ABSTRACT

Title of Thesis: A SYSTEM METHODOLOGY FOR FAULT DETECTION AND FAULT RESTORATION IN DISTRIBUTED ELECTRIC POWER SYSTEMS

Estefany Carrillo, Master of Science, 2017

Thesis directed by: Professor Huan Xu
Department of Aerospace Engineering

Due to advances in electronic technologies, many systems are becoming increasingly dependent on electric power for control and safety critical features [1]. When power failures occur, fault detection and fault restoration processes are essential to reestablishing the proper function of the system. This thesis presents a system methodology that addresses fault detection and fault restoration for distributed electric power systems and a case study on applying the developed methodology to an aircraft electric power system.

This thesis implements sensor placement algorithms in order to obtain the most information about the state of the system. These algorithms rely on a state estimation algorithm developed in [21] in order to quantify the information obtained by the sensor measurements. Then, the sensor placement and state estimation algorithms are combined with a controller synthesis method. This results in a system methodology that periodically monitors the system for faults and brings the system back to safe operation.

A SYSTEM METHODOLOGY FOR FAULT DETECTION AND FAULT RESTORATION IN DISTRIBUTED ELECTRIC POWER SYSTEMS

by

Estefany Carrillo

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2017

Advisory Committee:
Dr. Huan Xu, Chair/Advisor
Dr. Derek A. Paley, Co-Advisor
Professor Robert W. Newcomb

Dedication

To my family.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Huan Xu for giving me an invaluable opportunity to work on challenging and interesting projects. Her encouragment, help and advice have supported my work and made it possible to present this thesis. Everytime I needed to talk with her, she was available and ready to listen to my questions and ideas. It has been a pleasure to work with and learn from such an extraordinary individual. I would also like to thank my co-advisor, Dr. Derek A. Paley and Professor Newcomb for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

I owe my deepest thanks to my family - my mother and father who have always stood by me and supported my academic endeavors, and have made personal sacrifices to see my dreams turn into reality. I would also like to thank my husband who has served as a role model and advisor throughout my graduate studies. Words cannot express the gratitude I owe them. My classmates have been a crucial factor in my finishing smoothly. I'd like to express my gratitude to Mukul, Raji and Vidya for their friendship and support. It is impossible to remember all, and I apologize to those I've inadvertently left out.

Last but not least, thank you all and thank God!

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1: Introduction

It has been shown that many systems such as aircraft and marine systems are becoming increasingly dependent on electric power for control and safety critical features [29]. This move to more-electric architectures, such as electrical power actuators, has been motivated by advancements in electronics technology that can result in improved reliability, life cycle cost and performance [1]. For instance, recent advances in solid-state power electronics and reductions in the weight and volume of controls for high-speed electric machines have enabled the utilization of electrical power instead of hydraulic, pneumatic, and mechanical power in aircraft [30].

In this thesis, a system methodology is presented that is capable of maintaining a highly reliable, fault tolerant and autonomously controlled electrical power system. For instance, we show that the methodology can produce accurate fault detection results, i.e. components with an erroneous voltage or no voltage output are identified in 80% to 100% of all failure configurations. This methodology implements a process that addresses fault detection and fault restoration for an electrical power system in a closed-loop fashion such that the results from the fault detection step are passed to and used by the fault restoration step and the process transitions from one step to the other periodically or as needed. After accurately identifying the location of

the fault, a plan to redistribute power is devised and the system is reconfigured back to safe operation.

1.1 Motivation

Due to the increasing reliance of subsystems on electrical power, it is critical the electric power system functions properly [32]. This makes knowing when failures happen, and determining the proper response when failures occur, to be of utmost importance. Ensuring the correct behavior of the electrical power system can be framed as a two-step problem in which the first step involves fault detection and the second step involves fault restoration. Fault detection deals with identifying the components that have failures, equivalent to estimating the discrete state of the system, while fault restoration is applied to guarantee electrical power redistribution in order to satisfy safety requirements. Throughout this thesis, a discrete model for the electrical power system is used in order to make integration between fault detection and fault restoration easier and reduce the state space size as well as computation time [31].

Efforts have been made to design fault detection methods for electrical power systems including those that use wavelet transform, neural networks, and genetic algorithms (GAs) combined with fuzzy logic [2]. Wavelet neural networks have been applied to power system fault detection due to the combined advantages of the wavelet transform and neural networks. Wavelet analysis is a powerful mathematical tool for fault signal analysis and neural networks are capable of dealing with

pattern-recognition and self-learning [3]. For instance, artificial neural networks (ANNs) have been used to treat fault diagnosis as a pattern-recognition problem. After determining the network parameters and configuration, the network can be trained with different fault patterns in order to estimate the location of the fault in the system. The drawback of neural networks is that they come with severe workload for the training task, making them inefficient in real large scale power system applications [4].

A hybrid combination of genetic algorithms and fuzzy logic has also been applied to detect faults in electrical power systems. A genetic algorithm is used to fine-tune fuzzy parameters that determine fuzzy subsets to then construct a membership function to measure the likelihood of a possible faulty component and its degree of membership to each of the fuzzy subsets based on states of circuit breakers [5]. The accuracy of this algorithm relies on the fine-tuning of the fuzzy parameters, which has been shown to be too time consuming for on-line use. Other fault detection methods make use of model-based reasoning (MBR) techniques [6]. These techniques use artificial intelligence (AI) to learn from existing models and their diagnostics. Diagnostics are utilized to provide predictions about the behavior of devices. Then, these predictions are compared with the results of other predictions and observations. A discrepancy between the predictions and the observations may be an indication of a faulty component in the system [6]. One of the major drawbacks of this technique is the lack of reusable and extensible software tools that would facilitate the development of MBR systems for problems within a particular engineering application. In addition, the existing models may have their own

limitations and assumptions making them less applicable for certain problems [7].

In the area of fault restoration, various methods have been developed to restore the power supply to affected areas in the system after a fault occurs. One of these methods includes a recovery path search based on the genetic algorithm. This method formulates the fault restoration problem as a multi-objective, multi-constrained and combinatorial problem which solves for the optimal recovery path that minimizes the number of switch operations and power losses [8]. A drawback of this method is local convergence which prevents the optimal strategy from achieving the global minimum of the objective function. Another method proposed to obtain a recovery path is based on matrix elimination. This method considers a subsystem switch model and defines an incidence matrix based on the connectivity of the subsystems. Then, by elimination of the elements where the related subsystems are represented in the matrix, a recovery path search can be quickly obtained [9]. This approach requires searching for a complete recovery path as far ahead as possible to provide power restoration, making it inefficient for on-line use.

So far, various fault detection and fault restoration methods have been discussed. These methods can be applied to separately address fault detection and fault restoration, each with advantages and drawbacks. However, fault detection and fault restoration methods can be integrated into a process that periodically monitors the electrical power system for faults and devises a recovery strategy as soon as a fault is discovered. This integration facilitates the rapid exchange of information between the fault detection and fault restoration methods. Once the state of the system is estimated, it can be passed to the restoration method without

extra processing, and once the restoration strategy is applied, the state estimation method can be run again. Similarly, related work on combining offline fault-aware task scheduling algorithms and online fault restoration algorithms for a single point of failure is explored in [33], [34].

1.2 Thesis Contribution

This thesis presents a system methodology that combines fault detection and fault restoration techniques in order to monitor the system for faults and restore power for electrical power systems within the context of aircraft applications. The proposed methodology solves the problem of detecting and handling failures holistically and systematically with applicability to a wide range of electrical power systems and minimal dependency on extra hardware. The methodology functions in two modes. In one mode, the required pre-processing which entails determining a sensor placement strategy for the system is performed off-line. The sensor placement strategy relies on the results of the state estimation method which will be explained in detail in section II. The second mode is executed on-line. While in this mode, the methodology performs state estimation periodically to identify faulty components in the system. The resulting state estimate is passed to the fault restoration step in which a strategy is devised to restore power in the system according to safety and operation requirements.

The research presented in this thesis consists of three main components: sensor placement, state estimation, and controller synthesis. In Figure 1.1, the workflow of

the implementation of these components with their respective inputs and outputs is shown.

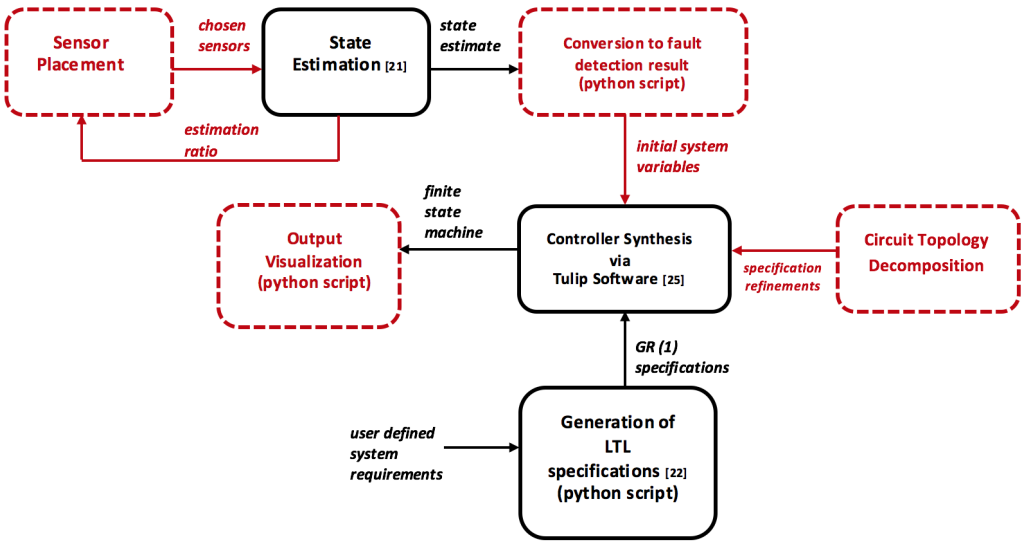

Figure 1.1: Workflow diagram implementing all the steps of the system methodology. Blocks with dashed lines correspond to algorithms and code developed in this thesis. Blocks with solid lines correspond to algorithms and code developed in previous work with the corresponding references.

The following sections provide a quick overview of related work in the areas of sensor placement, state estimation and controller synthesis as well as a summary of the thesis contribution to each one of these areas.

1.2.1 Sensor Placement

A significant amount of work has been done in the area of sensor placement for different applications. For instance, the work in [10] addresses the problem of selecting control nodes for a symmetric complex network in order to drive the

network to a target state by optimizing a function of the observability Gramian. The work in [11] explores some heuristics that decide on sensor placement for a power system based on the degree of observability of the system states quantified by an empirical observability Gramian. Both methods are shown to be computationally demanding.

Different approaches are shown in [12], [13], and [14], in which greedy algorithms with sub-optimality guarantees are provided. A drawback of most of these approaches is that they assume a model of the system with accurate parameters, which makes them subject to modeling errors. In addition, these algorithms may not find the optimal solution but can guarantee a sub-optimal solution even in worst case scenarios. For other applications of sensor placement, [15] and [16] look at the problem of choosing sensors in Wireless Sensor Networks (WSN) with the objective of minimizing the use of constrained resources or maximizing sensor coverage and quality. Other applications in robotics, alternatively, look at the coupling of sensor placement and state estimation [17].

In this thesis, we take an approach based off the theoretical concepts of the greedy and combinatorial methods explored in [13] and [15] respectively and implement sensor placement algorithms that seek to maximize a given function. Our contribution is in the determination of this function, which is defined as the estimation success ratio. The estimation success ratio is a function of the placement of sensors which affects state estimation performance. This state estimation process was developed in previous work [23] and will be discussed in more detail in section II. For a given sensor placement, the value of the estimation success ratio will vary

according to how many state configurations the state estimation process is able to produce a unique state estimate. Modifying the sensor placement based on results obtained from the state estimation process creates a coupling of these two algorithms, which is represented by the connections between the sensor placement and state estimation blocks as shown in Figure 1.1. The algorithms compute a sensor placement strategy iteratively until the required number of sensors are selected or a state estimation percentage is reached.

1.2.2 State Estimation

In the area of state estimation, some methods use optimization-based estimation in which the state vector of an electrical power system is determined by a least squares fit of nonlinear equations relating state variables (bus voltage magnitudes and phase angles) and measurements (voltage and current magnitudes) [18]. A drawback of these methods is that an efficient solution with global convergence is difficult to achieve. Other methods in [19], [20] rely on hardware redundancy in which multiple sensors and actuators are utilized to measure and determine the state of the system. However, the extra hardware makes these methods undesirable for aircraft applications which have restrictive weight requirements. In this thesis, a state estimation approach developed in [21] without the drawbacks mentioned is used and modified to work within the proposed methodology.

In this thesis, a function to systematically iterate through all possible state configurations and obtain a state estimate for each configuration is implemented in

Python 2.7. This function then computes and returns the estimation success ratio. Another function is implemented to translate a state estimate to a fault detection result in order to initialize the input to the controller synthesis. This function is represented in the conversion to fault result block as shown in Figure 1.1.

1.2.3 Controller Synthesis

In the area of fault restoration, instead of searching for a complete recovery path, a different strategy is utilized and incorporated to the methodology. Linear temporal logic (LTL) language is used to automatically design a correct-by-design controller that provides a formal guarantee of system correctness, i.e. proper power distribution to essential components in the electrical power system. This system correctness can be described by a set of user-defined system requirements. The translation from these system requirements written by the user to LTL specification language is done automatically by a Python script developed as part of the work in [22]. This script outputs LTL specifications and converts them to generalized reactivity (1) form, which is explained in [24]. Then, these GR(1) specifications are passed to the controller synthesis block as shown in Figure 1.1. This synthesis is performed using Tulip software tools. This software is explained in detail in [25]. The controller synthesized by Tulip then outputs a reactive protocol that ensures the system can be reconfigured dynamically to maintain system correctness even in the presence of failures as demonstrated in [22]. The reactive protocol is represented by a finite state machine that can be used to extract a sequence of actions that will

reconfigure the system back to safe operation.

In this thesis, the results from the state estimation method are integrated to the controller synthesis method and a graphic simulation of the cicuit topology changes dictated by the state transitions in the finite state machine outputted by the controller synthesis. This corresponds to the output visualization block as shown in Figure 1.1. A decomposition of the circuit is proposed for the implementation of the distributed controller synthesis, including refinements for specifications to handle the interfacing between the subcomponents. This is executed in the circuit topology decomposition block as shown in Figure 1.1.

1.3 Thesis Overview

The scope of this thesis covers the design, simulation and results of the proposed system methodology and its application to an aircraft electrical power system. Chapter 2 provides background information on electrical power systems and presents the problem set up. It also discusses the state estimation technique previously developed in [21] and the linear temporal logic language and formalisms used for the reactive and distributed controller synthesis part of the methodology. The main contribution of Chapter 3 is the presentation of the sensor placement algorithms. Results from the algorithms are simulated on representative electrical power system topologies. These algorithms are then compared in terms of accuracy and performance and the chosen approach is tested further by considering unreliable readings from sensors. Chapter 4 provides a detailed description of the system methodology

design flow. Additionally, it gives an overview of the application of formal specifications in synthesizing centralized and distributed controllers. Finally, a case study on applying the system methodology on an aircraft electric power system is presented in Chapert 5 along with simulation results for centralized and distributed approaches. Finally, Chapter 6 concludes the work and discusses directions for research in the future. In summary, these are the key contributions of this thesis:

1. The determination of a function, which is referred to as the estimation success ratio, for the implementation of combinatorial and greedy sensor placement methods.

2. The integration of a given state estimation method, for which sensor placement algorithms are implemented to improve its performance, and controller synthesis methods using Linear Temporal Logic into a system methodology that addresses the problem of fault detection and fault restoration holistically.

3. The demonstration of the application of the system methodology to a large electric power system for aircraft.

Chapter 2: Background

2.1 Electrical Power Systems

In this thesis, an electric power system for an aircraft is considered. This system is composed by a certain number of generators that serve as primary power sources to a set of loads through dedicated AC buses. Then, each AC bus delivers power to a DC bus through a rectifier unit or a transformer rectifier unit.

To control the flow of power, the system uses contactors (high-power switches). These contactors establish connections between components. In the case of a generator or switch failure, the system can be reconfigured by changing the open or closed status of contactors, hence redirecting power to the different buses or loads.

A representative electrical power system topology is shown in Figure 2.1. This topology includes a combination of generators G, contactors C, buses B, rectifier units R and sensors S. The following is a brief description of the components used througout this work.

Generators: AC generators can operate at either high voltages, which can connect to the high-voltage AC buses, or low voltages, which feed directly to the low-voltage buses.

Buses: High-voltage and low-voltage AC and DC buses deliver power to a

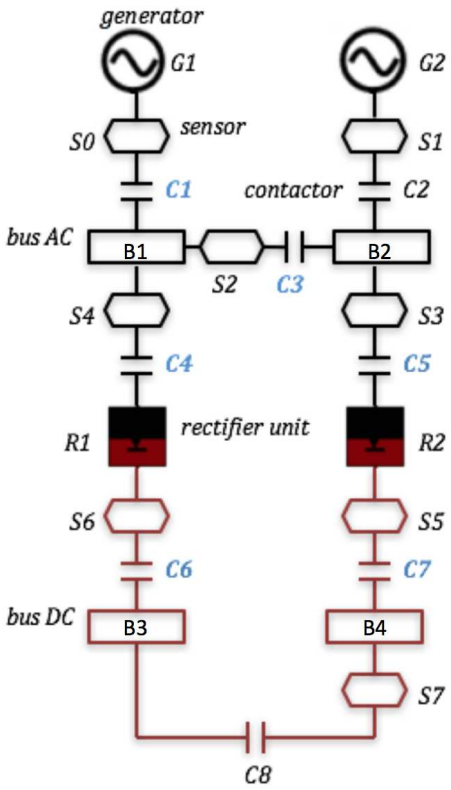

Figure 2.1: A single-line diagram of a simple circuit with AC components and DC components. In this circuit, controllable contactors are shown in blue (C1;C3;C4;C5;C6;C7) and uncontrollable contactors are shown in black (C2;C8).

number of sub-buses, loads, or power conversion equipment. Buses can be essential or non-essential depending on the power requirements on the loads. For example, essential buses supply loads that should always remain powered, such as the flight actuation subsystem, while non-essential buses have loads that may be shed in the case of a fault or failure, such as cabin lighting.

Contactors: Contactors are high-power electronic switches that connect the flow of power from sources to buses. Depending on the power status of generators and buses, contactors can reconfigure, i.e., switch between open and closed. Con-

tactors provide the actuation for reconfiguration of the topology of the electrical power system. By changing the paths through which power is delivered from generators to loads, contactors are used to bring the system back to safe operation. Throughout this thesis, the symbol for contactors should not be confused with that for capacitors.

Transformer Rectifier Units: Rectifier Units (RUs) convert three-phase AC power to DC power. Transformer Rectifier Units (XFMRs) combine a rectifier unit and a step-down transformer to additionally lower the voltage.

Batteries: Batteries can be used as auxiliary generation sources. They provide short-term power during emergency conditions while alternative sources are being brought online.

The following is a detailed description of the model of the representative electrical power system in Figure 2.1 for the purpose of applying state estimation, sensor placement and control synthesis.

2.2 Problem Description

A discrete model is considered for all electrical power systems used in this work. Using a discrete model has the advantage of making the integration of the state estimation and controller synthesis easier since the output of the state estimation process becomes the input to the controller synthesis algorithm. The discrete model of the system provides an abstract model which allows for a discrete controller to be synthesized. Additionally, a discrete model results in a much smaller state

space than a continuous model, which leads to a reduced computational complexity. Therefore, continuous values of voltage and current, as well as health statuses of components in the system are discretized before performing state estimation and sensor placement.

Additionally, the electric power system topology can be represented by a graph data structure $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ refers to the set of nodes and $\mathcal{E}$ refers to the set of edges in the graph. The set of nodes $\mathcal{N} = \{n_1, ..., n_n\}$ includes the following components: generators $\mathcal{G}$, rectifier units $\mathcal{R}$, and voltage sensors $\mathcal{S}$. The set of edges $\mathcal{E} = \{e_1, ..., e_e\}$ contains all contactors between components. The status of contactors in $\mathcal{E}$ can either be open (0) or closed (1). All edges in the graph are bidirectional, except for the incoming only edges on the AC side and the outgoing only edges on the DC side of the nodes that correspond to rectifier units.

The states of generators $\mathcal{G} \in \mathcal{N}$ and rectifier units $\mathcal{R} \in \mathcal{N}$ are uncontrollable. These states can take values of unhealthy (2) if the component is online and outputs an inadmissible voltage value, healthy (1) if the component is online and outputs the correct voltage, and offline (0) if the component does not output any voltage. The states of sensors $\mathcal{S} \in \mathcal{N}$ depend on the status of generators, rectifier units, and contactors. A live path occurs when there exists a simple path in G that connects two nodes with no offline component along the path (including end nodes) and with all contactors in the path closed. Readings from a sensor s can take the following values (i) *improper voltage*: if there is a live path between sensor s and a generator $g \in \mathcal{G}$, but either generator g or rectifier $r \in \mathcal{R}$ along such a path is unhealthy; (ii) *admissible voltage*: if for all $g \in \mathcal{G}$ that have a live path to s, all g and all rectifier

units along such path are healthy; (iii) or *no voltage*: if there exists no live path between sensor s and any generator $g \in \mathcal{G}$.

2.3 Dynamic State Estimation

In [21], the authors utilize software-based dynamic estimation strategies to obtain high-accuracy state estimates with a limited number of sensors. The proposed state estimation employs a greedy strategy that adaptively generates switching actions that reconfigure the system as to maximize the one-step expected uncertainty reduction. These actions are performed by a set of controllable contactors. In the estimation process, the state of the system is expressed as a valuation of the state of individual components $n \in \mathcal{G} \cup \mathcal{R}$ and uncontrollable contactors $e \in \mathcal{C}' \subseteq \mathcal{C}$. Controllable and uncontrollable contactors are user-defined and remain unchanged throughout the estimation process. The dynamic state estimation method can be executed by an embedded controller, which will be referred to as the fault detection controller. This controller only controls a subset $\mathcal{C} \setminus \mathcal{C}'$ of contactors. In Figure 2.1, controllable contactors are shown in blue and include (C1;C3;C4;C5;C6;C7). Thus, the state estimation process runs the fault detection controller to adaptively estimate the discrete state of the circuit. The controller takes *actions* like closing or opening controllable contactors, and then reads voltage sensor measurements. Hence, the controller can reconfigure the system and collect measurements to gain information about the unknown state.

Let x be the state of the circuit, modeled as a random variable X that can

only be determined by sensor measurements mapped back to a set of possible states of the circuit. First, the circuit is assumed to be in an unknown state $x_0 \in \Omega$, where Ω represents the set of all possible states. System faults are assumed to be independent and x_0 is assumed to remain fixed during the estimation process. Let $V = \{(v_i)_{i\in\{0,...,t\}}\}$, represent the set of actions that can be performed on the controllable contactors. Let Y be the set of sensor measurements, with $y = \mu(v, x) \in Y$ representing the unique outcome of performing action v for a system state x. For actions $\{v_0, ..., v_t\}$ and outcomes $\{y_0, ..., y_t\}$ observed up until step t are represented by the partial realization $\psi_t = \{(v_i, y_i)_{i\in\{0,...,t\}}\}$.

The estimation process adaptively gets to the actual state x_0. Let $D(y, v)$ with $y = \mu(v, x_0)$ be the set of states that are indistinguishable from x_0 under action v. Let $S_t = h(v_{0:t}, x_0)$ be an extension of this set containing states that produce the same set of outcomes as x_0 under the same set of actions $\{v_0, ..., v_t\}$. At each time step t that a new action $v' \notin \psi_t$ is taken, there exists a recursive relation between the two sets of states:

$$h(v_{0:t} \cup \{v'\}, x_0) = h(v_{0:t}, x_0) \cap D(\mu(v', x_0), v'), \tag{2.1}$$

and S_t becomes:

$$S_t = \cap_{i\in\{0,...,t\}} D(\mu(v_i, x_0), v_i). \tag{2.2}$$

A strategy π is defined as a function of partial realizations to actions where $\pi(\psi_t) = v_{t+1}$. For an initial congifuration with x_0, v_0 and y_0, the sequence of

decision, measurement and update operations of the estimation process are expressed in Equations (2.3a)-(2.3c) respectively:

$$v_i = \pi(\psi_{i-1}) \tag{2.3a}$$

$$y_i = \mu(v_i, x_0) \tag{2.3b}$$

$$\psi_i = \psi_{i-1} \cup (v_i, y_i) \tag{2.3c}$$

The goal is to reduce the uncertainty in the state estimate, thus minimizing the number of indistinguishable states, by performing $k \in \{0, ..., t\}$ actions that maximize the objective function f. This function maps the set of actions $A \subseteq V$ under state x_0 to reward $f(A, x_0)$, which measures the reduction in uncertainty of X represented by the probability distribution $\mathbb{P}[x]$ through performing k actions:

$$f(v_{0:k}, x_0) = -\mathbb{P}[S_k] = -\sum_{x \in S_k} \mathbb{P}[x]. \tag{2.4}$$

Thus, the estimation will find the strategy π that allows the best expected estimate for the state as shown in (2.4). We denote $|\tilde{V}(\pi, x_0)| \subseteq V$ the set of all the actions performed under the strategy π, the state of the system being x_0.

$$\pi^* \in \arg\max_{\pi} \mathbb{E}[f(\tilde{V}(\pi, X), X)], \tag{2.5}$$

subject to $|\tilde{V}(\pi, x)| \leq k$ for all x, and with expectation taken with respect to $\mathbb{P}[x]$. A greedy strategy is then used to select, at each step t, the action v_{t+1} that maximizes

the expected one-step gain in uncertainty reduction with respect to all previous actions:

$$v_{t+1} \in \arg\max_{v \in \tilde{V}} \mathbb{E}[f(v_{0:t} \cup \{v\}, X) - f(v_{0:t}, X)|\psi_t]. \tag{2.6}$$

The overall goal is to design a strategy the fault detection controller runs to adaptively estimate the discrete state of the circuit by taking actions (i.e., closing and opening controllable contactors), and then reading voltage sensor measurements. By changing the number and locations of sensors, it may be possible to improve state estimation performance. In section III, a sensor placement algorithm is proposed to determine the number of sensors and locations with the goal of maximizing the performance of the state estimation method. The estimation process is summarized in Algorithm 1.

Algorithm 1 Adaptive Greedy Strategy

Input: Probability measure $\mathbb{P}[x]$ on Ω, number of actions to perform k. The system is in the state $x_0 \in \Omega$, fixed and unknown, and the controlled contactors are in some configuration v_0.

Output: Partial realization ψ_k and the set S_k of compatible states after k actions are taken based on the strategy π_{greedy}

Take the measurement $y_0 = \mu(v_0, x_0)$.
$\psi_0 = \{(v_0, x_0)\}$
for all $t \in \{1, ..., k\}$ **do**
    $v_t = \pi_{greedy}(\psi_{t-1})$
    Perform action v_t
    Take measurement $y_t = \mu(v_t, x_0)$
    $\psi_t = \psi_{t-1} \cup \{v_t, y_t\}$
    $S_t = S_{t-1} \cap D(y_t, v_t)$
    Compute $\mathbb{P}[x|\psi_t]$ (Bayesian update)
end for
return $(\psi_k, S_k, \mathbb{P}[x|\psi_k])$

2.4 Linear Temporal Logic (LTL)

LTL is a version of temporal logic, a specification language that can be used to formally specify and verify behavioral properties of a system. LTL is well suited for describing certain properties of electrical power systems and solving the type of problems in which systems must react to a dynamic, a priori unknown environment [27]. These systems are referred to as reactive systems. To understand how LTL works, a brief overview of some basic definitions is presented below.

Definition 1: A system consists of a set V of variables. The domain of V, denoted $dom(V)$, is the set of valuations of V. A state of the system is an element $v \in dom(V)$.

Definition 2: An atomic proposition is the building block of LTL. It is defined as the statement on a valuation $v \in dom(V)$ with a unique truth value (True or False) for a given v. Let the valuation $v \in dom(V)$ be a state of the system, and p be an atomic proposition. Then $v \Vdash p$, read v satisfies p, if p is True at that state v. Otherwise, $v \nVdash p$.

In the electric power system domain, the set of variables includes, for instance, generator and rectifier unit statuses. Valuations of these variables include the health values of generators and rectifier units. An atomic proposition could state that each generator in the system be healthy as well as each rectifier unit. In addition, LTL includes Boolean connectors like negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$), material implication ($\rightarrow$), and two basic temporal modalities *next* ($\bigcirc$) and *until* ($\mathcal{U}$). Using these operators and propositions, it is possible to specify a wide range

of requirements on the desired behavior of a system and environment assumptions. A formula is either an atomic proposition or a sentence comprised of atomic propositions connected by logical operators. For a given set π of atomic propositions, an LTL formula is defined as follows:

1. any atomic proposition $p \in \pi$ is an LTL formula.

2. given LTL formulas φ and ψ over π, $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\bigcirc\varphi$ and $\varphi \,\mathcal{U}\, \psi$ are also LTL formulas.

For a set of valuations V and a set π of atomic propositions over valuations $v \in domn(V)$, LTL formulas over π are interpreted over infinite sequences of states. For instance, the formula $\bigcirc\varphi$ holds for a sequence of states at the current step of the sequence if in the next step φ is true. Some formulas derived from the formulas presented above include *eventually* ($\Diamond$) and *always* ($\Box$). For instance, $\Diamond\varphi$ is equivalent to $(true \,\mathcal{U}\, \varphi)$. Also, the operators $\Diamond$ and $\Box$ are logical duals as $(\Box\varphi)$ is equivalent to $(\neg\Diamond\neg\varphi)$ and $(\Diamond\varphi)$ is equivalent to $(\neg\Box\neg\varphi)$. The formula $\Diamond\varphi$ states that φ will be true at some point in the future, while $\Box\varphi$ is satisfied if and only if φ is true for all points.

The semantics of LTL is given as follows. Let $\sigma = v_0 v_1 v_2 ...$ be an infinite sequence of valuations of variables in V. Let φ and ψ be LTL formulas. We say that φ holds at position $i \geq 0$ of σ, written $v_i \models \varphi$, if and only if φ holds for the remainder of the execution σ starting at position i. Then, the satisfaction of φ by σ is inductively defined as:

1. for atomic proposition p, $v_i \models p$ if and only $v_i \Vdash p$;

2. $v_i \models \neg\varphi$ if and only if $v_i \not\models \varphi$;

3. $v_i \models \varphi \vee \psi$ if and only if $v_i \models \varphi$ or $v_i \models \psi$;

4. $v_i \models \bigcirc\varphi$ if and only if $v_{i+1} \models \varphi$; and

5. $v_i \models \varphi \,\mathcal{U}\, \psi$ if and only if $\exists\ k \geq i$ such that $v_k \models \psi$ and $v_j \models \varphi$ for all j, $i \leq k < j$.

Based on this definition, $\bigcirc\varphi$ holds at position i of σ if and only if φ holds at the next state v_{i+1}, $\Box\varphi$ holds at position i if and only if φ holds at every position in σ starting at position i, and $\Diamond\varphi$ holds at position i if and only if φ holds at some position $j \geq i$ in σ.

Let Σ be the collection of all sequences of valuations of V. Then, the truth value of a formula is determined by an assignment $\mathcal{A}$, which is a mapping of atomic propositions in the domain Σ to truth values: $\mathcal{A} : \Sigma \to \{0, 1\}$. Thus, if there are n atomic propositions in the domain, there are 2^n possible assignments for a given formula. A formula φ is satisfied by an assignment $\mathcal{A}$ that causes the overall formula to evaluate to true. A system composed of the variables V is said to satisfy φ if $\sigma \models \varphi$ for all $\sigma \in \Sigma$. A set of models Σ satisfies φ if every model in Σ satisfies φ.

2.4.1 Centralized Control Synthesis

We can now proceed to show how to use Linear Temporal Logic (LTL) to translate system specifications to temporal logic language in order to formally state the reactive synthesis problem. Let us define the system model as follows. The

system variables are classified into sets of environment variables E and controlled variables P. Let $s = (e, p) \in dom(E) \times dom(P)$ be the state of the system. Consider an LTL specification φ of assume-guarantee form

$$\varphi = \varphi_e \rightarrow \varphi_s, \tag{2.7}$$

where φ_e characterizes the assumptions on the environment and φ_s characterizes the system requirements. The synthesis problem is then concerned with designing a correct-by-construction controller that can reconfigure the electrical power system to address different failure scenarios according to all system specifications. System specifications are provided to define correct behaviors of the system. In addition, the synthesis problem can be viewed as a two-player game between the environment and the controlled plant: the environment attempts to falsify the specification in (2.7) and the controlled plant tries to satisfy it.

The controller outputs a strategy, a partial function $f : (s_0 s_1 ... s_{t-1}, e_t) \rightarrow p_t$, that chooses the move of the controlled variables based on the state sequence so far and the behavior of the environment so that the system satisfies φ_s as long as the environment satisfies φ_e. Thus, the output of the controller will be a different sequence of actions in each execution since the environment may be different. This captures the reactive nature of the controller. The formula φ is true if φ_s is true, and the system specifications are satisfied. When φ_e is false, i.e. the environment is inadmissible, the synthesis problem becomes unrealizable and there is no guarantee about the behavior of the system.

It has been shown that the synthesis problem has a doubly exponential complexity in [23]. For a subset of LTL, namely generalized reactivity (1) (GR(1)), it has been shown that this problem can be solved in polynomial time (polynomial in the number of valuations of the variables in E and P) [24]. We describe GR(1) and the need for it in more detail at the end of this chapter. GR(1) specifications restrict φ_e and φ_s to take the following form, for $\alpha \in \{e, s\}$,

$$\varphi_\alpha := \varphi^\alpha_{init} \wedge \bigwedge_{i \in I^\alpha_1} \Box \varphi^\alpha_{1,i} \wedge \bigwedge_{i \in I^\alpha_2} \Box \Diamond \varphi^\alpha_{2,i}, \tag{2.8}$$

where φ^α_{init} is a propositional formula characterizing the initial conditions; $\varphi^\alpha_{1,i}$ are transition relations characterizing safe, allowable moves and propositional formulas characterizing invariants; and $\varphi^\alpha_{2,i}$ are propositional formulas characterizing states that should be attained infinitely often. The GR(1) specifications are then passed to the synthesizer available in the temporal logic planning (TuLiP) toolbox. For a thorough description of this tool, see [25]. The synthesizer returns a finite-state machine in which states are valuations of environment and controlled variables and transitions are actions that the controller can take to reach a desired state.

When a controller has access to all environment and controlled variables, it functions as a centralized controller. As the number of components increases, this can result in a large computational complexity for the controller. In order to address this issue, a distributed controller strategy is considered as follows.

2.4.2 Distributed Control Synthesis

By using a distributed controller, the system is divided into subsystems and a local controller is synthesized for each one of them. Thus, the synthesis task is divided into smaller subproblems and the computational complexity of the problem is reduced. Also, the distribution of power can be controlled and reestablished for each subcomponent of the system without affecting other subcomponents, leading to more robustness and resilience to failures. Synthesizing a local controller requires decomposition of global specifications into local specifications. For instance, let a system S be decomposed into S_1 and S_2. For $i = 1, 2$, let E_i and P_i be the environment variables and controlled variables for S_i such that $P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$. The overall environment assumptions φ_e and system guarantees φ_s are distributed into the two systems S_1 and S_2. Let φ_{e_1} and φ_{e_2} be LTL formulas containing variables in E_1 and E_2, respectively. Let φ_{s_1} and φ_{s_2} be formulas in terms of $E_1 \cup P_1$ and $E_2 \cup P_2$, respectively. Then, as long as the following conditions are satisfied, the distributed control protocol will satisfy the global specifications:

1. any sequence of actions from the environment that satisfies φ_e also satisfies $(\varphi_{e_1} \wedge \varphi_{e_2})$,

2. any sequence of actions of the system that satisfies $(\varphi_{s_1} \wedge \varphi_{s_2})$ also satisfies φ_s, and,

3. there exists two control protocols that realize local specifications $(\varphi_{e_1} \longrightarrow \varphi_{s_1})$ and $(\varphi_{e_2} \longrightarrow \varphi_{s_2})$.

By dividing the synthesis problem into smaller pieces, a distributed approach will enable the design of flexible control architectures in terms of modularity and integrability [26]. However, it also introduces the need for coordination between subsystems in order to ensure that the overall problem is solved. The interaction between subsystems may be needed for the synthesis to become realizable. This interaction can be achieved through an extra set of local guarantees. For example, let S_1 have an extra set of local guarantees ϕ_1 that interact with S_2 as environment assumptions denoted as ϕ_1', while ϕ_2 are the local guarantees provided by S_2 that interact with S_1 as environment assumptions denoted as ϕ_2'. Thus, if the local specifications in (2.9) and (2.10) hold, then the global specification $\varphi_e \longrightarrow \varphi_s$ is realizable.

$$\phi_2' \wedge \varphi_{e_1} \longrightarrow \varphi_{s_1} \wedge \phi_1, \tag{2.9}$$

$$\phi_1' \wedge \varphi_{e_2} \longrightarrow \varphi_{s_2} \wedge \phi_2. \tag{2.10}$$

2.4.3 Generalized Reactivity (1)

Using a "specify and synthesize" aproach, we can translate text-based system specifications to a temporal logic specification language. The goal is to automatically synthesize centralized and distributed controllers as well as to reduce the computational complexity. To do so, a fragement of LTL known as Generalized Reactivity (1) is considered. Generally, the synthesis method for a given LTL specification φ starts by constructing a Büchi automaton $\mathcal{B}_\varphi$, which is then determinized into a deterministic Rabin automaton. This double translation may reach complexity of

double exponent in the size of φ [27]. On the other hand, it has been shown that realizability and synthesis problems for GR(1) specifications can be solved efficiently in polynomial time and GR(1) is expressive enough to provide complete specifications of many designs [28]. In GR(1), the approach to solve the synthesis problem can be viewed as the solution of a two-player game. In addition, specification of the design to be synthesized is restricted to partial fragments of LTL and formulas in (GR(1)) can be expressed as:

$$(\square\diamond p_1 \wedge ... \wedge \square\diamond p_m) \longrightarrow (\square\diamond q_1 \wedge ... \wedge \square\diamond q_n) \tag{2.11}$$

Since the synthesis problem is framed as a two-player game between a system and an environment, in which the realizability of an LTL formula can be reduced to the decision of winner in the game, the goal of the system is to satisfy the specification regardless of the actions of the environment. Let φ be the winning condition, given by an LTL formula. A play σ, which is a sequence of states, is winning for the system if it is infinite and it satisfies φ. Otherwise, φ is winning for the environment. Then, the game is solved by attempting to decide whether the game is winning for the environment or the system. If the environment is winning, the specification is unrealizable. Otherwise, we can synthesize a winning strategy. This can be characterized by the formula in (2.11) which expresses that the system forces the game to visit p-states infinitely often and q-states infinitely often.

Chapter 3: Sensor Placement

3.1 Overview

A significant amount of work has been done in the areas of sensor and actuator placement for various applications. Some approaches use control theory to address such problems. For instance, the authors in [10] look at the problem of selecting control nodes (actuators) for a symmetric complex network in order to achieve controllability to a target state. Choosing the optimal set of control nodes is framed as a maximization among the minimum eigenvalues of the controllability Gramians corresponding to each set of control nodes as follows,

$$\max_{\mathcal{K}\subseteq\{1,\ldots,n\}} \lambda_{min}(\mathcal{W}_{\mathcal{K},\infty}), \tag{3.1}$$

where $\mathcal{K}$ is the set of control nodes, n refers to the number of network partitions and $\mathcal{W}_{\mathcal{K},\infty}$ corresponds to the controllability Gramian. In [14], the authors study the problem of minimal actuator placement in a linear system subject to an average control energy bound. To address this problem, they propose an algorithm that selects a minimal number of actuators as to achieve the energy bound, which is computed using the smallest eigenvalue of the controllability Gramian. In [11], the

authors propose the use of the observability Gramian instead. They present some heuristics that decide on sensor placement for a power system based on the degree of observability of the system states quantified by an empirical observability Gramian. Due to the reliance of these methods on the Gramian, they have been shown to be computationally demanding. Another drawback of these approaches is that they assume a model of the system with accurate parameters, which makes them subject to modeling errors.

Other approaches are built on the concept of submodularity. For example, authors in [12] present the problem of sensor selection in sensor networks and frame it as a maximization of a submodular function over uniform matroids. The authors demonstrate how a greedy sensor selection algorithm achieves performance within $(1 - \frac{1}{e})$ of the optimal solution. In [13], a simple greedy heuristic is developed to address the sensor and actuator placement problem for large complex networks. It is shown that the mapping from possible placements to scalar functions of the controllability and observability Gramians for the network is a *modular* or *submodular set function*. Thus, a greedy heuristic is used to find the placements that maximize this set function.

For the application of sensor placement in Wireless Sensor Networks (WSN), [15] and [16] seek to choose sensor locations that will minimize the use of constrained resources or maximize sensor coverage and quality. Other applications for sensor placement are in robotics as presented in [17]. The authors in [17] explore the coupling of sensor placement and state estimation. This thesis also seeks to explore this coupling and demonstrate how sensor placement algorithms can rely on state

estimation results for improvement. The combinatorial algorithm from [15] and the greedy algorithm from [13] are modified and adapted to address the sensor placement problem presented in this work due to their performance and applicability to the sensor placement problem in discrete models of electrical power systems. A description of the chosen algorithms is provided below.

3.2 Combinatorial Algorithm

It is shown in [15] that the combinatorial approach outperforms other algorithms in choosing the optimum sensor locations in a wireless sensor network. For this thesis, the combinatorial algorithm is adapted to choose the sensor locations that maximize the performance of the state estimation algorithm discussed in Section 2.3 for electrical power systems. This performance is quantified by computing a function of the results obtained by the state estimation algorithm when run for all possible state configurations for a given sensor placement. This function is referred to as the estimation success ratio throughout this thesis. The estimation success ratio is computed for a given sensor placement strategy S as follows,

$$f(S) = \frac{n_s}{n_t}, \tag{3.2}$$

where n_s is the number of state configurations for which the state estimation algorithm returns a unique state (i.e. number of successful runs) and n_t is the total number of state configurations. The combinatorial algorithm considers all possible combinations of the desired number of sensor nodes out of the total available sensor

nodes and outputs the combination of sensors that results in the maximum estimation success ratio. The pseudocode for the combinatorial algorithm is as follows:

Algorithm 2 Combinatorial Sensor Placement Algorithm

1: **procedure** MAXSELECTION(sensors)
2: Loop through the set of all possible combinations of the sensors,

$$s = \binom{n}{k}$$

where n is the total number of sensors and k is the number of required sensors
3: For each combination in s, attach the chosen sensors to the circuit configuration
4: Load in the database of inverse mapping from sensor measurements to compatible states of the circuit for the modified configuration
5: Calculate the estimation success ratio and store it in an array
6: Select the combination of sensors for which the estimation success ratio is maximum
7: **return** sensor placement strategy
8: **end procedure**

3.3 Greedy Algorithm

In [13], the problem of sensors and actuators placement is framed as a set function optimization problem of the form

$$\underset{S \subseteq V,\ |S|=k}{\text{maximize}} \quad f(S), \tag{3.3}$$

where $V = \{1, ..., M\}$ is a finite set and $f : 2^V \to \mathbb{R}$ is a set function. V represents the potential locations for the placement of sensors or actuators and f is a metric for how controllable the system is for a given set of placements. After proving that f is a monotone increasing submodular function, a greedy heuristic is used to obtain

a sub-optimal solution with guaranteed worst-performance bounds. For f to be submodular and monotone increasing, the following equations need to be satisfied respectively:

$$f(A \cup \{s\}) - f(A) \geq f(B \cup \{s\}) - f(B), \tag{3.4}$$

for all subsets $A \subseteq B \subseteq V$ and all elements $s \notin B$,

$$A \subseteq B \Rightarrow f(A) \leq f(B), \tag{3.5}$$

for all subsets A, $B \subseteq V$.

The greedy algorithm starts with an empty set, $S_0 \leftarrow \emptyset$, computes the gain $\Delta(a|S_i) = f(S_i \cup \{a\}) - f(S_i)$ for all elements $a \in V \setminus S_i$ and adds the element with the highest gain

$$S_{i+1} \leftarrow S_i \cup \left\{ \arg\max_a \Delta(a|S_i) | a \in V \setminus S_i \right\}. \tag{3.6}$$

Thus, the greedy algorithm obtains a sensor placement strategy by making a sequence of choices. At each step i, it assumes that i sensor locations are fixed and makes a greedy choice where to place the (i + 1)-th sensor. The sensor location added at each step is the one that maximizes the gain in the estimation success ratio. The estimation success ratio is a monotone increasing submodular function since adding a new sensor can never reduce the amount of information obtained and adding a new sensor to a smaller set of sensors increases the estimation success ratio by at least as much as adding the new sensor to a superset of sensors. Moreover,

this algorithm implements the greedy heuristic with sub-optimality guarantees as described above. The pseudocode for the greedy estimation success algorithm is shown in Algorithm 2.

Algorithm 3 Greedy Sensor Placement Algorithm

1: **procedure** GREEDYSELECTION(n, k,sensors)
2: Initialize the maximum success ratio to 0.0
3: Attach the sensors in the sensor placement strategy to the circuit configuration
4: Loop through each of the unselected available sensors and attach it to the circuit
5: Load in the database of inverse mapping from sensor measurements to compatible states of the circuit for the modified configuration
6: Calculate the success ratio, if it is greater than the current maximum success ratio, set it as the maximum success ratio
7: Go to the next sensor and repeat from step 4
8: Add the chosen sensor that corresponds to the maximum success ratio to the sensor placement strategy
9: Go back to step 2 and repeat from there until k sensors are selected or the state estimation success threshold is satisfied
10: **return** sensor placement strategy
11: **end procedure**

3.4 Experimental Results

3.4.1 Circuit Topology

The circuit in Figure 2.1 consists of basic high-voltage AC components: two generators G1-G2 and two AC buses B1-B2, DC components: two rectifier units R1-R2 and two DC buses B3-B4, and contactors C1-C8. Its corresponding adjacency matrix is

$$A = \begin{pmatrix} 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 1&0&0&1&1&0&0&0 \\ 0&1&1&0&0&1&0&0 \\ 0&0&1&0&0&0&1&0 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&1&1&0 \end{pmatrix}$$

Using matrix A, a network graph structure is created for the circuit as shown in Figure 3.1. Each component corresponds to a node in the network and each connection between any two components corresponds to an edge in the graph.

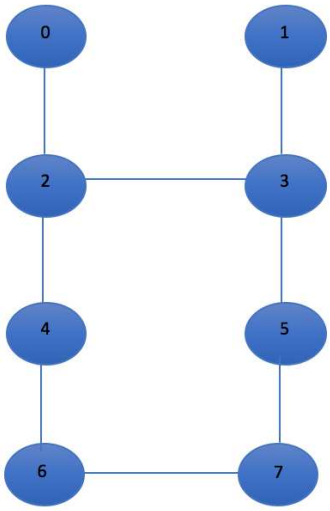

Figure 3.1: A network graph diagram derived from a circuit topology of Figure 2.1 with nodes 0-7 representing each of the circuit components, i.e. node 0 represents generator G1.

3.4.2 Variables

In this model, generators and rectifier units can be in three states: offline (0), healthy (1) and unhealthy (2). Contactors can be in two states: open (0) and closed (1). Buses can be in two states: healthy (1) and unhealthy (0). Component variables and status variables are distinguished by upper and lower cases, e.g., the first generator is represented by G1, while its status is represented by g_1.

The first step of the implementation is to create the databases storing the inverse mappings from sensor measurements to compatible states of the circuit for

each possible sensor placement configuration. For example, if a sensor were placed at $S0$, then the database reads a 0 for a fault at G1. For the given circuit, it is possible to place up to 8 voltage sensors, $S0 - S7$.

Using the databases obtained for each sensor placement configuration, we can execute the sensor placement algorithms which calculate the estimation success ratio. For this calculation, we use two versions of the state estimation algorithm for comparison. In one version, the algorithm uses the full greedy strategy in which all possible configurations of the circuit components are simulated systematically. In the other version, the algorithm uses the reduced greedy strategy in which the configuration for the controlled components is fixed. The only difference between these two versions is the number of configurations simulated to calculate the estimation success ratio.

3.4.3 Results

In this simulation, the greedy strategy is run with a horizon length of $k = 6$, the number of actions for the state estimation process. The status of each sensor is set to healthy. The combinatorial and greedy algorithms are run in Python 2.7 on a 2.3 GHz Intel Core i7 64 bits CPU. The results are shown in Tables 3.1-3.4 for various numbers of sensors required and a given configuration of controllable and uncontrollable components. In the given configuration, components (G1;G2;R1;R2;C2;C8) are set as uncontrollable and (C1;C3;C4;C5;C6;C7) as controllable. The following results are obtained.

Table 3.1: Sensor placement strategy obtained by the combinatorial algorithm with the full version of the adaptive greedy strategy.

No of Sensors	Sensors Selected	Success Ratio	Execution Time (s)
1	S7	0.15	1660.76
2	S6, S7	0.39	5576.294
3	S5, S6, S7	0.76	11627.31
4	S4, S5, S6, S7	0.80	14651.73
5	S3, S4, S5, S6, S7	0.80	11905.79
6	S2, S3, S4, S5, S6, S7	0.80	5969.48

Table 3.2: Sensor placement strategy obtained by the combinatorial method with the reduced version of the adaptive greedy strategy.

No of Sensors	Sensors Selected	Success Ratio	Execution Time (s)
1	S7	0.15	28.23
2	S6, S7	0.39	90.86
3	S5, S6, S7	0.76	187.83
4	S4, S5, S6, S7	0.80	247.02
5	S3, S4, S5, S6, S7	0.80	202.99
6	S2, S3, S4, S5, S6, S7	0.80	101.53

Table 3.3: Sensor placement strategy obtained by the greedy method with the full version of the adaptive greedy strategy.

No of Sensors	Sensors Selected	Success Ratio	Execution Time (s)
1	S5	0.15	1656.72
2	S5, S1	0.38	2904.79
3	S5, S1, S7	0.76	2904.79
4	S5, S1, S7, S4	0.80	4800.33
5	S5, S1, S7, S4, S6	0.80	5424.95
6	S5, S1, S7, S4, S6, S3	0.80	5722.11

Table 3.4: Sensor placement strategy obtained by the greedy method with the reduced version of the adaptive greedy strategy.

No of Sensors	Sensors Selected	Success Ratio	Execution Time (s)
1	S5	0.15	27.55
2	S5, S1	0.38	48.16
3	S5, S1, S7	0.76	67.46
4	S5, S1, S7, S4	0.80	74.96
5	S5, S1, S7, S4, S6	0.80	85.59
6	S5, S1, S7, S4, S6, S3	0.80	94.63

3.5 Chosen Approach

First, it is important to note that running the sensor placement algorithms with the reduced version of the adaptive greedy strategy for state estimation produces the same estimation ratios as those obtained by running the sensor placement algorithms with the full version of the adaptive greedy. However, utilizing the reduced version of the adaptive greedy strategy leads to a much lower execution time. In regards to the performance of the sensor placement algorithms, we can observe that even though both sensor placement algorithms obtain the same success ratio, the greedy algorithm performs at least twice as fast as the combinatorial algorithm. In additon, the greedy algorithm is well-suited for addressing the problem of selecting a required number of sensor locations as well as the problem of finding the minimum number of sensor locations that would satisfy an estimation success ratio. Meanwhile, the combinatorial algorithm is better suited for selecting the required number of sensors. Overall, we can conclude that the greedy algorithm is better.

For the circuit in Figure 2.1, both sensor placement algorithms with both versions of the state estimation process obtained a maximum estimation success ratio of 0.80. This result is the same as the one obtained after placing all available sensors and running the state estimation by brute force method (i.e. where all possible configurations and actions are tested). Thus, for the given circuit set up, placing all available sensors will only get the state estimation algorithm to uniquely identify the state in 80 percent of all possible states.

To improve this percentage, we can consider changing the topology of the cir-

cuit. For instance, we can allow some uncontrollable contactors become controllable. When contactor C2 is set to be controllable and sensors $S5$ and $S6$ are placed in the circuit, an estimation success ratio of 1 is obtained. The effect of allowing more contactors to be controllable can be observed in Table 3.5, which shows the estimation success ratio obtained for an increasing number of controllable contactors and the same number of sensors (N = 4). As more contactors are set to be controllable, the dynamic estimation process has access to a wider range of possible actions to reconfigure the circuit and consequently can gather more information about the state. Hence, for a given number of sensors, the state estimation performs better for a configuration that has more controllable contactors.

Table 3.5: Sensor placement strategy obtained by the greedy method for various controllable and uncontrollable contactor configurations.

Controllable Contactors	Uncontrollable Contactors	Sensors Selected	Success Ratio	Execution Time (s)
C1, C2, C3, C4	C5, C6, C7, C8	S7, S6, S5, S4	0.1075	1209.82
C1, C2, C3, C4, C5	C6, C7, C8	S7, S6, S5, S4	0.225	224.85
C1, C2, C3, C4, C5, C7	C6, C8	S7, S6, S5, S4	0.80	74.71
C1, C2, C3, C4, C5, C6, C7	C8	S7, S6, S5, S4	1.0	25.62

We now present more results obtained from running the greedy algorithm for two different contactor configurations. In the first contactor configuration, let components (G1;G2;R1;R2;C7;C8) be set as uncontrollable and (C1;C2;C3;C4;C5;C6) as controllable. It can be noted that the furthest contactors from the generators are

chosen to be uncontrollable for this configuration. The results from this configuration are shown in Table 3.6. In the second contactor configuration, let contactors (C3;C4;C5;C6;C7;C8) be set as controllable. In this case the closest contactors to the generators are chosen to be uncontrollable. The results from this configuration are shown in Table 3.7. From these results, we can observe that choosing contactors closer to the generators to be uncontrollable achieves a higher estimation ratio.

Table 3.6: Sensor placement strategy obtained by the greedy method with the reduced version of the adaptive greedy strategy.

No of Sensors	Sensors Selected	Success Ratio	Execution Time (s)
1	S7	0.20	23.16
2	S7, S5	0.25	44.3
3	S7, S6, S5	0.25	69.73
4	S7, S6, S5, S4	0.25	82.48

Table 3.7: Sensor placement strategy obtained by the greedy method with the reduced version of the adaptive greedy strategy.

No of Sensors	Sensors Selected	Success Ratio	Execution Time (s)
1	S7	0.07	28.76
2	S7, S1	0.30	50.91
3	S7, S1, S0	0.51	74.17
4	S7, S1, S0, S4	0.53	91.07

3.6 Reliability Results

With the greedy algorithm as the chosen approach, we proceed to test it further for a scenario in which selected sensors become unhealthy. In this scenario, the greedy algorithm can be run again with new databases. These databases are obtained by simulating erroneous sensor readings corresponding to unhealthy sensors. For example, if an unhealthy sensor were placed at $S0$, then the database reads a 1 for a fault at G1. In this simulation, contactors (C1;C2;C3;C5;C6;C7;C8) are set as controllable and contactor C4 is uncontrollable. Given an estimation success ratio of 0.6, the greedy algorithm can output a new sensor placement strategy to satisfy this ratio once sensors become unhealthy. We test this for a given sensor placement in which sensors S7 and S4 are initially selected. The results obtained are shown in Table 3.8 below.

When sensor $S7$ becomes unhealthy, the new sensor placement strategy adds sensor $S5$ in order to achieve a ratio of 0.6. If instead sensor $S4$ becomes unhealthy, the new sensor placement strategy adds sensors $S6$ and $S3$ in order to obtain the same ratio. If both sensors $S4$ and $S7$ become unhealthy, then the sensor placement strategy adds $S3$ and $S5$. From these results, we can conclude that some sensor locations are more informative than others, for instance, when sensor $S4$ became unhealthy, two more sensors were needed to reach the desired ratio as opposed to just one more sensor when $S7$ became unhealthy.

Table 3.8: Sensor Placement Strategy obtained by the greedy method with unhealthy sensors.

Sensors Initially Selected	Unhealthy sensors	Sensors Newly Selected	Threshold Obtained
S7, S4	S7	S5	0.6
S7, S4	S4	S6, S3	0.6
S7, S4	S7, S4	S5, S3	0.6

Chapter 4: System Methodology

4.1 Overview

The system methodology proposed combines fault detection and fault restoration techniques previously discussed. These techniques are based on formal methods and mathematical language which allow for automatically synthesizing a correct-by-construction control protocol that reroutes power according to system requirements. The methodology functions in two modes: off-line mode and on-line (real-time) mode. Figure 4.1 depicts the methodology workflow in which the first step occurs in off-line mode and the second step occurs in on-line mode. These modes are explained in more detail as follows.

4.1.1 Off-line mode

In this mode, the methodology performs the sensor placement algorithm to obtain the sensor placement strategy by choosing the sensor locations that will maximize the percentage of state configurations for which the state estimation algorithm is able to produce a unique state estimate. This percentage is obtained by running the state estimation method for all possible state configurations given a fixed sensor

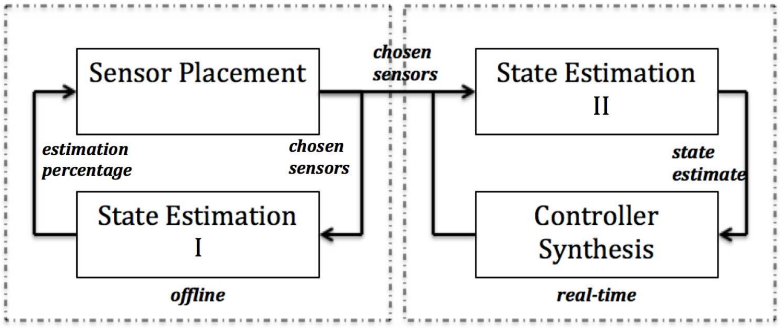

Figure 4.1: Design of closed loop approach to automate a fault-tolerant system. Sensor placement strategy is passed from off-line mode to on-line mode in which fault detection and fault restoration are performed.

placement strategy, which is initialized with one sensor. Each time a new sensor is added to the strategy, this percentage is recalculated using new databases required for state estimation. These databases contain inverse mappings from sensor measurements to compatible states of the system and must be rebuilt to account for new sensor measurements. The process of generating these databases can be too computationally demanding, therefore it is performed in the off-line mode. Once the databases are loaded in, the state estimation is run systematically for all state configurations and a percentage is generated. Using the percentage results, the sensor placement algorithm is computed iteratively until a given number of sensors are selected or the desired state estimation percentage is reached with the minimum number of sensors. Then, the chosen sensors are connected to the circuit in order to proceed to the on-line mode of the methodology.

4.1.2 On-line mode

While in this mode, the methodology implements the state estimation method initiated with the inverse mappings database computed for the sensor placement strategy obtained in the off-line mode. The result from the state estimation method is converted to a fault detection result, i.e. the state of the system lists out the states of each component which are then mapped to statuses: healthy or unhealthy. The state estimate is then passed to the controller synthesis method. This method translates system requirements written in English to Linear Temporal Logic specification language. Then, it generates a control protocol that outputs a sequence of actions (i.e. opening and closing switches) that will satisfy the given specifications and reconfigure the system back to safe operation, if needed. After the control protocol is executed, the methodology can monitor for faults again by running the state estimation method. In the case of sensor failures, the methodology can go from this mode back to the off-line mode to compute a different sensor placement strategy.

4.2 Case Study

In order to implement the methodology, we examine the simple circuit topology in Figure 2.1. First, the sensor placement strategy obtained off-line in Section 2 is applied to the circuit. This is, sensors $S4-S7$, which achieved an estimation success ratio of 1, are attached to the circuit. Next, the on-line step of the methodology is performed. In this step, state estimation algorithm is run for the circuit with a fixed fault configuration. The resulting state estimate is passed to the control

synthesis method. In order to test the control synthesis method, first we define the system variables, assumptions on admissible environments and the desired system specifications for the circuit.

4.2.1 Variables

Variables used in this formulation are classified as environment, controlled, or dependent.

Environment Variables: Let AC generators G1 and G2 as well as rectifier units R1 and R2 be defined as environment variables. The health status of these components (g_1;g_2;r_1;r_2) can each take values of healthy (1) and unhealthy (0).

Controlled Variables: Let all contactors C1-C8 be the controlled variables. Each of the contactors (c_1-c_8) can take value of open (0) and closed (1).

Dependent Variables: The power statuses (b_1, b_2, b_3, b_4) corresponding to the AC and DC buses can be either powered (1) or unpowered(0) depending on the status of neighboring contactors and generators.

The overall goal of the controller synthesis method is to reconfigure the controlled variables so that power will be delivered to buses and guarantee the system specifications described in the next section.

4.2.2 Formal Specifications

Environment Assumption: At least one generator and rectifier unit is always healthy. Also we assume that once a generator becomes unhealthy, it will

remain unhealthy. The specifications to accomplish this are:

$$\Box\{(g_1 = 1) \vee (g_2 = 1)\}, \tag{4.1}$$

$$\Box\{(r_1 = 1) \vee (r_2 = 1)\}, \tag{4.2}$$

$$\Box \bigwedge_{i=0}^{1} \{(g_i = 0) \longrightarrow (\bigcirc(g_i = 0))\}. \tag{4.3}$$

Unhealthy Sources: All neighboring contactors to generators and rectifier units that become unhealthy should be set to open. Set of neighboring contactors to generators are $\mathcal{N}$(G1) = C1 and $\mathcal{N}$(G2) = C2. Neighboring contactors to rectifier units are $\mathcal{N}$(R1) = {C4, C6} and $\mathcal{N}$(R2) = {C5, C7}. For example, if generator G1 becomes unhealthy, contactor C1 will be set to open. In this instance, the specification for G1 is:

$$\Box\{(g_1 = 0) \longrightarrow (c_1 = 0)\}. \tag{4.4}$$

No Paralleling of AC Sources: In Figure 2.1 there is one generator pair {G1,G2}. We avoid instances of paralleling ac sources. For example, a live path for the pair G1,G2 exists if contactors C1, C2 and C3 are all closed. The specification to disallow paralleling of AC sources is:

$$\Box\neg\{c_1 = 1 \wedge c_2 = 1 \wedge c_3 = 1\}. \tag{4.5}$$

Power Status of Buses: A bus will be powered if a live path exists between itself and a generator. A DC bus will be powered if it is connected to a healthy

rectifier unit and a live path exists between itself and a powered AC bus. Otherwise, the bus will be unpowered. For instance, the specifications for AC buses B1 and B2 and DC buses B3 and B4 to be powered are:

$$\square\{(c_1 = 1 \ \wedge \ g_1 = 1) \longrightarrow (b_1 = 1))\}, \tag{4.6}$$

$$\square\{(g_1 = 1 \ \wedge \ c_2 = 1 \ \wedge \ b_2 = 1 \ \wedge \ c_3 = 1) \longrightarrow (b_1 = 1)\}, \tag{4.7}$$

$$\square\{(b_1 = 1 \ \wedge \ c_4 = 1 \ \wedge \ r_1 = 1 \ \wedge \ c_6 = 1) \longrightarrow (b_3 = 1))\}, \tag{4.8}$$

$$\square\{(b_2 = 1 \ \wedge \ c_5 = 1 \ \wedge \ r_2 = 1 \ \wedge \ c_7 = 1 \wedge b_4 = 1 \ \wedge \ c_8 = 1) \longrightarrow (b_3 = 1)\}. \tag{4.9}$$

Essential Buses: In this problem, we consider buses B1 and B2 to be connected to safety-critical loads, and can be unpowered for no longer than two time steps. Each increment of the clock variable θ_{B_1} and θ_{B_2} represents one time step δ=1. If bus status B1 is unpowered, then at the next time step, clock θ_{B_1} increments by one:

$$\square\{(b_1 = 0) \longrightarrow (\bigcirc\theta_{B_1} = \theta_{B_1} + 1)\}. \tag{4.10}$$

If bus status B1 is powered, then at the next time step, reset clock θ_{B_1} to zero:

$$\square\{(b_1 = 1) \longrightarrow (\bigcirc\theta_{B_1} = 0)\}. \tag{4.11}$$

To ensure that the status of B1 is never unpowered for more than two steps, we have:

$$\square\{\theta_{B_1} \leq 2\}. \tag{4.12}$$

We also require that all DC buses must always remain powered by:

$$\square\{b_3 = 1 \ \wedge \ b_4 = 1\}. \tag{4.13}$$

We now turn to define the initial, transition, and goal values for the synthesis in GR(1) form:

$$\varphi_i^e = \bigwedge_{i=0}^{1}\{(g_i = 1), (r_i = 1)\} \tag{4.14}$$

$$\varphi_i^s = \bigwedge_{i=1, i\neq 3}^{8}\{(c_i = 0)\} \tag{4.15}$$

$$\varphi_t^e = \vee_{i=0}^{1}\{(\bigcirc\neg g_i), (\bigcirc\neg r_i)\} \tag{4.16}$$

$$\varphi_g^e = \square\diamond(True) \tag{4.17}$$

$$\varphi_g^s = \square\diamond(True) \tag{4.18}$$

4.3 Results

In our simulation, the synthesis method is initiated with the following fault configuration: G1 = 0, G2 = 1, R1 = 1, R2 = 0, C2 = 0, C7 = 1. After running the state estimation algorithm with the reduced version of the greedy strategy, an state estimate is obtained in 6.48s. Then, the controller synthesis is run with the

specifications and variables set as described above. Generating the specifications and obtaining the controller strategy took 0.58s. In this simulation, a centralized approach is used in which the controller has access to all system variables and the status of each of these variables is known.

The output of the controller is a Büchi automaton, i.e. a finite state machine with states and transitions [27]. In Figure 4.2, a portion of the automaton generated by the controller is shown. This figure shows three states, each with a set of values for the controlled and dependent variables. These values dictate the actions that the controller must perform until a change in the environment occurs. Such change is captured in the transitions between a pair of states. Depending on the environment action, the controller transitions from a set of controlled variables to another, which guarantees that the properties of the system remain satisfied in the presence of a change in the environment.

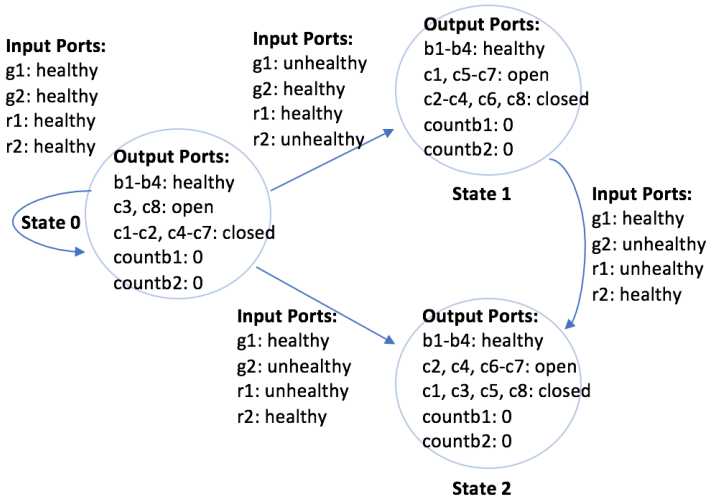

Figure 4.2: Automaton generated by the controller synthesis. Only three out of 26 states with a sample of their transitions are shown here.

In Figures 4.3 - 4.5, we show a simulation trace for the reactive controller. Assume that at time $t = 0$, all components of the circuit are in a healthy state as shown in Figure 4.3. At time $t = 1$, G1 and R2 become unhealthy. The controller then outputs a sequence of actions that reconfigure the circuit by opening contactors C2, C5 and C7 and closing contactor C3 as shown in Figure 4.4. Then at time $t = 2$, G2 and R1 become unhealthy and the controller reacts by opening contactors C2, C4 and C6 and closing contactor C3 as shown in Figure 4.5.

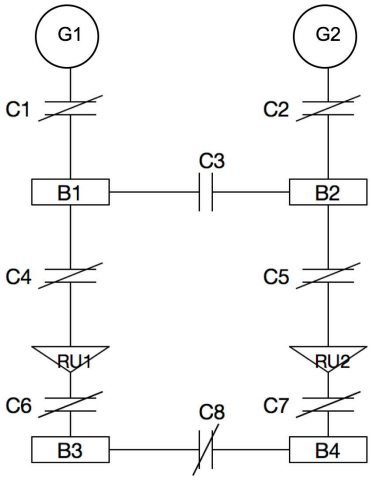

Figure 4.3: State at t = 0, G1 and R2 (RU2) are healthy. Initial configuration.

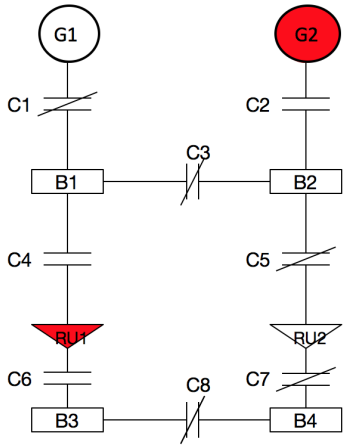

Figure 4.4: State at t = 1, G2 and R1 (RU1) become unhealthy (red). The controller opens C2, C4 and C6. It closes C3 to maintain B1 powered.

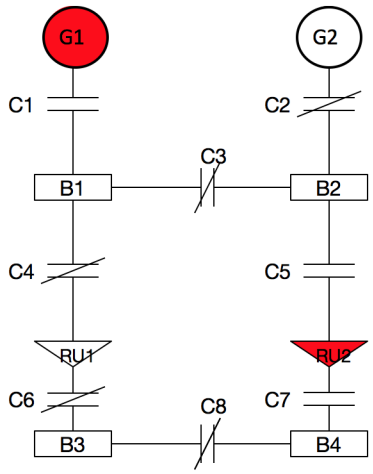

Figure 4.5: State at t = 2, G1 and R2 (RU2) become unhealthy (red) and the controller opens C1, C5 and C7. It closes C3 to maintain B2 powered.

4.4 Conclusion

In this chapter, the framework for translating text-based system specifications into a temporal logic specification language is presented. Using this framework, a controller is synthesized for the circuit in Figure 2.1. The output of the controller is an automaton with state transitions that capture changes in the environment, corresponding to a health status of each generator and rectifier unit. In each state of the automaton, the closing and opening of contactors allows generators and rectifier units to connect and disconnect from buses. Hence, the controller is guaranteed, by construction, to satisfy the system requirements even in the presence of failures.

Chapter 5: Results and Analysis

In this section, we present a case study on applying the proposed system methodology to perform fault detection and fault restoration for the large circuit topology shown in Figure 5.1. This circuit topology is adapted from a typical electric power system for a passenger aircraft [35]. The first step in the simulation is to determine the sensor placement strategy to obtain state estimates successfully for as many fault configurations as needed to satisfy a given threshold. Then, the state estimation algorithm is run for various fault configurations and the state estimates are passed to the controller synthesis step. From this step, we obtain all correct configurations of contactors that satisfy the system requirements provided.

The system requirements considered in this case study include safety and reliability specifications. Safety specifications serve to constrain the length of time AC buses can tolerate power shortages and ensure the non-paralleling of AC sources to avoid synchronization problems between multiple AC generators. Reliability specifications are used to limit the number of generators with failures at each step of the execution. The challenge in this implementation is the increased computational complexity given the large number of states and fault configurations that need to be accounted for. The circuit topology is described next.

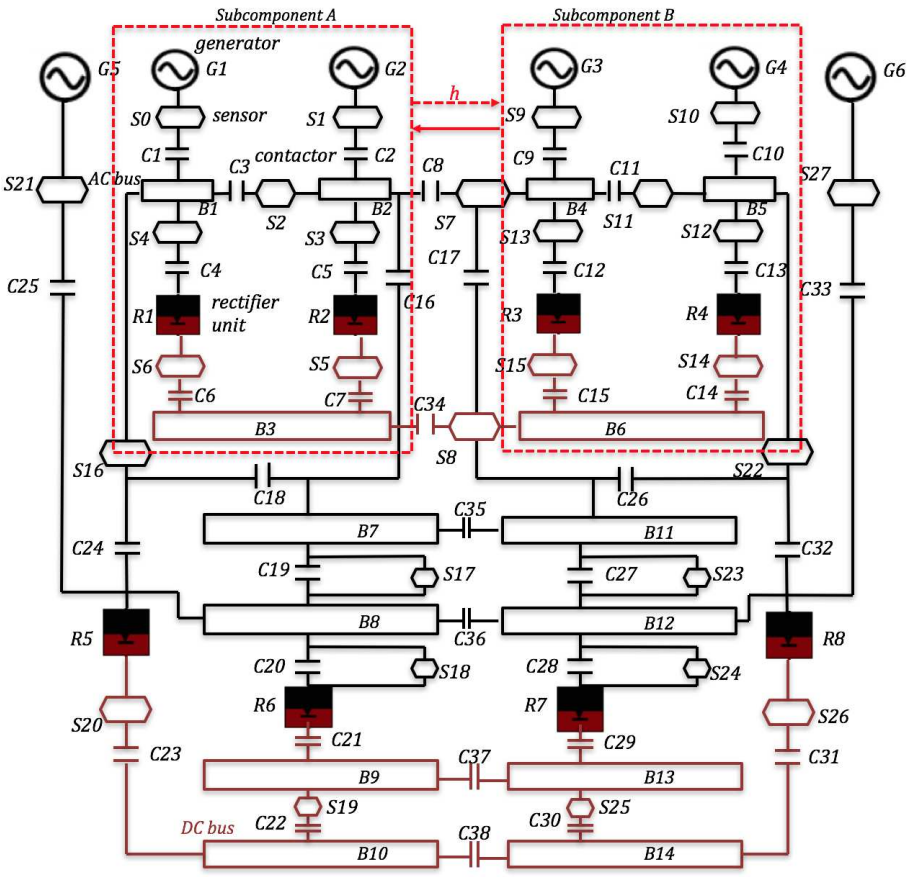

Figure 5.1: A circuit topology diagram. DC components are shown in red and AC components are shown in black. Buses are represented as rectangles. There are 6 generators connected to AC buses and 8 rectifier units connected to DC buses. A distributed design decomposition is shown using dashed rectangles. Components enclosed within the dashed rectangles are controlled by their own respective controllers and are given their own sensor placement strategies in the distributed approach. The dashed arrow represents information flow, in the form of a health status variable h, directed from subcomponent A to subcomponent B. The solid arrow represents the physical transfer of power from subcomponent B to subcomponent A.

5.1 Circuit Topology

There are six AC generators G1-G6 in the top section of the circuit topoloy. An AC bus is connected to each AC generator. There are 8 AC buses in total. The circuit also consists of rectifier units R1-R8, each connected to a DC bus. R1 and R2 are connected to the same DC bus B3. R3 and R4 are also connected to the same DC bus B6. All other rectifier units connect to one DC bus each. In total, there are 6 DC buses and 38 contactors. The circuit topology presented is symmetric along contactors C8 and C34-C38 in that the right section of the circuit has the same components and topology as the left side.

5.1.1 Subcomponents

The entire circuit is divided into four subcomponents when using the distributed approach for the implementation of the methodology. We take advantage of the symmetric properties of the circuit and select its subcomponents accordingly. Subcomponent A has the same components and topology as subcomponent B, while subcomponent C has the same components and topology as subcomponent D. The top section of the circuit is composed of subcomponents A and B. Subcomponent C is located in the bottom right part of the circuit and subcomponent D is located in the bottom left part of the circuit as shown in Figure 5.2.

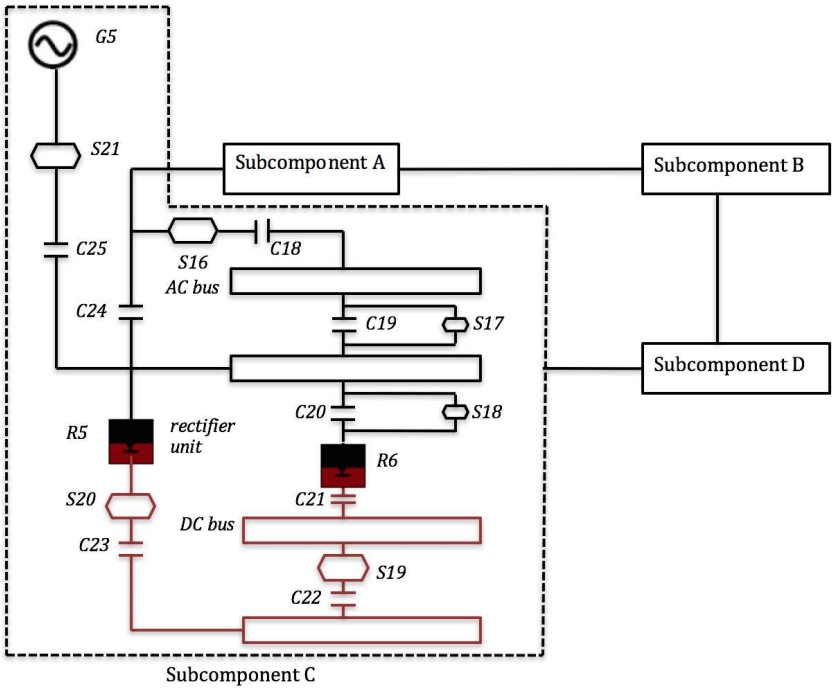

Figure 5.2: The circuit topology consists of subcomponents A, B, C and D. Each subcomponent is connected to two other components. Subcomponent C enclosed within the dashed rectangle is the mirror image of Subcomponent D, which is not explicitly shown.

5.1.2 Variables

Environment Variables: The environment variables consist of all generators G1-G6 and all rectifier units R1-R8. The health statuses for generators and rectifier units can take values of healthy (1) or unhealthy (0). At each step of the simulation, the environment variables can take a status of healthy or unhealthy subject to the assumption that one generator in the set {G1, G2, G5} and one generator in the set

{G3, G4, G6} remain healthy. A similar assumption is put into place for rectifier units, at least one rectifier unit in the set {R1, R2, R5, R6} and one rectifier unit in the set {R3, R4, R7, R8} remain healthy. These assumptions are imposed so that the right side of the circuit as well as the left side can be kept powered at each step independently of each other.

Controlled Variables: The controlled variables consist of all contactors C1-C38. Contactors can each take values of open (0) or closed (1). All contactors are assumed to be directly controlled without delays.

Dependent Variables: The power statuses ($b_1 - b_{14}$) corresponding to the AC and DC buses can be either powered (1) or unpowered(0). AC and DC buses are considered as dependent variables since their statuses can depend on the status of neighboring contactors and generators.

In the following sections, we present the psuedocode and implementation details to perform all three steps of the methodology.

5.1.3 Fault Detection

In order to perform state estimation, we first need to determine a sensor placement strategy. In the centralized approach, the sensor placement algorithm is run for the entire circuit. In the distributed approach, the greedy sensor placement algorithm is performed for each subcomponent following the procedure demonstrated in Section 3. In order to apply the sensor placement algorithm for each subcomponent, we simply isolate each subcomponent by opening the contactors between subcom-

ponents. For instance, when analyzing subcomponent A, contactors C8, C16, C24 and C34 are set to open, preventing power flow from subcomponents B or D. The topology for subcomponent A used by the sensor placement algorithm is shown in Figure 5.3. Similarly, when analyzing subcomponent B, its connections to subcomponents A and D can be set to open and the sensor placement strategy obtained for A can be applied to B. For subcomponent C, power can flow from A and D. Since R5 in subcomponent C is connected to subcomponent A in way that disconnecting it from A would make it a floating element, we include the input from A in the analysis of C. A can be modeled as a power source in the topology of C as shown in Figure 5.4. In this topology, contactors that connect C to D are set to open. Since C and D have the same topology, the same sensor placement strategy can be applied to both. Once the sensor placement strategy is determined for each subcomponent, some fault configurations are simulated and the fault detection algorithm is performed for each configuration. The pseudocode for this process is described in Algorithm 4.

Algorithm 4 Fault Identification Algorithm

Input: Sensor placement strategy
Output: State estimate of the system

```
1: Attach sensors indicated by the sensor placement strategy
2: Run estimation algorithm
3: while state estimate is not unique do
4:     if number of sensors available > 0 then
5:         Run sensor strategy algorithm to add an extra sensor
6:     end if
7: end while
8: return state estimate
```

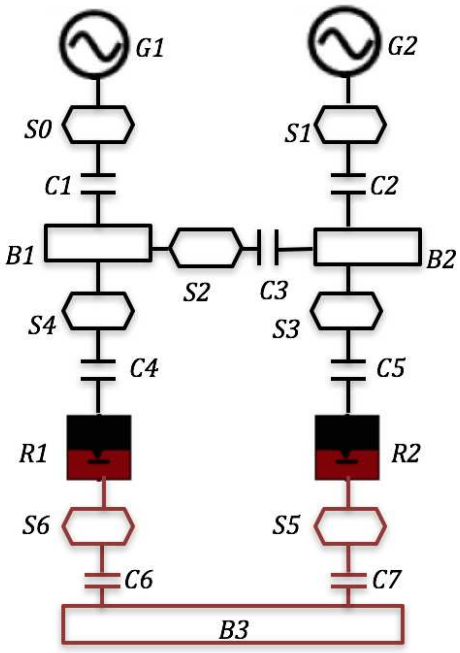

Figure 5.3: Topology of subcomponent A when disconnected from subcomponents B and C in order to apply the sensor placement and state estimation algorithms using a distributed approach.

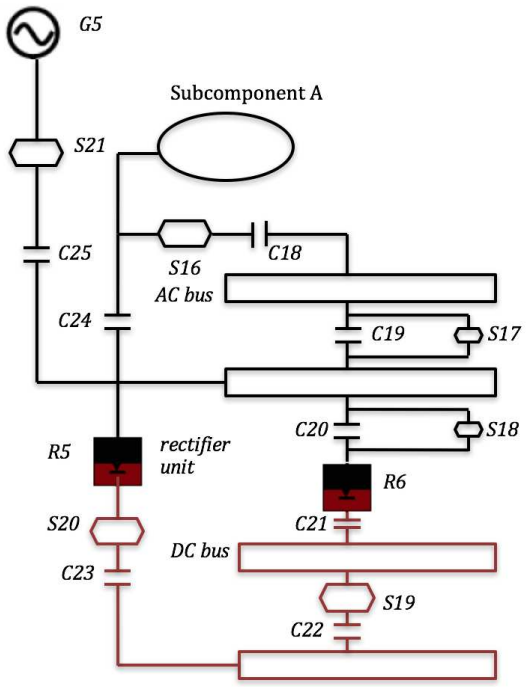

Figure 5.4: Topology of subcomponent D when disconnected from subcomponent C. Subcomponent A is modeled as a power source in order to apply the sensor placement and state estimation algorithms using a distributed approach.

5.1.4 Fault Restoration

For the circuit in this case study, the following is a pseudocode that generates the formal specifications as described in Section 4 to be transformed into GR(1) specifications. Then, the controller synthesis is performed with these specifications and we obtain a sequence of states or actions at each iteration such that the system

satisfies the specifications over the full execution horizon. Synthesis can be performed repeatedly, either periodically or when the expected environment variables change status.

Algorithm 5 Specifications Generation Algorithm

Input: State Estimate
Output: Formal Specifications

Initialize system variables
for each generator g_i, rectifier unit r_i, contactor c_i, bus b_i and path p_i **do**
    $env_{init} = \{g_i, r_i\}$
    $sys_{init} = \{c_i, !c_j, countb_i, b_i\}$
    g_i, r_i, b_i, c_i and c_j are set as boolean variables
    $countb_i$ is set as an integer variable
    c_i's are set to closed, c_j's are set to open to avoid parallel AC sourcing
    Set safety assumptions and requirements:
    $sys_{safe} = \{((b_i g_i p_i) \longrightarrow (bi))$, paths to healthy sources guarantee power
                $(!((0 = 1)||(b_i g_i p_i)) \longrightarrow (!bi))$, disconnect buses
                $((!b_i) \longrightarrow (\bigcirc countb_i = countb_i + 1))$, unhealthy bus
                $((b_i) \longrightarrow (\bigcirc countb_i = 0))$, reset count
                $(countb_i \leq 1)$, bus power guarantees
                $((!g_i) \longrightarrow (!c_i))$, disconnect unhealthy generators
                $((!r_i) \longrightarrow (!c_i))$, disconnect unhealthy rectifier units
                $(!((c_j \wedge c_k \wedge ...c_n)))$, avoid parallel sourcing
                $(b_i \leq 1)$, for essential buses only, keep them on at all times
                $(countb_i \leq 1)\}$
end for
return formal specifications

After obtaining the formal specifications for the circuit, we use Tulip to generate a discrete controller as described in the following pseudocode:

Algorithm 6 Controller Synthesis Algorithm

Input: Formal Specifications, *specs*
Output: Automaton

1: **procedure** SYNTHESIZE($env_{vars}, sys_{vars}, env_{init}, sys_{init}, env_{safe}, sys_{safe}$)
2: Set controller to use a Moore machine which reads current state but not next environment variable value
3: Set quantification of initial variable values
4: strategy = synth.synthesize('omega', *specs*)
5: Check realizability of output
6: **return** strategy
7: **end procedure**

5.2 Centralized Approach Results

As described in Section 4, the circuit used in this case study can be represented as a system with environment variables E and system variables P, which include controlled and dependent variables. In the centralized approach, it is assumed that all system variables are known and can be accessed. In the simulation, the three steps of the proposed methodology are run on a 2.6 GHz Intel Core i7 processor with 16 GB memory. First, the sensor placement algorithm is run with sensors $S0-S27$ as the available sensor locations. The algorithm run for more than 4 hours without completion. This was due to the computational complexity of building the databases for the entire circuit as required by the sensor placement and state estimation algorithms. This challenge is overcome by opting for a distributed sensor placement strategy which will be discussed in the next section.

The next step was to run the controller synthesis algorithm. Given environment assumptions φ_e as in (4.1)-(4.3) as well as system assumptions φ_s as the

conjunction of all specifications from (4.4)-(4.18) in Section 4, the synthesis problem is to find a control protocol such that $\varphi = \varphi_e \rightarrow \varphi_s$ holds. The controller synthesis algorithm was run for the entire circuit taking more than 10 hours after which it was stopped without completion. Similar to the choice made for the sensor placement strategy, a distributed controller synthesis strategy is considered to reduce the computational complexity of the simulation.

Since simulation results were not attainable within a reasonable time, here we present an example of one possible trace of the simulation: at time = 0, the statuses of each generator G1-G6 and each rectifier unit R1-R8 are set to healthy. Contactor statuses $c1, c3, c4, c6, c8, c11, c18, c19 - c24$ and $c34 - c38$ are set to closed. All other contactors are set to open to avoid paralleling of AC sources. Setting the statuses like this ensures that all AC and DC buses are initially powered and the power source being used is $G1$. At time t = 1, the statuses for generator G1 and rectifier unit R1 become unhealthy. This would require the controller to switch contactor statuses $c1$, $c4$ and $c6$ to open while statuses $c2$, $c5$ and $c7$ are switched to closed in order to keep all AC and DC buses powered.

5.3 Distributed Approach Results

5.3.1 Sensor Placement Results

In order to apply the methodology using a distributed approach, the entire circuit is divided into four subcomponents. Then, the methodology is applied to each subcomponent. For the sensor placement step, a state estimation threshold

of 0.8 and a horizon length k of 5 actions are provided. For subcomponent A as shown in Figure 5.3, contactors (C1;C2;C3;C4;C5;C7;C8) are set as controllable while contactor C6 is set as uncontrollable. The sensors chosen by the greedy sensor placement algorithm run for A were S5 and S6. The algorithm took 57.49 seconds to build and load the databases and 7.55 seconds to output the chosen sensor locations. This sensor placement strategy resulted in an estimation ratio of 1.0 satisfying the required threshold. In addition, the sensor placement strategy for B can be derived from the sensor placement strategy obtained for A, hence sensors S14 and S15 are chosen for B. For subcomponent C as shown in Figure 5.4, contactors (C18;C19;C20;C21;C22;C23;C24;C25) are set as controllable. The sensors chosen by the greedy sensor placement algorithm for C were S16, S17, S20 and S21. The algorithm took 549.15 seconds to build and load the databases and 38.57 seconds to output the sensors. This sensor placement strategy results in an estimation ratio of 0.8 which satisfies the required threshold. The sensor placement strategy for D can be derived from the sensor placement strategy obtained for C, hence sensors S24 and S26 are chosen for D.

5.3.2 Fault Detection Results

For the state estimation step, state estimates are obtained for different system configurations. The state estimation process is run first for subcomponents A and B, and then for subcomponents C and D since these rely on information from A and B. For subcomponents A and B, the state estimation is run for each of them

independently after setting contactors C8 and C34 to open. Once the state estimates are obtained for A and B, we can then proceed to use these state estimates to derive the input to the state estimation process for C and D. For instance, if the state estimate of A indicates that either bus B1 or B2 is powered, this will make the state of the source connected to subcomponent C be set to healthy, otherwise it will be set to unhealthy.

Similarly, if the state estimate of B indicates that either bus B4 or B5 in B is powered, this will make the state of the source connected to D be set to healthy, otherwise to unhealthy. The state estimates of A and B can be used for the state estimation process for C and D respectively. Table 5.1 and Table 5.2 show the results of the fault detection step for different configurations tested. The system state shown in the first columns of Table 5.1 and Table 5.2 represents the statuses of all circuit subcomponents in the following order: status of subcomponent A, $X_a = [g_1,g_2,b_1,b_2,r_1,r_2,b_3]$; status of subcomponent B, $X_b = [g_3,g_4,b_4,b_5,r_3,r_4,b_6]$; status of subcomponent C, $X_c = [g_5,source_A,b_7,b_8,b_9,b_{10},r_5,r_6]$; and status of subcomponent D, $X_d = [g_6,source_B,b_{11},b_{12},b_{13},b_{14},r_7,r_8]$. Thus, the system state can be represented as $X = [X_a,X_b,X_c,X_d]$.

Table 5.1: State estimates for subcomponents A and B obtained by the fault detection process.

States set for A and B	State estimates	Execution time (s)
X_a;X_b = 1,1,1,1,1,1,1; 1,1,1,1,1,1,0	X_a;X_b = 1,1,1,1,1,1,1; 1,1,1,1,1,1,0	14.71
X_a;X_b = 1,0,1,1,0,1,1; 0,1,1,1,1,0,1	X_a;X_b = 1,0,1,1,0,1,1; 0,1,1,1,1,0,1	7.7
X_a;X_b = 0,0,1,1,0,1,1; 0,0,1,1,1,0,1	X_a;X_b = 0,0,1,1,0,1,1; 0,0,1,1,1,0,1	0.183
X_a;X_b = 0,0,0,1,0,1,1; 0,0,1,1,0,0,1	X_a;X_b = 0,0,0,1,0,1,1; 0,0,1,1,0,0,1	0.185
X_a;X_b = 0,0,0,0,0,0,0; 0,0,0,0,1,0,1	X_a;X_b = 0,0,0,0,0,0,0; 0,0,0,0,1,0,1	0.183
X_a;X_b = 2,1,1,1,1,1,1; 1,2,1,1,2,1,1	X_a;X_b = 2,1,1,1,1,1,1; 1,2,1,1,2,1,1	2.19

Table 5.2: State estimates for subcomponents C and D obtained by the fault detection process.

States set for C and D	State estimates	Execution time (s)
X_c; X_d = 1,1,1,1,1,1,1,1; 0,1,1,1,1,1,1,1	X_c; X_d = 1,1,1,1,1,1,1,1; 0,1,1,1,1,1,1,1	11.03
X_c; X_d = 2,1,1,1,1,1,1,1; 1,2,1,1,1,1,1,1	X_c; X_d = 2,1,1,1,1,1,1,1; None	6.29
X_c; X_d = 1,1,1,1,1,1,0,1; 1,1,1,1,1,1,1,0	X_c; X_d = 1,1,1,1,1,1,0,1; 1,1,1,1,1,1,1,0	9.76
X_c; X_d = 1,1,1,1,1,1,2,2; 1,0,1,1,1,1,1,0	X_c; X_d = None; 1,0,1,1,1,1,1,0	4.12
X_c; X_d = 1,1,1,1,1,1,1,2; 1,1,1,1,1,1,2,1	X_c; X_d = 1,1,1,1,1,1,1,2; 1,1,1,1,1,1,2,1	11.73

The state estimation method uniquely identified 12 out of 12 simulated states for subcomponents A and B. On average the state estimation process took 3.9 seconds. For subcomponents C and D, the state estimation method only identified

8 out of 10 simulated states. On average the state estimation process for C and D took 8.58 seconds.

5.3.3 Controller Synthesis Results

After generating the LTL specifications as described in Section 4, the controller synthesis method is first run for subcomponents A and B independently. This required contactors C8 and C34 to be set to open. The controller synthesis took 0.48 seconds to solve for a control protocol with 25 states for each subcomponent. The control protocol output for A keeps its DC bus B3 powered as long as one generator and one rectifier is kept healthy at each execution step. The same occurs with B. In this setup, subcomponents A and B can function independently supplying power to components C and D. Figure 5.5 shows three states of a specific trace of the automaton resulting from the control protocol for A.

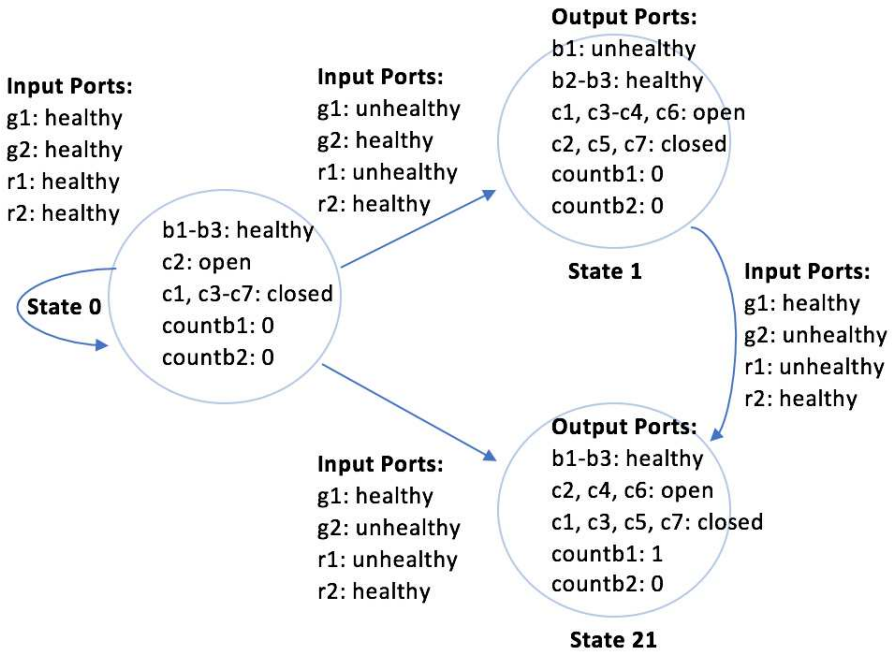

Figure 5.5: Simulation trace shows actions generated by the control protocol for subcomponent A for different sets of inputs. The same control protocol can be applied to subcomponent B.

We then proceed to run the controller synthesis method for subcomponents C and D. Since subcomponent C is connected to A, A is modeled as an environment variable s_a in the controller synthesis for C. The value of s_a will vary according to the states of the AC buses in A. If both AC buses in A are unhealthy, s_a will be set to unhealthy, otherwise to healthy. An additional specification considered for the controller synthesis for subcomponent C is that when power can be supplied by both G5 and subcomponent A, this is when g_5 is healthy and s_a is healthy, preference is given to the power flowing in from subcomponent A requiring g_5 to

be disconnected. A simulation trace of the automaton generated by the control protocol for C is shown in Figure 5.6. The synthesis took 0.43 seconds with a total of 17 states.

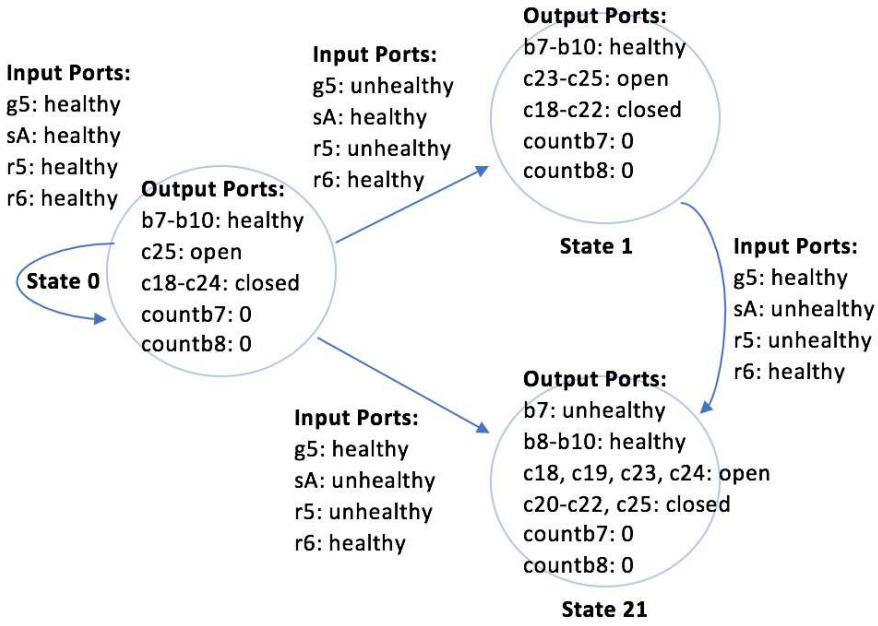

Figure 5.6: Simulation trace shows actions generated by the control protocol for subcomponent C for different sets of inputs. The input from subcomponent A is included as an environment variable. The same control protocol can be applied to subcomponent D with the input from subcomponent B modeled as an environment variable.

In the next simulation, the requirement of having at least one generator and one rectifier unit healthy at all times in both subcomponents A and B is relaxed. In this case, the controller synthesis is run with the requirement that only one subcom-

ponent has at least one generator and one rectifier unit healthy at each execution step. Therefore, the subcomponent with power at all times will supply power to the other subcomponent. This set up is known as the *master/slave* setup. The subcomponent with the requirement of having at least one generator and one rectifier unit healthy would behave as the *master*. Having a subcomponent acting as the *master* also prevents a deadlock situation due to power being able to flow between subcomponents in either direction. By opting for a *master/slave* architecture, the *master* subcomponent can control the power flow in an unidirectional manner and the *slave* subcomponent can only receive power from the *master*.

Let subcomponent B be the *master* and subcomponent A be the *slave*. By placing the assumption that B has at least one generator and one rectifier unit, we ensure that for any allowable sequence of environment actions, the controller is able to supply power to subcomponent A at any step by closing contactors $c8$ and $c34$ as needed. Health status information for G1 and G2 is sent to subcomponent B via a health status variable h. The variable is set to 0 if neither source is healthy, and 1 if either g1 or g2 is healthy. If health status $h = 0$, i.e., both g1 and g2 are unhealthy then, whenever AC bus B4 in B is powered, c8 will be set to closed. Similarly, we can have another health status variable $h2$ such that if r1 = 0 and r2 = 0, $h2$ will be set to 0 causing the controller to close contactor c34. It is required to have DC buses B3 and B6 powered at each step in the execution, with the assumption that at least one generator and one rectifier unit in the *master* subcomponet is healthy at all times. Running the controller synthesis in this setup took 1.87 seconds obtaining a control protocol with 53 states. Some states of a simulation trace of the automaton

generated by the resulting control protocol for B are shown in Figure 5.7.

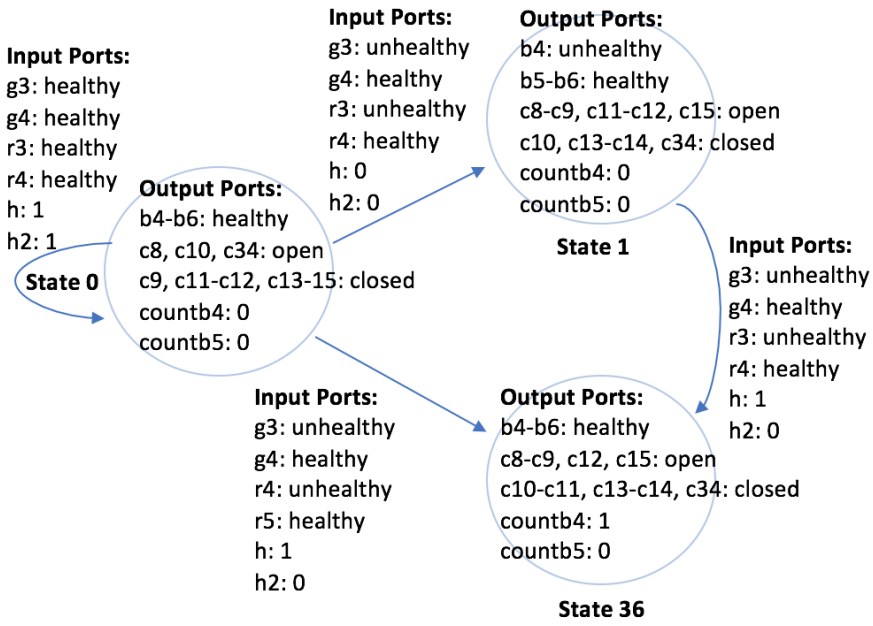

Figure 5.7: Simulation trace shows actions generated by the control protocol for subcomponent B in a master/slave configuration for different sets of inputs. Subcomponent B is the master and subcomponent A is the slave.

When running the controller synthesis for subcomponents C and D combined, the inputs from A and B are modeled as environment variables s_{a1}, s_{a2}, s_{b1} and s_{b2}. If AC bus B1 is healthy, then s_{a1} is set to healthy and is able to supply power to C via contactor C18. If AC bus B2 is healthy, then s_{a2} is set to healthy and is able to supply power to C via contactor C16. If AC bus B4 is healthy, then s_{b1} is set to healthy and is able to supply power to D via contactor C17. If AC bus B5 is

healthy, then s_{b2} is set to healthy and is able to supply power to D via contactor C26. These environment variables can then be mapped to one variable s_{ab}, which specifies whether power flows or not from either subcomponent A or B. If at least one of the environment variables s_{a1}, s_{a2}, s_{b1} and s_{b2} is healthy, then s_{ab} is set to healthy, otherwise to unhealthy. An additional specification considered here is that when power can be supplied by s_{ab}, G5 or G6, preference is given to s_{ab}. Running the controller synthesis method for both subcomponents C and D took 52.02 seconds to solve for a control protocol with 209 states. Some states of a simulation trace of the automaton generated by the resulting control protocol for C and D are shown in Figure 5.8.

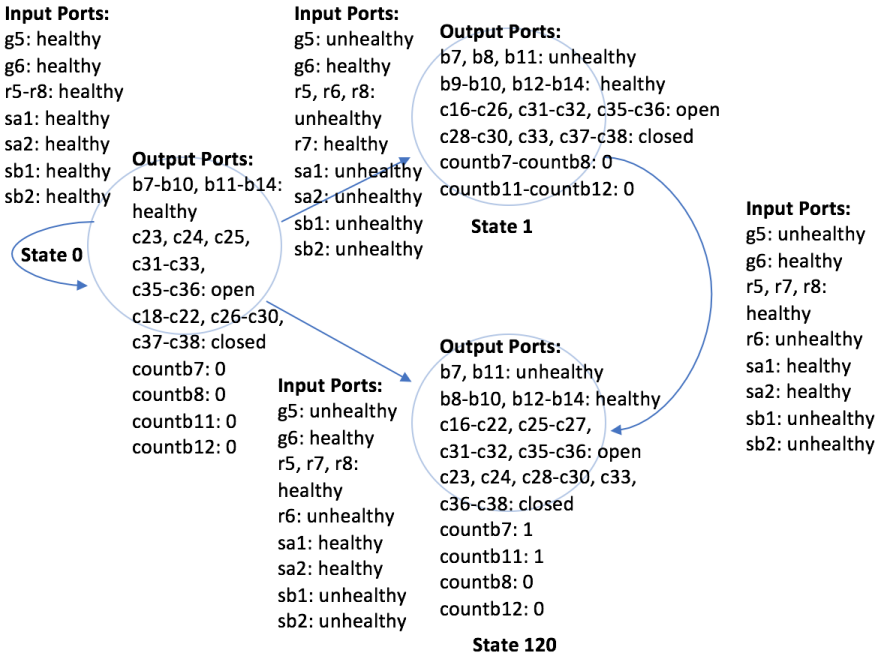

Figure 5.8: Simulation trace shows actions generated by the control protocol for subcomponents C and D combined for different sets of inputs. The input from subcomponents A and B are combined into a single environment variable.

5.4 Analysis

In this section, it is demonstrated how the system methodology proposed can perform fault detection and fault restoration for the large scale circuit in Figure 5.1. Using a centralized approach, the sensor placement, state estimation and controller synthesis steps of the methodology are run assuming access to all circuit components.

However, this approach proved to be too computationally demanding, particularly for the task of building the databases of inverse mapping from sensor measurements to compatible states of the circuit. A distributed approach was explored and tested in order to reduce this computational complexity. For the distributed approach, the circuit is divided into subcomponents and the sensor placement, state estimation and controller synthesis tasks are performed for each subcomponent separately. These results can then be combined into fault detection and fault restoration strategies that work for the entire circuit.

The subcomponents were chosen by taking advantage of the symmetric properties of the circuit topology. Therefore, for the circuit in Figure 5.1, four subcomponents were reduced to two pairs of subcomponents with the exact same topology, allowing for the sensor placement algorithm to be run only twice. Once the sensor locations were decided on, fault detection is run for each subcomponent and the state estimates are obtained within seconds. The estimation process is done in a specific order due to the nature of the power flow in the circuit. State estimates are obtained first for subcomponents A and B and then for subcomponents C and D.

For the controller synthesis step, applying a distributed approach required additional refinements to the topologies of the circuit subcomponents and the system specifications for the controller synthesis step. Subcomponents A and B were included and modeled as power sources in the topologies of subcomponents C and D. In addition to new elements in the subcomponent topologies, new variables were introduced for the controller synthesis step. For instance, for the master/slave configuration, a health status variable was included in the synthesis for subcomponent

B in order to account for its interaction with subcomponent A. Since B was set as the master, the control protocol for B had full control of the actions on the contactors connecting A and B, hence routing power from B to A when needed. This set up was preferred over a decentralized controller design in order to prevent deadlock situations and the need for timing specifications. This choice was made with the goal of reducing computational complexity and state space size.

From the results obtained, it is clear that not only it is feasible to run all the steps of the methodology using the distributed approach, it also takes much less computational time due to smaller state spaces in each subcomponent. One of the drawbacks of the distributed approach is having to account for refinements and added specifications that can handle the interaction between subcomponents. This also limits the scope of the actions the distributed controller can generate compared to a centralized controller given that it has to satisfy some extra assumptions. For instance, the assumption that a specific subcomponent must have at least one generator and rectifier unit healthy at all times places a restriction on how the system is expected to behave. In such cases in which an assumption is not satisfied, the controller would not be able to handle a system failure. Hence, there is a trade off between computational complexity and the ability of the controller to address all types of failures in the system.

Chapter 6: Future Work and Conclusions

6.1 Future Work

The methodology proposed can perform fault detection and fault restoration in a closed loop fashion in real time. There is, however, a heavy preprocessing task of building measurement mapping databases that is performed offline. The sensor placement algorithm relies on these databases and can run much faster if the databases are built beforehand. Another aspect of the sensor placement algorithm that can affect its speed is the number of contactors set as controllable. From the results obtained, it was demonstrated that if more contactors were set to be controllable, the estimation results would improve, but computational time would also increase. Setting contactors as controllable was done in adhoc manner as to optimize fault detection results and reduce computational time. Since there is a trade off between computational time and higher estimation success ratios, machine learning algorithms could be developed to learn how to choose these contactors for different circuit topologies based off the results obtained by the state estimation method.

For the application of the methodology, in particular the sensor placement and state estimation steps, it was assumed that sensors were healthy at each step

of the execution. Future work can relax this assumption and integrate information about the states of the sensors in the controller synthesis in order to account for sensors becoming unhealthy. For instance, extra set of specifications may be needed to isolate components that are connected to faulty sensors. Some tests were run for scenarios in which sensors become faulty and the sensor placement strategy was changed to remove unhealthy sensors and replace them with available healthy sensors. However, placing specifications that can react to sensor failures in case the sensor placement strategy cannot be changed might prove to be useful. Also, in the synthesis step, knowledge of the system states is provided but may become unattainable due to sensor failures. Again, specifications could address these failure cases by isolating the components with unknown states.

Another area of improvement is in the process of distributing a given topology among subsystems and generating interface specifications. In this thesis, this distribution and the formulation of interface specifications are done manually. A possible solution is automatizing the generation of interface specifications as well as the distribution process for different topologies using graph theory. For instance, the circuit can be represented by nodes and branches and decomposed into smaller circuits using k-factorization partition techniques. In addition to modifying the circuit toplogy, taking into account the continous dynamics of the circuit can also be an area of future work. For this electric power system design problem, continuous dynamics were abstracted and discretized. Other areas of future work include testing the system methodology on hardware and extending the formal language used for control synthesis algorithms to support timed logic, continuous-time specifica-

tions and more complex system behaviors that are able to model network transients and delays. An interface similar to PSpice can also be created to draw the circuit topology and facilitate the abstraction from any given topology to a graph data structure.

6.2 Conclusions

This thesis presents a system methodology that addresses fault detection and fault restoration holistically for an electric power system by integrating three algorithms: sensor placement, state estimation and reactive controller synthesis. The methodology can function in real-time going from fault detection to fault restoration periodically or as specified by the user. To be able to run fault detection in real-time, the sensor placement step must occur off-line. The developed sensor placement algorithm is designed to maximize the performance of the state estimation algorithm. The results obtained by the sensor placement algorithm show that as more sensors are added to the circuit, more state configurations can be estimated. However, the improvement in uniquely identifying the state diminishes as more sensors are added to the circuit demonstrating the submodularity nature of the sensor placement problem. The goal of the sensor placement algorithm is then to minimize the number of chosen sensors as to satisfy a given state estimation performance threshold and return if and when the state estimation stops improving. The number of sensors can also be varied according to the weight, space and power consumption requirements for the system. For the fault detection step, we perform discrete state estimation

using active control of switches within the electric power system in a distributed control architecture.

Given a sensor strategy and a state estimate, we are then able to automatically synthesize a control protocol according to a set of system requirements. The controller reacts to changes in the environment and is guaranteed, by construction, to satisfy the desired behavior as long as the environment is admissible. The algorithms are able to find a controller, if one exists, that will satisfy the specifications given in any modeled environment. The correct-by-construction guarantees with respect to the specification and abstraction of the synthesized controller eliminate human-error in implementation and are more robust than a manual composition of controllers. Overall, we demonstrate how the proposed methodology, combining different tools and techniques, can be applied to a large circuit. It is shown that the distributed approach makes running the methodology steps in real-time feasible. Furthermore, results obtained for each subsystem can easily be integrated and combined in order to solve the problem for the entire circuit.

Bibliography

[1] J. Rosero, J. Ortega, E. Aldabas, and L. Romeral. Moving towards a more electric aircraft. In *Aerospace and Electronic Systems Magazine, IEEE*, vol. 22, no. 3, pages 39, 2007.

[2] P. Bunnoon. Fault Detection Approaches to Power System: State-of-the-Art Article Reviews for Searching a New Approach in the Future. In *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 3, no. 4, pages 553-560, 2013.

[3] Y. Xiaohua, Z. Yadong, C. Z. Faqi, and X. Zhongmei. Wavelet neural network based fault detection method in power system. In *Second International Conference on Mechanic Automation and Control Engineering (MACE)*, pages 1864-1867, 2011.

[4] Y. Huang. Power transformer fault detection using intelligent neural networks. In *IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 3, pages 1761-1764, 2002.

[5] D. Srinivasan, R. L. Cheu, Y. P. Poh, and A. K. Ng. Automated fault detection in power distribution networks using a hybrid fuzzy genetic algorithm approach. In *Engineering Applications of Artificial Intelligence*, vol. 13, no. 4, pages 407-418, 2000.

[6] E. M. Davidson, S. D. McArthur, and J. R. McDonald. A toolset for applying model-based reasoning techniques to diagnostics for power systems protection. In *IEEE Transactions on Power Systems*, vol. 18, no. 2, pages 680687, 2003.

[7] K. Keller, K. Swearingen, J. Sheahan, M. Bailey, J. Dunsdon, K. W. Przytula, and B. Jordan. Aircraft electrical power systems prognostics and health management. In *2006 IEEE Aerospace Conference*, Big Sky, MT, page 12, 2006.

[8] C. Xueting, N. Ke, D. Weiwen, L. Mengzan, P. Jun, and S. Zeyuan. Design and realization for fault restoration system based on the genetic algorithm. In *2016 IEEE International Conference on Information and Automation (ICIA)*, Ningbo, pages 905-909, 2016.

[9] X. Cheng, H. Zheng, W. Duan, X. Hao, and W. Duan. Fault restoration based on the path analysis for distribution grid. In *2015 IEEE International Conference on Information and Automation*, pages 552-556, 2015.

[10] F. Pasqualetti, S. Zampieri, and F. Bullo. Controllability metrics, limitations and algorithms for complex networks. In *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pages 40-52, 2014.

[11] J. Qi, K. Sun, and W. Kang. Optimal PMU placement for power system dynamic state estimation by using empirical observability Gramian. In *IEEE Transactions on Power Systems*, vol. 30, no. 4, pages 2041-2054, 2015.

[12] M. Shamaiah, S. Banerjee, and H. Vikalo. Greedy sensor selection: Leveraging submodularity. In *49th IEEE Conference on Decision and Control (CDC)*, pages 2572-2577, Atlanta, GA, 2010.

[13] T. H. Summers, F. L. Cortesi, and J. Lygeros. On Submodularity and Controllability in Complex Dynamical Networks. In *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pages 91-101, March 2016.

[14] V. Tzoumas, M. A. Rahimian, G. J. Pappas, and A. Jadbabaie. Minimal Actuator Placement With Bounds on Control Effort. In *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pages 67-78, March 2016.

[15] V. Gupta, K. Kapoor, and R. Devi. Wireless Sensor node selection strategies for effective surveillance. In *2015 IEEE International Advance Computing Conference (IACC)*, pages 924-929, 2015.

[16] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *2006 5th International Conference on Information Processing in Sensor Networks*, pages 2-10, 2006.

[17] A. W. Mahoney, T. L. Bruns, P. J. Swaney, and R. J. Webster. On the inseparable nature of sensor selection, sensor placement, and state estimation for continuum robots or where to put your sensors and how to use them? In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4472-4478, 2016.

[18] A. Abur and A. G. Exposito. *Power System State Estimation: Theory and Implementation*. New York, NY: Marcel Dekker, 2004.

[19] R. Isermann. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Berlin: Springer, pages 10-17, 2006.

[20] N. Meskin and K. Khorasani. *Fault Detection and Isolation: Multi-Vehicle Unmanned Systems*. New York, NY: Springer New York, pages 1-13, 2011.

[21] Q. Maillet, H. Xu, N. Ozay, and R. M. Murray. Dynamic state estimation in distributed aircraft electric control systems via adaptive submodularity. In *52nd IEEE Conference on Decision and Control*, pages 5497-5503, Firenze, 2013.

[22] H. Xu, U. Topcu, and R. M. Murray. Specification and Synthesis of Reactive Protocols for Aircraft Electric Power Distribution. In *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pages 193-203, June 2015.

[23] A. Pnueli. Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends. In *Current trends in Concurrency*, pages 510584, 1986.

[24] N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive (1) designs. In *Verification, Model Checking, and Abstract Interpretation*, pages 364380. Springer, 2006.

[25] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray. Tulip: a software toolbox for receding horizon temporal logic planning. In *Proceedings of the 14th international conference on Hybrid systems: computation and control (HSCC)*, pages 313-314, 2011.

[26] T. Laengle, T. C. Lueth, and U. Rembold. A distributed control architecture for autonomous robot systems. In *Series in Machine Perception and Artificial Intelligence*, pages 384402, 1995.

[27] K.Y. Rozier. Linear Temporal Logic Symbolic Model Checking. In *Computer Science Review*, vol. 5, no. 2, pages 163-203, May 2011.

[28] R. Alur, S. Moarref, and U. Topcu. Counter-Strategy Guided Refinement of GR(1) Temporal Logic Specifications. In *FMCAD*, pages 2633, 2013.

[29] M. G. J. W. Howse. More Electric Technologies for the 21st Century. In *Power Electronics Machines and Drives*, pages 16-18, April 2002.

[30] B. K. Bose. Global energy scenario and impact of power electronics in 21st century. In *IEEE Transanctions Industrial Electronics*, vol. 60, no. 7, pp. 2638-2651, July 2013.

[31] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis. Failure diagnosis using discrete-event models. In *IEEE Transactions on Control Systems Technology*, vol. 4, no. 2, pages 105124, 1996.

[32] J. A. Weimer. Electrical Power Technology for the More Electric Aircraft. In *Digital Avionics Systems Conference 1993 DASC AIAA/IEEE* pages 445-450, October 1993.

[33] Lakshmisowjanya M., Swetha A., and Pillay V.R. Fault Tolerant Scheduling - Dual Redundancy in an Automotive Cruise Control System. Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1. Advances in Intelligent Systems and Computing, vol. 337, Springer, Cham, 2015.

[34] Wei T., Mishra P., Wu K., Online Task-Scheduling for Fault-Tolerant Low-Energy Real-Time Systems. ICCAD'06, November 5-9, 2006, San Jose, CA.

[35] R. Michalko. Electrical starting, generation, conversion and distribution system architecture for a more electric vehicle, 10 2008.