real world. This cyclic nature of my model facilitates not only the clarification of task-level measures of performance but system-level measures of effectiveness.

1. **Observe**: The raw sensor data is recorded and pre-processed at this stage to describe the phenomenon being measured. Firstly, the data is processed to attain a common spatial and time frame. It is called the data

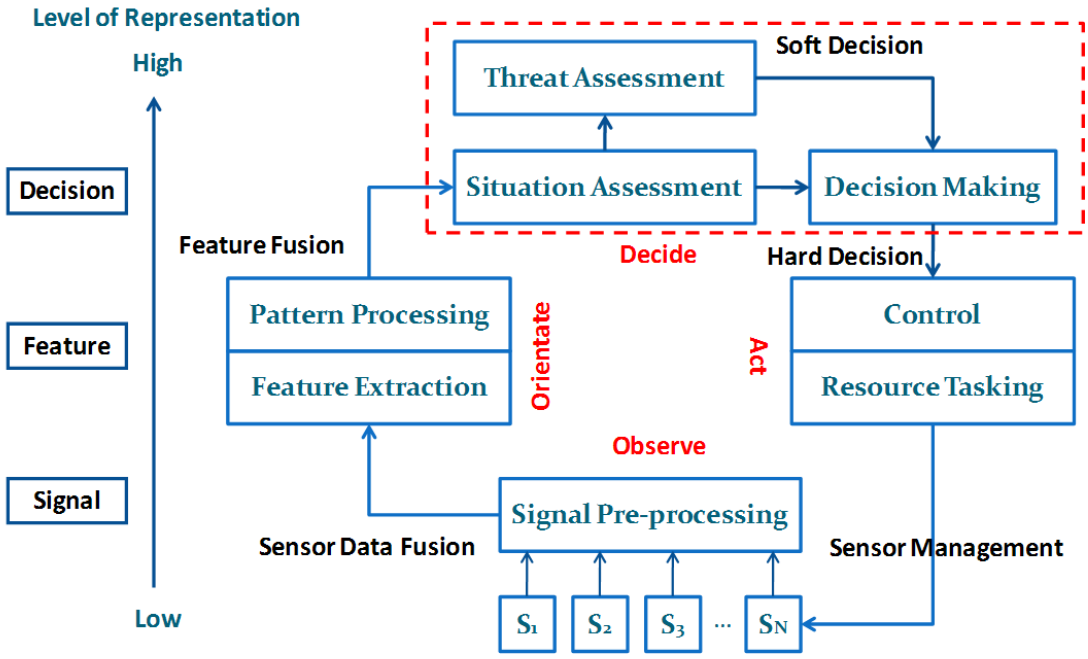


Figure 1.2: The *Intelligent data fusion model*, extended from the *Omnibus model* by adding a threat assessment.

alignment. Then, the data is analyzed as association performed among data units of the same variable and between data units of different variables, based on the degree of proximity among variables. For example, when continuously monitoring wind turbine performance to detect failures at an early stage, both the measurements of generated power and pitch angle are highly correlated to the wind speed, but uncorrelated to each other. So, two individual tests of power and pitch angle are designed against wind speed, and executed separately to describe the turbine current working condition from two different perspectives. This data pre-processing also includes object refinement. The object refinement is concerned with the identification and estimation of continuous (*e.g.* location and velocity of aircraft) or discrete (*e.g.* behavior or attributes of turbine performance) states of objects.

2. **Orientate**: At this stage, features are extracted by functions of the raw or pre-processed measurement data. These functions are application-

specific feature extractors. This feature extraction process is applied to generate a minimum set of features that contain the maximum information relevant to the problem. Then, the features are fed into pattern processing, or pattern classification, which is a mapping from the extracted features to the defined pattern classes in this problem domain. The class labels represent meaningful and distinct categorization of the problem. For example, to describe one interesting turbine working condition at current time, multiple performance tests are designed on generated power, rotor speed, pitch angle of this turbine, and also on the wind speed difference from its neighboring turbine. Each test result is one feature of current turbine status. All the test results are fused to reach a final working-condition pattern of this turbine, which may be complete shutdown, under-performing, abnormally frequent default, as well as normal working.

3. **Decide**: The current situation is assessed to obtain a decisional level of inferences based on a statistical model, such as a Bayesian network. This process is called threat assessment. Its output is a set of action estimates with probabilities associated with them. The decision-making component delineates this set of possible courses of action and the effect they would have in the current situation, and then choose the action as a hard decision which optimizes the system performance. For example, a wind turbine diagnostic unit can automatically monitor the turbine working status and identify the potential failures with a percentage certainty based on any abnormal changes in the turbine performance.

4. **Act**: The action is undertaken to continuously update the system with feedback information arriving from the decision-making component. The feedback is an element of resource management and used to close the loop by reallocating resources (*e.g.* sensors, communications and processing) in order to support the objectives of the mission. For example, once predict when and what types of failures are going to happen to a turbine, the wind farm operators can fix the failures at an earlier stage to avoid major breakdowns. With the recommended maintenance actions,

the system controller can come up with an optimal maintenance schedule to minimize the repair costs.

Table 1.1: Characteristics of data fusion levels

Characteristics	Signal Level	Feature Level	Decision Level
Representation Level of Information	Low	Medium	High
Type of Sensory Information	Multi-dimensional signal	Features extracted from signal	Logic decision from features
Model of Sensory Information	Random variables with noise	Non-invariant form of features	Decision with degree of uncertainty

As shown in Figure 1.2, there are three representation levels of information: raw data or signal, feature and decision. Table 1.1 illustrates a comparison of the different fusion levels classified by the representation of information. The *Intelligent data fusion model* comprises a process flow chart from the signal level to the decision level, and certain fusion algorithms can be applied among different levels. Usually, the fusion process is described based on the input and output information types, such as the fusion modes of *data in - feature out*, and *feature in - decision out* mentioned above.

This dissertation proposes the system-based implementation of the *Intelligent data fusion model* to support decision-making in two real-world problems: one is the sensor network management for air traffic control; the other is wind turbine failure detection and maintenance. Due to the dynamically changing environment and the complexity of system in both problem domains, artificial intelligence algorithms, such as Bayesian inference and particle swarm optimization, are employed to make the data fusion system more adaptive and cognitive to solve problems quickly and in real-time.

1.2 Research Objective

Many real-world problem domains have dynamically changing environments. For example, in the air traffic control system, the aircraft movement is affected by the unconstant strength of airflow; and the varied weather condition is always an important influence to the energy production of a wind farm (9). Moreover, with the growing complexity of problem domain, it requires a decision support system capable of dealing with the rapidly increasing amount of multiple data sources. Therefore, the main objective of this dissertation is to design an automatic adaptive decision support system, and implement it in solving real-world problems that normally would require human intelligence.

The form of decision support system which is generally used in industry is the rule-based system, also known as the expert system. It uses human expert knowledge to make decisions. Expert knowledge is often represented in the form of rules or as data stored in the computer (10). However, the rule-based expert system exhibits human-level performance in some very narrow area, which is incapable of learning or expanding its expertise as the external environments changing. Also, it has to be supported and maintained by technical people.

To overcome these limitations, the *Intelligent data fusion model* for decision support I propose in this dissertation has the following properties,

- This data fusion model has a loop structure that returns the feedback of decisions into the next time interval, so that it can respond to the dynamically changing environment in real time, and execute the actions meeting the system requirements with respect to the current circumstance.

- This model uses an artificial intelligent mechanism, which makes the decision support system more cognitive and reliable (11). The cognitive aspects of decision support systems focus on modeling the feedback-loop process of human decision making by which a preferred option or a course of actions is chosen from among a set of alternatives based on certain criteria. For instance, I apply Bayesian network for threat assessment in the system, in that it can create a set of possible operations based

on current situation, and present confidence or uncertainty of taking one course of action over another. I also apply particle swarm optimization algorithm to make decisions more objectively and optimally. This algorithm has simple computations, so that it can converge to an optimal solution very quickly.

- This model is a data-driven approach, which makes the decision support system more automatic and adaptive. It only needs the expert knowledge in the system design phase, but requires no human in the loop execution. The intelligent mechanism learns from the system behavior, and automatically adapts the system parameters to the new changes. In addition, due to its data-driven nature, this model is easily transferable from one problem domain to another.

1.3 Research Contribution: System-based Design and Implementation of Decision Support System

The design and implementation of an automatic adaptive decision support system form the main thrust of this dissertation. The system parameters are optimized and controlled in real time to improve the system performance, by integrating the technologies of data fusion, signal processing and intelligent algorithms. This methodology has been applied into two real-world problems: the sensor network management for air traffic control, and the failure detection and maintenance for wind turbines.

1.3.1 Sensor Network Management for Air Traffic Control

The objective of sensor network management is to determine sensor modes, sensor search patterns and many other network parameters to increase their collective effectiveness. Typically, sensors are used to observe phenomena in a region of interest. The sensor settings and network settings are managed in

real time to improve the system performance (12). A few examples of the settings of the sensor network that can be controlled include the detection threshold, waveform parameters in active sensors such as pulse repetition interval, communication sequences, sensor schedules and other network parameters.

Data fusion has been an active area of research for sensor network management, which strives to increase the quality of information gathered, and decrease the amount of data transferred (13). The primary hurdle in applying fusion technology to wireless sensor networks is designing an efficient, real-time, and low power system capable of transferring all critical information content to the optimal processing and system output location. Algorithms capable of reducing the message size without destroying information are critical to this area of sensor management.

In addition, the data fusion model of communication and signal processing is constructed to link the system performance to the sensor control settings. The model is the key to successfully achieving optimum performance. In order to select the appropriate settings, the sensor or network manager must be able to accurately predict the impact that new operating parameters have on the global performance. The manager model includes estimates of a dynamically varying environmental state, i.e. the observable events (scenarios) and the sensor current performance. These estimates are incorporated into the model, making the model more probabilistic than deterministic.

As more sensors and data become networked together, the management problem is getting more complicated by the number of sensors, spectrum resource allocation, sensor placement, calibration, and fusion processing, etc. To provide an automatic control of a group of sensors, the efficient intelligent algorithms are needed in the sensor network management system (14). These algorithms make management meet the real-time requirements adapted by the dynamically changing missions and situations. The ability of self-organization and optimization enables networks to be employed in a wide variety of applications. For example, if one considers global performance as the numerical measure of the system efficacy, a robust multi-objective optimization algorithm can be designed to efficiently manage the sensor measurement process, communications and fusion process. So, several artificial intelligent algorithms are

employed with data fusion model to solve the real-world problems quickly and dynamically.

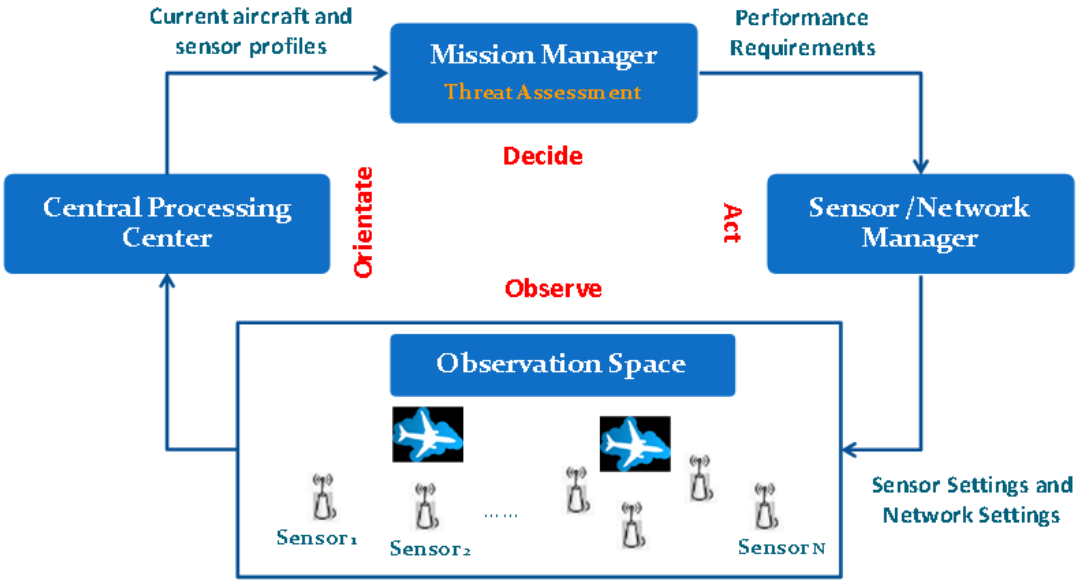


Figure 1.3: Sensor network management system framework for air traffic control problem.

Figure 1.3 presents the sensor network management framework for air traffic control problem in this dissertation. The sensors and network are controlled by the sensor manager (15). Chapter 4 describes in detail the proposed sensor management system dealing with the efficient resource allocation to meet mission objectives of air traffic control. The sensors process information messages and send them to the central processing center, which fuses the observations to assess the current situation, including the aircraft profiles, such as its position and velocity, and the sensor capabilities. The threat assessment is achieved at the mission manager with the information fed by the processing center. The mission manager, for example designed as a Bayesian network, determines the situation specific parameters, goals, objective functions and requirements for the network as time progresses. These network requirements are sent to the sensor manager to determine the sensor and network settings. Eventually, the sensor manager, for example applying the particle swarm optimization algorithm, decides the controls and sensor settings to configure the

network, such as a sensor schedule.

In the air traffic control application, a sensor schedule is constructed every time interval by integrating a Bayesian network with a particle swarm optimizer, which simultaneously meets the measurement accuracy and update rate, while minimizing the transmissions from the sensor. The Bayesian network, as a mission manager, automatically determines the management requirements for individual aircraft by monitoring the dynamically changing situations. This component maintains the best overall performance possible from the entire system. The particle swarm optimizer is used to form, test and search potential sensor schedules for an optimal solution that maximizes the weighted performance estimates, in which the weights are selected as the management requirements sent by the Bayesian network.

1.3.2 Cognitive Monitoring and Maintenance for Wind Turbines

Wind energy is expected to play an increasingly important role in the future national energy scene. With the development of the wind turbine installed capacity and increasing of the single unit capacity, the safety and reliability of wind turbines have become more significant (16). However, without an effective monitoring system, under-performing or faulty turbines will cause a huge loss in revenue (17, 18). Early detection of such failures help prevent these undesired working conditions, and allow the operators to develop maintenance plans with prioritized tasks (19). If failures are detected at an early stage, the consequent damage is minimized or mitigated, and also repairs are better scheduled. This leads to shorter down-times and lesser revenue losses. Therefore, diagnosis and prognosis of potential faults are crucial to maintain and improve the efficiency of the wind energy generation system.

Currently, condition monitoring technologies are used for wind turbine maintenance and repair, but their monitoring is based on vibration, noise and acoustic emission signals sensed from turbines. To avoid the lack of monitoring information, extra diagnostic systems have to be installed in each single turbine. The cost of extra systems is a problem. Also, it is difficult to transmit a

large amount of data between the wind turbine and the central control room. Moreover, there are other built-in diagnostic units developed for physical model based faults such as bearing faults (20), and there are some effective features to diagnose specific faults, such as wavelet transformation for spectrum decomposition in mass unbalance fault diagnosis (21), but other minor also detrimental faults, such as anemometer and wind vane faults, are not fully studied yet (3, 22).

This dissertation proposes an intelligent data-driven approach to monitor the wind turbine performance at real-time by fusing multiple test results, and detect the turbine abnormalities by tracking the turbine status variations. The description of this approach is fully expanded in Chapter 5. This approach applies data fusion technologies for reducing the influence of uncertainties associated with the measurement process. The data-driven and adaptable nature of this approach also makes it capable of monitoring a large wind farm.

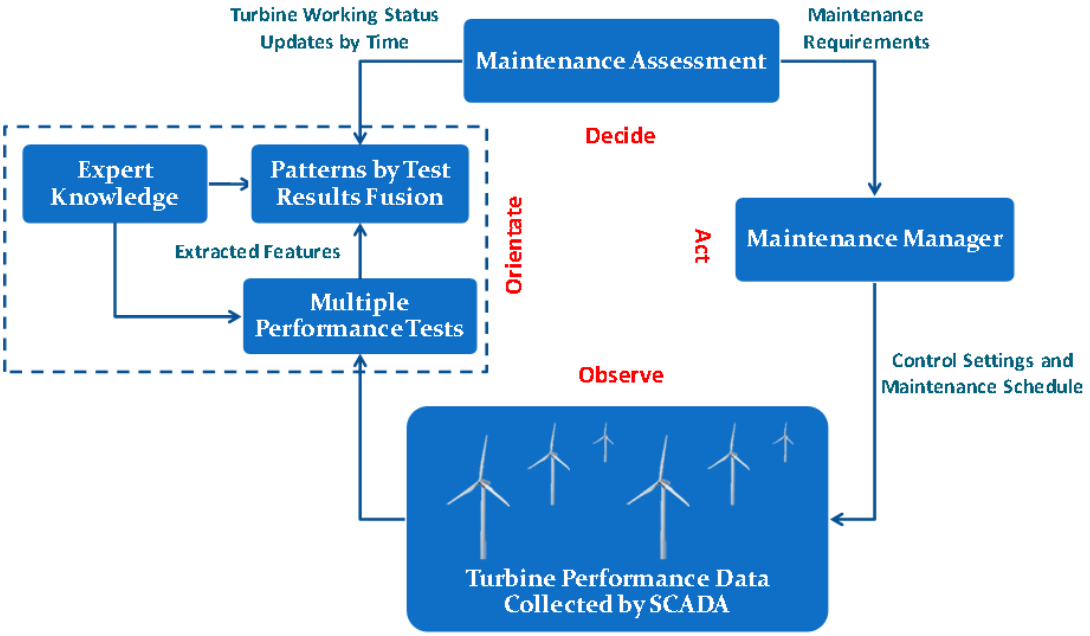


Figure 1.4: Wind turbine monitoring and maintenance system framework.

The turbine performance data are collected by the SCADA (Supervisory Control And Data Acquisition) system. It is very straightforward to analyze turbine performance to detect its faults. As shown in Figure 1.4, I develop multiple

performance tests on power generated, rotor speed, pitch angle of individual turbine, and also on the wind speed difference between two neighboring turbines. In each test, multiple states are defined as features to distinguish different performance patterns. Then, all the test results are fused to reach a final conclusion, which describes the turbine working status at current time, including complete shutting down, under-performing, abnormally frequent default, as well as working properly. Fusing multiple tests is more effective than any single test. For example, through extensive data mining of historical data and verification from farm operators, some state combinations are discovered to be strong indicators of failures, such as spindle failures, lighting strikes, anemometer faults, etc, for fault detection. Next, a Bayesian network (BN) is applied to automatically monitor the turbine working status and identify the potential failures with a percentage certainty based on any abnormal changes in the turbine performance. The main goal is to predict when and what types of failures are going to happen, so that the farm operators can fix the failures at an earlier stage to avoid major breakdowns. Hence, BN recommends the maintenance actions to the system controller to come up with the optimal maintenance schedule, which minimizes the repair costs by using the particle swarm optimization (PSO) algorithm.

1.4 Dissertation Outline

This dissertation is organized as follows:

Chapter 1 has given an introduction to the theory of decision support system using data fusion technologies. This chapter proposes a new data fusion model with intelligent mechanism, which is capable of optimally responding to the dynamically changing environment. The design and implementation of this methodology is the main thrust of this dissertation. In this chapter, I briefly present the applications of the decision support system into two real-world problems: the sensor network management for air traffic control, and the failure detection and maintenance for wind turbines.

Chapter 2 describes the Bayesian network algorithm, which is an artificial intelligence approach applied in the proposed data fusion model. Currently, Bayesian network (BN) becomes a extremely popular probabilistic graphic model for encoding and reasoning uncertain knowledge. This chapter includes the mathematic theory of the presentation, inference and parameter learning of Bayesian networks.

Chapter 3 presents a detailed description of the particle swarm optimization (PSO) algorithm for continuous search space. PSO has been applied successfully to solve various real-world optimization problems in engineering, especially the NP-hard problems. This chapter also includes a presentation of the numerous key features in PSO.

Chapter 4 proposes a new sensor management system for a complicated and highly dynamic air traffic control problem. This chapter focuses on demonstrating that the system performance with the Bayesian mission manager has a significant improvement for the highest priority region, such as the airstrip area. This research work was supported by Sensis Corporation, conducted in a collaborative fashion (1).

Chapter 5 proposes an intelligent data-driven approach to monitor the wind turbine performance in real-time by fusing multiple test results, and to detect the turbine failures by tracking the turbine status variations. The test results have verified the effectiveness of detecting turbine failures at an early stage, especially for anemometer failures. This research work was collaborated with AWS Truepower, Inc. (2).

Chapter 6 provides a conclusion of this dissertation and recommendations for future work.

2

Bayesian Networks

Bayesian networks (BNs), also known as belief networks, belong to the family of probabilistic graphical models. These graphical structures are used to represent knowledge about an uncertain domain. In particular, each node in the graph represents a random variable, while the edge between the nodes represent conditional dependencies among the corresponding random variables. These conditional dependencies in the graph are often estimated by using known statistical and computational methods. Hence, graphical models provide a principled approach to dealing with uncertainty through the use of probability theory, and an effective approach to coping with complexity through the use of graph theory.

BNs are known as a directed acyclic graph (DAG) that has become a powerful tool in statistics, machine learning, and artificial intelligence communities. BNs are both mathematically rigorous and intuitively understandable (23). They enable an effective representation and computation of the joint probability distribution (JPD) over a set of random variables. BNs became extremely popular models for encoding and reasoning uncertain knowledge in the last decade. They have been used for applications in various areas, such as text mining, natural language processing, speech recognition, signal processing, bio-informatics, error-control codes, medical diagnosis, weather forecasting, and cellular networks. I introduce the BNs algorithm in this chapter, and then I adapt it as the mission manager in the sensor management system for air

traffic control problem in Section 4.2, and also for identifying potential turbine failures with probability at an early stage in Section 5.7.

2.1 Representation

The Bayesian network has a directed acyclic graph (DAG) structure providing causal information. The structure of a DAG is defined by two sets: the set of nodes and the set of directed edges. The nodes represent random variables in the domain, and are drawn as circles labeled by the variable names. The edges represent direct dependence or causal relation among the variables and are drawn by arrows between nodes (24). In particular, an edge from node X_i to node X_j represents a statistical dependence between the corresponding variables. Thus, the arrow indicates that a value taken by variable X_j depends on the value taken by variable X_i, or roughly speaking that variable X_i influences X_j. Node X_i is then referred to as a parent of X_j, and, similarly, X_j is referred to as the child of X_i, as shown in Figure 2.1. Note that although the arrows represent direct causal connection between the variables, the reasoning process can operate on BNs by propagating information in any direction.

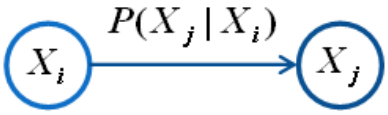


Figure 2.1: Two-node Bayesian network demonstrating the causal relationship between two random variables. The causal relation is represented by a conditional probability.

In addition to the DAG structure, which is often considered as the qualitative part of the model, it is necessary to specify the quantitative parameters of the model. The parameters are described as the conditional probability distribution (CPD) at each node depends only on its parents. For example, the CPD for X_j, given its parent X_i in Figure 2.1, is denoted as $P(X_j|X_i)$. The CPD captures the conditional probability of the random variable, given its parents in the graph. For discrete random variables, this conditional probability is often represented by a table, listing the local probability that a child node takes on each

of the feasible values (called states of the variable), given each combination of values of its parents. The joint distribution of a collection of variables can be determined uniquely by these local conditional probability tables (CPTs).

Figure 2.2 shows one example of a Bayesian network. Each node in the graph corresponds to a discrete random variable in the domain. An edge describes a causal relation between two variables, such as $W \rightarrow Y$ in which W is the parent and Y is the child. In addition to the graph, each node has a conditional probability table (CPT) specifying the probability of each state of the node given each possible combination of parent's states. The root node contains no parent, then the table contains marginal probabilities. For example, the root node W has two states, $w1$ and $w2$. The CPT of its child Y contains four conditional probabilities, $P(y1|w1), P(y2|w1), P(y1|w2)$ and $P(y2|w2)$.

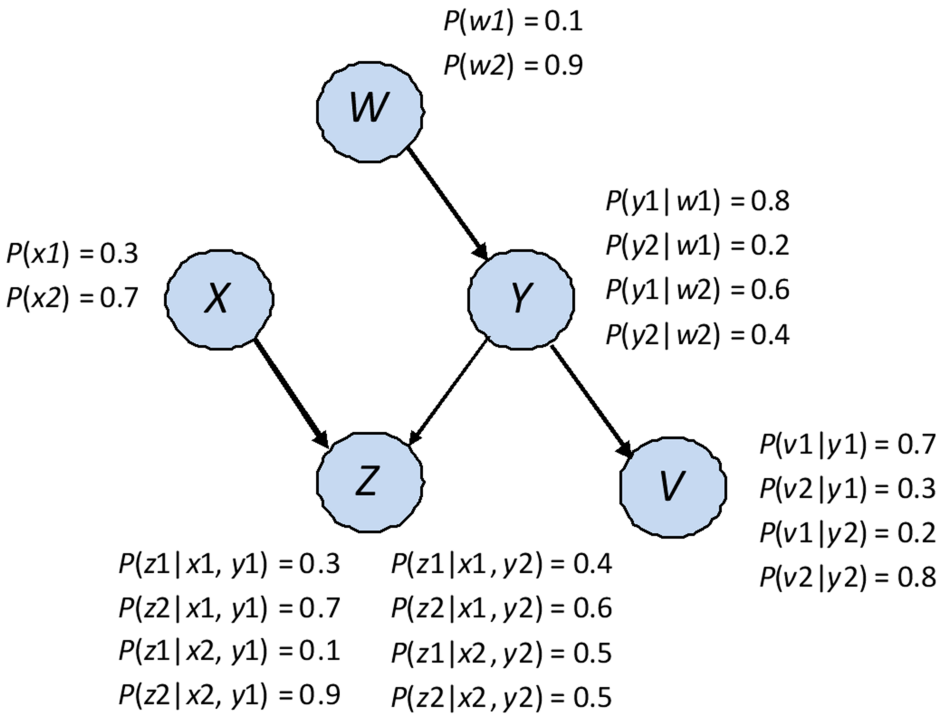


Figure 2.2: Bayesian network example Its DAG has five discrete random variables, W, X, Y, V, Z. In addition to DAG, it contains CPTs for each node.

The goal of graphical models is to efficiently represent a joint distribution P

over a set of random variables $X_1, X_2, ..., X_n$. Based on the chain rule (23),

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_1, X_2, X_{i-1}). \tag{2.1}$$

Even in the simplest case where these variables are binary-valued, a joint distribution requires the specification of 2^n numbers - the probabilities of the 2^n different assignments of values $x_1, x_2, ..., x_n$. For example, the joint probability of variables in Figure 2.2 is shown as following,

$$P(W, X, Y, V, Z) = P(W)P(X|W)P(Y|W, X)P(V|W, X, Y)P(Z|W, X, Y, V). \tag{2.2}$$

However, BNs can simplify Equation (2.1) by reducing the number of the model parameters, based on the independence properties exploited by BN structure. These independence properties are called the local Markov assumptions (23): Given a BN structure over random variables $X_1, X_2, ..., X_n$, for each variable X_i, it has that

$$(X_i|NonDescendants_{X_i}|\Pi_{X_i}), \tag{2.3}$$

where $NonDescendants_{X_i}$ denotes the variables in the graph that are not descendants of X_i, and Π_{X_i} denotes the parent set of X_i. In other words, the local Markov assumptions state that each node X_i is independent of its non-descendants, and only dependent on its parents. The joint probability P can be factorized according to BN structure. Equation (2.4) shows that P is expressed as a product of set of CPTs associated with BN's nodes. Each factor represents a conditional probability of the variable only given its parents in the network.

$$P(X_i, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|\Pi_{X_i}). \tag{2.4}$$

If X_i has no parents, its local probability distribution is said to be unconditional, otherwise it is conditional. If the variable represented by a node is observed, then the node is said to be an evidence node, otherwise the node is said to be hidden or latent.

Bayesian networks are used to reduce, sometimes significantly, the number of parameters that are required to characterize the JPD of the variables. This reduction provides an efficient way to compute the posterior probabilities given the evidence. For example, according to the BN structure in Figure 2.2, the variable X is not a descendant of variable Y, and Y only has one parent W, so that based on the local Markov assumptions, $P(Y|W,X) = P(Y|W)$. Then, the Equaiton (2.2) can be simplified as,

$$P(W,X,Y,V,Z) = P(W)P(X)P(Y|W)P(V|Y)P(Z|X,Y). \tag{2.5}$$

2.2 Inference

Given a BN that specified the JPD in a factored form, one can evaluate all possible inference queries by summing out over irrelevant variables (25). Two types of inference support are often considered: *predict support* for node X_i, based on its observed parent nodes (also called *top-down reasoning*); and *diagnostic support* for node X_i, based on its observed children nodes (also called *bottom-up reasoning*).

In general, the full summation over discrete variables is called *exact inference*, and know to be an *NP-hard problem* which is exponential in the number of nodes. Some efficient algorithms exist to solve the exact inference problem in restricted classes of networks. The inference method used in this dissertation is variable elimination, which is summing out the irrelevant variables over the joint probability distribution.

Given the example in Figure 2.2, one might consider the diagnostic support for the belief on variable Y, given the observation of its child Z. Such a support is to infer causes from observed effects, which is formulated as follows:

$$P(Y|Z) = \frac{P(Y,Z)}{P(Z)} = \frac{\sum_{W,X,V} P(W,X,Y,V,Z)}{\sum_{W,X,Y,V} P(W,X,Y,V,Z)}, \tag{2.6}$$

where

$$P(Y,Z) = \sum_{V}[P(V|Y)\sum_{X}[P(Z|X,Y)P(X)\sum_{W}P(Y|W)P(W)]], \tag{2.7}$$

and

$$P(Z) = \sum_{Y}[\sum_{V}[P(V|Y)\sum_{X}[P(Z|X,Y)P(X)\sum_{W}P(Y|W)P(W)]]]. \tag{2.8}$$

2.3 Learning

Due to the good performance on probability inference, Bayesian networks are popular within the community of artificial intelligence. However, in many practical problems, the Bayesian network is unknown and one needs to learn it from the data. To describe a Bayesian network, two parts are needed to be specified: its graph topology (network structure) and the parameters of CPT for each node (26). This problem is known as the BN learning problem: Given training data and prior information, such as expert knowledge, estimate the graph topology and the parameters of the CPT in the BN.

Learning the BN structure is much harder than learning BN parameters. In this dissertation, all the Bayesian networks, applied for real-world problems, have a known structure, which is designed based on the expert knowledge and evidence data. However, another obstacle arises in situations of partial observability when nodes are hidden or when data is missing. Therefore, two BN learning cases are considered here, to which different learning methods are proposed, as seen in Table 2.1.

Table 2.1: Two cases of BN learning problems with known structure

Case	BN Structure	Observability	Learning Method
1	Known	Full	Maximum Likelihood Estimation
2	Known	Partial	Expectation Maximization

2.3.1 Learning with Known Structure and Full Observation

The goal of learning in this case is to find the values of the parameters of each CPT that maximize the likelihood of the training dataset. The dataset contains m cases that are often assumed to be independent. Given the training dataset $\mathbf{D} = \{\mathbf{d_1}, ..., \mathbf{d_m}\}$, where $\mathbf{d_l} = (x_{l1}, ..., x_{ln})^T$, and the parameter set $\Theta = (\theta_1, ..., \theta_n)$, where θ_i is the vector of parameters for the conditional probability table of variable X_i, and n is the total number of variables in BN, the log-likelihood of the training dataset is a sum of terms, one for each node:

$$\log L(\mathbf{\Theta}|\mathbf{D}) = \sum_m (\sum_n (\log P(x_{li})|pi_i, \theta_i)). \tag{2.9}$$

The log-likelihood scoring function decomposes according to the graph structure; hence, one can maximize the contribution to the log-likelihood of each node independently (27).

Usually, maximizing the likelihood of training dataset amounts to counting frequencies of each event happening. For example, consider estimating the CPT for the node Y of the network shown in Figure 2.2. Given the training dataset, one can count the number of times Y takes value $y1$ when its parent W is equal to $w1$, denoted by $N(Y = y1, W = w1)$. Similar, $N(Y = y1, W = w2)$ represents the number of times Y takes value $y1$ when W is equal to $w2$. Then, with these two counts, one can find the maximum likelihood estimate of one condition probability value of the node Y as following,

$$P(Y = y1|W = w1) = \frac{N(Y = y1, W = w1)}{N(Y = y1, W = w1) + N(Y = y1, W = w2)}. \tag{2.10}$$

2.3.2 Learning with Known Structure and Partial Observation

In the second case with known structure and partial observation, one can use the expectation maximization (EM) algorithm to find a locally optimal maximum-likelihood estimate of the parameters (23).

Given a Bayesian network consisting of a set **X** of observed data, a set of unobserved hidden data **Z**, and a vector of unknown parameters θ, along with a likelihood function

$$L(\theta; \mathbf{X}, \mathbf{Z}) = P(\mathbf{X}, \mathbf{Z}|\theta), \tag{2.11}$$

EM is an iterative method which alternates between performing an expectation (E) step, which calculates the expected value of the log-likelihood function with respect to the conditional distribution of **Z** given **X** under the current estimates of the parameters $\theta^{(\mathbf{t})}$:

$$Q(\theta|\theta^{(\mathbf{t})}) = E_{\mathbf{Z}|\mathbf{X},\theta^{(\mathbf{t})}}[\log L(\theta; \mathbf{X}, \mathbf{Z})], \tag{2.12}$$

and maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step:

$$\theta^{(\mathbf{t+1})} = argmax_{\theta} Q(\theta|\theta^{(\mathbf{t})}). \tag{2.13}$$

These parameter estimates are then used to determine the conditional distribution of the hidden variables in the next E step. This iterative procedure is guaranteed to converge to a local maximum of the likelihood surface.

For example, to learn the conditional probability $P(Y = y1|W = w1)$ with missing data, I replace the observed counts of the events in Equation (2.10) with the number of times that each event is expected to happen.

$$P(Y = y1|W = w1) = \frac{E[N(Y = y1, W = w1)]}{E[N(Y = y1, W = w1)] + E[N(Y = y1, W = w2)]}, \tag{2.14}$$

where $E[N(x)]$ is the expected number of times event x occurs in the whole training dataset, given the current estimates of the parameters. These expected counts can be computed as following,

$$E[N(x)] = \sum_{k=1}^{m} I(x|D(k)), \tag{2.15}$$

where $I(x|D(k))$ is an indicator function, which is 1 if event x occurs in kth training case, and 0 otherwise.

2.4 Features of Bayesian Networks

2.4.1 Representation of Uncertainties

It is very common that the real-world data contains uncertainties. Bayesian network, as a probabilistic graphical model, can manage and process the data with probabilities. This statistical model has the advantage at expressing the relative certainty of many different possible answers rather than only one. So, Bayesian network can produce more reliable results when it is utilized as a component of a large system. For example, Bayesian network is used to perform statistical inference with random patterns of missing data.

2.4.2 Model of Dependence

When combining evidence from multiple sources, Bayesian network can properly model the correlation and statistical dependence between variables in the problem domain. For example, if two similar sensors are applied to machine failure detection, their outputs are highly correlated, so that the same evidence will be counted twice with simple mathematic methods. In practice, this results in a phenomenon known as over-confidence. One of the sensor outputs is almost redundant, and including it in the process might actually decrease the accuracy of detection. This effect has been observed in many real-world problem fields, where evidence has to be combined, such as medical diagnosis. Bayesian network is one powerful tool for avoiding the over-confidence effect by model the variables with conditional dependence.

3

Particle Swarm Optimization for Continuous Variables

3.1 Optimization Problems

Optimization is the process of finding the optimum value of a given objective function on a particular domain, possibly with a number of additional constraints (28). An optimum can be either a maximum or a minimum depending on the problem formulation; maximization of an objective function f is equivalent to minimization of the opposite of the function $-f$.

Mathematically, a minimization task is generally defined as:

Given f: $\mathbb{R}^n \to \mathbb{R}$ Find $x^* \in \mathbb{R}^n$ such that $f(x^*) \leqslant f(x), \forall x \in \mathbb{R}^n$

Similarly, a maximization task is defined as:

Given f: $\mathbb{R}^n \to \mathbb{R}$ Find $x^* \in \mathbb{R}^n$ such that $f(x^*) \geqslant f(x), \forall x \in \mathbb{R}^n$

The domain $\mathbb{R}^n$ is referred to the search space. Each element of $\mathbb{R}^n$ is called a candidate solution in the search space, and x^* represents the optimal solution. The value n denotes the number of dimensions in the search space. The objective function f maps the search space to a one-dimensional fitness space, providing a single fitness value for each candidate solution. The final goal of an optimization task is to find the parameters in the search space that maximize or minimize the fitness function.

In this dissertation, the domain of optimization problem is continuous, and the objective function often has at least a first order derivative, which means the gradient vector

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix}, \tag{3.1}$$

is assumed to exist. There are many popular approaches to solve optimization problems, such as gradient descent based methods and linear programming. However, these approaches have their own limits. For example, to use gradient descent, the objective function should be differentiable; linear programming only deals with optimization problems with a linear objective function, which is subject to linear constraints. This chapter introduces particle swarm optimization algorithm, belonging to evolutionary algorithms, which overcomes those limitation. Particle swarm optimization algorithm cannot guarantee finding the global optima (29). However, inverse problems (the problems that convert observed measurements into information/model about a physical object or system) are very often ill-posed for standard optimization methods, and this is where particle swarm optimization is useful (30).

3.2 Algorithm Description

Inspired by the fact that simple behaviors of individuals lead to much complicated societal phenomenon, Kennedy and Eberhart first proposed the particle swarm optimization (PSO) algorithm in 1995 (31). Particles in PSO are randomly deployed to effectively explore the solution space, while the fitness function (the notation of “fitness” function is by convention, more accurately, it should be called an objective function without the implication that the higher the value, the more “fit”) guides the particles to exploit the promising regions with randomness. PSO has been applied successfully to solve various real-world optimization problems in engineering, especially the NP-hard problems, when other optimization algorithms do not work. I introduce the generic PSO algorithm for continuous variables in this chapter, and then I adapt it for the

wind speed difference test in Section 5.4, and determining the fusion rules of turbine daily working status in Section 5.6.

Without losing generalizability, assume that the optimization is a minimization problem (negating a maximization yields minimization), and the function $f(\mathbf{x})$ of multivariate **x** is to be optimized,

$$\mathbf{x} = \mathbf{arg}min f(\mathbf{x}). \tag{3.2}$$

Then the particle in PSO is **x**, and the function value $f(\mathbf{x})$ is the fitness of the solution **x**. Suppose that the search space is composed of D dimensions, and there are N particles in the swarm with the fitness of each particle as $Fitness_i$ ($i \in [1, N]$). For the ith particle, there are its current position $\mathbf{x}_i = \{x_{i1}, x_{i2}, \cdots, x_{iD}\}$, its previous best position $\mathbf{p}_i = \{p_{i1}, p_{i2}, \cdots, p_{iD}\}$, and its velocity $\mathbf{u}_i = \{u_{i1}, u_{i2}, \cdots, u_{iD}\}$.

The particle's movement is affected by its inertia, its cognitive awareness (p_{best}, the best location that the particle has been before) and social influence (g_{best}, the best location within the population during the iterations). The algorithm of PSO is as follows:

1. Initialize a population of N particles with random positions and velocities. Each particle is a solution, $\mathbf{x}_i$, $i \in [1, N]$. $Fitness_i$, p_{best_i}, and g_{best}, are all initialized to be infinity.

2. Until some termination criteria are satisfied, such as

   - the iteration index t reaches the maximum iterations t_{max};
   - there is a sufficiently good fitness. For a minimization problem, $g_{best} \leqslant \epsilon$, where ϵ is a small value.

   Repeat the following:

   (a) Evaluate $Fitness_i = f(\mathbf{x}_i)$.

   (b) Compare the particle's fitness with $p_{best_{i,t-1}}$. If the current value is smaller, copy it to $p_{best_{i,t}}$, and set $\mathbf{p}_i$ equal to the current position $\mathbf{x}_i$). Then, update $p_{best_{i,t}}$ for each particle, as cognitive awareness.

Algorithm 1 Particle Swarm Optimization

Require: :

$Initialize\ a\ population\ of\ N\ particles\ with\ random\ positions\ and\ velocities.$
$Each\ particle\ is\ a\ solution,\ \mathbf{X}_i,\ i \in [1, N].$
$Initialize\ the\ fitness\ of\ all\ particles\ to\ infinity.$
for $t = 1$ *to the number of generations* **do**
  for $i = 1$ *to the population size* **do**
    for $d = 1$ *to the problem dimensionality* **do**
      $Update\ Velocity:$
      $V_{id}^{t+1} = \omega \times V_{id}^{t} + \Psi_1 \times (p_{id} - X_{id}) + \Psi_2 \times (p_{gd} - X_{id})$
      $where\ P_i\ is\ the\ best\ position\ visited\ so\ far\ by\ X_i,$
      $and\ P_g\ is\ the\ best\ position\ visited\ so\ far\ by\ any\ particle;$
      $Update\ Position:$
      $X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1}$
    end for
    $Compute\ the\ fitness\ of\ X_i^{t+1}.$
    if $Fitness(X_i^{t+1}) < Fitness(P_i)$ **then**
      $P_i = X_i^{t+1}$
    end if
    if $Fitness(X_i^{t+1}) < Fitness(P_g)$ **then**
      $P_g = X_i^{t+1}$
    end if
  end for
  $Terminate\ if\ P_g\ meets\ problem\ requirements.$
end for

(c) Update g_{best_t} for the population, as social influence.

(d) Move the particles by

$$\mathbf{x}_{i,t+1} = \mathbf{x}_{i,t} + \mathbf{u}_{i,t+1}, \tag{3.3}$$

where $\mathbf{u}_{i,t+1}$ is the influence defined by

$$\mathbf{u}_{i,t+1} = \omega \cdot \mathbf{u}_{i,t} + c_1 \cdot r_1 \cdot (p_{best_{i,t}} - \mathbf{x}_{i,t}) + c_2 \cdot r_2 \cdot (g_{best_t} - \mathbf{x}_{i,t}), \tag{3.4}$$

where $\mathbf{u}_{i,t}$ is the velocity of the ith particle at time instance t, $\mathbf{u}_{i,t+1}$ is the velocity of the next step, and ω is an inertial constant, less than 1, to retain the information of previous velocity. $\mathbf{x}_{i,t}$ is the current particle's location that needs to be updated. c_1 and c_2 are constants to weigh these influences. r_1 and r_2 are uniform random numbers to randomize the influences.

(e) $t = t + 1$.

The flowchart of PSO algorithm is shown in Figure 3.1.

3.3 Features of Particle Swarm Optimization

The following features of PSO have given it increasing popularity in the field of optimization since it was created in 1995. PSO has been wildly applied in the areas of system design, multi-objective optimization, classification, pattern recognition, scheduling, and decision making.

3.3.1 Simplicity

The particle swarm optimization algorithm uses simple computations to travel through the search space. The algorithm structure just requires 5 simple steps for each iteration:

1. Initialize each particle;

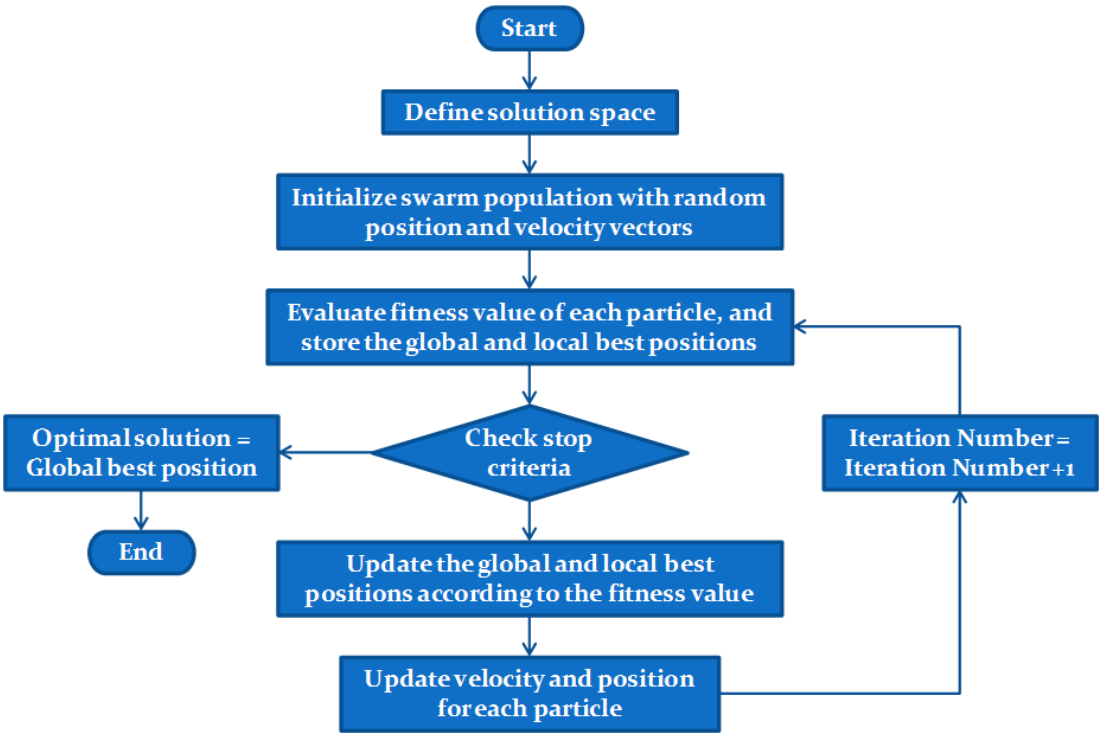


Figure 3.1: The flowchart of particle swarm optimization (PSO).

2. Update particle's velocity;

3. Update particle's position;

4. Evaluate particle's fitness;

5. Update p_{best} and g_{best}.

Therefore, the particle swarm optimization algorithm can converge to an optimal solution in a relatively short time. For example, comparisons between PSO and the standard Genetic algorithms (another kind of optimization algorithm) have been done analytically based on performance in (32). Compared to Genetic algorithms, the PSO tends to converge more quickly to the optimal solution.

3.3.2 Adaptivity

The core of PSO algorithm lies in the velocity update function. The particles travel in the continuous space by updating their velocity to obtain an equivalent position. This velocity update function does not need to be changed for implementation of different problems. Hence, essentially the search procedure of PSO algorithm does not change neither.

Moreover, the PSO algorithm also treats each dimension of the problem independently, which enables this algorithm to search for a solution made up with a combination of continuous variables. In Genetic algorithm, however, the crossover operator violates this constraint and does not allow hybridization of several individual dimensions.

3.3.3 Independence of Objective Function

The PSO algorithm separates the objective function as a black-box from the particle population. It queries the objective function to estimate the performance for each particle. This separation of the objective function enables the particles to be used for a variety of problems without any additional changing steps.

4

Intelligent Sensor Network Management for Air Traffic Control System

With the advancements in sensor technology and sensor networking, decisions regarding efficient allocation of sensor resources are quickly becoming important. Sensor management is the automatic control of a group of sensors including the data fusion processing in a sensor network to achieve a system goal. Its objective is to improve the efficiency of the sensors and communications, simultaneously. Large, complex systems of sensors are emerging due to new networking technology, but advances in sensor management are needed to make the communication system viable.

This chapter proposes a new sensor management system for a complicated and highly dynamic air traffic control (ATC) sensor network. This network is responsible for collecting information from aircraft landing, from aircraft taking off and taxiing to/from gates at the airport, or from aircraft moving above the airport's surrounding regions. Wireless communication is established between sensors and aircraft. Then based on this information, the system comes up with a sensor schedule for every certain period that simultaneously meets the measurement accuracy and update rate, while minimizing the transmissions from the sensors. This sensor management system cohesively integrates a Bayesian Network (BN) with Particle Swarm Optimization (PSO).

As a mission manager of the sensor management system for ATC application, shown in Figure 1.3, BN maintains the best overall performance possible from the entire sensor network system. The contribution of Bayesian mission manager component is the main thrust of this chapter as the following aspects,

1. It automatically updates the sensors' real-time operating requirements based on monitoring system performance, dynamically changing environment, and current sensor capabilities;

2. It predicts the impact of new operating requirements on global performance, and sends them to the PSO to select the appropriate sensor settings;

3. It balances the overall system performance by allocating more sensor resources to the regions with higher priority.

These three aspects are demonstrated with the comparison of running the whole sensor management system with and without Bayesian mission manager. As a result, the system performance with the Bayesian mission manager has a significant improvement for the highest priority region, such as the airstrip area. This research work was supported by Sensis Corporation, conducted in a collaborative fashion (1).

4.1 Air Traffic Control System

4.1.1 Problem Description

The main problem of air traffic control system is how to schedule discrete-time, two-way communications between sensors and transponder-equipped aircraft over a given coverage area (33), including both the airport and the area surrounding it. The surveillance sensor network at airports can assist air traffic controllers in guiding aircrafts as they approach and land. Since many of the aircraft primarily carry people, the reliability and accuracy of the system must be maintained at all times (34).

To perform the communication, the system interfaces with aircraft transponders using sensors that are located within the coverage area. There are two types of sensors: those that can both receive and transmit (RT) and those that can receive only (RO). The power with which the RT sensors can interrogate is variable, and can be used to control the area over which the RT transmits. Both the RT and RO sensors have a fixed range over which they can receive a reply reflected from the aircraft, with the probability of detecting the reply diminishing with range. The location of an aircraft is determined through an interrogation sequence of messages sent by the RT sensors. The RT sends an interrogation message to the aircraft and can only transmit a certain amount of messages per second. Then, a subset of sensors, usually including one RT and several RO sensors, receives and interrogates the reflected messages to locate the aircraft by using multi-lateration algorithms, as shown in Figure 4.1.

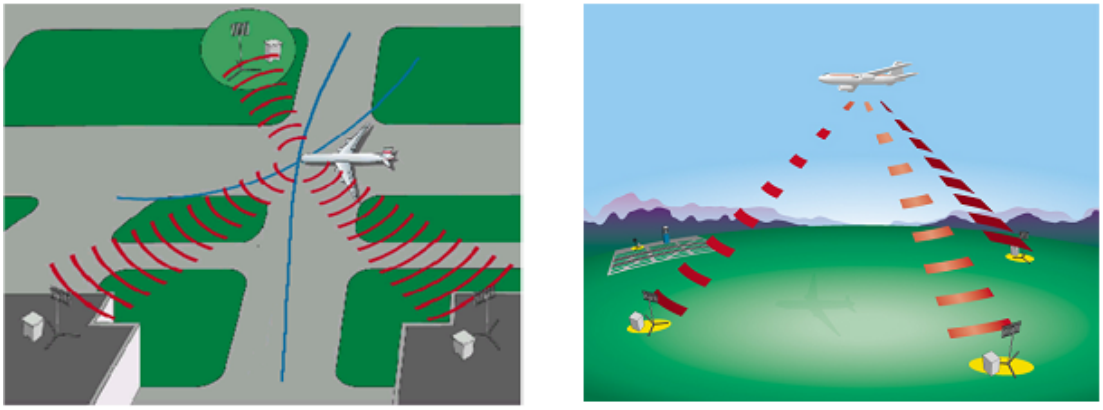


Figure 4.1: Demonstration of using integrated multi-lateration to estimate aircraft position.

Within the coverage area, each aircraft must be interrogated with an overall probability of successful communication. A *successful communication* is defined as three steps: the aircraft detecting an interrogation sent by an RT sensor, replying to the interrogation, and the aircraft's reply being detected by a given number of sensors. An *update period* is defined as the rate of interrogations, at the end of which the communication must be finished. To assure this system's quality, the coverage area is divided into smaller regions. Sensors are located in this coverage area to detect the moving aircraft. Each subdivided

region has its own communication requirements to be met for any aircraft in it. The region requirements include meeting a detection probability within a certain update period. Aircrafts in a higher priority region have shorter update periods, while those in a lower priority region have longer update periods. For example, an important region, like the airport, might have an update rate of 1 per second with a required detection probability of 97%, while a lesser priority region might have an update period of 4 seconds with a detection probability of 90%.

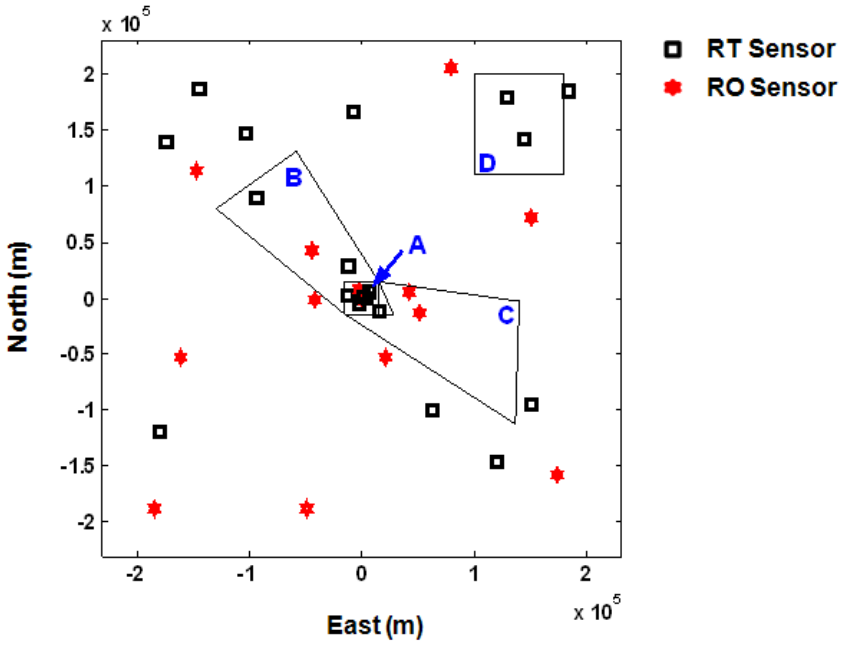


Figure 4.2: An example scenario of sensors and regions placement, including five subdivided regions and 40 sensors (1).

Figure 4.2 shows an example scenario of air traffic control problem. The coverage area for this scenario consists of a 250-by-250 nmi (nautical mile) square. This is also a $5 \times 10^5 - by - 5 \times 10^5\ m^2$ square (1 nmi = 1,852 meters), which is the unit of Figure 4.2. There are 40 sensors placed throughout the area, with 15 of them as RT sensors and the rest as RO sensors. Within the coverage area, five subdivisions, from *A* to *D* and the remainder of the coverage area, represent different regions of interest. Each region has its own default required update period, detection probability and the number of receiving

Table 4.1: Default requirements for each subdivided region in coverage area (1)

Region	Required Update Period	Required P_D	Minimum Number of Receiving Sensors	Aircraft Density ($No./nmi^2$)
A (Airport)	1 second	0.97	4	0.20
B	1 second	0.95	4	0.10
C	2 seconds	0.92	3	0.08
D	2 seconds	0.90	3	0.15
Surround Area	4 seconds	0.80	1	0.05

sensor for a successful communication, as shown in Table 4.1. The effective transmit range of any RT sensor can be varied from 2.5 nmi out to 50 nmi, with a step size of 2.5 nmi. Dependent on the distance from RT sensor to an aircraft, the RT transmit power is adjustable to make an aircraft constantly receive the messages with probability 97%, $P_{tgt} = 0.97$. The fixed range of a receiving sensor is equal to 50 nmi, denoted by $D = 50nmi$. So, the probability of any RT or RO sensor to receive a response from the aircraft is dependent on the range to the aircraft, denoted by R. The receiving probability is given by the following equation:

$$P_{rec} = \begin{cases} 1 - (\frac{R}{D})^2 & \text{for} \quad R \leq D \\ 0 & \text{for} \quad R > D \end{cases} \tag{4.1}$$

The probability of at least one successful communication in Q attempts is given as:

$$P_D = 1 - (1 - P_{tgt} - P_{NofM})^Q, \tag{4.2}$$

where N is the required number of sensors receiving the reply in the clear, and M is the total number of sensors commanded to listen. So, the P_{NofM} is the probability of a least N of M sensors received the reply in the clear given the sensor's individual P_{rec}.

4.1.2 Research Objective

The research objective of this air traffic control problem is to develop an adaptive sensor management system that schedules communications with moving aircraft. The sensor management system can deal with the efficient sensor resource allocation by generating a sensor schedule to meet mission objectives of the application (35, 36). The mission objective, in this case, is that the generated schedule must minimize the total number of interrogations needed to conserve power, while maintaining update interval constraints and detecting aircraft with a detection probability as high as possible. The main contribution of the proposed sensor management system is that it automatically updates the system operation requirements in real-time based on monitoring its current performance and dynamically changing environment. Figure 4.3 describes a realistic scenario of the sensor management system for air traffic control.

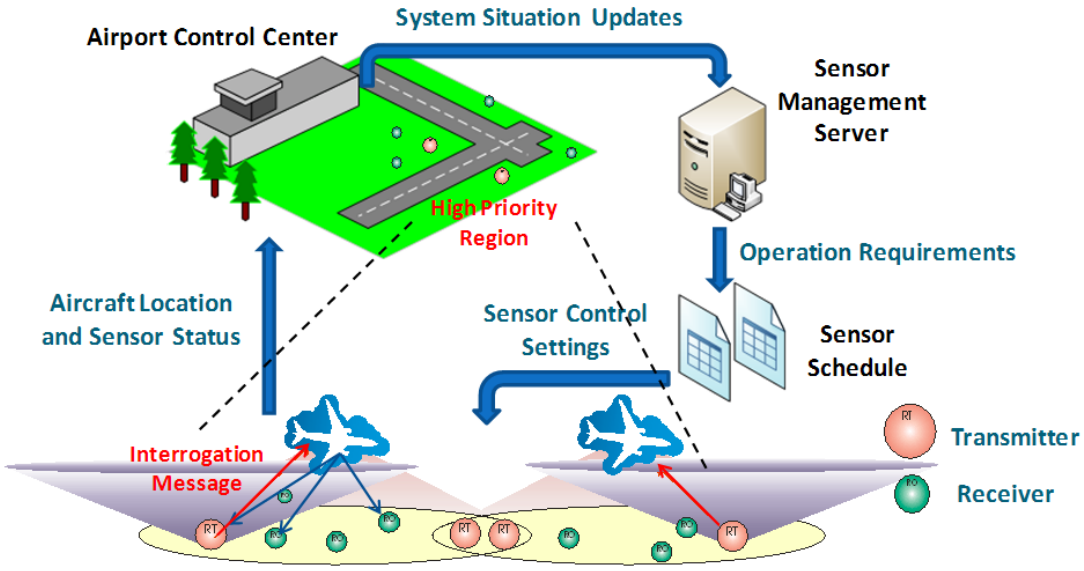


Figure 4.3: The realistic scenario of sensor management system for air traffic control application.

As mentioned in the previous section, the region requirements include meeting a detection probability within a certain amount of time. Also, the trade-off of this problem is minimizing the interrogation density but meeting the requirements for each region at the same time. Bayesian network, as a statistic

graphic model, usually can find a balance between those two objectives by making decisions of system adjustment. In my proposed sensor management system, as shown in Figure 1.3, the Bayesian mission manager updates region requirements in real-time after monitoring the current system performance. Each requirement can be modified to be either more stringent or looser, if the modification is beneficial for the efficiency of sensor resource allocation. Although this is counter intuitive, the test results demonstrate that this can improve coverage and efficiency. Some regions may have more stringent detection requirements than others, due to a higher priority.

Eventually, a schedule is generated based on the period of the longest update rate for any region. The schedule determines which sensor transmits the message, as well as, determines when and which sensors will receive it, and determines the transmit power needed for each interrogation in any region at a given time. The scheduling interval is divided by the update rate, which is the period of time between two interrogations. During each scheduling interval, the sensor schedule controls the following settings:

- which RT sensors are interrogating;
- which RT or RO sensors are listening;
- the power of each interrogating RT sensor, which determines its effective range;
- the time slot RT sensors are interrogating;
- the time slot each RO or RT sensors is listening.

As the number of available sensors and aircraft increase, the scheduling problem grows exponentially, making exhaustive search techniques difficult in the time allocated. Scheduling is an NP complete problem, which has no known deterministic solution reducing the search to polynomial time. Thus, the techniques of mission management for the sensor network must be separated from the multi-objective optimization for scheduling, to properly manage the solution. This design idea is specifically illustrated in Figure 1.3. The Bayesian network can make the high level decisions regarding each region's

performance goals to support the mission management in real-time. These adaptive decisions would include the required update rate and probability of successful communication between aircraft and sensors for each region.

4.1.3 Sensor Management System Design

The sensor management system is scheduling discrete-time, two-way communications between the aircraft and sensors in specified regions. Figure 4.4 shows the block diagram of the sensor management system for air traffic control application. This design has the following advantages,

1. The loop nature of this block diagram takes account of the effect of current sensor model into next scheduling interval, which makes the sensor management system be able to respond to the dynamically changing environment in real-time;

2. The mission manager uses a Bayesian network to come up with operating requirements for each region every scheduling interval, so that it efficiently balances the allocation of sensor resources according to different region priorities.

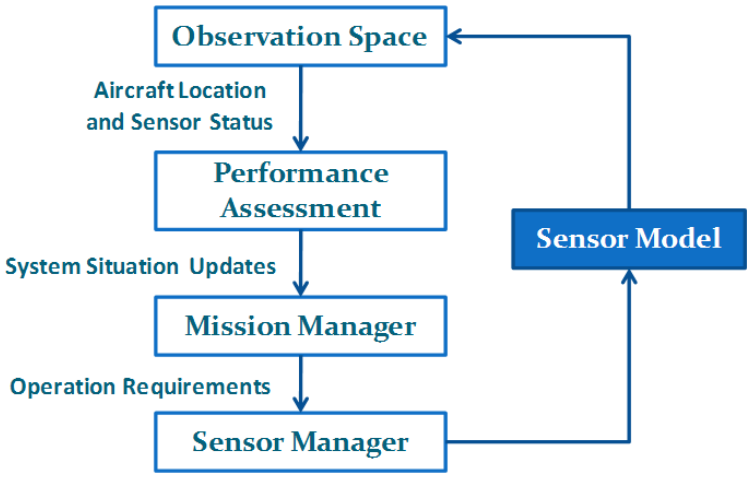


Figure 4.4: The block diagram of sensor management system for air traffic control application.

Real aircraft and sensor information could be used to test the system effectiveness. However, it is unfeasible to get the real cases of sensors detecting aircraft. Thus, we develop a simulation environment to build up an air traffic control model. It has aircraft and sensor structures defined, including the placement of sensors and regions, as well as the aircraft trajectories moving above the regions, and going on the ground in space. The simulation environment is used to support testing of the sensor management system.

The sensor simulation dynamically varies the sensor states from fully functional to completely nonfunctional. This is necessary to test adaptability of the scheduling algorithms to variations in sensor status. At initialization, the sensors are set as either RT or RO sensors. This information is stored in a sensor status vector.

The aircraft simulation randomly initializes aircrafts in space and moves them as a function of time. Each aircraft also has a state vector of its current position and region. The mission manager determines the performance requirements for each region as a function of the aircraft. The aircraft identify the region it lies in, based on the summation of the angles resulting from the lines drawn from the aircraft to the corners of the region. If these angles sum to 360°, the aircraft is in that region. This is a simple and computationally fast way to determine the aircraft's region.

The whole system data flow of sensor management system in the simulation environment is shown in Figure 4.5.

First of all, the Simulation Sensor Communication Processor (SSCP) gets the information of aircraft and sensor actions during the scenario. Aircraft's profiles contain all the aircraft locations in position, velocity, and acceleration as functions of time. The sensor status uploads the sensor operation schedule generated in the last update interval. The SSCP reacts to the ground truth and sensor information that dynamically varies as the simulation runs. Once the schedule is known for a given time period, the SSCP generates successful or unsuccessful communication exchanges between the aircraft and sensor sets given in the real-time schedule.

Then, the information of aircraft location measurements and sensor status directly goes to the Performance Assessment Processor (PAP), which main-

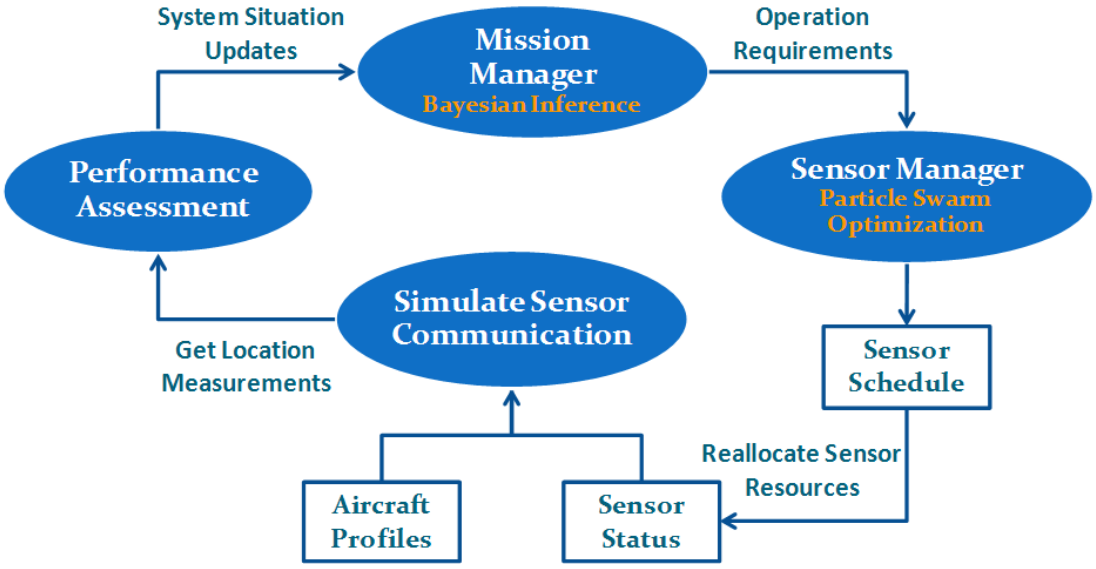


Figure 4.5: The data flow of sensor management system with simulation environment of air traffic control.

tains statistics on successful communication attempts and update times. Faster update times are needed when the aircraft is moving at high speeds through a region. Also, a sharp maneuver will require higher probability of successful communications since missed measurements may cause the controllers to lose the aircraft. This processor calculates important statistic parameters from system performance, including the mean and standard deviation of both aircraft velocity and acceleration, and the maximum difference in aircraft density within a region. All these parameters impact the system operating requirements which are distinct among regions. PAP uploads them to the mission manager to obtain the update time and probability of successful communication requirements for each region.

As a mission manager, the Bayesian inference assists in making the decisions of choosing the priorities to meet required performance parameters using both expert knowledge and evidence data collected from simulation. It analyzes aircraft performance assessments and sensor status, and then determines the operating requirements for various regions during the next scheduling interval. These requirements are used by the PSO to select the optimum sensor operating parameters, which best achieve the specified performance

for each region.

In this ATC application, as mentioned above, the PSO receives its requirements from the Bayesian mission manager. Its fitness function is designed to penalize the following: high interrogation density, interrogations close in time and space, transmitters used repeatedly, aircraft missed and interrogates less than required probability of detection (Pd). Moreover, the solution, as a sensor schedule, consists of sensor sets assigned to each aircraft target and a time when the communication between the sensors and aircraft must occur.

Once the schedule is generated, the sensor model is updated including the probability of success in communicating position between the sensors and aircraft for the next schedule.

4.2 Bayesian Mission Manager

4.2.1 Network Structure Design

For this sensor management system, the Bayesian mission manager assists in updating the mission requirements in real-time by using both expert knowledge and evidence data collected from simulations. This Bayesian network receives performance data characterizing current aircraft motion and sensor status. Then, it converts these to performance parameter priorities for various regions (37). These priorities are sent to the PSO to find the sensor schedule that can best handle the current scenario conditions, requirements, and sensor suite status.

Initially, the Bayesian network is organized based on expert knowledge. The graphical model is illustrated in Figure 4.6. Its DAG is a tree of depth three, with some additional edges between some of the nodes. The meanings of the nodes are in Table 4.2.

When building this probabilistic model, the components of interest are identified as the operating parameter requirements for the system. In this case, they are aircraft detection probability requirements and update time requirements for each subdivided region, which are represented by two root nodes

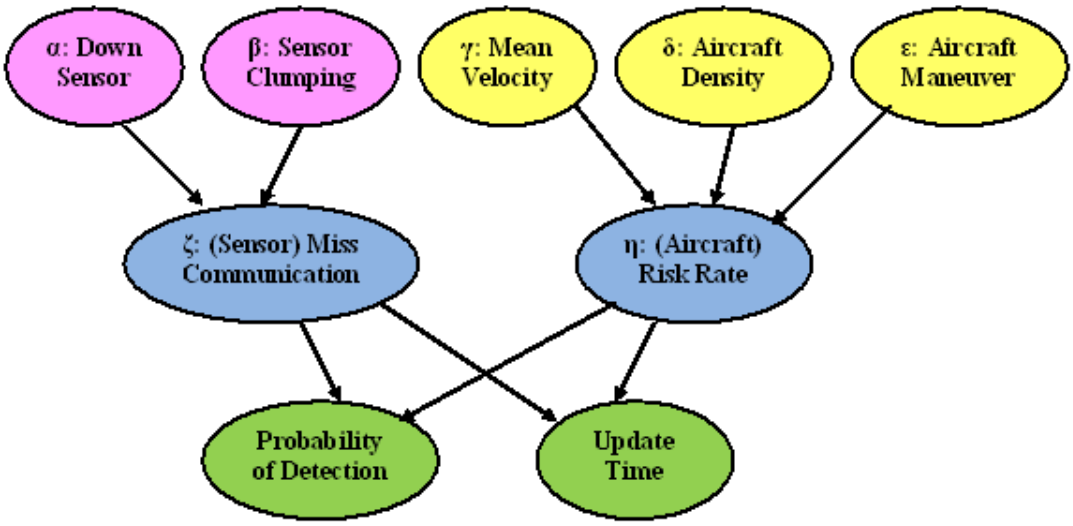


Figure 4.6: Bayesian network structure for decision making in sensor network management system.

Pd and Ut, respectively, in the Bayesian network. The root node Pd corresponds to a discrete variable that has 10 states, ranging from 0.91 to 1.00 for region A, B, C, D, and another 10 states from 0.80 to 0.90 for the surrounding region, while the root node Ut corresponds to a discrete variable having 3 states which are updating sensor schedule in 1 second, 2 seconds, and 4 seconds. These two root nodes are the final hypothesis that has to be evaluated for each region, and they are determined by the sensor status and aircraft profiles at the current time.

The two nodes at the second level of the tree represent independent aspects of system current situation: the sensor status and aircraft profiles. The sensor status determines the probability of missing communication, represented by the a discrete variable ζ, and the aircraft profiles determines the risk rate of aircraft, represented by the a discrete variable η in the network. It is expected that these two variables are largely uncorrelated, because they use different types of system information. They also each correspond to clusters of sensor-related variables and aircraft-related variables whose values are correlated.

Table 4.2: Description of variables in the Bayesian network

Variable	Description	Unit
α	down sensors	$\%(\frac{No.\ of\ down\ sensors}{Total\ No.\ of\ sensors})$
β	sensor clumping	$No.\ of\ sensors/nmi^2$
γ	mean aircraft velocity	m/s
δ	aircraft density	$No.\ of\ aircraft/nmi^2$
ε	aircraft maneuver percentage	$\%(based\ on\ aircraft\ acceleration)$
ζ	missing communication probability	$\%$
η	aircraft risk rate	$\%$

The sensor status includes the percentage of down or off-line sensors, represented by a discrete variable α, and sensor clumping, represented by a discrete variable β. A higher percentage of off-line sensors causes a higher probability of missing communications. Sensor clumping, or non-uniform distribution of sensor concentration, puts the aircraft at a risk of receiving no observation. Hence, as shown in Figure 4.6, the percentage of down sensors and sensor clumping are parent nodes of missing communications.

Moreover, the operating requirements are also influenced by aircraft profiles, such as aircraft density, mean aircraft velocity, and percentage of aircraft maneuvering. If aircraft density is too high in one region, extra interrogations are scheduled to guarantee an accurate position estimate. Thus, an aircraft may receive more communications as a result. On the other hand, extremely low aircraft density may also cause more interrogates as a result of attempting to maintain detection on aircraft. According to the mean aircraft velocity, the system will increase the number of interrogates for slow aircraft. The fast aircraft also requires more interrogates to maintain location prediction accuracy. For example, faster update times are needed when aircrafts are moving at high speeds through a region to maintain accuracy. Also, a sharp maneuver will require higher probability of successful communications because missed measurements may cause the system to lose the aircraft.

4.2.2 Parameter Estimation with Partial Observation of System Performance

With the known structure of the Bayesian mission manager, I need to learn its parameters from the evidence data before applying it for inference. The evidence data is the system performance collected from training simulations as a function of time, including the sensor status and aircraft profiles. The Bayesian parameters are the conditional probability distributions associated with each directed edge in the network structure. This parameter-estimation procedure is executed off-line.

As mentioned in Section 4.1.1, the given coverage area is divided into 5 smaller regions, region A, B, C, D and the surrounding region. There are 40 sensors uniformly located in the whole area with 15 of them as RT and the rest RO, and also 30 aircraft moving above this area. The simulation designed for collecting evidence data is modeled over 1,000 seconds, and lets the system run without the Bayesian mission manager.

The final decisions of Bayesian mission manager are the operating requirements for different regions. Each region has its own specific requirements, including the probability of detection for aircraft belonging to this region and the update time for sensors located in it. The probability of detection requirement Pd has 10 states ranging from 0.91 to 1.00 for region A, B, C, D, and another 10 states from 0.80 to 0.90 for the surrounding region; and the update time requirement Ut has 3 states which are updating sensor schedule in 1 second, 2 seconds and 4 seconds. Since each region has its own operating requirements, the following procedure is repeated for every particular region. Then, take region A as an example:

1. Given every possible state combination of Pd and Ut to all the aircraft that belong to region A, calculate the aircraft positions based on interrogation information and record the sensor status in region A at current time.

2. Assess the sensor-related variables and aircraft-related variables in the Bayesian network using sensor status and aircraft position information:

(a) Percentage of down sensors (α):

$$Percentage\ of\ down\ sensors = \frac{No.\ of\ down\ sensors\ in\ region\ A}{Total\ No.\ of\ sensors\ in\ region\ A}; \tag{4.3}$$

(b) Sensor clumping (β): The sensors are located uniformly in each region, so that the sensor clumping is assumed to be a constant;

(c) Mean aircraft velocity (γ), the average velocity of all aircraft in region A;

(d) Aircraft density (δ)

$$Aircraft\ density = \frac{No.\ of\ aircraft\ in\ region\ A}{Area\ of\ region\ A}; \tag{4.4}$$

(e) Aircraft maneuver percentage (ε), measured as the percentage of aircraft of which acceleration is beyond certain threshold.

3. The training dataset collected for region A are the partial observation of aircraft profiles and sensor status, and also contains the true label of Pd and Ut as the prior knowledge. So, expectation maximization (EM) algorithm is applied to learn the conditional probability distribution for each node in the Bayesian mission manager of region A. For example, the result parameter for the node *"missing communication"* (ζ) is $P(\zeta|Pd_i, Ut_j)$, and the parameter for the node *"aircraft risk rate"* (η) is $P(\eta|Pd_i, Ut_j)$, where the subscripts represent the state numbers of variable Pd and Ut, so that $i = 1, 2, ..., 10$ and $j = 1, 2, 3$.

4.2.3 Inference of Management Requirements in Real-time

Based on the evidence data collected from simulation, the directed edge of Bayesian mission manager is quantified by the learned conditional probability distribution to specify the strengths of the causal influence. According to Equation (2.4), the joint probability of Bayesian mission manager is as follows:

$$P(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta, Pd, Ut) = \\ P(Pd)P(Ut)P(\zeta|Pd, Ut)P(\eta|Pd, Ut)P(\alpha|\zeta)P(\beta|\zeta)P(\gamma|\eta)P(\delta|\eta)P(\varepsilon|\eta). \tag{4.5}$$

The final decisions are the operating requirements, aircraft detection probability (Pd) and update rate (Ut), given current sensor status (ζ) and aircraft performance (η), which are represented by $P(Pd|\zeta, \eta)$ and $P(Ut|\zeta, \eta)$. Then, to obtain the final decisions, I apply Bayesian inference by summing out the joint probability over irrelevant variables,

$$P(Pd|\zeta, \eta) = \frac{P(Pd, \zeta, \eta)}{P(\zeta, \eta)} = \frac{\sum_{\alpha,\beta,\gamma,\delta,\varepsilon,Ut} P(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta, Pd, Ut)}{\sum_{\alpha,\beta,\gamma,\delta,\varepsilon,Pd,Ut} P(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta, Pd, Ut)}, \tag{4.6}$$

and

$$P(Ut|\zeta, \eta) = \frac{P(Ut, \zeta, \eta)}{P(\zeta, \eta)} = \frac{\sum_{\alpha,\beta,\gamma,\delta,\varepsilon,Pd} P(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta, Pd, Ut)}{\sum_{\alpha,\beta,\gamma,\delta,\varepsilon,Pd,Ut} P(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta, Pd, Ut)}. \tag{4.7}$$

4.3 Simulated Experiments and Results

4.3.1 Scenario Description

In the testing scenario, same as the training simulation, there are 5 subdivided regions in the given coverage area, region A, B, C, D and the surrounding region. The testing scenario is also modeled over a 1,000 second period. A complete schedule is required to be generated every 4 seconds. There are 40 sensors uniformly located in the whole area with 15 of them as RT and the rest as RO. The sensor positions remain close to a hexagonal arrangement, and are assigned uniformly. Figure 4.7 shows the sensor placement. The sensors are simulated to have a certain probability of failure, which affects the detection performance.

There are 30 aircrafts moving around this area. The aircraft motion model returns the aircrafts' positions as a function of time. In this chapter, the aircraft motion is simulated in the three-dimensional space. This is done by designing

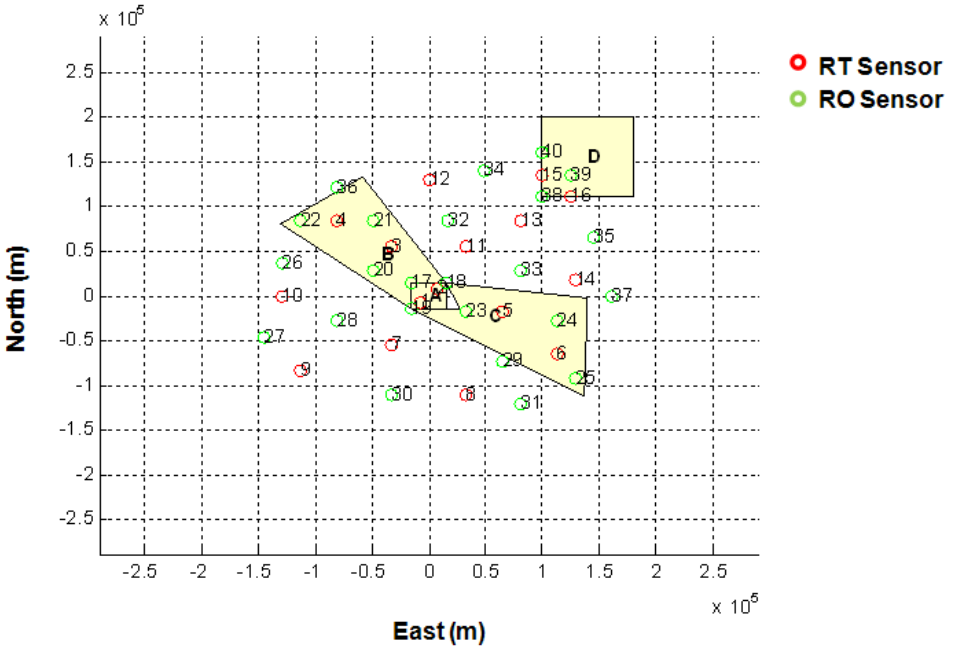


Figure 4.7: Sensor placement.

actual flight paths for each aircraft by choosing key positions, which are random with respect to distance and phase from the center. Periodically, each aircraft comes close to the center in region A. At this point, the aircraft descend closer to the ground, as if landing, and ascend, as it takes off on the runway. Figure 4.8 shows a snapshot of the aircraft while in motion.

4.3.2 Simulation

Prior to running the Bayesian mission manager, the sensor status and aircraft profiles are collected from the training simulation to initialize the Bayesian network off-line, as described in Section 4.2.2. The main objective of the testing simulation is to run the system with the Bayesian mission manager, so that the mission manager automatically updates the operating requirements in real-time for each region by monitoring the aircraft's movements, current sensor capabilities, and dynamically changing environment. The Bayesian mission manager maintains the best overall performance possible for the entire sensor network system.

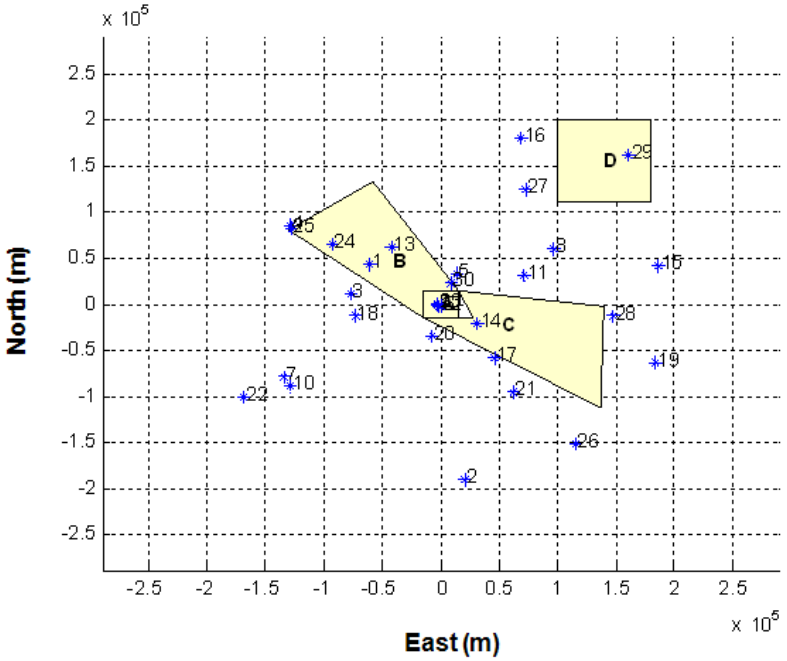


Figure 4.8: Snapshot of aircraft positions.

The probability of detection (Pd) requirement is divided into 10 states, ranging from 0.91 to 1.00 for region A, B, C, D, and another 10 states from 0.80 to 0.90 for the surrounding region. The update time (Ut) requirement is one of three states: updating sensor schedule every 1 second, 2 seconds, and 4 seconds. The final decisions of Bayesian mission manager for each region are selecting the Pd and Ut with maximum posterior probability given by the sensor status and aircraft profiles at current time.

The prior probabilities are set uniformly for detection probability and update time requirements: $P(Pd_i) = 1/10$ and $P(Ut_j) = 1/3$, where $i = 1, 2, ..., 10$, and $j = 1, 2, 3$. The states of both variables, *missing communication* (ζ) and *aircraft risk rate* (η) are defined as 4 ranges from 0.0 to 1.0 with a step of 0.25. They are conditionally independent to each other given Pd and Ut. In a given region, with the learned Bayesian mission manager, the joint probabilities of current performance are analyzed as follows:

$$P(\zeta_k, \eta_l, Pd_i, Ut_j) = P(\zeta_k|Pd_i, Ut_j)P(\eta_l|Pd_i, Ut_j)P(Pd_i)P(Ut_j), \tag{4.8}$$

where the subscript represents the state number of each variable happening at that time, and the conditional probabilities of $P(\zeta_k|Pd_i, Ut_j)$ and $P(\eta_l|Pd_i, Ut_j)$ are the Bayesian parameters learned from training dataset. These joint probabilities are then given as inputs to the Bayesian mission manager during the testing simulation.

When simulating the whole system with Bayesian mission manager in real-time, the state combination of missing communication percentage and aircraft risk rate, (ζ_k, η_l) is chosen based on the system performance at current time. Then, the interesting posterior conditional probabilities are evaluated as follows,

$$P(Pd_i|\zeta_k, \eta_l) = \frac{P(Pd_i, \zeta_k, \eta_l)}{P(\zeta_k, \eta_l)} = \frac{\sum_{Ut_j} P(\zeta_k, \eta_l, Pd_i, Ut_j)}{\sum_{Pd_i, Ut_j} P(\zeta_k, \eta_l, Pd_i, Ut_j)}, \tag{4.9}$$

and

$$P(Ut_j|\zeta_k, \eta_l) = \frac{P(Ut_j, \zeta_k, \eta_l)}{P(\zeta_k, \eta_l)} = \frac{\sum_{Pd_i} P(\zeta_k, \eta_l, Pd_i, Ut_j)}{\sum_{Pd_i, Ut_j} P(\zeta_k, \eta_l, Pd_i, Ut_j)}. \tag{4.10}$$

At last, the values of Pd and Ut are returned to PSO, which correspond to the states with maximum posterior conditional probability.

4.3.3 Results of Bayesian Mission Manager Performance

The overall system is run over a 1,000 second period with and without the Bayesian mission manager, respectively. All aircraft belonging to one region have their own operating requirements associated with this region. In the circumstance of without Bayesian mission manager, the requirements are set as default values from Table 4.1 for each region, which are constant through the whole testing period. Conversely, when the system is run with Bayesian mission manager, the requirements are automatically updated by the mission manager every scheduling interval.

As mentioned before, each region updates its own operating requirements to obtain more successful communication exchanges between aircraft and sensor sets in it. In other word, one successful communication happens only if the

aircraft detecting the required amount of interrogation messages sent by sensors, within a certain update period. Therefore, to evaluate the system performance for each scheduling interval, I compute the probability of unsuccessful communication for each region, which is the ratio of the number of aircraft that have not met the requirements to the total aircraft number in this region.

$$\begin{aligned} &Probability\ of\ unsuccessful\ communication \\ &= \frac{No.\ of\ Aircraft\ if\ Pd\ <\ Pd_{required}}{Total\ No.\ of\ Aircraft\ in\ Region}. \end{aligned} \tag{4.11}$$

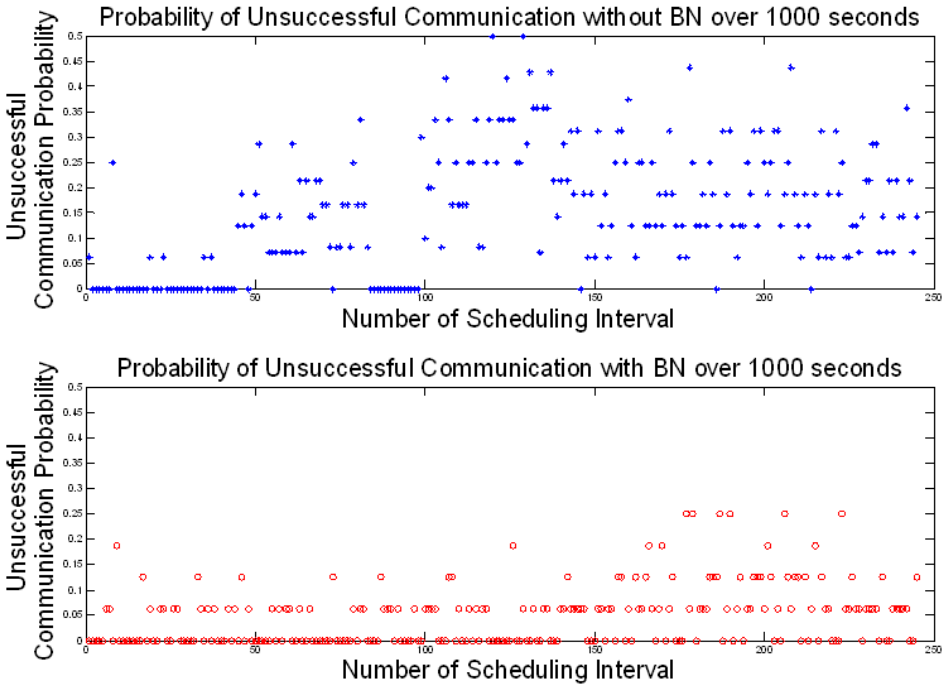


Figure 4.9: Probability of unsuccessful communication with and without Bayesian mission manager over 1,000 seconds in region A.

The performance of Bayesian mission manager is evaluated by the probability of unsuccessful communication, which is expected to be as small as possible. Figure 4.9 is the comparison between the probabilities of unsuccessful communication with and without the Bayesian mission manager over 250 scheduling intervals for region A. In the upper plot, each blue point rep-

resents a percentage value of unsuccessful communication in one scheduling interval without the Bayesian mission manager. Similarly, the red points in the lower plot represent percentage values of unsuccessful communication with the Bayesian mission manager. As shown in this figure, there are more red points falling on the *x-axis* than the blue ones. This result means that more scheduling intervals have completely successful communication with the Bayesian mission manager than without it. Moreover, the probability of unsuccessful communication without the Bayesian mission manager reaches 50% for several scheduling intervals, whereas, the highest probability with the Bayesian mission manager is no greater than 25%.

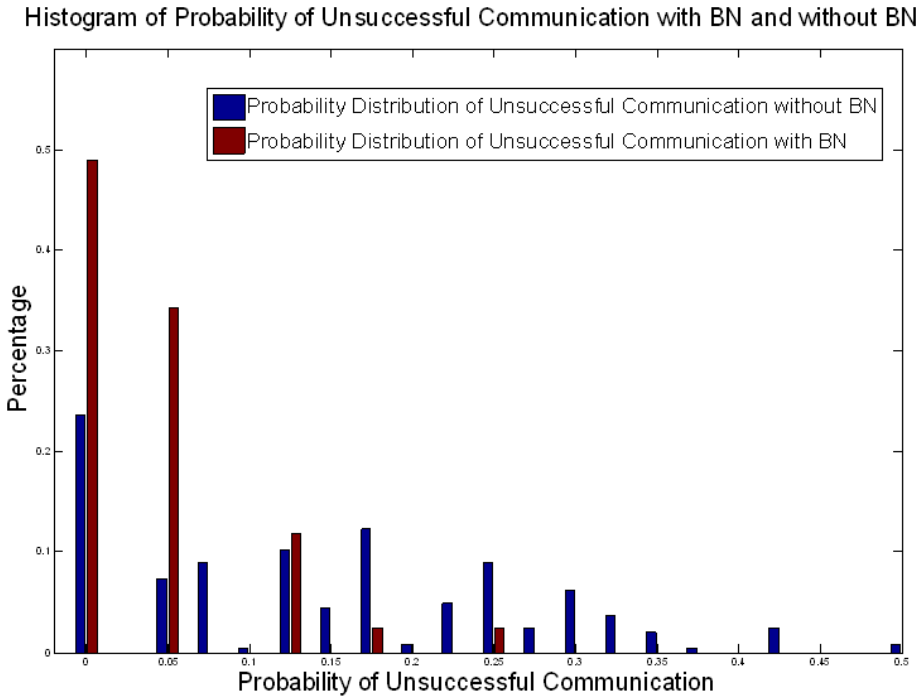


Figure 4.10: Probability distribution of unsuccessful communication with and without Bayesian mission manager over 1,000 seconds in region A.

Figure 4.10 is another way to illustrate the performance of Bayesian mission manager. It shows the probability distribution of unsuccessful communication with and without Bayesian mission manager in region A. In this figure, the red bars are much higher than the blue ones when closing to value zero. It indi-

cates that the probability of unsuccessful communication is equal to zero more frequently with Bayesian mission manager than without it. In other words, the Bayesian mission manager improves the system performance enormously by increasing the percentage of completely successful communication in region A from 24% to 49%. Moreover, the red bars disappear when probability of unsuccessful communication is greater than 25%, which is consistent with Figure 4.9.

Table 4.3: Unsuccessful communication grouped by regions, cumulative over all scheduling intervals

	A	B	C	D	Surrounding
Without BN	0.0249	0.1077	0.0149	0.6892	0.1155
With BN	0.0064	0.0770	0.0109	0.7159	0.1459
Improvement	74.30%	28.51%	26.85%	-3.87%	-26.32%

Table 4.3 compares the probabilities of unsuccessful communication averaged over all scheduling intervals in the two situations mentioned above. The Bayesian mission manager gives better performance for regions with higher priority, such as region A, B, and C. The percentage of unsuccessful communications in region A has decreased by 74.30% in the best situation. This better performance is due to the fact that aircrafts are landing or taking off in region A, which in turn causes a higher aircraft density and maneuver concentration. The higher aircraft density and sharper maneuvers trigger the Bayesian mission manager to assign more interrogations reducing the risk of no observations. Region D experiences a degradation in performance due to the poor placement of sensors and relatively lower aircraft density. In the surrounding region, the region is too large causing averaging of the statistics over many aircrafts causing the mission manager to make bad decisions. In a word, the Bayesian mission manager is more effective for the crucial regions, such as region A, B, and C in this case.

4.4 Conclusion

In most models and systems of sensor management designed for air traffic control, a common critical problem is to achieve simultaneously computing efficiency and communication efficiency between sensors and aircraft. In the complex environment where message exchange among sensors and aircraft is intensive, to decide the best action of sensors for new arriving message from aircraft during on-line control is a difficult task.

In practice, most sensor management units are modeled as a rule-based expert system, in which decisions are made by inferring answers from domain-specific principles. For example, in Table 4.1, the default requirements are obtained by interviewing domain experts, such as airport controller and sensor scheduler, and/or learning from experience. The rule-based sensor management system collects interviews and observations into a knowledge base, and uses an inference engine to set the operating requirements that are appropriate for certain situation (38, 39). The knowledge base is usually formalized as a set of *if-then* rules, such as:

- *If region = A, then required detection probability = 0.97*.
- *If update rate = 1 second, then minimum number of receiving sensors = 4*.

The conventional rule-based sensor management system has ability to preserve human experience, and is easy to be developed. However, the air traffic control problem continuously exhibits different levels of uncertainties, such as air turbulence, sensor noise, and the change of the flight mode. Therefore, the rule-based expert system can hardly balance the allocation of sensor resources and on-going changing operating requirements due to region priorities.

The new sensor management system, in this chapter, uses Bayesian network as a mission manager to automatically update the sensors' real-time operating requirements based on monitoring system performance, dynamically changing environment, and current sensor capabilities. Hence, it efficiently balances the overall system performance by allocating more sensor resources to the regions with higher priority.

Another advantage of this new system is that its loop nature of system design returns the feedback of current sensor settings into next scheduling in-

terval, which makes it be able to respond to the dynamically changing environment in real-time. By contrast, to achieve a respond to dynamic environment with the rule-based sensor management system, decisions for every specific situation have to be contained in its rule base. The rule-based system undoubtedly increases the search difficulties and reduces the computing efficiency.

These advantages are demonstrated with the comparison of running the whole sensor management system with and without the Bayesian mission manager. In fact, the experiments without Bayesian mission manager are the rule-based expert system with a knowledge base shown in 4.1. As a result, the system performance with the Bayesian mission manager has a significant improvement for the highest priority region, such as the airstrip area.

5

Intelligent Failure Detection for Wind Turbines

Wind is an important renewable energy source. The energy and economic return from building wind farms justify the expensive investments in doing so. However, without an effective monitoring system, under-performing or faulty turbines will cause a huge loss in revenue (17, 18). Therefore, to make wind energy more competitive in the future, efforts are required to enhance the availability, reliability and lifetime of the wind turbines.

Early detection of such failures helps prevent these undesired working conditions and allows the operators to develop maintenance plans with prioritized tasks (19). If failures are detected at an early stage, the consequent damage is minimized or mitigated, and also repairs are better scheduled. This leads to shorter down-times and lesser revenue losses. Therefore, diagnosis and prognosis of potential faults are crucial to maintain and improve the efficiency of the wind energy generation system (17, 40).

This chapter proposes an intelligent data-driven approach to monitor the turbine performance at real-time by fusing multiple test results and to detect the turbine abnormalities by tracking the turbine status variations. Optimization algorithms are applied to determine the fusion rules or the detection boundary in particular tests, objectively and optimally. This procedure is adaptable to each turbine using Supervisory Control And Data Acquisition (SCADA) system

data, automatically. My approach is advantageous in its applicability and data-driven nature, to monitor a large wind farm. Also, the test results have verified the effectiveness of my approach. This research work was collaborated with AWS Truepower, Inc. (2).

5.1 Wind Turbine Failure Diagnosis

5.1.1 Problem Description

A wind farm consists of multiple turbines, even hundreds, in some cases. The coverage of a farm can be on a complicated landscape with major obstructions due to local topographic features (41). This implies that each turbine may have its own operational characteristics, which complicates the fault detection (42). The turbine performance data is collected by the Supervisory Control And Data Acquisition (SCADA) system. The data is collected at certain time intervals using dozens of built-in sensors on the turbines that measure various physical quantities.

Some failures, such as the bearing and gearbox failures (43), cause the turbines to completely shut down, which is obviously detrimental to the turbine's performance, and they, therefore, have been studied thoroughly by many manufactures. Their diagnostic units are built into the wind turbines. These failures are called hard failures. For example, the data currently available from SCADA includes not only turbine measurements, but also some flagging of events from the existing built-in diagnostic units. Such flagging is designed for indicating hard failures, but it is not comprehensive in detecting other faults, especially the degradation of the faulty components. So, many soft failures, which degrade the turbine's performance but do not necessarily stop the turbine from running, are often overlooked, such as anemometer faults. This kind of soft failure can also be very harmful to the long term efficiency of the turbines, in the same way.

Among the soft failures, anemometer failure is hard to be detected but most badly effect the turbine production. Anemometer measures the wind speed as

detected by the turbine. The wind speed measurement is used for configurations and control settings of the turbine and, hence, very important for online monitoring of turbine performance (44). Firstly, a damaged or out of tolerance anemometer will effect the estimated energy production of a site. The inability to detect bad anemometers can dramatically effect estimated return on investment. For example, Figure 5.1 shows how a 2% error in an estimate of average annual wind speed can result in a 6% difference in power production, which directly effects the return on investment.

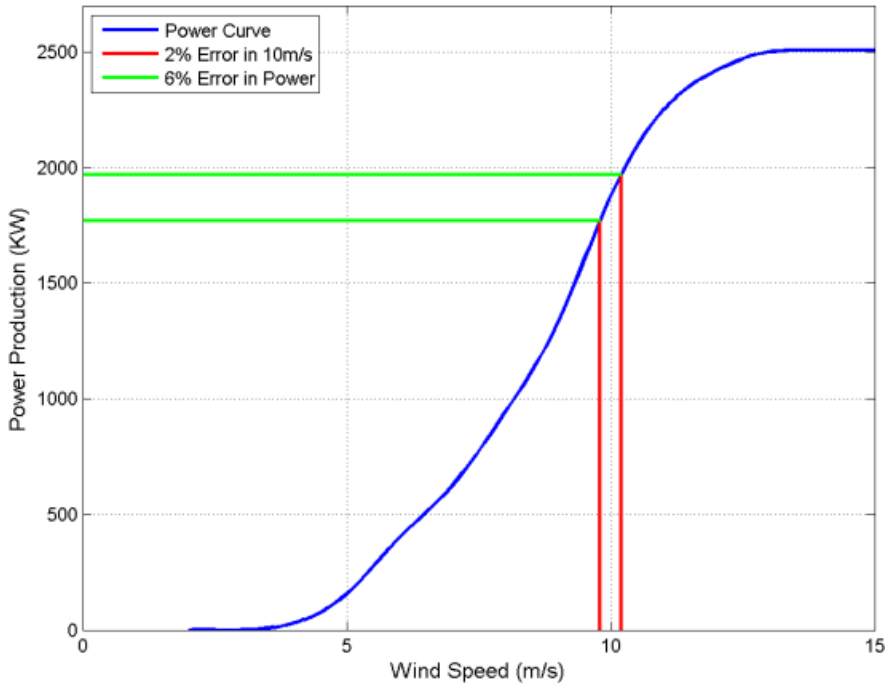


Figure 5.1: Power production curve for a typical large wind turbine, showing the effect of error in wind speed measurement on estimated power production (2).

Secondly, the wind speed measurement is an important control parameter for real-time operation, which is highly correlated with rotor speed, pitch angle, exported power, etc. Wind turbines have different operating modes that match each possible wind speed. The turbine starts generating electricity when the wind speed exceeds a lower-bound threshold, such as 5 m/s. Then the turbine increases its rotation speed as it reaches its maximum power production at a wind speed of approximately $15 - 18$ m/s. If the wind speed exceeds an

upper-bound threshold, such as 20 m/s, the wind turbine is stalled or braked to prevent damage or an accident. So, a faulty anemometer reading can cause severe damage because, if the turbine does not shut down at the right time, the turbine may not keep up and over heat, or a blade may come loose. Further, incorrect readings may also result in false alarms that the wind speed is higher than it really is and may frequently brake causing unnecessary downtime.

5.1.2 Research Objective

Currently, condition monitoring (CM) techniques are used to detect some specific hard failures at an early stage with separate systems designed by different manufacturers. These techniques, such as vibration, lubrication oil, acoustic emission signals, and generator current analysis, require the development of a variety of sensors and computationally intensive analysis techniques (45). Despite their applications in the wind industry, the condition monitoring systems have not yet proven their effectiveness due to the peculiarities of each wind turbine. For example, commercial condition monitoring systems mostly employ vibration-based techniques, which are sophisticated, and the sensors and cabling are costly. Lubrication oil analysis is becoming more popular for detecting gearbox tooth and bearing wear, but it cannot detect any failures outside the gearbox. Each manufacturer has basically chosen their own way of designing a condition monitoring system (46); therefore, the present condition monitoring techniques may not be suited to all wind turbine types and faults. Moreover, developing reliable wind turbine condition monitoring techniques require complex and lengthy collaboration between the farm operators and manufactures in the field.

Hence, the main objective of this research is to design a data-driven approach by only analyzing the turbine performance data collected from existing built-in sensors on each turbine, which makes this approach more adaptive to each turbine automatically. This approach can recoup some of the losses by catching not just hard failures but also soft failures early on, and then support maintenance plans to minimize the failures' impact. If a fault could be

detected at an early stage, the potential damage could be minimized or mitigated through early repair avoiding drastic breakdown of the wind turbine, which leads to less downtime and more revenue; meanwhile, the maintenance can be optimally scheduled with the regular maintenance trips to minimize the repair costs (47).

5.1.3 Failure Detection System Design

This chapter proposes to use data-driven approaches to learn the normal and abnormal patterns from the available data, which helps both sensor validation and diagnostics in an integrated way. The abnormalities are detected by tracking the turbine state variations (48). The data flow of the failure detection system is illustrated as Figure 5.2, which has 4 steps as follows:

1. Pre-processing:

   I develop multiple performance tests on several important turbine variables, such as power generated, rotor speed, pitch angle of individual turbine, and also on the wind speed difference between two sibling turbines that most likely see a similar wind speed, if they both work normally. These variables are illustrated in Table 5.1.

Table 5.1: Wind Turbine Variables Used in Designing Multiple Performance Tests (2)

Turbine Variable	Unit	Definition
Wind Speed (Vw)	m/s	Average measurement seen by the anemometer
Generated Power (P)	kw	Average real power produced
Rotor Speed (Vr)	revolutions per min	Average rotational speed of the rotor
Pitch Angle (Dp)	degrees	Average pitch angle of the blades
Wind Speed Difference (wsd)	m/s	Absolute wind speed difference between two sibling turbine $wsd = \|ws_A - ws_B\|$

2. Feature extraction:

   In each test, multiple states are defined to distinguish different working condition features, including complete shut-down, under-performing, abnormally frequent default, as well as, normal working. Through extensive data mining of historical data and verification from farm operators, some state combinations are discovered to be strong indicators of failures, such as spindle failures, lighting strikes, anemometer faults, etc., for fault detection.

3. Pattern recognition:

   Then, all test results are fused to reach a final conclusion used as a pattern, which describes the turbine working status at current time. It is more effective than any single test. I apply the particle swarm optimization (PSO) algorithm to recognize several patterns with fusion rules, which are determined in the feature space objectively and optimally.

4. Failure prediction:

   In this step, a Bayesian network (BN) is applied to automatically monitor the turbine working status and identify the potential failures with a percentage certainty based on any abnormal changes in the turbine performance. The main objective is to predict when and what types of failures are going to happen, so that the farm operators can fix the failures at an earlier stage to avoid major breakdowns.

The design of this failure detection system has the following advantages,

- The state-of-the-art Supervisory Control And Data Acquisition (SCADA) system in industry can only answer the question whether there are abnormal working states, but my evaluation of multiple states in multiple turbine performance tests is also promising for diagnostics and prognostics.
- These multiple tests are combined to reach a final conclusion, which is more effective than any single test. From the fused test results, one can gain a qualitative understanding of turbine performance status to detect

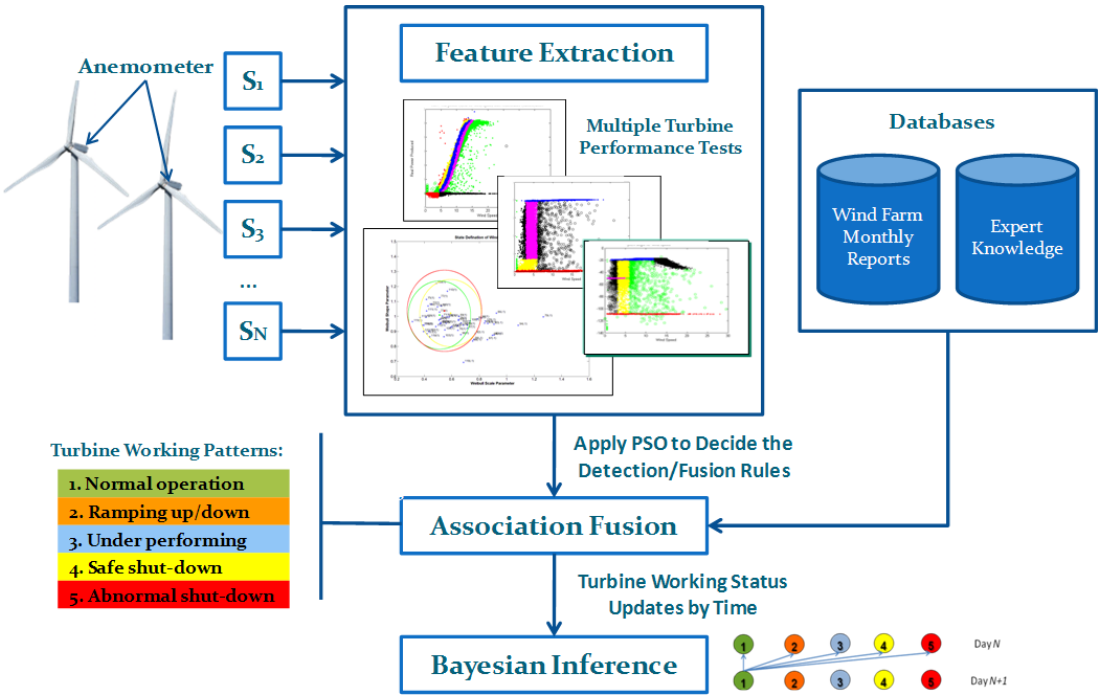


Figure 5.2: Data flow of wind turbine failure detection system, consisting 4 steps: pre-processing, feature extraction, pattern recognition and failure prediction.

faults and can also provide explanations on what has happened for detailed diagnostics.

- My approach is adaptable to each turbine automatically and is advantageous in its applicability and data-driven nature to monitor a large wind farm.

As described in Section 5.1.1, the wind turbine systems require a more reliable anemometer reading to ensure their efficiency. It is crucial to monitor the anemometer status and detect its failure at an early stage. One solution is to pair up all the turbines so that each turbine has a sibling. Two sibling turbines would most likely measure a similar wind speed at the same time. If the measured wind speed is drastically different, it could be due to anemometer faults, turbine malfunctioning or wind blockage. In this chapter, I also propose a new wind speed difference test as one of the multiple performance tests for particularly detecting anemometer failures. This test uses wind speed difference between a turbine pair to detect faults and fuse the results from multiple pairs to identify the turbine at fault relative to anemometer (49). To improve the detection accuracy of this wind speed difference test, I also design a new method to determine a circle-shaped decision boundary between the normal and abnormal states in its feature space using PSO algorithm. The weeks inside the circle are decided to be *normal*, when both turbines are functioning properly. The ones outside the circle are decided *faulty* or *idle*, where the *faulty* state often indicates soft failures, like anemometer faults, and the *idle* state often indicates hard failures that cause turbines to completely shut down. Compared to differential evolution (DE) and evolution strategy (ES) algorithms, PSO achieves better solution with lower fitness value in a shorter convergence time.

5.2 Multiple Turbine Tests on Critical Performance Variables

It is straightforward to analyze turbine performance to detect its faults. I choose three variables – power production, rotor rotating speed and blade pitch angle – significant to adjudicate on whether the turbine working properly (50). First of all, power production is one of the main yardstick to measure turbine performance. Once there is a failure occurring in a turbine, the power it produces will be effected badly. Second, the rotor component converts wind energy to low speed rotational energy. The faster it rotates, the more energy the turbine produces. Moreover, the blade pitch gives the turbine blades the optimum angle of attack. Allowing the angle of attack to be remotely adjusted gives greater control, so that the turbine collects the maximum amount of wind energy for the time of day and season. I measure these variables against wind speed, since the turbine performance is also determined according to how fast the wind blows.

5.2.1 Test 1: Generated Power vs. Wind Speed

The power curve, a plot of the generated power of the wind turbine averaged over unit time interval against the wind speed, is a key test for turbine health, since power production is the ultimate goal of the wind turbine.

The nominal power curve, provided by the manufacturers, is a discrete set of power versus wind speed $\{(w_1, p_1), (w_2, p_2), \ldots, (w_m, p_m)\}$. Yan (3) first fit the nominal power curve using a Gaussian cumulative distribution function (CDF), where the parameters of the Gaussian CDF are optimized by particle swarm optimization (PSO) algorithm. A Gaussian CDF fitting function is defined by

$$\hat{P}(w|c, a, s, m) = -m + s \cdot \int_{-\infty}^{w} \frac{1}{\sqrt{2\pi}a} e^{-\frac{(x-c)^2}{2a^2}} dx, \tag{5.1}$$

where w is the wind speed, $\hat{P}$ is the estimated power at specific w. In the Gaussian CDF fitting, c is its mean, a is the standard deviation, s is the scaling factor, and m is an extra shift. The common digits of fitting parameters found

by PSO are provided in Table 5.2. Then we construct the inverse function of the Gaussian CDF, $\hat{w} = f^{-1}(\hat{P})$, to linearize the nominal power curve.

Table 5.2: Gaussian CDF Fitting Coefficients (3)

	a	c	s	m
Gaussian CDF	2.684	9.020	1.087	0.026

The multiple states relative to the linearized nominal power curve are listed in Table 5.3. This linearization method simplifies the state definition because the nominal power curve needs to be fitted only once; meanwhile, the states, in the linearized domain, are defined by thresholds, rather than, the complicated boundary curves in the power curve domain, where those boundary curves require multiple fittings.

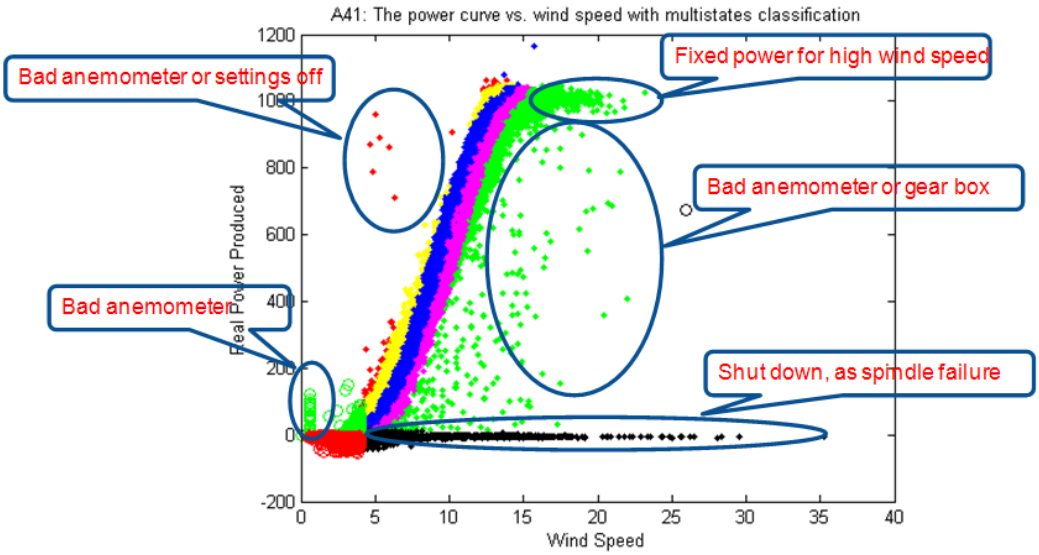


Figure 5.3: State diagram for power curve vs. wind speed.

The definition of the multiple states relative to the linearized power curve is shown in Table 5.3. The colored sections in Figure 5.3 identify different regions in the power curve that represent potential problems in the wind turbine, if the measurements change between regions (51). For example, the power measurements should reside in the yellow, blue, and pink regions. However, the

Table 5.3: State Definition of Linearized Power Values ($\hat{P}$) vs. Wind Speed (Vw)

State	Definition	Working Condition
1	$\hat{P} > 1.5$	Under-performing with soft failures
2	$0.5 < \hat{P} \leq 1.5$	Normal working
3	$-0.5 < \hat{P} \leq 0.5$	Normal working
4	$-1.5 < \hat{P} \leq -0.5$	Normal working
5	$\hat{P} \leq -1.5$	Under-performing with soft failures
6	Horizontal power	Shut down with hard failures
7	$Vw < 4 and \hat{P} > 0$	Shut down with soft failures
8	$Vw < 4 and \hat{P} \leq 0$	Abnormal defaults

scattered red points in the upper left are typical of a faulty anemometer in the turbine. So, if the measurements move to this upper left region, there is some soft failures, which are insufficient to stop operations yet still cause power production losses, such as faulty anemometer or gear box. Without timely maintenance, the turbine is continuously breaking, until it completely shuts down. Some hard failures, like spindle failure and lightning strike, will also cause the turbine to completely shut down. When it happens, the measurements move to the red or black horizontal region.

5.2.2 Test 2: Rotor Speed vs. Wind Speed

The next variable analyzed is rotor speed as a function of measured wind speed. The turbine generally operates at a constant rotor speed if there is adequate wind. More power is produced as the wind force increases but the rotor speed remains constant. The cycle for a productive wind turbine is to begin with a rotor speed increasing linearly for wind speed between 0 m/s and 3 m/s, then reaches a speed of 20 cycles/s from 3 m/s to 6 m/s, and finally stays at 20 cycles/s. Based on this working cycle of wind turbines, I classify 7 states of the rotor speed curve, as shown in Table 5.4. If the turbine is down or off due to some hard failures, the rotor speed is 0 m/s for all wind speeds, corresponding to the red horizontal region in Figure 5.4. Some of the mea-

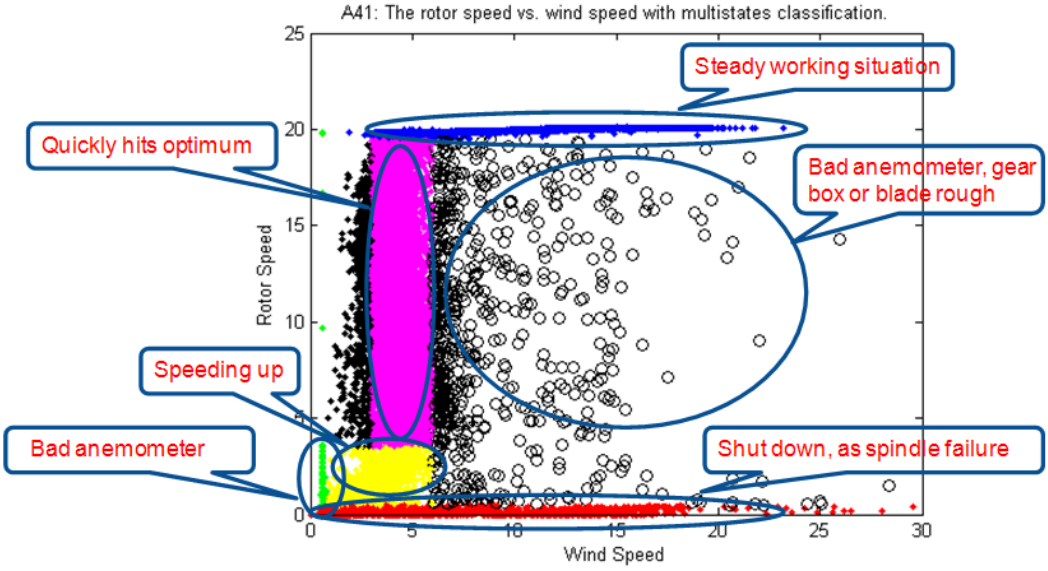


Figure 5.4: State diagram for rotor speed vs. wind speed.

surements in the black region are caused by various types of failures, such as bad anemometer or rough blade.

5.2.3 Test 3: Pitch Angle vs. Wind Speed

The third test analyzes the pitch angle against the measured wind speed. Similar to the previous test, pitch angle keeps around -20 degree when the turbine is steadily working, as shown in the blue region in Figure 5.5. If the wind speed is really high, the pitch turns away the coming wind direction to avoid the damage. This condition is given in the upper right black region. Some of the measurements are in the pink line corresponding to -50 degree pitch angle. Since the pitch is set to -50 degree as preventative angle before the turbine is going down, this state is used to do prognostics of the turbine faults. The state two, the lower vertical green line, is a critical indicator of the lightning event happening to the turbine.

Table 5.4: State Definition of Rotor Speed (Vr) vs. Wind Speed (Vw)

State	Definition	Working Condition
1	$Vr < 0.5$ (Horizontal state)	Shut down with hard failures
2	$Vw < 0.5$ (Vertical state)	Anemometer failures
3	$0.8 \leq Vw \leq 6\ and$ $0.5 \leq Vr \leq 3.5$	Ramping up/down
4	$3 \leq Vw \leq 6\ and$ $3.5 \leq Vr \leq 19.5$	Normal working
5	$Vr > 19.5$	Normal working in high wind speed
6	$AbnormalVrwhenVw < 3$	Under-performing, Low wind speed but relatively high rotor speed
7	$AbnormalVrwhenVw > 6$	Under-performing, High wind speed but relatively low rotor speed,

5.3 Anemometer Failure Detection by Wind Speed Difference Test

An anemometer measures the wind speed as detected by the turbine. This measurement is affected by several factors like elevation, turbulence at the turbine, other topographical factors or simply noise. However, in a wind farm, there are turbine groups that see similar wind flow, either due to their physical proximity, or their similar configuration in a cluster of turbines including surrounding geography (52). These turbines are called sibling turbines. It is fair to assume that if the sibling turbines all perform properly, then they should measure similar wind speed. Based on this similarity, I begin to compare wind measurements between turbine pairs. If they begin to differ, this is a good indication that one of the wind turbines requires maintenance. I collected a week's worth of data and model the wind speed difference using the Weibull distribution, as the Weibull distribution is very practical for reliability testing (53, 54). The detector uses the estimated Weibull parameters to define the normal and abnormal states of wind turbines. The abnormal wind speed difference patterns are caused either by a faulty anemometer, directly, or by other faults, indirectly. For instance, if a turbine is shut down due to major component fail-

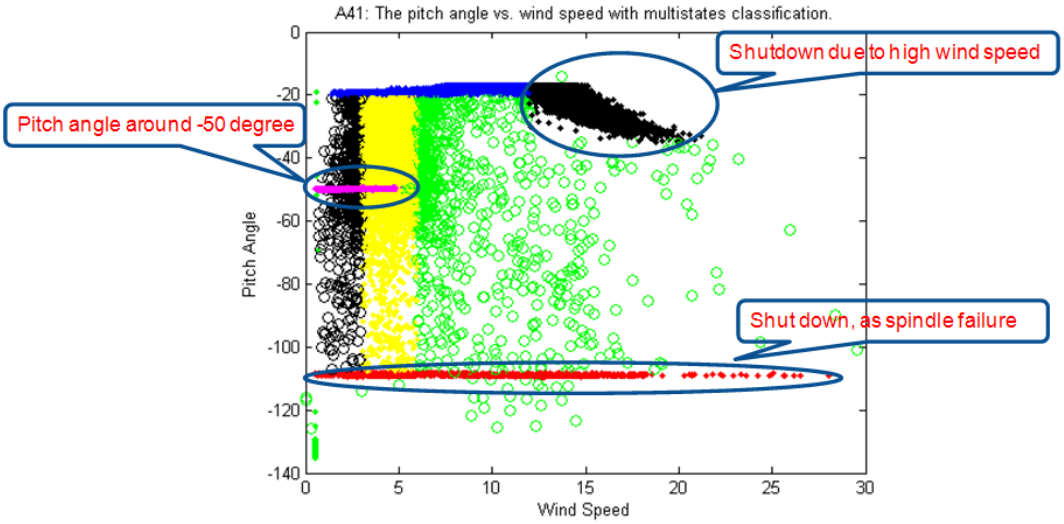


Figure 5.5: State diagram for pitch angle vs. wind speed.

ure, or if the turbine is under the influences of lightning, the anemometer of that particular turbine will produce very small readings causing detectable discrepancies in the wind speed measurements of the turbine pairs. Earlier in this chapter, I have designed multiple tests on other sensor data to exclude faults not caused by anemometers. Due to the high variation of wind speed, a direct point-to-point comparison of wind speed differences easily triggers false alarms. Therefore, a week's worth of wind speed data is aggregated to increase the robustness of the detection. This method is employed by C. A. Cassity and D. Parker, who won the Prognostics and Health Management (PHM) 2011 Data Competition (55).

5.3.1 Two-Dimensional Failure Detector

The wind speed measurement is used as an independent scale variable in my modeling efforts. According to (44), wind speed is Weibull distributed. Going forward, we assume that wind speeds at different wind turbines are uncorrelated Weibull random variables.

I choose a turbine, denoted as A, and its closest neighboring turbine, B, to evaluate the difference in their wind speed measurements. These two turbines

Table 5.5: State Definition of Pitch Angle (Dp) vs. Wind Speed (Vw)

State	Definition	Working Condition
1	$-110 < Dp < -108$ (Horizontal state)	Shut down with hard failures
2	$0.5 < Vw < 0.7$ (Vertical state)	Anemometer failures
3	$Dp = -50\ degree$	Default setup
4	$3 \leq Vw \leq 6\ and$ $-108 \leq Dp \leq -20$	Normal working or ramping up/down
5	$Vw \leq 12\ and\ Dp > -20$	Normal working
6	$12 \leq Vw \leq 22\ and$ $-35 \leq Dp \leq -15$	Switch pitch angle to avoid damage caused by high wind speed
7	$AbnormalDpwhenVw < 3$	Under-performing in low wind speed
8	$AbnormalDpwhenVw > 6$	Under-performing in high wind speed

are closest to each other, and there are no other turbines around them to interfere with the air flow around them. They make a sibling turbine pair. We divide the data set into individual weeks that contain n data points or less, if data is missing. For each week, we calculate the absolute values of wind speed difference between A and B as

$$wsd = |ws_A - ws_B|. \tag{5.2}$$

where ws_A and ws_B are uncorrelated Weibull random variables.

Since wind speed as seen by the turbine follows a Weibull distribution, the weekly data of wind speed difference wsd is also fitted into a two-parameter Weibull distribution of

$$pdf(wsd; \lambda, k) = \frac{k}{\lambda}(\frac{wsd}{\lambda})^{k-1} e^{-(wsd/\lambda)^k}, \tag{5.3}$$

where k and λ are the estimated shape parameter and scale parameter, respectively.

Since both turbines ideally should see similar wind speeds, the Weibull probability density distribution should match and hence produce little spread. The less spread of the Weibull probability density distribution of wind speed

difference, the better the anemometer functioning on the two turbines. Figure 5.6 (1), (2), and (3) demonstrate the *normal*, the *faulty* and the *idle* states of weighted histograms of weekly wind speed difference, respectively, which are most confidently Weibull distributed. I consider a Weibull distribution with one pair of estimated shape and scale parameters as a single state for the wind speed difference variable under analysis. It is obvious that there are significant differences among the spread of the three distributions.

The *normal* state, such as in week 31 in Figure 5.6 (1), has a sharp distribution. Its default condition is defined as

$$\text{If } k_i > 0.8 \text{ and } 0 < \lambda_i < 0.8, \text{ then } pdf(wsd_i; k_i, \lambda_i) \in \textit{normal} \text{ week,}$$

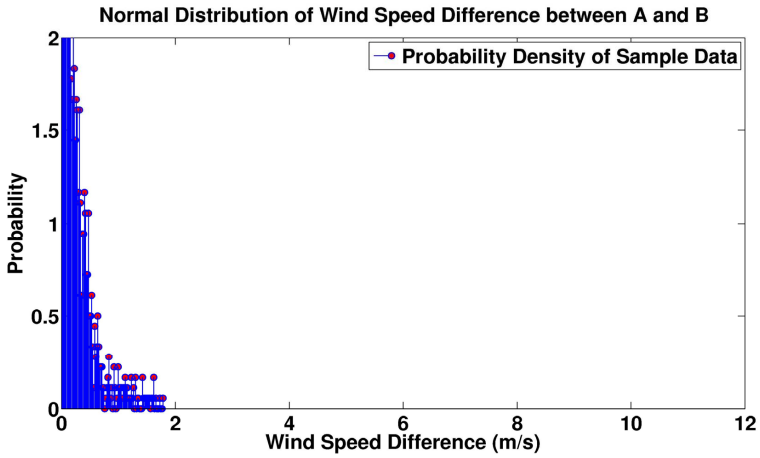
where i is the week index. The wind speed difference in this week is all less than 2 m/s, and the probability of zero wind speed difference is close to 1, which means that the wind speed difference is consistently small. Moreover, it has a better performance since we have a good estimate of the wind speed difference with a high probability. If we need to predict the wind speed difference within 1 m/s, we know the difference will be 1 m/s with nearly 100% probability. This implies that both turbines work similarly, and the probability that they both work well is high.

The *faulty* state, such as in week 79 in Figure 5.6 (2), has a more spread distribution between 0 m/s and 6 m/s with a larger scale parameter, and its shape parameter is greater than 1. Its default condition is defined as

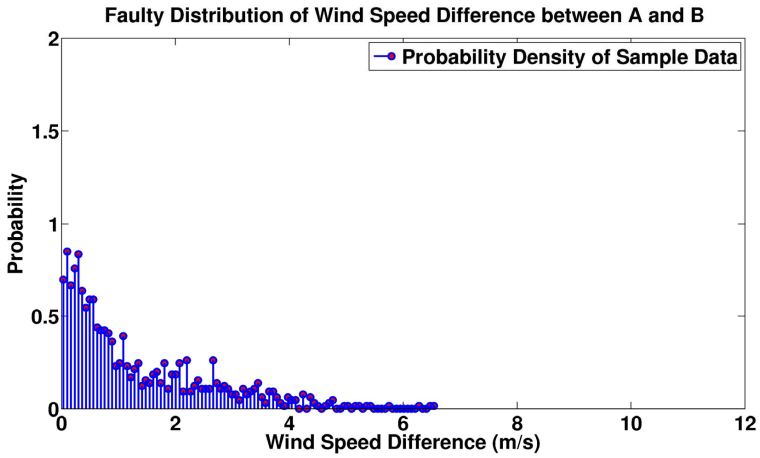
$$\text{If } k_i > 1 \text{ and } \lambda_i > 0.8, \text{ then } pdf(wsd_i; k_i, \lambda_i) \in \textit{faulty} \text{ week.}$$

This distribution is indicative of a soft failure in one of the turbine anemometers, mostly due to aging. The faulty turbine's performance degrades gradually. However, we cannot conclusively say, which turbine in fact has an anemometer problem without a further wind speed difference test.

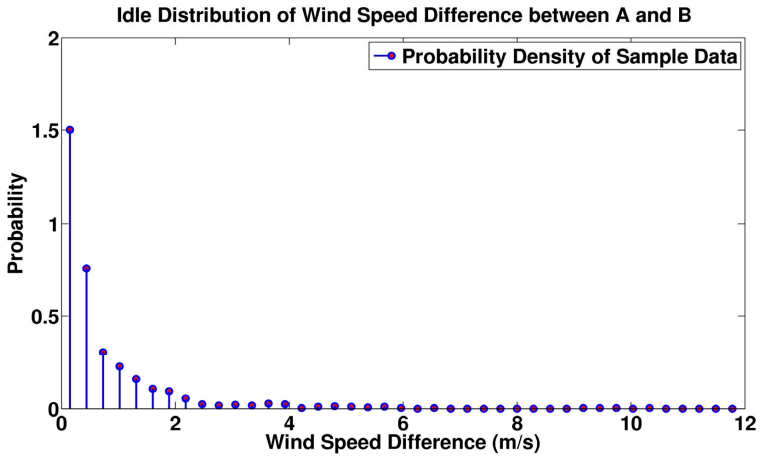
The *idle* distribution in Figure 5.6 (3), as an example of week 118, is flattened out with a long tail. The probabilities for wind speed difference higher than 2 m/s are significantly lower. This kind of Weibull distribution has a large scale parameter and a shape parameter less than 1. Its default condition is defined as



(1) normal in week 31



(2) faulty in week 79



(3) idle in week 118

Figure 5.6: Examples of normal, faulty, and idle distributions of wind speed difference between turbine A and B.

If $0 < k_i < 1$ and $\lambda_i > 0.8$, then $pdf(wsd_i; k_i, \lambda_i) \in$ *idle* week.

An intuitive explanation of such a scenario is that one turbine completely shuts down because of hard failures such as lightning or major component failure, and its anemometer is also turned off to read near-zero wind speed. As a result, the wind speed difference distribution is almost the same as the wind speed distribution of the other normal turbine.

To sum up, the turbine working status is reflected by the estimated Weibull distribution of the weekly wind speed difference data, represented by the scale and shape parameters. Based on this phenomenon, we can detect turbine failures by plotting the pairs of Weibull scale and shape parameters, used as a single state of turbine working condition. And, the main goal is, in the 2-dimensional plot of Weibull scale and shape parameters, to reveal the normal and abnormal regions of turbine performance.

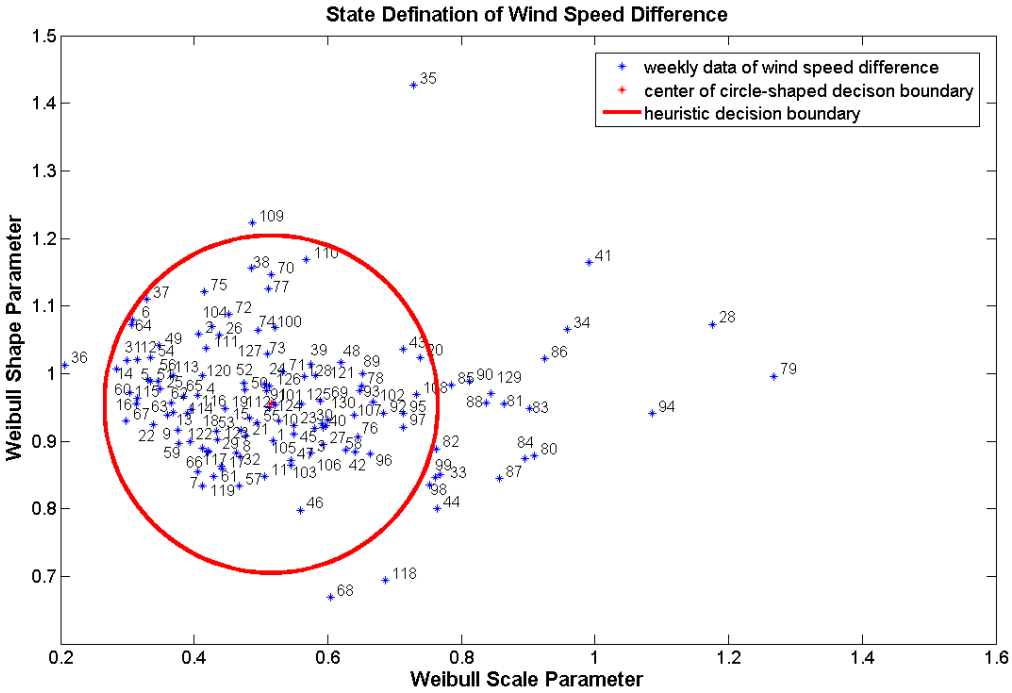


Figure 5.7: Estimated Weibull parameters of weekly wind speed difference between turbine A and B.

I analyze the Weibull probability density function for the 130 weeks worth of data. If the scale parameter λ is larger, the distribution is more spread out, such as in week 79 and week 118. If λ is smaller, the distribution is more concentrated, such as in week 31. On the other hand, if the shape parameter k is close to 1, the distribution is sharper giving small wind speed differences a higher probability. Figure 5.7 shows the 2-dimensional plot of the estimated Weibull scale and shape parameters of weekly wind speed difference between the turbine A and B. The red spot is the cluster center of all parameters. A circle with this center represents the *normal* functioning of the anemometers on both the turbines. In fact, the circle acts as a detection threshold for *faulty* and *idle* states. The weeks inside the circle pertain to a *normal* state, when both turbines are functioning properly, and the ones outside the circle are functioning less optimally in a *faulty* or *idle* state. Notice that week 79 and week 118 are both outside the circle. The two anomalous points in 2-dimensional space are indicative of faulty functioning. However, with two distinct characteristics, they represent different types of failures as discussed above.

In the 2-dimensional plot, the *faulty* state with the shape parameter k greater than 1 and scale parameters λ greater than 1 on the top-right portion of the circle, which we can expect abnormal functioning of the anemometer. However, the ones with the shape parameter k less than 1 on the bottom-right of the circle may or may not have a failure in the anemometer. For example, verified with the corresponding flags of true events from SCADA data, the abnormal performance in both week 68 and week 118 are due to lighting strikes. If considering the points with both shape and scale parameters higher than 1, I get Weibull distributions that spread over a wide range of wind speeds with a close resemblance to a uniform distribution (distribution which is more flat across the wind speed differences). This is indicative of an abnormal functioning of anemometer as well as poorly performing other parts of the turbine. In this approach, I focus on the points above the value of 0.9 for both scale and shape parameters. The weeks that fall in this category are weeks 86, 34, 41, 94, 28, and 79 from left to right in Figure 5.7. Through the verification from farm operators, except that the weeks 34 and 41 are due to missing data, the bad performance in all the other weeks are caused by the failures related to the anemometer.

5.3.2 One-Dimensional Failure Detector

As a way to measure the integrated effects of the scale and shape parameters, I design another test based on the area under the Weibull cumulative distribution function (CDF). The area definition is given later in Equation (5.5). The Weibull CDF function is defined as

$$\text{cdf}(wsd; \lambda, k) = 1 - e^{-(wsd/\lambda)^k}, \tag{5.4}$$

where wsd is the Weibull random variable of wind speed difference. If both turbines work well and similarly, the cumulative probability function rises to value 1 fast. In such case, the two turbines rarely have different wind speeds. Otherwise, if one turbine is faulty, then the wind speed difference is larger, and the Weibull cumulative distribution approaches 1 slower, which corresponds to a diagonal or nearly linear function. For instance, Figure 5.8 illustrates that the Weibull cumulative distribution of wind speed difference in week 31 is steeper than those in week 79 and week 118, and week 31 is a *normal* week when both turbines work well.

As shown in Figure 5.8, when the Weibull cumulative distribution rises to 1 fast, the curve covers more area under it. So we consider the normalized total area under the CDF curve (AUC) as an indicator of such a phenomenon,

$$\text{AUC}(wsd = wsd_{max}; \lambda, k) = \frac{\int_0^{wsd_{max}} \text{cdf}(w; \lambda, k) dw}{\int_0^{wsd_{max}} 1 dw}, \tag{5.5}$$

where the infimum of the area is

$$\text{AUC}(wsd = wsd_{max}; \lambda, k) \to \frac{\int_0^{wsd_{max}} 1 dw}{\int_0^{wsd_{max}} 1 dw} = 1, \tag{5.6}$$

and in our application, we have $wsd_{max} = 25$ here. The bigger the AUC, the better both turbines work. Figure 5.9 plots the AUC curves versus week numbers. An area of $\int_0^{25} 1 - e^{-(wsd/0.9)^{0.9}} dw = 0.96382$, shown as the green straight line, serves as a reference when the scale and shape parameters are both 0.9. The line is used as the threshold to declare problematic weeks for assigning the

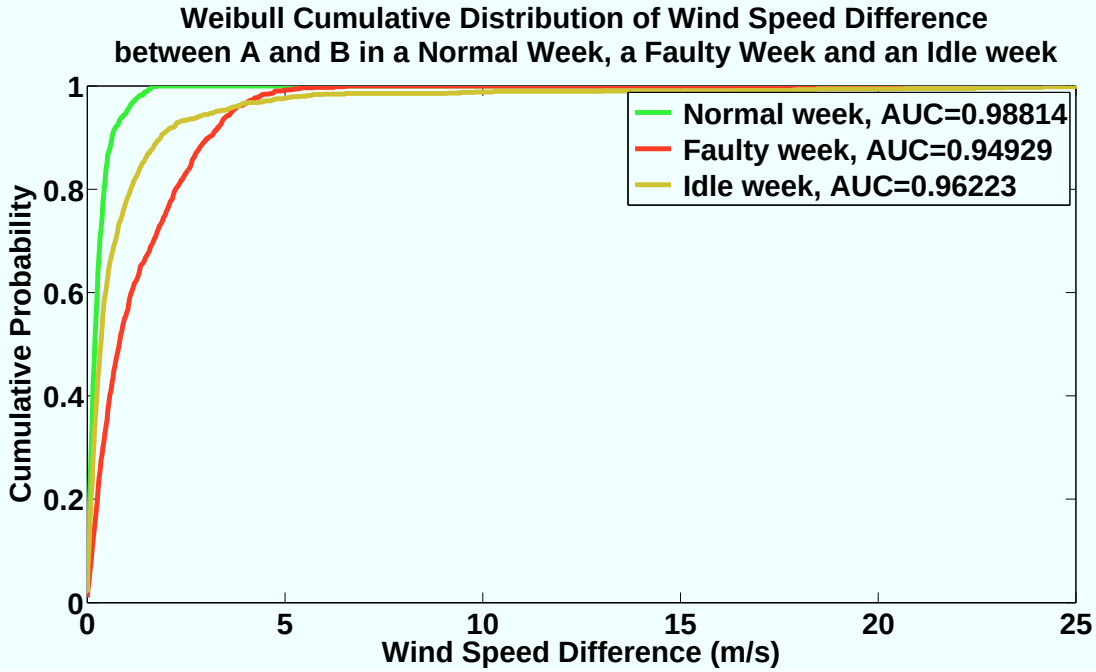


Figure 5.8: Weibull cumulative distribution of wind speed difference between A and B in week 31, 79, and 118.

weekly health flags later on. The one-dimensional failure detector obtains the same detection results as the detector of two dimensions, which is declaring weeks 86, 34, 41, 94, 28, and 79 to be abnormal weeks.

5.3.3 Relation between 2D and 1D Failure Detectors

In this chapter, the 2-dimensional method uses 1-dimensional method to help training the decision boundary, so 2-dimensional method is enhanced by 1-dimensional method. On the other hand, if only using 1-dimensional method, we can't tell the hard failures and soft failures apart. These two methods complement each other, and we do not compare them.

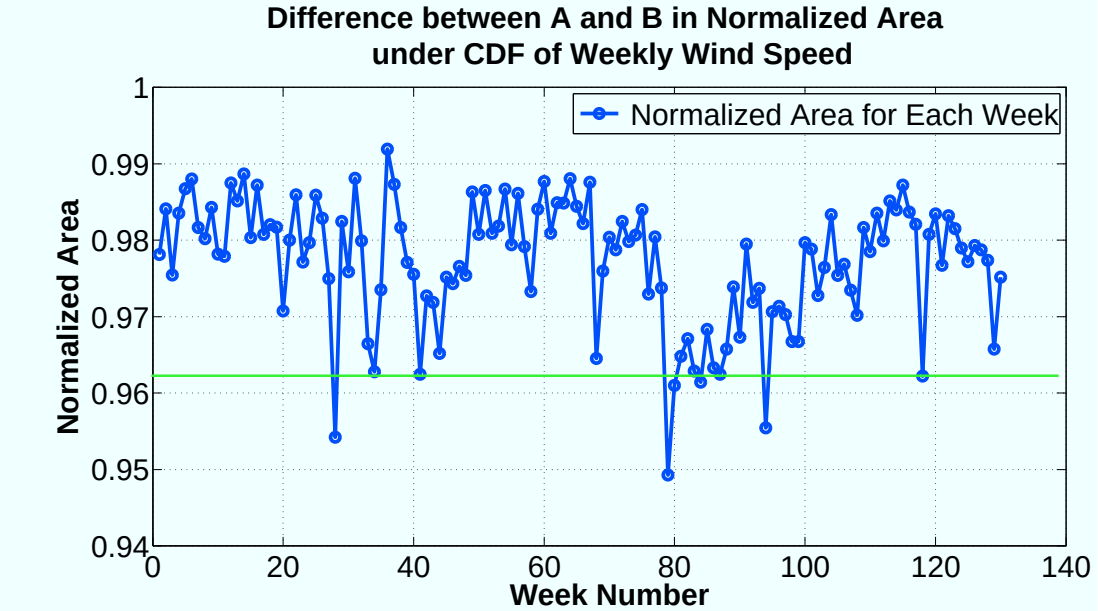


Figure 5.9: Normalized area under Weibull cumulative distribution of weekly wind speed difference, AUC, between turbine A and B.

5.4 Particle Swarm Optimization Based Method for Failure Detection Boundary Determination

The previous section sums up the turbine working status which is reflected by the estimated Weibull distribution of the weekly wind speed difference data, represented by the scale and shape parameters. Based on this phenomenon, the main goal is to reveal the normal and abnormal regions of turbine performance, in the 2-dimensional plot of Weibull scale and shape parameters (49).

In order to determine the decision boundary between the normal and abnormal states in the wind speed difference failure detector objectively and optimally, I propose to use the particle swarm optimization (PSO) algorithm to learn from the historical data, in this section. This new method determines a circle-shaped decision boundary between the normal and abnormal states. The weeks inside the circle are decided to be *normal*, when both turbines are functioning properly. The ones outside the circle are decided *faulty* or *idle*, where the *faulty* state often indicates soft failures, and the *idle* state often indi-

cates hard failures.

Model parameterization is critical for accurate diagnosis (56), and PSO is an effective optimization algorithm to automate the parameter estimation. PSO is inspired by social behaviors, where a population of particles search through the solution space to find the global optimum of a fitness function defined for each specific application. Each particle represents a complete solution and moves in the search space using both cognitive awareness and social influence. The PSO algorithm has been widely applied in NP-hard optimization problems.

The optimal decision boundary between normal and abnormal states is designed as the boundary to minimize the errors due to missed detection and false alarms on the training data. I also compare the PSO algorithm with differential evolution (DE) and evolutionary strategy (ES) algorithms. It turns out that the PSO algorithm achieves the lowest fitness value in a shorter converging time than the other two algorithms. The same decision rule is then tested on an exclusive testing data set for the same turbine. I verify my test results by the automatic failure flagging on the turbines and the monthly operational reports from the wind farm operators.

5.4.1 Application of PSO to Determine the Decision Boundary

A major problem in the wind speed difference test is to determine the decision boundary between the normal and abnormal states in the feature space objectively and optimally. The feature space spanned by the shape and scale parameters of the estimated Weibull distribution of the weekly wind speed differences is illustrated in Figure 5.10.

When soft failures such as the anemometer faults happen, the wind speed difference is not drastic, but the difference is consistently there, and hence the distribution is skewed towards the bigger wind speed difference. In this case, both the shape parameter and the scale parameter are large numbers, pointing to the upper right region.

When hard failures such as major component failures happen, one turbine completely shuts down, and its anemometer reads zero wind speed, causing maximum wind speed difference between a turbine pair. The distribution is thus nearly flattened out with a very heavy tail, represented by a bigger scale parameter but small shape parameter, namely, the lower right region.

Based on the above analysis, we note that the features of the normal weeks are clustered together, yet the features of the abnormal weeks can spread around anywhere outside of that cluster. I propose a heuristic that the features of the normal weeks are clustered in a circular region, which is the least informative shape, shown as Figure 5.7 in the previous section. But, the location and size of such a circle in the feature space is turbine specific. In order to determine the origin and the radius of the circle, I utilize PSO algorithm to learn from historical data. Then I use this decision rule to test on future data.

The particle in our application is defined as

$$\{P_i : x_i, y_i, r_i\}, \tag{5.7}$$

where the particle index is i going from 1 to N, the origin of its circle is (x_i, y_i), and the radius of its circle is r_i.

Each SCADA data record is associated with a manual or automatic label indicating the event happening at that time, which is used to label whether the turbine works normally at that instance. If the labels associated with faults account for more than 20% percentage of the weekly data, this week is regarded as a problematic week, and hence a health flag of -1 is applied to this week. Otherwise, this week is assigned with a health flag of 1. An exception is assigned a health flag of 0, if there is too much missing data and the data is not sufficient, which does not effect the optimization procedure.

However, the SCADA flagging is not completely dependable, especially on the degradation of the faulty components. This is also why we need to design better diagnostics and prognostics algorithms. Another factor to label whether the week is healthy or problematic is the area value under the Weibull CDF curve of the wind speed difference in that week, as defined in Equation (5.5). A health flag of -1 overrides a flag of 1, if the area value is less than 0.96382.

Note that labeling is based on both the reported events and AUC, where AUC serves as a data-based evaluation to enhance the reliability of event flagging for labeling, which is for training purpose. In testing, we often do not have enough resources to label, and then testing is based on classification.

After the health flag of each week is assigned, any decision boundary may incur two types of errors. One error is miss detection, defined as the number of weeks with a health flag of -1 but located inside the circle as being normal,

$$(E_{\text{miss detection}})_i = I(week_j \; is \; \text{normal} \; (inside \; circle), \; when \; \text{health flag}_j = -1), \tag{5.8}$$

where j is the set of week indices in the training set. $I(\cdot)$ is a counting function, with value 1 when the argument is true, or else 0. The other error is false alarm, defined as the number of weeks classified as abnormal outside the circle but with a health flag of 1,

$$(E_{\text{false alarm}})_i = I(week_j \; is \; \text{abnormal} \; (outside \; circle), \; when \; \text{health flag}_j = 1). \tag{5.9}$$

The fitness function is the summation of the above two errors associated with each particle,

$$Fitness_i = f(P_i) = (E_{\text{miss detection}})_i + (E_{\text{false alarm}})_i. \tag{5.10}$$

The optimal solution minimizes the fitness function.

5.4.2 Experiments and Results

There are 130 weeks' worth of data, and I split them into training and testing sets. The training set includes the first 70 weeks, and each week is labeled with 1, -1, or 0, indicating whether that week is healthy, problematic, or missing-too-much-data. In Figure 5.10, the green points represent healthy weeks and the red ones problematic weeks. If one green point is outside the candidate decision boundary, it causes false alarm error; however, if one red point is inside the boundary, it causes a miss detection error. I use 200 particles and let them search in 100 iterations. Note that the number of particles could be reduced

to maintain a similar performance, and my analysis indicates that more particles do not necessary improve performance, and hence I use 200 particles. I have tried different numbers of iterations. PSO often converges in less than 50 iterations, and I choose 100 iterations to allow some tolerance.

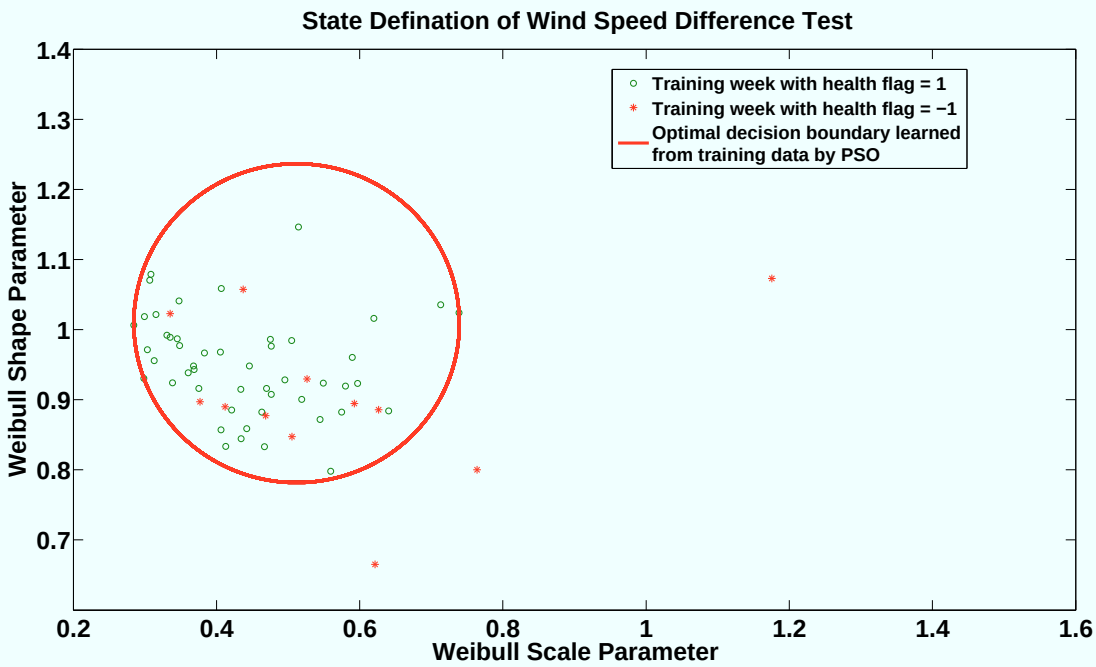


Figure 5.10: 2D plot of weekly estimated Weibull parameters of wind speed difference between turbine A and B. Apply the first 70 weeks as training data. Learn the optimal decision boundary of normal and abnormal states by PSO.

In each iteration, the fitness of each particle is evaluated by Equation (5.10). The best solution seen by each particle is used to update p_{best_i}, and the best solution seen by the whole population is used to update g_{best}. Through iterations, all particles move toward optimal locations, driven by their own cognitive awareness and social influence.

Figure 5.10 shows the 2-dimensional plot of Weibull scale and shape parameters of wind speed difference between turbine A and B. Each point represent one week. I use the first 70 weeks as training data, and then apply PSO to determine the optimal decision boundary with the lowest error amount. In Figure 5.10, the red circle represents the optimal decision boundary obtained by PSO algorithm from the training data. This circle separates the normal and

abnormal regions of the wind speed difference test incurring minimum errors. In this feature space, the weeks associated with soft failures are located to the upper right of the circle, and the weeks associated with hard failures are located to the lower right of the circle.

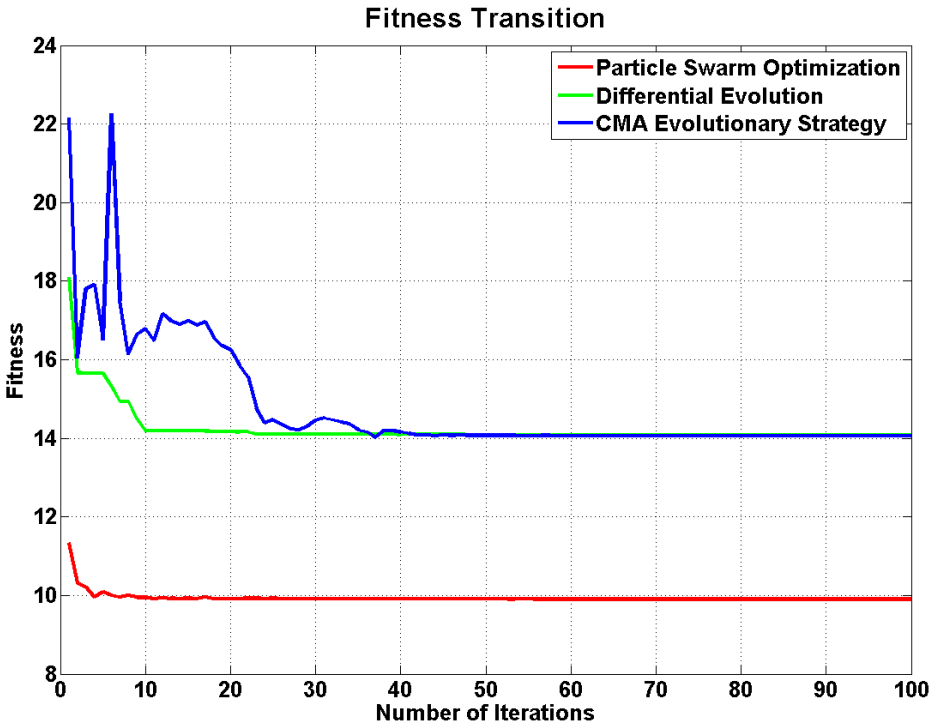


Figure 5.11: Comparison of fitness transition among PSO, DE and CMA-ES.

I compare PSO algorithm with the other two optimization algorithms, Differential Evolution (DE) (57) and Evolution Strategy (ES) (58). Differential Evolution algorithm is an optimization method in evolutionary algorithms (EAs), capable of handling non-differentiable, nonlinear and multi-model objective problems. The crucial idea behind DE is a scheme for generating new candidate solutions by combining existing ones according to its formulae of vector crossover and mutation. Then, DE adds the weighted difference between two solution vectors to a third one, which makes the scheme completely self-organizing. In each iteration, DE updates the solution which has the best fitness value on the optimization problem. On the other hand, Evolution Strategy (ES) algorithm

is a stochastic, derivative-free numerical optimization method for nonlinear or non-convex problems. In ES algorithm, new candidate solutions are sampled according to a multivariate normal distribution. The pair-wise dependencies between the variables in this distribution are described by a covariance matrix, which is updated by the covariance matrix adaptation (CMA) method. So, this algorithm is also called CMA-ES.

Figure 5.11 compares the progression of g_{best} versus iterations among three mentioned algorithms. As shown in Figure 5.11, the PSO algorithm achieves the lowest fitness value in a shorter converging time than both DE and CMA-ES (59). The fitness value by PSO converges below 10 around the 5th iteration, but the other two algorithms only converge to 14. Table 5.6 shows that both DE and CMA-ES need more CPU time to hit the lowest fitness value than PSO does.

Table 5.6: Compare PSO with DE and CMA-ES on CPU Time of Iteration and Convergence

	PSO	DE	CMA-ES
CPU time per iteration	0.134s	0.151s	1.42s
No. of iterations to converge	5	10	40
CPU time per convergence	0.67s	1.51s	56.80s

After the location and size of the decision boundary circle are determined for each turbine, I use this decision rule to test future data for the same turbine. I use the data from week 71 to week 130 in testing. In Figure 5.12, the red circle is the optimal decision boundary learned from the first 70 weeks. All the weeks outside this circle are classified as abnormal weeks. Except for week 118, which falls into the *idle* state due to the lighting strike, all the other abnormal weeks are caused by the fact that the anemometer is degraded as time goes on. According to the monthly report, the anemometer barely functions since week 79 until it is replaced in week 95, same as Figure 5.12 showing. So, the test results are consistent to the monthly report. With the red decision boundary learned from the earlier data, and even through there are false alarms, miss detections are avoided.

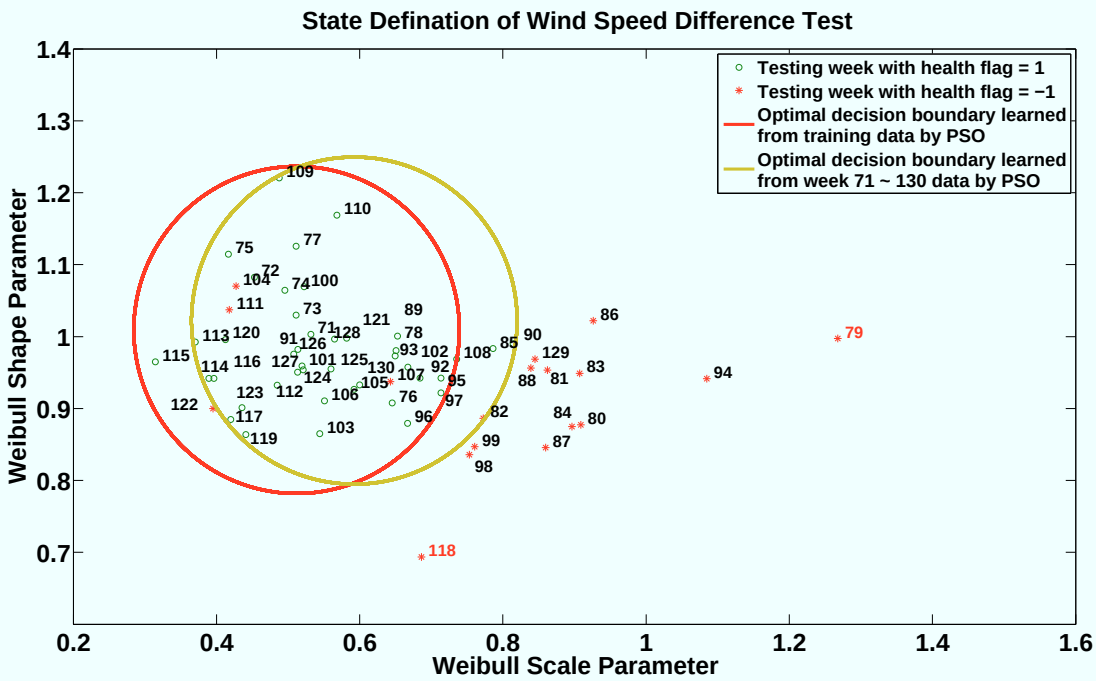


Figure 5.12: 2D plot of weekly estimated Weibull parameters with optimal decision boundary. Apply the week 71 to week 130 as testing data. The average system performance gets worse due to the aging anemometer.

To demonstrate the system degradation due to anemometer aging, we apply the same decision boundary procedure on the testing data from week 71 to week 130 to obtain the yellow circle. This procedure on the latter data is not used to train or test the system but only to show the average system performance. The center of later data set is shifted to the right, indicating that the anemometer shifts towards worse performance.

5.5 Failure Detection by Fusing Information from Multiple Neighboring Turbines

Once a failure is detected using the previous methods, it cannot be conclusively said which of the two turbines have an abnormal condition. To further enhance the performance of my detector, I evaluate the wind speed differences with respect to multiple turbines for the turbine under test. I consider the turbine A and evaluate its wind speed difference with respect to B, C, D, E, F, G, and H. Order these turbines numerically with a notation of i. For instance, turbine B is the 1st turbine in comparison, and turbine C is the 2nd turbine in comparison, etc., until turbine H is the 7th turbine in comparison. In my example, I compare the wind speed of turbine A with seven other turbines. In practice, one can use more or less turbines for comparison. For i goes from 1 to 7, the ith wind speed difference is given by

$$wsd_i = |ws_A - ws_i|. \tag{5.11}$$

Similar to my analysis in pervious sections I estimate the probability density function and the cumulative density function for each of the wind speed difference. Further I will attempt to fuse these multiple sources of information to determine whether turbine A has an abnormal condition or not. If turbine A functions abnormally, then it is highly likely that some or all of the wind speed difference CDFs are in bad states. On the other hand, if turbine A functions properly, then it is unlikely that all of the CDF tests would be bad.

Due to the differences in the physical distances, wakes, or potential faults, the wind speed difference between multiple turbine pairs would differ. The CDF plots of the weekly wind speed difference is averaged over the full data duration, and these average CDF plots between turbine A and multiple turbines are shown in Figure 5.13 to show the variation. Note that the best in this average is turbine B. The bigger the area under the CDF plot, the smaller the difference between the wind speed measured on the turbine pair, and hence the more important this turbine is in comparison for telling when turbine A may function abnormally. Based on this idea, the areas under the average CDF curve are used as the weights for later fusion.

$$w_i = \int_0^\infty \overline{CDF(wsd_i)} dwsd_i, \tag{5.12}$$

where wsd_i is defined in Equation (5.11).

For each turbine pair, the areas under the empirical weekly CDF curve are first evaluated. In the jth week, the ith turbine assumes an area under the CDF curve as below,

$$A_{i,j} = \int_0^\infty CDF(wsd_{i,j}) dwsd_{i,j}, \tag{5.13}$$

where $wsd_{i,j}$ is the wind speed difference between the turbine A and the ith turbine in the jth week. Then the maximum area within all these weeks is used to normalize the weekly area, so that the normalized area is in the range of 0 to 1,

$$a_{i,j} = \frac{A_{i,j}}{max_j\{A_{i,j}\}}, \tag{5.14}$$

where $max_j\{A_{i,j}\}$ takes the maximum value over week number j.

Finally the multiple turbine tests are fused as below,

$$FA_j = \sum_{i=1}^{m} w_i a_{i,j}, \tag{5.15}$$

where m is the number of turbines in comparison. In my example, $m = 7$. FA_j is the weekly fused result, as shown in Figure 5.14. In Figure 5.14, the test

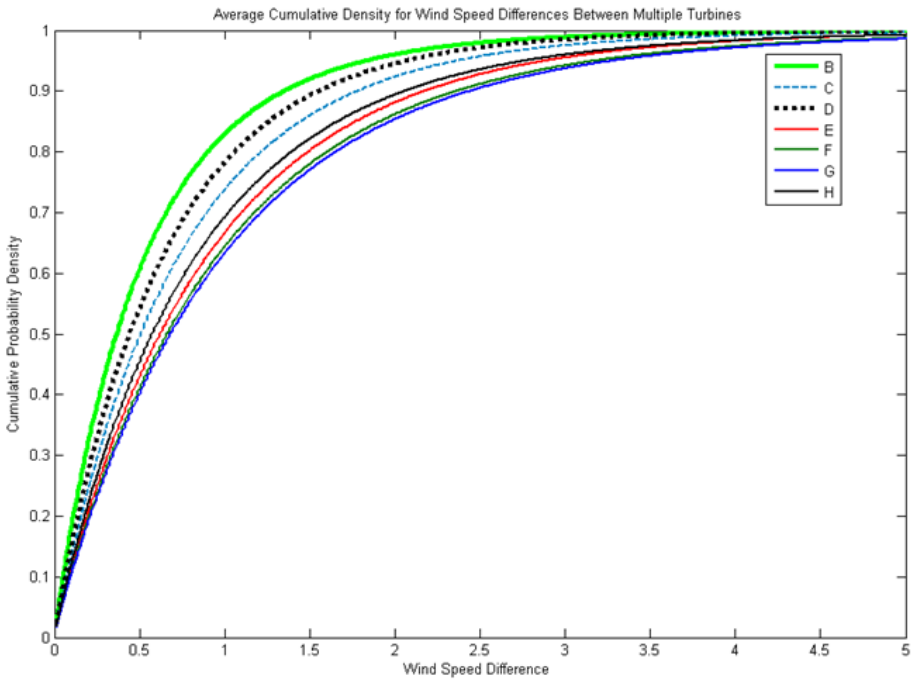


Figure 5.13: Variation in Weibull cumulative density of weekly wind speed difference between turbine A and multiple turbines.

result between the single turbine pair, in red line with circle markers, is taken from Figure 5.9, which is compared with the fused result, in blue line with star markers, between the multiple turbine pairs.

From Figure 5.14, we can see that the single turbine pair test indicates that week 79 may have a problem. With the multiple turbine pairs test, this week is excluded. This result indicates that the low value in the single turbine pair test in week 79 is due to turbine B, instead of turbine A. With a single turbine pair test, we cannot conclusively say which turbine causes the problem. However, with the multiple turbine pairs test, we can determine which turbine is faulty in that week.

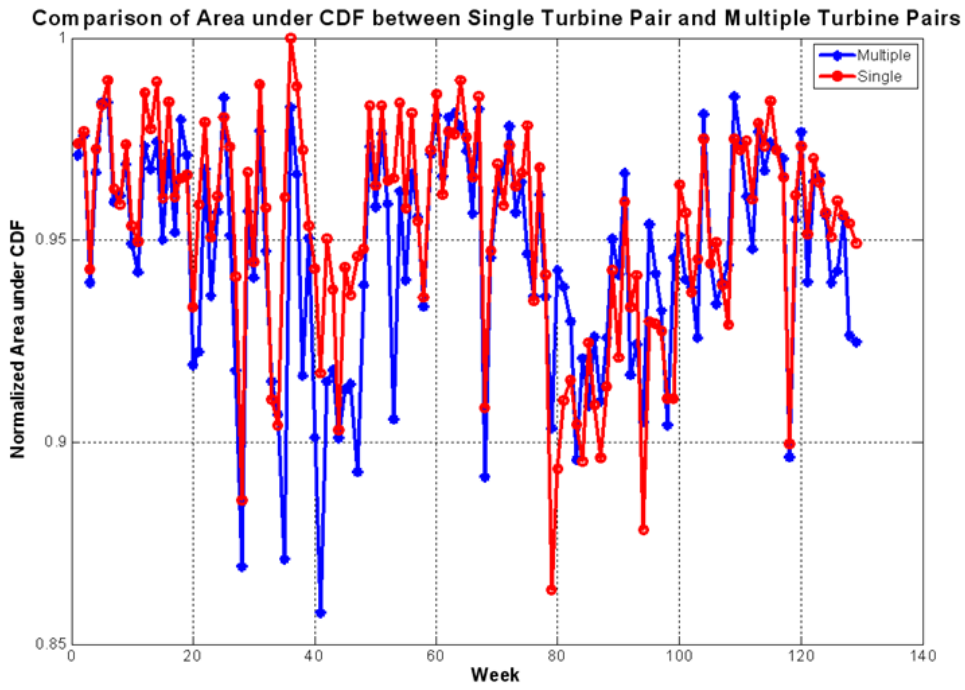


Figure 5.14: Comparison of area under CDF between single turbine pair and multiple turbine pairs.

5.6 Pattern Recognition by Fusing Multiple Test Results with PSO

After extracting turbine working features from the multiple designed performance tests, the next step is to recognize different patterns of turbine working status at current time by fusing all the test results.

5.6.1 Turbine Working Status Analysis and Classification

I take the data for each day as a unit to detect on which days the turbine behaves abnormally. Assuming that there are n observations per day, I run the multiple tests, designed for describing turbine performance, on each observation point. In each test, multiple states are defined to extract different working condition features, such as complete shut-downs, under-performing states, abnormally frequent default states, as well as, normal working states.

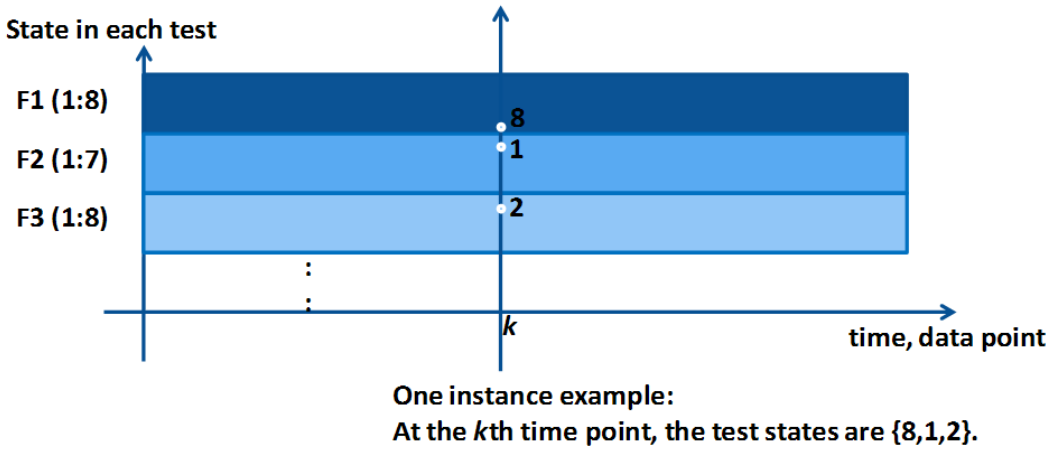


Figure 5.15: Turbine working status analysis procedure.

Through extensive data mining of historical data and verification from farm operators, every state combination corresponds to one type of turbine working status. Some state combinations are also discovered to be strong indicators of turbine abnormalities. Figure 5.15 demonstrates the procedure of testing turbine performance at the kth time point. Functions $F1, F2, F3$ are defined for

the power curve test, the rotor speed test, and the pitch angle test, respectively. Each test can represent one feature of turbine working status at the kth time point. So, more tests extended, more specifically the turbine performance can be described. At the kth time point, the corresponding state combination is $8, 1, 2$, in which the three numbers orderly represent the result states of the power curve test, the rotor speed test, and the pitch angle test. The result of the rotor speed test belongs to the horizontal state 1, which means that the rotor speed is relatively low while the wind speed is quite high; meanwhile, the power produced is negative at this time point, which is assigned as the abnormal working state 8. These two features indicate that this turbine is barely rotating due to failures. Moreover, the pitch angle also stays in the vertical state 2. This is because the turbine makes its blades get the least wind area to avoid a destructive break. All of the multiple tests tell that the turbine is in an abnormal working status due to a bad windy weather. The detection results have been verified as a lighting strike to this turbine.

Based on the knowledge from both wind energy experts and observation data, we classify the turbine working status into 5 different categories:

1. Normal Operation

2. Ramping Up/Down

3. Degradation

4. Safe Shutdown

5. Abnormal Shutdown

Category 2 is defined as the working status when the turbine is ramping up or down caused by its normal reaction to wind speed varying, different control settings, etc. Category 3 describes the situation that even the turbine still keeps running but its performance has already been degrading. It is a warning sign to the later complete shut-down. The safe shutdown is due to some allowable reasons, such as annual maintenance; on the contrary, the abnormal shutdown happens unexpectedly because of several harmful faults, such as spindle failure. Each state combination that has ever occurred to turbines is assigned to its relative working status category, as shown in Table 5.7.

Table 5.7: Turbine Working Status Category Assignment

Category	Relative State Combinations
1. Normal Operation	254, 255, 256, 354, 355, 356, 444, 455, 456, 556
2. Ramping Up/Down	813, 833
3. Degradation	378, 558, 578, 754, 755, 834, 837, 854, 855
4. Safe Shutdown	811, 812, 818, 911
5. Abnormal Shutdown	611, 633, 644, 618, 671

5.6.2 Applying PSO to Determine the Fusion Rules of Turbine Daily Working Status

I divide the whole data set into daily data for the interesting turbine, with n observations per day. As mentioned above, fault detection through data fusion is more robust than using individual tests. So, after running multiple tests as time goes by, each observation point is assigned one particular state combination. To monitor the turbine performance in daily base and understand turbine daily working status better, the main problem is to fuse the $n - point$ state combinations into one category for each day, which requires the construction of a weight matrix as the fusion rules. I utilize PSO to determine the weight matrix learning from historical working days objectively and optimally. Then I use the fusion rules to test future data.

Based on the previous analysis, the particle in this application is defined as a weight matrix,

$$Particle_l = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \end{bmatrix}_l , \tag{5.16}$$

where the particle index is l going from 1 to L which is the total number of particles, and the weight w_{ij} means how determinant it is to daily category i when there are state combinations belonging to category j happened. Usually, the weights in the diagonal line are much higher than the others on the same

row, since they are in a dominant role for defining the significance of the row-corresponding category.

The total daily percentage P_i of category i is defined in Equation (5.17),

$$P_i = \sum_{j=1}^{5}(w_{ij} \times \sum_{k=1}^{K} p_{jk}),, \tag{5.17}$$

where p_{jk} is the percentage of kth state combination which is classified into category j. Then set the category with the maximum percentage to the working status for this day by following Equation (5.18),

$$category \equiv argmax(P_i).. \tag{5.18}$$

After the working status category of each day is assigned, any weight matrix may incur inconsistency with identified daily category in the training set. The fitness function is the total amount of these un-matching days,

$$Fitness_l = \sum_{n} I(category_n \neq truecategory_n), \tag{5.19}$$

where, for each particle solution, n is the set of day indices in the training set, I is a counting function, with value 1 when the argument is true, or else 0. The optimal solution minimizes the fitness function.

After fusing the daily state combinations, with the learned weight matrix, each day will have one category to describe its turbine working status. My approach gains a qualitative understanding of turbine performance status to detect faults, and it also provides explanations on what has happened for detailed diagnostics and prognostics.

5.7 Data-Driven Bayesian Inference for Turbine Failure Prediction

The main purpose of applying Bayesian inference is to identify and predict any particular failure with statistical certainty by online monitoring turbine perfor-

mance. I assume that the faulty turbine follows specific degradation patterns due to differential failure before and after it happens. Based on turbine degradation, the Bayesian network is able to tell the wind farm operators what type of failures is happening to this turbine, and tell the time when it is probably going to shut down. Then, after receiving these maintenance requirements sent by BN, PSO can establish a schedule with priorities on each maintenance task.

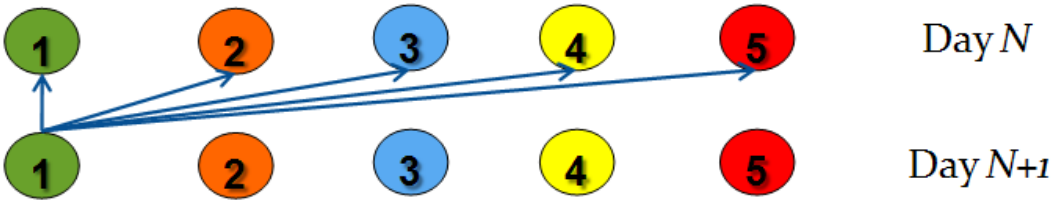


Figure 5.16: Bayesian inference of turbine status variation between two adjacent days.

For the purpose of prognostics, I model the degradation pattern by analyzing the working status among 25 days backwards from the day when the interesting event starts. The network is given as a fully connected structure, in which each level of variables represents the all working status ever occurred on this day, and each variable is binary, with value 1 when this working status happens; otherwise, it is 0. The problem focuses on learning the joint probability distribution of each two status between two adjacent days.

Let θ_{ijk} to denote $p(x_i = j|\Pi_i = k)$, where x_i is one of the 5 working status categories in day N. If j is 1, this status appears, or else does not. And, k is a combination of the working status happened in the next day $N + 1$. The parameter vector Θ is estimated from a collection **D** of independent data cases $D_1, D_2, ..., D_m$ when the interesting event arises. The joint probability table is shown as

$$\Theta = [p_{ij}(x_i = 1, y_j = 1)], \tag{5.20}$$

where $i, j \in 1, 2, 3, 4, 5$. In this joint probability table, the row and column represent the category indices of turbine working status in the current day N and its next day $N + 1$, respectively. The value of p_{ij} is the joint probability that both status x_i in day N and y_j in day $N + 1$ happen. If p_{ij} is equal to 0, there is no link between the nodes x_i and y_j.

Considering spindle failure as an example, Figure 5.17 shows the joint probability transition of turbine working status among a total of 25 analysis days before it happens. The red points are joint probability samples of variable $(1, 1)$, meaning that the turbine keeps working properly in two adjacent days; whereas, the samples of variable $(5, 5)$, represented by the green points, indicate that the turbine is shutting down completely in two adjacent days. As shown in this figure, the abnormal shut-down is gradually increasing, since 20 days before the spindle failure happens. Meanwhile, the turbine barely operates normally during the same time, especially around 10 days before this failure is verified. In other words, the good performance is decreasing and bad performance is increasing since 20 days before the spindle failure really happens.

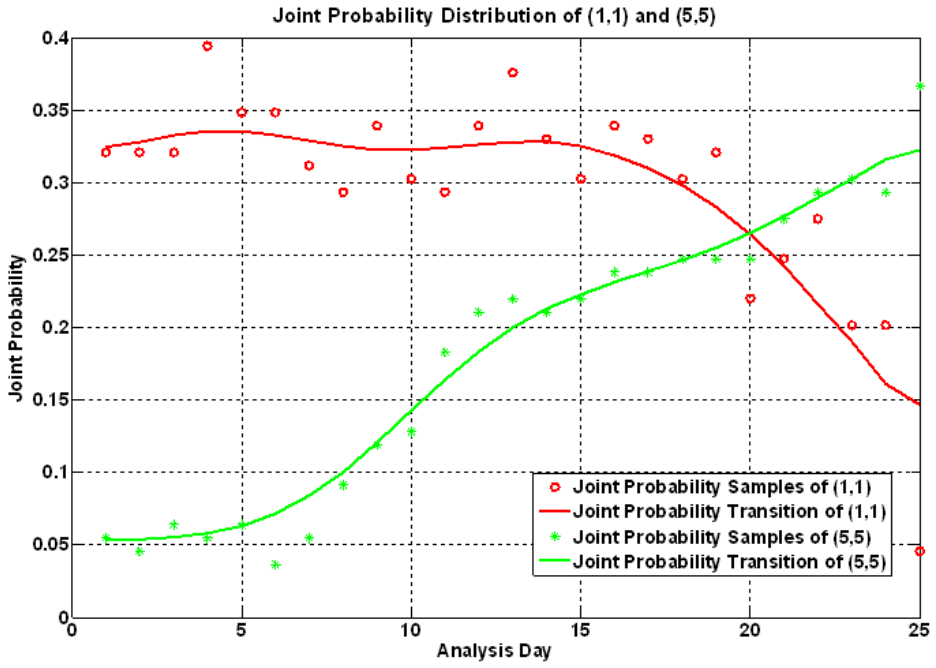


Figure 5.17: Joint probability transition of turbine working status among 25 days before spindle failure happens.

To sum up, the Bayesian network automatically monitors turbine performance and predicts any particular failure. As a result, failures can be detected at an early stage, so that the potential damage could be minimized or mitigated

through early repairs, avoiding drastic breakdowns of the wind turbine. Also, maintenance can be optimally scheduled, which leads to less downtimes and more revenue.

5.8 Conclusion

Wind energy is expected to play an increasingly important role in the future national energy scene. Wind turbines are used to tap the potential of wind energy, which is available in millions of MW. Reliability of wind turbines is critical to extract this maximum amount of energy from the wind. This chapter proposes a data-driven methodology developed to monitor the global performance of wind turbine as well as for an early fault detection to keep away the wind turbines from catastrophic conditions due to sudden breakdowns.

Currently, one means of enhancing reliability and robustness of wind turbine is to implement efficient, adaptable and responsive systems of condition monitoring. Autonomous online condition monitoring systems (CMSs) with integrated fault detection algorithms allow early warnings of mechanical and electrical defects to prevent major component failures. It can reduce costs by enabling necessary repair actions to be planned in time. Most methodologies of CMSs being used today are separately designed for each particular component or subsystem (60). For example, vibration analysis and acoustic monitoring are often applied for rotating equipments, such as bearing systems of wind turbine, while thermography technology and electrical effects are applied for monitoring electronic components (61).

For wind turbines, however, the system to be monitored is complex with high number of subsystems, and the margins for investments are small. The implementation of CMS techniques on subsystem level normally requires the initial cost of installation and adaptation to existing system. Also, the local monitoring is often focused on a very limited number of aspects, but, when to define relationships among subsystems, the whole system becomes more complicated. Hence, with relatively low costs, the global performance monitoring of wind turbine can be added, which makes early fault detection based on

trend analysis possible.

In this chapter, the trending of wind turbine main parameters, such as the relationship of power, wind velocity, rotor speed and pitch angle, are analyzed, which gives global insight in the operation in the turbine. By application of more advanced methods of signal analysis and artificial intelligence, significant changes in turbine behavior are detected at an early stage. For example, in case of large deviations on trends of representative signals or combination of signals, an alarm is generated. This approach is cost effective as no additional investment is required, but can accomplish the task of monitoring the turbine as a whole.

To establish a condition-based maintenance and repair, there is a need to develop an efficient fault detection algorithm. Many faults can be detected while the defective component is still operational. Numbers of techniques are available for identification of faults, which are distinguished between model-based algorithms and data-driven algorithms. The model-based algorithms encode human knowledge via a hand-coded representation of the system, including rule-based expert systems and finite-state machines. These hand-coded model uses qualitative, rather than numerical, variables to describe the physics of wind turbine. To detect specific failure, this chapter proposes an intelligent data-driven classification approach to automatically distinguish turbine behavior between normal operation and faulty condition. The test results have verified the effectiveness of detecting turbine failures at an early stage, especially for anemometer failure. I also suggest the use of a Bayesian network for estimating the turbine degradation pattern due to different failures.

6

Conclusion and Future Work

6.1 Conclusion

Data fusion technologies have been used across a wide range of applications, to support a real-time decision making in complex, dynamically changing environments. In this dissertation, I develop an intelligent data fusion model for decision support systems applied to real-world problems. It makes the decision support system adaptable, and can be applied in various problem structure and problem types.

To sum up the advantages of this proposed data fusion model, I list its three attracting features as follows:

- This data fusion model has a loop structure that returns the feedback of decisions into the next time interval, so that it can respond to the dynamically changing environment at the real-time, and execute the actions meeting the system requirements with respect to the current circumstance.

- This model uses an artificial intelligent mechanism, which makes the decision support system more cognitive and reliable. For instance, I apply Bayesian network (BN) for threat assessment in the system, in that it can come up with a set of possible operation based on current situation, and present confidence or uncertainty of taking one course of action over another. I also apply particle swarm optimization (PSO) algorithm to make

decision more objectively and optimally. This algorithm has simple computations, so that it can converge to an optimal solution very quickly.

- This model is a data-driven approach, which makes the decision support system more automatic and adaptive. It only needs the expert knowledge in the system design phase, but requires no human in the loop execution. The intelligent mechanism learns from the system behavior, and automatically adapts the system parameters to the new changes. In addition, due to its data-driven nature, this model is easily transferable from one problem domain to another.

To demonstrate its advantages, I implement the intelligent data fusion model into two real-world problems: one is the sensor network management for air traffic control; the other is the failure detection and maintenance for wind turbines.

1. **Sensor network management for air traffic control:**

   The research objective of air traffic control problems is to develop a sensor manager that can allocate sensor resources efficiently. The main contribution of the proposed sensor management system is that it automatically updates the system operating requirements in real-time based on monitoring its current performance and dynamically changing environment. As a mission manager in the system, BN maintains the possibly best overall performance by allocating more sensor resources to the regions with higher priority.

2. **Failure detection and maintenance for wind turbines:**

   To detect turbine failures at an early stage, this dissertation proposes an intelligent data-driven approach to monitor the turbine global performance at real-time by fusing multiple test results, and identify the turbine abnormalities by tracking the turbine status variations in real time. This approach is adaptable to each turbine automatically, and is advantageous in its applicability and data-driven nature to monitor a large wind farm. The test results have verified the effectiveness of detecting turbine failures before they really happen, especially for anemometer failures.

6.2 Future Work

Despite its wide usage in the real-world applications, the data fusion model presents formidable challenges to those interested in the implementation of system design and algorithms through theoretical analysis. So, to date a fully comprehensive and adaptable model of decision support system is still not available. We would like to explore both theoretical and practical sides of the data fusion model and gain more insight on it. The research on data fusion model and its application in the following directions are going to be useful:

6.2.1 Applying Hybrid Methods of Other Artificial Intelligent Algorithms

One important area of active research is applying hybrid methods of computational intelligence techniques to solve a single problem. This is often used to solve complex real-world problems where one technique is typically used to fix the weaknesses of the other. In this proposed decision support system, I particularly combine BN for uncertainty assessment and PSO for optimization. However, for different types of problems, it might be more effective to apply the hybrid methods of other artificial intelligent algorithms.

For example, another popular paradigm of swarm intelligence is called ant colony optimization (ACO). The algorithm is inspired by the stigmergistic communication system employed by ants to evaluate alternative choices and take decisions in dynamic optimization problems. Moreover, one of the most popular machine learning approaches to wind energy prognostics is to use neural networks to model the system (62). Neural networks are a type of model that establishes a set of interconnected functional relationships between input stimuli and desired output where the parameters of the functional relationship need to be adjusted for optimal performance.

6.2.2 Applying Could Computing to Make System More Scalable

Decision support with data fusion technology, even with sparse data, is computationally intensive. In the most of real-world problems, data is usually very noisy, also has few training examples.

To mitigate the problems of sparse and noisy data, our strategy will be to generate multiple data fusion models which express as many distinct, useful explanations of the data as possible. This will involve factoring the data fusion technology to simultaneously investigate multiple strategies using different model accuracy metrics, model complexity metrics and factoring of the training data. This will be computationally expensive and can only be achieved by distributing the computation process on a massive compute resource like the cloud.

Cloud computing has emerged as a new paradigm for commercial, scientific, and engineering computation. A cloud allows an organization to own or rent efficient, pooled computer system instead of acquiring multiple isolated large computer systems each commissioned and assigned to particular internal projects. It is rapidly becoming well understood how to get the most out of the cloud as an application technology platform for developing complex web services or database servers. This is because the cloud and internet have grown together. However, when one considers cloud-based Artificial Intelligence, how to transition to the cloud is not entirely clear. This is because a lot of the technology supporting cloud application development is still itself being developed. Hence, there is a vast potential for the future development of deploying and updating cloud-based AI algorithms into real-world applications.

Bibliography

[1] "Automatic sensor scheduling," in *Project Contract of Syracuse University with Sensis Corporation*, from July, 2007 to July, 2008. i, xi, 14, 34, 36, 37

[2] "Wind turbine intelligent diagnostics and prognostics," in *Project Contract of Syracuse University with AWS Truewind, LLC*, from May, 2008 to Jan, 2010. ii, xi, 14, 58, 59, 61

[3] Y. Yan, L. A. Osadciw, G. Benson, and E. White, "Inverse data transformation for change detection in wind turbine diagnostics," in *Proceedings of 22nd IEEE Canadian Conference on Electrical and Computer Engineering*, (Delta St. Johns, Newfoundland and Labrador, Canada), May 2009. xi, 12, 65, 66

[4] S. Paradis, R. Breton, and J. Roy, "Data fusion in support of dynamic human decision making," in *Proc. IEEE Int'l. Conf. on Information Fusion*, 1999. 1

[5] A. M. Bisantz, R. Finger, Y. Seong, and J. Llinas, "Human performance and data fusion based decision aids," in *Proc. IEEE Int'l. Conf. on Information Fusion*, vol. II, 1999. 1

[6] J. Esteban, A. Starr, R. Willetts, P. Hannah, and P. Bryanston-Cross, "A review of data fusion models and architectures: towards engineering guidelines," in *Journal of Neural Computing and Applications*, vol. 14, December 2005. 2

[7] L. A. Klein, *Sensor and data fusion: a tool for information assessment and decision making*. SPIE-The International Society for Optical Engineering, 2004. 2

[8] M. Bedworth and J. O'Brien, "The omnibus model: A new model of data fusion," in *Aerospace and Electronic Systems Magazine*, vol. 15, pp. 30–36, 2000. 2

[9] M. Ticha and T. Ranchin, "A case based reasoning data fusion scheme: application to offshore wind energy resource mapping," in *Proceedings of 9th International Conference on Information Fusion*, July 2006. 7

[10] A. A. Oklahoma and A. Abraham, "Rule-based expert systems," in *Handbook for Measurement Systems Design*, pp. 909–919, 2005. 7

[11] R. Krzysztofowicz, "Decision criteria, data fusion, and prediction calibration: a bayesian approach," *IEEE Transactions on Hydrological Sciences*, vol. 55, March 2010. 7

[12] L. A. Osadciw, "Emerging sensor networks can make sense for your business," in *Technical Report 2003*, (Syracuse, NY), 2003. 9

[13] G. A. McIntyre, "A comprehensive approach to sensor management and scheduling," in *Doctoral Dissertation*, (George Mason University, Fairfax, VA), 1998. 9

[14] L. Osadciw and K. Veeramachaneni, "A controllable sensor management algorithm capable of learning," in *Proceedings of SPIE Homeland and Defense Symposium*, April, 2005. 9

[15] L. Rothman and S. Bieri, "Evaluation of sensor management systems," in *Proceedings of National Aerospace and Electronics Conference*, vol. 4, pp. 1747–1752, 1998. 10

[16] T. Burton, D. Sharpe, N. Jenkins, and E. Bossanyi, *Wind Energy Handbook*. Wiley, 2001. 11

[17] Z. Chen and F. Blaabjerg, "Wind energythe worlds fastest growing energy source," *IEEE Transiations of Power Electron Soc Newslett*, vol. 18, no. 3, 2006. 11, 57

[18] S. M. Kay, "Fundamentals of statistical signal processing: Detection theory," vol. II, 1998. 11, 57

[19] X. Wei and M. Verhaegen, "Fault detection of large scale wind turbine systems," in *IEEE Proc. of 16th Mediterranean Conference on Control and Automation*, (Ajaccio, France), pp. 1675–1680, June 25-27, 2008. 11, 57

[20] A. Stefani, A. Bellini, and F. Filippetti, "Diagnosis of induction machines' rotor faults in time-varying conditions," *Industrial Electronics, IEEE Transactions on*, vol. 56, pp. 4548 –4556, nov. 2009. 12

[21] W. Yang, P. Tavner, C. Crabtree, and M. Wilkinson, "Cost-effective condition monitoring for wind turbines," *Industrial Electronics, IEEE Transactions on*, vol. 57, pp. 263 –271, jan. 2010. 12

[22] B. Lu, Y. Li, X. Wu, and Z. Yang, "A review of recent advances in wind turbine condition monitoring and fault diagnosis," pp. 1 –7, jun. 2009. 12

[23] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, July 31, 2009. 15, 18, 21

[24] D. Heckerman, "A tutorial on learning with bayesian networks," in *Microsoft Research Technical Report*, March 1995. 16

[25] I. Ben-Gal, "Bayesian networks," in *Encyclopedia of Statistics in Quality and Reliability*, Wiley, 2007. 19

[26] R. E. Neapolitan, *Learning Bayesian networks*. Pearson Prentice Hall, 2004. 20

[27] B. Das, "Representing uncertainties using bayesian networks," in *Scientific and Technical Report*, December 1999. 21

[28] E.Polak, *Optimization: algorithms and consistent approximations*. New York: Springer, 1997. 25

[29] A. E. R. Brits and F. V. den Bergh, "Locating multiple optima using particle swarm optimization," vol. 189, pp. 1859–1883, 2007. 26

[30] J. P. S. Kiranyaz and M. Gabbouj, "Multi-dimensional particle swarm optimization for dynamic environments," in *Proceedings of 12th International Conference on Innovations in Information Technology*, pp. 34–38, Dec. 2008. 26

[31] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int'l. Conf. on Neural Networks (Perth, Australia)*, vol. IV, (IEEE Service Center, Piscataway, NJ), pp. 1942–1948, 1995. 26

[32] J. D. Papastavrou, *Decentralized Decision Making in a Hypothesis Testing Environment*. The MIT Press, 1990. 30

[33] R. V. Denton, E. I. Alcaraz, J. Linas, and K. J. Hintz, "Towards modern sensor management systems," in *Science of Command and Control, Part III: Coping with Change*, pp. 118–134, AFCEA International Press, Fairfax, 1994. 34

[34] L. Osadciw, X. Ye, and G. Kamath, "Smart sheduler for air traffic control application," in *Techniqual report for Sensis Corporation*, September. 34

[35] T. Jun and S. Gan-lin, "Multi-sensor management: Optimal allocation of tracking resources," in *Proceedings of Electric Information and Control Engineering (ICEICE)*, pp. 532–535, April. 38

[36] T. Brhard and E. Duflos, "Optimal policies search for sensor management," in *Proceedings of IEEE Fusion Conference*, 2008. 38

[37] X. Ye, G. Kamath, and L. Osadciw, "Using bayesian inference for sensor management of air traffic control system," in *IEEE Symposium on Multicriteria Decision Making*, (Nashville, TN), March. 43

[38] J. P. Wangermann and R. F. Stengel, “Principled negotiation between intelligent agents: A model for air traffic management,” vol. 12, pp. 177 – 187, 1998. 55

[39] D. P. A. Li Weigang, Marcos V. P. Dib and A. Crespo, “Intelligent computing methods in air traffic flow management,” vol. 18, pp. 781 – 793, 2010. 55

[40] J. Ribrant and L. M. Bertling, “Survey of failures in wind power systems with focus on swedish wind power plants during 1997 ndash;2005,” *Energy Conversion, IEEE Transactions on*, vol. 22, pp. 167 –173, mar. 2007. 57

[41] A. Tindal, C. Johnson, M. LeBlanc, K. Harman, E. Rareshide, and A. Graves, “Site-specific adjustments to wind turbine power curves,” in *AWEA WINDPOWER Conference*, (Houston, TX, USA), 2008. 58

[42] J. Ribrant, *Reliability performance and maintenance - a survey of failures in wind power systems*. PhD thesis, XR-EE-EEK, 09 2006. 58

[43] D. Robb, “Gearbox design for wind turbines improving but still face challenges,” *Windstat Newsletter*, vol. 18, no. 3, 20045. 58

[44] T. Burton, D. Sharpe, N. Jenkins, and E. Bossanyi, *Wind Energy Handbook*. Wiley, 2001. 59, 70

[45] W. Lu and F. Chu, “Condition monitoring and fault diagnostics of wind turbines,” in *Proceedings of IEEE Prognostics and System Health Management Conference year=2010,*. 60

[46] W. Yang, P. J. Tavner, C. J. Crabtree, and M. Wilkinson, “Cost-effective condition monitoring for wind turbines,” vol. 57, pp. 263 –271, Jan 2010. 60

[47] Z. Chen, X. Lian, H. Yu, and Z. Bao, “Algorithm of data mining and its application in fault diagnosis for wind turbine,” vol. 2, pp. 240 –243, nov. 2009. 61

[48] A. Zaher and S. McArthur, "A multi-agent fault detection system for wind turbine defect recognition and diagnosis," pp. 22 –27, jul. 2007. 61

[49] X. Ye, K. Veeramachaneni, Y. Yan, and L. A. Osadciw, "Unsupervised learning and fusion for failure detection in wind turbines," in *Proceedings of 12th International Conference on Information Fusion*, (Seattle,Washington, USA), July 2009. 64, 78

[50] X. Ye, W. Gao, Y. Yan, and L. A. Osadciw, "Multiple tests for wind turbine fault detection and score fusion using two-level multidimensional scaling (mds)," in *Proceedings of SPIE symposium on Defense, Security, and Sensing*, (Orlando, FL, USA), April 2010. 65

[51] L. A. Osadciw, Y. Yan, X. Ye, G. Benson, and E. White, "Wind turbine diagnostics based on power curve using particle swarm optimization," in *book Wind Power Systems: Applications of Computational Intelligence*, Springer, 2010. 66

[52] Y. Yan, G. Kamath, L. A. Osadciw, G. Benson, P. Legac, P. Johnson, and E. White, "Fusion for modeling wake effects on wind turbines," in *Proceedings of 12th International Conference on Information Fusion*, (Seattle,Washington, USA), July 2009. 69

[53] K. Hisada and F. Arizino, "Reliability tests for weibull distribution with varying shape-parameter, based on complete data," *Reliability, IEEE Transactions on*, vol. 51, pp. 331 – 336, sep. 2002. 69

[54] T.-H. Yeh and L. Wang, "A study on generator capacity for wind turbines under various tower heights and rated wind speeds using weibull distribution," *Energy Conversion, IEEE Transactions on*, vol. 23, pp. 592 –602, jun. 2008. 69

[55] C. A. Joshua Cassity and D. Parker, "Applying weibull distribution and discriminant function techniques to predict damaged cup anemometers in the 2011 phm competition," in *Proceedings of 2011 Prognostics and Health Management*, (Montreal, Quebec, Canada), September 25-29 2011. 70

[56] O. Bennouna, N. Heraud, H. Chafouk, and G. Notton, "Influence of model parameters on the diagnosis of the wind turbine generator," pp. 1 –5, jul. 2009. 79

[57] F. Gao and H. Tong, "Differential evolution: An efficient method in optimal pid tuning and on–line tuning," in *Proceedings of the First International Conference on Complex Systems and Applications*, (Wuxi, China), 2006. 83

[58] N. Hansen, "The cma evolution strategy: A tutorial," tech. rep., March 7, 2010. 83

[59] X. Ye, Y. Yan, and L. A. Osadciw, "Learning decision rules by particle swarm optimization (pso) for wind turbine fault diagnosis," in *Proceedings of Prognostics and Health Management*, (Portland, OR, USA), October 10-15 2010. 84

[60] Y. M. C. S. H. A. C. K. S. Z. Hameed, Y. S. Hong, "Condition monitoring and fault detection of wind turbines and related algorithms: A review," vol. 13, pp. 1 – 39, 2009. 96

[61] X. W. Bin Lu, Yaoyu Li and Z. Yang, "A review of recent advances in wind turbines condition monitoring and fault diagnosis," in *Proceedings of 2009 Power Electronics and Machines in Wind Application*, Juner 24-26 2009. 96

[62] M. Schwabacher and K. Goebel, "A survey of artificial intelligence for prognostics," (Arlington VA), 2007. 101

VITA

NAME OF AUTHOR: Xiang Ye

MAJOR: Electrical and Computer Engineering

EDUCATION:

Ph.D. in Electrical and Computer Engineering	June 2012	Syracuse University Syracuse, NY
M.S. in Electrical Engineering	Aug. 2009	Syracuse University Syracuse, NY
M.S. in Engineering Management	July 2006	Pittsburg State University Pittsburg, KS
B.S. in Electrical Engineering	June 2005	Wuhan University of Technology, China

PUBLICATIONS:

1. Daniel Nikovski, Alan Esenther, Xiang Ye, "Bayesian networks for Matcher Composition in Automatic Schema Matching", 28th IEEE International Conference on Data Engineering, Washington, DC, USA, April 1-5, 2012.

2. Xiang Ye, Yanjun Yan, Lisa Ann Osadciw, "Learning Decision Rules by Particle Swarm Optimization (PSO) for Wind Turbine Fault Diagnosis," in *Proceeding of Prognostics and Health Management*, Portland, OR, October 10-15, 2010.

3. Lisa Ann Osadciw, Yanjun Yan, Xiang Ye, Glen Benson and Eric White, "Wind Turbine Diagnostics based on Power Curve Using Particle Swarm Optimization," accepted book chapter in *Wind Power Systems: Applications of Computational Intelligence*, Springer, 2010.

4. Xiang Ye, Weihua Gao, Yanjun Yan and Lisa A. Osadciw, "Multiple Tests for Wind Turbine Fault Detection and Score Fusion Using Two-level Multidimensional Scaling (MDS)," in *Proceeding of SPIE symposium on Defense, Security, and Sensing*, Orlando, FL, April 2010.

5. Xiang Ye, Kalyan Veeramachaneni, Yanjun Yan, and Lisa A. Osadciw, ”Unsupervised Learning and Fusion for Failure Detection in Wind Turbines,” in *Proceedings of 12th International Conference on Information Fusion*, Seattle, Washington, USA, July 6-9, 2009.

6. Xiang Ye, Ganapathi Kamath and Lisa Osadciw, ”Using Bayesian Inference for Sensor Management of Air Traffic Control System”, *IEEE Symposium on Multicriteria Decision Making*, Nashville, TN, March 30 - April 2, 2009.

7. Ganapathi Kamath, Xiang Ye and Lisa A. Osadciw, ”Using Swarm Intelligence and Bayesian Inference for Aircraft Interrogation”, in *Proceeding of Wireless Communication and Networks Conference 2008*, Las Vegas, NV, April 2008.

8. Rajani Muraleedharan, Xiang Ye and Lisa A. Osadciw, ”Prediction of Sybil Attack on WSN Using Bayesian Networks and Swarm Intelligence”, in *Proceeding of Wireless Sensing and Processing 2008*, Orlando, FL, March 2008.