EFFICIENT DETECTION ON STOCHASTIC FAULTS IN PLC BASED AUTOMATED ASSEMBLY SYSTEMS WITH NOVEL SENSOR DEPLOYMENT AND DIAGNOSER DESIGN

A Dissertation

by

ZHENHUA WU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Mechanical Engineering

Efficient Detection on Stochastic Faults in PLC Based Automated Assembly Systems

with Novel Sensor Deployment and Diagnoser Design

EFFICIENT DETECTION ON STOCHASTIC FAULTS IN PLC BASED AUTOMATED ASSEMBLY SYSTEMS WITH NOVEL SENSOR DEPLOYMENT AND DIAGNOSER DESIGN

A Dissertation

by

ZHENHUA WU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Mechanical Engineering

ABSTRACT

Efficient Detection on Stochastic Faults in PLC Based Automated Assembly Systems with Novel Sensor Deployment and Diagnoser Design.

(May 2012)

Zhenhua Wu, B.S., M.S., Hefei University of Technology;

M.S., The University of Texas-Pan American

Chair of Advisory Committee: Dr. Sheng-jen Hsieh

In this dissertation, we proposed solutions on novel sensor deployment and diagnoser design to efficiently detect stochastic faults in PLC based automated systems.

First, a fuzzy quantitative graph based sensor deployment was called upon to model cause-effect relationship between faults and sensors. Analytic hierarchy process (AHP) was used to aggregate the heterogeneous properties between sensors and faults into single edge values in fuzzy graph, thus quantitatively determining the fault detectability. An appropriate multiple objective model was set up to minimize fault unobservability and cost while achieving required detectability performance. Lexicographical mixed integer linear programming and greedy search were respectively used to optimize the model, thus assigning the sensors to faults.

Second, a diagnoser based on real time fuzzy Petri net (RTFPN) was proposed to detect faults in discrete manufacturing systems. It used the real time PN to model the manufacturing plant while using fuzzy PN to isolate the faults. It has the capability of handling uncertainties and including industry knowledge to diagnose faults. The proposed approach was implemented using Visual Basic, and tested as well as validated on a dual robot arm.

Finally, the proposed sensor deployment approach and diagnoser were comprehensively evaluated based on design of experiment techniques. Two-stage statistical analysis including analysis of variance (ANOVA) and least significance difference (LSD) were conducted to evaluate the diagnosis performance including positive detection rate, false alarm, accuracy and detect delay. It illustrated the proposed approaches have better performance on those evaluation metrics.

The major contributions of this research include the following aspects: (1) a novel fuzzy quantitative graph based sensor deployment approach handling sensor heterogeneity, and optimizing multiple objectives based on lexicographical integer linear programming and greedy algorithm, respectively. A case study on a five tank system showed that system detectability was improved from the approach of signed directed graph's 0.62 to the proposed approach's 0.70. The other case study on a dual robot arm also show improvement on system's detectability improved from the approach of signed directed graph's 0.61 to the proposed approach's 0.65. (2) A novel real time fuzzy Petri net diagnoser was used to remedy nonsynchronization and integrate useful but incomplete knowledge for diagnosis purpose. The third case study on a dual robot arm shows that the diagnoser can achieve a high detection accuracy of 93% and maximum detection delay of eight steps. (3) The comprehensive evaluation approach can be referenced by other diagnosis systems' design, optimization and evaluation.

DEDICATION

This dissertation is dedicated to my parents who always love and support me.

ACKNOWLEDGEMENTS

Firstly, I would like to praise my heavenly Father who always watches me closely with his endless love.

I would like to thank my committee chair, Dr. Sheng-jen Hsieh, and committee members, Drs. Luis San Andres, Reza Langari, Steve Suh, and Yu Ding, for their insightful guidance and strong support throughout my research course.

Sincere acknowledgement will be extended to Dr. Hsieh for his patience, financial support and valuable suggestions in my Ph.D. years. Dr. Hsieh has set up a dedicated role model for me in teaching and research. The experience accumulated with him will profoundly influence my career and life journey.

I have met so many nice and excellent professors who always help and inspire me. I would like to acknowledge Drs. Jianzhi Li, Andrew Chan, Fujie Zhou, Banglin Chen and Jim Ji for their help and advice in every aspect on the meaning of life and study.

I would like to extend my appreciation to Mr. Ramnath Sekar's for his diligent co-working and interesting discussions on this diagnosis project, Laura Reinisch and Necia Perdue for their help on many administrative issues. Thanks also go to my friends for making my time at Texas A&M University a great experience.

Last but not least, thanks to my mother and father for their love and encouragement during my study in the United States.

NOMENCLATURE

PLC	Programmable Logic Controller
DES	Discrete Event System
PN	Petri Net
DG	Directed Graph
SDG	Signed Directed Graph
GA	Genetic Algorithm
SA	Simulated Annealing
TS	Tabu Search
IP	Integer Programming
DP	Dynamic Programming
MINLP	Mixed Integer Nonlinear Programming
FMEA	Failure Mode Effect Analysis
AHP	Analytical Hierarchy Process
MILP	Mixed Integer Linear Programming
IPN	Interpreted Petri Net
CTPN	Colored Timed Petri Net
FPN	Fuzzy Petri Net
MOM	Mean of Maximum
COA	Center of Area
RTFPN	Real Time Fuzzy Petri Net

MM	Markov Model
SFC	Sequential Function Chart
DOE	Design of Experiment
ANOVA	Analysis of Variance
LSD	Least Significant Difference
POD	Probability of Detection
POFA	Probability of False Alarm

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Motive

Programmable logic controller (PLC) based automated assembly systems are widely used in manufacturing lines, semiconductor fabrication facilities etc. The success of PLC automated assembly systems critically depends on fault-free operations and low machine down time. The faults involved in assembly systems can be induced by underlying causes such as hardware/software failures, design errors, manufacturing defects, improper application of parts, or users programs not following the protocols etc. [1]. Fault diagnosis is the action to identify whether a system is deviating from the normal behavior, and determine the fault types, locations and potential root causes for the abnormal behaviors. Traditionally, fault diagnosis on assembly systems was performed by humans. It was time-consuming and mainly depended on the technicians' experience and skills. When systems get more complex, the causality mapping between fault symptoms and root causes becomes highly nonlinear.

Besides this, for stochastically faulty systems, the exact fault time and modes are not completely known due to the insufficient input/output that can be observed from the

This dissertation follows the style of International Journal of Advanced Manufacturing Technology.

system. These stochastic faults may not be observed directly but "hiding" under the observed signals. This challenges the technicians' expertise and hinders the efficiency and accuracy of human-based machine maintenance. Design of a computer-aided diagnosis system on stochastic faults is desired for the purpose of improving accuracy, enhancing time-effectiveness and lowering the diagnosis cost.

1.2 Research question

When designing a computer based diagnosis system with excellent accuracy and efficiency, researchers and engineers need to consider many factors. In literature [2], authors summarized the main obstacles for the designers of diagnosis systems. They are listed as below:

1. Design an architecture that can integrate technologies including sensor, signal processing, communication etc to resolve the diagnosis work;
2. Select the types, numbers and locations of sensors.
3. Design the effective diagnoser along with the selected sensors;
4. Design an effective fusion algorithm to combine sensors and signal processing methods to improve performance;
5. Reduce the cost of the diagnosis system without losing the performance merits;
6. Automate the design process of a diagnosis system.

To address the difficulties proposed in [2] in diagnosing PLC based automated assembly system, we have investigated system architectures' effects on the diagnosis performance [3, 4].What remain incomplete are sensor deployment and diagnoser's effects on diagnosing PLC based automated systems. Sensors and diagnosers work in tandem to affect diagnosing performances, instead of individually affecting the diagnosis work, so the study of them together can help researchers to understand how they facilitate the diagnosis on systems. PLC based automated systems are typical discrete event systems (DES). A Discrete Event System (DES) is a discrete-state, event-driven system, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events over time [5]. Major systems' classification is shown as below Figure 1. The conventional differential or difference equation based diagnostics are not effective in analyzing discrete event systems. Therefore this problem calls for a unique methodology on deploying sensor and designing diagnoser on DES.

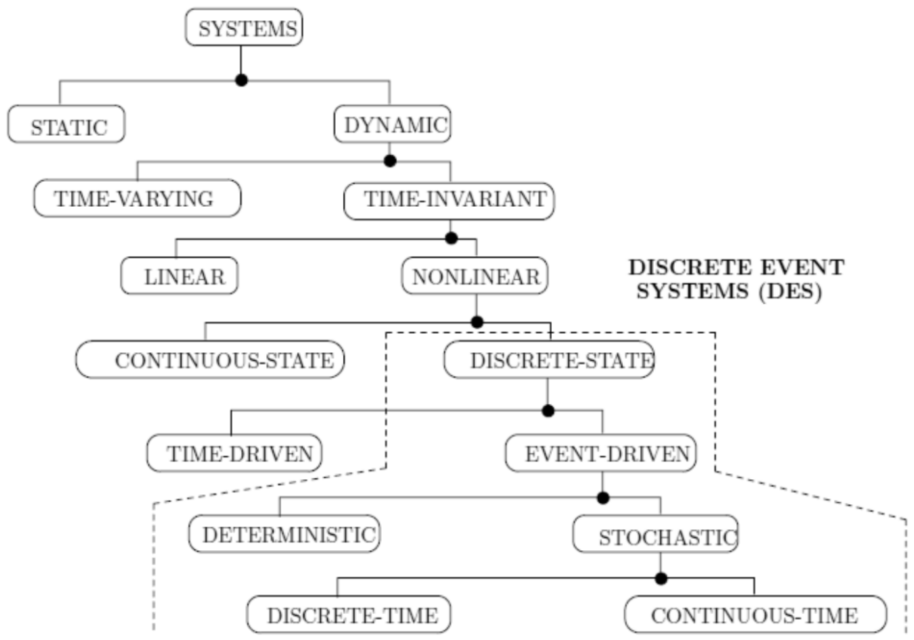


Figure 1 Major system classifications [5]

Firstly, sensors and sensing technology constitute the fundamental basis for the fault diagnosis. Diagnosis systems' performance critically depends on whether sensor measurements can monitor faulty symptoms accurately and efficiently. Sometimes one sensor can monitor several symptoms; sometimes one symptom needs to be detected by several sensors. Due to the budgetary or physical constraints, it is impossible to install every necessary sensor to monitor the fault signature. Insufficient or inaccurate measurements resulting from improper sensor deployment will significantly deteriorate fault diagnosis system's performance. Although redundantly sensing every physical system parameter can minimize information loss, the redundant sensor network may be cursed with overburden on data amount as well as analysis cost. A good sensor deployment strategy can result in a network configuration at the minimum cost while observing pre-specified performance criteria. Currently, the sensor deployment strategies in diagnosis are mainly based on *ad hoc* or heuristics method, it is mostly an "artistic" procedure, instead of a scientific technique [6]. Although analytical techniques on sensor deployment optimization have been suggested in recent years through qualitative methods such as directed graph [7] or signed directed graph [8]; or quantitative methods such as mathematic programming [9-10] or quantitative graph [11-12], literature survey [13] also noted that current research reported in the area of sensor deployment for fault diagnosis lacks a methodology to handle heterogeneous sensors-fault information and distribute sensors. Besides this, Sensor deployment for fault diagnosis is a delicate work which tackles multiple objectives including observability, reliability, accuracy and efficiency under the constraints of cost, resources and environment etc. Most sensor deployments only tackle a single objective such as either cost or reliability. Multiple-objective decision-making to optimize cost, reliability, accuracy and efficiency under the constraints of available resources and environment, is a prominent issue in sensor deployment.

Secondly, when PLC automated systems get more sophisticated and complex, the identification of fault root cause also becomes more complicated. This case is even worse for the stochastic faults, because it is hard to get the detailed fault modes [14]. Finite state automaton (FSA) and Petei net (PN) are the most popular approaches to model and diagnose DES systems. Diagnoser designers have worked on diagnosing deterministic faults with FSA [15-22] or PN [23-39], some researchers also tried to detect undetermined stochastic faults with stochastic timed automaton [40-41], stochastic automaton [42-43] or stochastic timed Petri net [39]. Both FSA and PN based diagnosis are both model based approaches. An important issue to the success of these diagnosers is whether they can incorporate the knowledge about the faultless and faulty system behavior [41]. For stochastic faults, it often has incomplete fault messages and data rendering analysis or diagnose automated system faults ineffective. In order to improve the efficiency of diagnosis, how can we integrate the useful but incomplete knowledge about the fault into the diagnoser design?

Initiated by these, we get our research question: can we propose a methodology including the selection of crucial sensors and the design of effective diagnosers to diagnose stochastic faults in PLC based automated assembly systems?

1.3 Research objective and methodology

1.3.1 Research objective

Based on the challenges identified in the section 1.2, the objective of this research includes four aspects:

1) Understand the effects of sensors and diagnosers on diagnosing stochastic faults in PLC based automated systems;
2) Develop a novel sensor deployment approach that can handle sensor heterogeneity and multiple-objective optimization in sensor allocation;
3) Design a capable diagnoser that can integrate knowledge from industrial experience to accurately diagnose stochastic faults on PLC based automated assembly systems;
4) Develop a comprehensive methodology to analyze performance parameters, thus evaluating how the sensor deployment and diagnoser design facilitate diagnosis work.

This goal is deemed complex, because the diagnosis work requires the integration of many disparate applications including sensors, signal processing, information fusion, and decision making, etc. However, if successful, the outcome of this research will provide designers with a tool that can aid the design process of diagnosis systems by arranging all the resources with high efficiency, short development time, and cost-effective numbers of sensors.

1.3.2 Research methodology

Before describing methodologies to reach the objective, we would like to propose the assumptions of this work:

1) We focus on the PLC based automated assembly system with discrete event systems (DES) characteristics which are event-driven and discrete input/output state spaces.
2) We assume that the faults in the system are stochastic. According to literature [14], the faults in the system can be classified as either deterministic or stochastic. The occurrence of deterministic failure is nonrandom. The deterministic failures can be observed instantaneously. For stochastic failure, it is hard to get the stochastic failure mode about the system due to the insufficient output that can be observed from the system; only part is available instead of complete failure mode about the system.
3) We focus on the system or equipment faults, rather than the product faults. The difference lies in that products are manufactured by systems; good products can be manufactured if and only if the manufacturing system behaves normal, but a good system can manufacture bad products as well due to improper process planning. The fault detection on products is usually solved with statistical process control (SPC) or defect analysis, which is not the emphasis of this dissertation.
4) In this dissertation, we use failures and faults interchangeably. In the literature [44], the researchers defined the difference between faults and failures. Failures occur when a resource--which is a collection of entities such as controllers, machine, tools and software program--ceases to deliver the expected task. An error occurs when some part of the resource reaches an undesired state. A fault is the cause of an error, a sequence of errors, or a failure.

5) The failure modes about the system including the failure rates, severities, priority and fault effects, are known or partially known to the researchers.

6) We focus on the single fault situation. The investigation on the simultaneous multiple faults scenario is the future research.

7) The proposed approach is for guiding sensor placement for a new designed system, not for improving the existed system.

8) The life time cost for sensors is not a factor considered as affect sensor deployment. How to include the sensor's life time cost into sensor deployment optimization is a future direction.

To target aforementioned objectives identified in section 1.3.1, we would like to propose solutions in three major stages: 1) optimize sensor deployment strategy that can handle heterogeneous sensor-fault information and target multiple objectives; 2) design a Petri net (PN) diagnoser for detecting stochastic faults in realtime; and 3) evaluate the proposed sensor deployment and diagnoser in a PLC controlled manufacturing system.

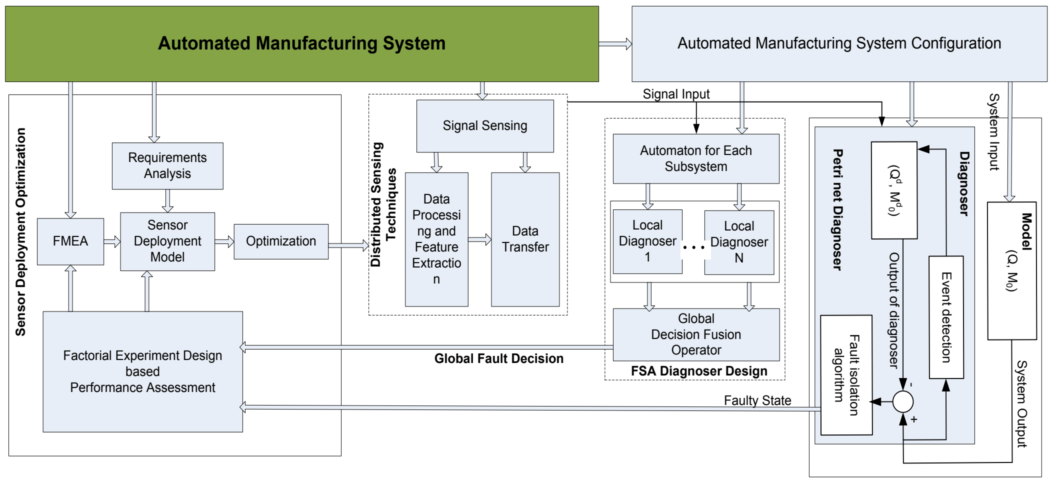

Automated Manufacturing System
Automated Manufacturing System Configuration
Sensor Deployment Optimization
Requirements Analysis
FMEA
Sensor Deployment Model
Optimization
Factorial Experiment Design based Performance Assessment
Distributed Sensing Techniques
Signal Sensing
Data Processing and Feature Extraction
Data Transfer
Signal Input
Automaton for Each Subsystem
Local Diagnoser 1
Local Diagnoser N
Global Decision Fusion Operator
FSA Diagnoser Design
Global Fault Decision
Faulty State
Petri net Diagnoser
Diagnoser
(Q^d, M^d_0)
Output of diagnoser
Event detection
Fault isolation algorithm
System Input
Model
(Q, M_0)
System Output

Figure 2 Architecture of proposed methods

As the research architecture shown in Figure 2, the detailed tasks for the dissertation have four aspects.

Task1) Propose sensor deployment based on fuzzy quantitative graph to handle heterogeneous properties for different sensors and multiple-objective optimization on distributing sensors to detect faults. This deployment strategy will optimize the cost and reliability of the sensor system under the constraints of detectability, limited resources etc.

Task2) Propose and implement fault diagnosis methodology based on real-time Petri net and fuzzy Petri net to detect faults in the discrete manufacturing system. This diagnosis approach should overcome the shortcoming initiated by partial information, computation complexity and knowledge integration etc.

Task3) Implement fault diagnoser based on finite state automaton and sequential function chart to detect faults in automated assembly systems. This is a benchmark diagnoser that is going to be compared with the proposed diagnoser in Task 2).

Task4) Design and analyze experiments based on the factors involved--sensors, diagnosers, and faults--to comprehensively evaluate the factors on detecting the faults with a dual robot system in the System Integration Laboratory at Texas A&M University.

1.4 Organization of the dissertation

The organization of the rest of the dissertation is as follows: Chapter II presents the state of arts in the area of: 1) faults with automated assembly systems, 2) sensor deployment, and 3) DES diagnoser design. Existing gaps were summarized in this chapter. Chapter III presents the sensor deployment design considering heterogeneity and multiple objectives for diagnosing manufacturing systems. The result of this chapter has been submitted to "*Robotics and Computer Integrated Manufacturing*", and is in the second round review. Chapter IV presents the realtime fuzzy Petri net diagnoser design. The result of this chapter has been published in "*International Journal of Advanced Manufacturing Technology*". Chapter V describes the diagnoser design based on finite state automaton model and sequential function chart. The result of this chapter has been published and presented in "*International Symposium of Flexible Automation*" 2010 and 2012. Chapter VI presents the comprehensive performance evaluation of sensor deployment and diagnoser for fault diagnosis in a discrete manufacturing system. Chapter VII highlights the contributions and suggests the directions for future work.

CHAPTER II

LITERATURE REVIEW

In this section, a literature review of the studies on faults in automated systems, sensor deployments for diagnosis purpose, and diagnoser on discrete event systems, is presented. It also summarizes the existing gap identified in literature and formulates the problems to be addressed.

2.1 Faults in automated assembly systems

The faults in automated assembly systems can be classified either according to the occurrence rate or root cause types.

2.1.1 Faults classification according to the occurrence rate

In literature [14], it classified the faults in systems into either deterministic or stochastic type with different occurrence rate.

1. **Deterministic faults:** The deterministic faults are new and can be observed instantaneously, its occurrence is nonrandom, and the aging faults (tearing or wearing) are clock time failure, not necessarily operational.
2. **Undeterministic faults:** This kind of fault is also referred to as the stochastic fault. It can be further classified as faults under risk or faults under uncertainty. For stochastically failing equipment under risk, it is impossible to predict the exact time of faults; but the distributions of the time to faults of each component of the system are known [14]. Typical fault rate can be constant, such as exponential distribution, or increasing such as Weilbull distribution, gamma distribution, etc. For

stochastically failing equipment under uncertainty, the exact time of failure and the distribution of the time to failure are not known [14]. It is hard to get the stochastic failure mode about the system due to the insufficient output that can be observed from the system. Thus stochastic fault diagnosis can give only an assessment of the likelihood about the faults. Sherif classified them into three situations: [14]

a) The system is new, so the information and failure data are not known.
b) Limited information about the system's failure characteristics is known.
c) Some subjective information (judgment, belief) about the system failure characteristics is known.

2.1.2 Faults classification according to the root cause type

The faults in assembly systems can also be classified as follows according to fault root causes:

1. **Hardware faults** [45]: The hardware faults are classified as input sensor faults and output actuator faults. Input sensor faults occur when a sensor is defective. Output actuator faults happen when an actuator is dysfunctional in some way from acting at all, or does not work within a prescribed period of time. They are also known as equipment faults [46].
2. **Software faults** [45]: They are caused by improper software design or implementation. They manifest in the form of a system or component fault. For example, an actuator's mis-timed action or system miss-initialization.

3. **Product faults** [45]: They are in the form of products manufactured not conforming to a specific standard. These are also known as quality faults [46] and refer to deterioration in product quality that is not normally detectable by the system sensors, which are conventionally used for control purposes. They could be caused by low quality in materials or components, or by a hardware fault in the manufacturing system, such as arrival of a faulty component or a component being dropped. As mentioned in Chapter I, product faults are not the main focus of this dissertation.
4. **Task faults** [46]**:** A task fault is defined as a deviation from the expected operation of the process due to unpredictability and lack of constraint (for example, failure of inserting a screw in a hole assembly). They can be detected if they are expressed at the sensor outputs as deviations from the normal operation. These can also be referred to as operational errors [47].
5. **Tolerance faults** [47]: These faults are caused due to defective parts, or parts that do not meet the specifications. These are errors attributed to the properties of parts.

The faults and their occurrence rate in a video tape recorder assembly line were presented in [48]. The assembly line considered in their study was made up of components such as conveyors, robots and part feeders. The faulty data of 89 such assembly cells showed that the part feeder system, robot grasp and insertion system and fixture location system were most susceptible to faults, followed by unqualified parts.

Data regarding the distribution of faults in robotic assembly [49] were acquired from three robotic assembly cells grouped under set A, set B and set C with 98, 392, and 368 samples of assembly actions, respectively. It was observed that the faulty cases registered were 31.6% for set A, 30.6% for set B and 13.3% for set C. All these faults except one in set A were attributed to failure of insertion or seating (insertion where gravity is intended to assist). Presentation faults were caused by the deviation in the part configuration as expected by the work cell. For sets B and C, faults were distributed as shown in Table 1 Fault distribution for robotic assembly cells.

Table 1 Fault distribution for robotic assembly cells

Fault (A)	Fault (B)	Percent	Fault (C)	Percent
Insertion or Seating (an gravity assisting insertion)	Insertion	51.3%	Insertion	71.4%
	Grasping	17.5%	Dropping	16.3%
	Sensing	16.7%	Grasping	6.1%
	Presentation	8.3%	Others	6.1%
	Flawed parts	5.8%		

The screw insertion process [50] was considered in detail wherein the causes for insertion failure of the screw were identified as a mismatch in the diameter of the hole in the base plate with the screw, which is inserted. Another type of insertion failure is jamming, which could occur due to several reasons, including manufacturing errors where the main body of the screw widens close to the head, a hole diameter reduction at the end of the insertion, or the presence of burrs in the hole etc.

The possible faults in robotic assembly [51] were identified as eccentric gripping of the peg due to loss of tolerance of the position of the gripper or fixture or impacts damaging the peg or fixture during extraction of the peg and presence of burrs on the edge of the base part or dirt on the chamfer of the bore, resulting in a fault. The causes for insertion failure were identified as dimensional errors of the peg or the hole (height and diameter), including the angular misalignment of the peg; presence of extraneous matter at the contact point resulting in high friction; and improper peg parts having ruts and burrs.

From these observations, we can conclude that inserting and grasping a part and keeping hold of it are among the most susceptible to faults. This is followed by sensor failures and bad parts. These failures would ideally represent typical situations that need to be addressed in the implementation of diagnosis system for automated assembly systems.

2.2 Literature review on sensor deployment

Sensor deployment problems usually involve four phases sequentially: 1) model the faults' cause-effect on sensor data variations; 2) set up the objective functions for sensor deployment based on the variation effects; 3) find approaches to optimize sensor deployment strategy; and 4) evaluate the optimized strategy. Among them, step 1) and 3) are the most important. Thus we also searched literature from aspects of: 1) modeling cause-effect relation between system faults and monitoring sensors, and 2) optimizing the cause-effect model.

Ding et al. [13] presented a comprehensive survey of inspection strategy and sensor distribution in discrete-part manufacturing processes. In his survey, he noted that diagnosis-oriented sensor distribution strategy is a relatively new problem with lots of research opportunities; especially no report has been found on how to systematically deploy the heterogeneous sensors.

Graph theory has been applied on optimal sensor deployment strategies from qualitative [7-8, 52-54] or quantitative [9-10] perspectives for sensor deployment's effects on assessing complex system status. Ali et al. used the spanning tree to model and optimize the sensor deployment for fault observability and detection reliability [52]. They defined the sensor deployment's process reliability as the smallest reliability among all of the process variables. Mass-flow and energy distribution balances in chemical plants are the basis for generating the spanning tree. Later this spanning tree procedure was extended for optimal design of a redundant sensor network for linear processes [53], as well as a nonredundant sensor network for bilinear processes [54]. Raghuraj and Bhushan et al. had qualitatively investigated the sensor deployment problem with directed graph (DG) [7] or signed directed graph (SDG) [8] for the chemical plant. The authors assumed that all faults had to be defined clearly along with their tolerances using a priori knowledge; then DG/SDG can be SDG/DG were applied to represent the cause-effect relation between faults and sensors and guide sensor placement. The only difference between DG and SDG is that signs are placed on the arcs of DG to get an SDG. However, the structures are exactly the same. In [7], fault observability or resolution was the single objective to be maximized through greedy search, so fault nodes would be covered under the constraints of sensor numbers or cost. In [8], various unique issue in SDG cause-effect model including presence of multiple paths, multiple faults occurrence, control loop were discussed and then optimization issue were extended from DG's approach. The DG/SDG based sensor deployment shows that the inclusion of signs may improve the fault resolution of SDG, but the signs of some of the arcs in the SDG might require plant-specific information that might not be available at the design stage. These two graph methods were largely qualitative without including quantified information from the system. Thus it was hard for them to model the fault propagation in complex systems only with the qualitative information on faults and sensors.

To overcome the deficiencies of qualitative method, Bhushan et al. proposed a comprehensive design strategy considering the quantitative information including fault occurrence and sensor failure probability into an integer programming formulation [9]. They tried to minimize the cost of the sensory monitoring system while ensuring that the solution provides threshold reliability. Following this paper, he presented a detailed application of applying the proposed approach to a large flowsheet, various issues involved in the application of the reliability maximization based optimization procedure were discussed [10]. They only optimized one objective, either minimum cost or maximum reliability at one time, and only mentioned a little bit on the multiple objective issue based on the one step optimization approach.

Zhang et al. also tried to attack this problem using quantified directed graph (QDG) [11, 12] and particle swarm optimization. Various fault properties including fault severity, criticality, occurrence rate, sensor properties including sensitivity, and sensor-fault relation including propagation time and gain were included into the graph modeling. These quantitative values were fused using empirical equation into single edge values representing sensors' detectability to faults. Zhang's method on quantitatively defining the sensor detectability did not well address the uncertainties involved in fault detection, and the optimization process is not clear on the criterion either.

From the discrete event system (DES) control perspective, researchers have used Petri net (PN) [55] or finite state automaton [56, 57] to model the sensor deployment. Ru [55] assumed the DES is partially observable, then used Petri Nets to model optimal sensor selection in DES, thus achieving a minimum number of sensors while maintaining structural observability to uniquely determine the system state based on sensor information. He divided this problem into optimal place sensor selection (OPSS) and optimal transition sensor selection (OTSS). To avoid NP-hard in OPSS problem, they first reduced the problem to linear integer programming, and then proposed two heuristic algorithms (top-down, and bottom-up) to approximate its solution with polynomial complexity. The OTSS problem is solvable with polynomial complexity. But their methods all strive to optimize the single objective: minimizing the total number of sensors, and they cannot specify where to locate the selected sensors.

From quality control or process variation perspective, Ding et al. have proposed several sensor deployment approaches based on different optimality [58, 59, 60], Then fast exchange routine with a sort-and-cut procedure was applied to place coordinate sensors for estimating variation means and variance, but their assumptions are for homogeneous sensor types such as coordinate measuring machines (CMMs). Nof et al proposed sensor economy principles and selection procedures in manufacturing systems, thus enabling flexible automation and quality control [61]. He noted that it was not possible to develop an accurate model on predicting sensor performance in the real world, so performance could only be derived in an empirically but systematic manner. Based on operational analysis and economical condition, he decided on a general design guideline for sensor numbers, types, locations, interaction modes and overall performance. Li had proposed a sensor deployment approach in order to detect system abnormality in time [62]. She formulated the casual relationship among the system's physical variables using Bayesian Networks (BN) model. The sensor allocation task was formulated into a "set covering" problem with the aim of minimizing cost while observing detectability requirement. An integrated algorithm combing pre-processing and greedy search was applied to optimally decide which physical variable deserved sensing. Khan et al proposed a methodology by configuring sensors to provide an optimally distinctive signature for detecting faults in discrete part assembly processes [63]. Based on assembly structure data derived from CAD files, a multi-level, two-step, hierarchical (bottom-up or top-down) optimization procedure was used to obtain a novel, distributed sensor configuration. He quantified diagnosability performance in the form of a defined index, so the diagnosability index can guide the optimization and establish the diagnostics worth of sensor distribution candidates.

After modeling the cause-effect relation between faults and sensors, we are starting to optimize the sensor deployment model. Various optimization algorithms, from heuristic search to mathematic programming, have been used for optimizing the sensor deployment. Most of the qualitative graph based sensor deployment approaches including spanning tree [52-54], DG [7] or SDG [8] were using heuristic search. In spanning tree based sensor deployment, authors used the concept of cutsets and hill climbing to identify the sensor set with optimal reliability. SDG and DG approaches were using the concepts of key component and greedy search to solve the sensor-fault cover problem. Other heuristic methods, including simulated annealing (SA), Tabu search (TS), and genetic algorithm (GA), have been applied to this problem. Both TS and SA evaluate a neighborhood on performance, and move to the neighbor with better performance. However, whereas SA relies on probabilistic events to search for good solutions, TS deterministically incorporates search history as well as structural features of the model to drive it towards higher quality solutions.

Kannan had proposed a two phase simulated annealing based localization (SAL) algorithm to address the flip ambiguity issue in wireless sensor network localization [64]. In the first phase, SA was used to obtain an accurate estimate of location. Then a second phase of optimization was performed only on those nodes that are likely to have flip ambiguity problem. Based on the neighborhood information of nodes, those nodes likely being affected by flip ambiguity were identified and moved to the correct position.

Kincaid [65] used a simple static Tabu Search method to seek the number and location of sensors for active controlling and/or sensing vibrations of truss structure. The search showed Tabu Search approach dominates the traditional approaches to finding D-optimal designs. Compared to other heuristic search methods GA can reduce the elapsed time required for solution of a large combinatorial problem because GA methods are easy to adapt for use on parallel computing. Swann had used GA and finite element analysis (FEA) technique to develop an optimal sensor placement procedure that is used to determine the optimal sensor pattern for detecting seeded delamination locations in a composite plate [66]. Sen et al. had developed a sensor network design approach based on graph theory and genetic algorithm [67]. The sensor network was designed to optimize a single criterion of cost, reliability or estimation accuracy, using a minimum number of sensors. However, Sen's algorithm cannot solve the multi-objective optimization problem. The heuristic optimization algorithms cannot guarantee convergence to the global optima, but can cover useful local optima after examining part of all possible combinations, so they are suitable for a small and simple sensor deployment problem, although it is straight forward and easy on implementation.

Sensor optimization problem have also been studied from mathematic programming such as Integer Programming (IP) [55, 68], Nonlinear Programming [69], Dynamic Programming (DP) [58] etc.

Bagajewicz proposed a mixed integer nonlinear programming (MINLP) problem to obtain cost optimal sensor networks for linear systems subject to constraints on precision, residual precision and error detectability [69]. He used the tree enumeration and branch first rule to optimize the nonlinear programming problem thus achieving cost-optimal sensor network under the constraint of precision, gross error, availability and resilience.

Projection matrix based sensor deployment such as fault signature matrix [70, 71] was used to classify fault variables. Using fault signature matrix, Fijany et al. [71] generated analytical redundancy relations between faults and sensors, then applied matrix operation such as Gauss-Jordan elimination to optimize sensor deployment. Yahya et al. [68] studied the problem of minimum sensor placement cost for directional wireless sensor networks. An integer linear programming model was proposed for the sensor placement problem in directional sensor networks to minimize the total sensor cost by properly choosing the type and direction for each sensor to be installed in the sensor field. However, the computation requirement for mathematic programming will increase considerably with the increasing of sensor numbers.

Table 2 Literature summary in sensor deployment

Modeling	**Classification**	**Reference**
Qualitative	Spanning tree	Ali (93') 52, Ali (95') 53, Ali (96') 54
	Direct graph	Raghuraj (99') 7, Bhushan (00') 8
	Signed direct graph	Bhushan (00') 8
	Petri net	Ru (10') 55
	Finite state automaton	Jiang (03') 56, Park (96') 57
Quantitative	Quantitative direct graph	Zhang (05') 11, Zhang (07') 12
	Mathematic programming	Bhushan (02') 9, Bhushan (02') 10
	Fault signature matrix	Osais (08') 68, Abed (08') 70
Optimization	**Classification**	**Reference**
Heuristic search	Simulated annealing	Kannan (06') 64
	Tabu search	Kincaid (02') 65
	Genetic algorithms	Swann (04') 66, Sen (98') 67
Mathematic programming	Integer programming	Bhushan (02') 9, Bhushan (02') 10
	Dynamic programming	Liu (04') 58
	Nonlinear programming	Bagajewicz (97') 69

The references were summarized in above Table 2. From the literature readings, we identified two typical issues in sensor deployment remain intact: 1) Heterogeneous properties of sensors in the diagnosis process. In a typical fault diagnosis system, it usually deploys sensors which generally have different sensing characteristics including uncertainty, accuracy, resolution and statistical property on physical signal data. Nevertheless, how to systematically select crucial and optimum sensor deployment for heterogeneous sensor system poses a unique problem in the automated assembly system, which has never been reported [13]. 2) Multiple-objective optimization. Sensor deployment for fault diagnosis is a delicate work which tackles multiple objectives including observability, reliability, accuracy and efficiency under the constraints of cost, resources and environment *etc*. Most of the sensor deployment researchers only targeted single objective such as either cost or reliability. A comprehensive method that considers the multiple-objective decision making involved in the sensor deployment is yet to be created. As such, these two issues call for a systematical procedure to design a cost-effective and highly reliable sensor deployment strategy. This approach should be able to incorporate the heterogeneous properties of sensors uncertainties such as sensor failure and accuracy into the cause effect model, also consider the multiple objectives including cost, reliability, observability, etc. involved in fault detection while subjecting to the constraints.

2.3 Literature review on diagnoser for discrete event system

Many methods, including the mathematical model based approach [72], fault tree [73, 74], artificial intelligence such as expert system [75], neural network [76] etc have been developed to deal with fault diagnosis. A comparison on these approaches' applications on diagnosis system is summarized in Table 3.

We focused on discrete event systems which have inherent discrete state space of logic values and event-driven. System models based on DES descriptions may give more efficient diagnosis algorithms than models based on continuous time system approaches [15]. FSA and PN are the two most commonly used methods on modeling and diagnosing the DES. Besides this, I focused more on diagnosing stochastic DES faults. How to incorporate "stochastic" and "uncertainty" properties into diagnoser were also key words when I was reviewing the literature. Based on these, the literature was searched from the aspects of: stochastic fault diagnosis based on 1) FSA or 2) PN.

Table 3 Comparisons on different fault diagnosis approaches

Approaches	**Pros**	**Cons**
Fault tree	Easy to read and understand, can be synthesis automatically [18]	Difficult to include information about ordering and timing information of events in fault tree. No way to treat common-cause failures resulting from fault propagation [18]
Expert system	Suit for systems that are difficult to model, i.e. systems involving subtle and complicated interaction whose outcomes are hard to predict [23]	A considerable amount of time may elapse before enough knowledge is accumulated to develop the necessary set of heuristic rules for reliable diagnosis, very domain dependent, difficult to validate [23]
Model based analytical redundancy	Mostly for continuous system, is able to detect abrupt faults and incipient fault [23]	Computation load for detailed online modeling of process, the sensitivity of detection process with respecting errors and measurement noise [23]
Finite state automaton	Easy to set up the component and system model, mostly for DES	Model complexity explosion by explicitly listing all possible states and events, lack of readily available software packages
Petri net	Mathematical capability and graph description of DES	Lack of readily available software packages

2.3.1 Literature on FSA diagnoser design

In order to identify the existing work related to the FSA diagnoser design, the literature review was further classified from two aspects: fault characteristic (stochastic or deterministic) and diagnoser architecture (centralized or decentralized). The summarization of FSA diagnoser design is in Table 4.

Table 4 Review on FSA based diagnoser design

Reference	Method	Fault Character	Diagnoser Architecture
Sampath [16]	The process to model the plant and to construct the diagnoser using classical DES theory	Deterministic	Centralized and Decentralized
Zad [18]	The automaton based diagnosis framework and scheme to reduce the diagnoser complexity	Deterministic	Centralized
Sampath [17]	Necessary and sufficient condition for diagnosability and I-diagnosability	Deterministic	Centralized
Holloway [19, 20]	Use discrete-event template to monitor the manufacturing system	Deterministic	Decentralized
Mouchaweh [22]	Decentralized diagnosis using Boolean discrete event model	Deterministic	Decentralized
Lunze [40]	state observation and diagnosis discrete event systems described by stochastic automata	Stochastic	Centralized
Thorsley [42]	Embed Markov model in DES to diagnose system, proposed the theory of A-diagnosability and AA-diagnosability	Stochastic	Centralized
Liu [43]	Extend Thorsley's stochastic diagnosability theory to decentralized diagnoser, propose the concept of co-diagnoser	Stochastic	Decentralized
Inagaki [77]	Use Bayesian network as global diagnoser and timed Markov model as local diagnoser, to diagnose event-driven controlled systems.	Stochastic	Decentralized
Anthanasopoulou [78]	Probabilistic algorithm to calculate the likehood of normal model and faulty model under partial observation	Stochastic	Centralized

There are two important papers [16, 18] that we need to highlight in the FSA diagnoser design, because other FSA diagnoser designs were extended from the framework proposed in these two papers. Sampath et al. proposed a method to model the system's faulty behavior in the plant model. This modeling process includes: 1) identification of possible states and events transitions with them; 2) sensor mapping between states and observing sensors; 3) estimation on occurrence of events based on the reading changes in the observing sensors; and 4) then use label propagation and events observations from 3) to construct the diagnoser [16]. They also provided necessary and sufficient conditions for the diagnoser's diagnosability as well as I-diagnosability [17]. Sampath's method required that the plant model and the diagnoser start from the same starting point which may not be easy to synchronize at the initial time for many industry applications. To overcome this deficiency, Zad et al. proposed a similar method as Sampath's, but without synchronizing requirement [18]. In Zad's work, he also suggested how to reduce the complexity of diagnosers with model reduction approach. By nature these two methods explicitly list the possible states and events, which have high computation requirements and do not fit for many practical manufacturing systems with distributed control requirements and information characteristics. Thus Lafortune also proposed a decentralized method to model the system and corresponding distributed properties [16] to lower computation load.

Holloway et al. used the distributed template model to monitor discrete event manufacturing systems [19, 20]. Forward and backward templates were developed as the diagnoser to predict future events following the triggering events and to indicate possible prior events before the trigger. They also combined templates with statistic process control (SPC) for discrete I/O signals in manufacturing systems [21]. Nevertheless, their template method does not analyze either fault isolation or the diagnosability of the diagnoser, it only alarmed the occurrence of faults. Sayed-Mouchaweh et al. proposed a decentralized diagnoser for manufacturing systems based on Boolean discrete event models [22]. Local diagnosers were designed using event sequences, time delays between correlated events and state conditions to detect abnormalities about system execution. To avoid partial observation of local diagnosis, a central diagnoser to coordinate local diagnoser was designed based on rules to isolate fault partition.

All these aforementioned FSA diagnosis methods are for deterministic models. Their faulty cases usually were abrupt, either normal or faulty, instead of probabilistic nature. However, in the real process, many systems' faulty processes are stochastic, instead of abrupt. The stochastic faults show undetermined properties and are even harder to be detected with partial information. To design a diagnoser that can locate such kind of unobservable probabilistic faults will be more realistic and meaningful to real applications.

Lunze is the first researcher studying the stochastic fault diagnosis with automaton. He solved the problems of state observation and diagnosis discrete event systems described by stochastic automata [40]. They assumed that systems were not observable but it was possible to reconstruct the state unambiguously. The observation problem was set up as the problem of determining the smallest possible set of states that are compatible with the measured input and output sequences. The diagnostic problem was solved as an observation problem. They also discussed the conditions for the observability and diagnosability of stochastic automata.

David Thorsley proposed an approach on applying stochastic techniques on the partially observed discrete event systems [42]. In his approach, Markov model was embedded with the FSA to model and diagnose DES. With Markov model's computation ability, the possibilities of the states in each discrete step were calculated using Bayesian law. Also it relaxed Sampath's diagnosability condition to A or AA diagnosability with probability. The difference between Thorsley and Lunze's stochastic diagnoser lies in that: Lunze clearly pointed out that the diagnostic problem of a stochastic automaton cannot be solved by another stochastic automaton. On the contrary, Thorsley inherited the diagnoser approach, where the diagnoser possesses a probabilistic structure by appending to each transition on a matrix that can be used to update the probability distribution on the state estimate. Besides this, Thorsley's diagnoser does not require detailed in-depth modeling of the system to be diagnosed.

Liu et al extended Thorsley's stochastic diagnoser to decentralized stochastic diagnoser. He investigated the decentralized diagnosis of stochastic discrete event systems (SDESs) by using multiple local stochastic diagnosers, each possessing its own sensors to deal with different information [43]. They formalized the notions of decentralized diagnosis for SDESs by defining the concept of codiagnosability for stochastic automata in which any communication among the local stochastic diagnosers or to any coordinators is not involved. A stochastic system being codiagnosable means that a fault can be detected by at least one local stochastic diagnoser within a finite delay. A codiagnoser was constructed from a given stochastic system with a finite number of projections, each of which contained a local diagnoser to complete the model of the system. Necessary and sufficient conditions of the codiagnosability, as well as computing method to check the codiagnosability for SDESs was presented, which generalizes the corresponding results of centralized diagnosis for SDESs.

Inagaki et al. presented a decentralized fault diagnosis strategy, which used timed Markov models (TMM) as local diagnosers and Bayesian networks as global diagnoser, for the event-driven controlled systems such as PLC control systems [77]. The relationship between two successive events observed in the corresponding subsystem is represented by TMM. The probability density function for the successive events in TMM was estimated with maximum entropy theory. The Bayesian network represents the causal relationship between the faults and observations from subsystem. They build the Bayesian network using the control logic through sensor actuator dependency graph and dependency tree.

To overcome the partial observation problem in the finite state machine, Anthanasopoulou et al. have proposed the approach of maximum likelihood diagnoser under unreliable observations [78]. In this research, they developed a probabilistic methodology for calculating the likelihood of an observed, possibly corrupted event sequence that was generated by two candidate FSAs (one representing normal model of operation and the other for failed model). They formulated the fault diagnosis problem as deciding which FSA is most likely to have generated the observed event sequence. The observed events may be corrupted by failures causing event insertions and deletions or transposing etc. Given the possibly erroneous observed sequence, they proposed an efficient recursive algorithm by extending Viterbi algorithm in HMM to obtain the most likely underlying FSA. Diagnosability analysis was also implemented for her proposed diagnoser based on "miss detection" measurement. However, she missed the "false alarm" analysis for her diagnoser, which means her diagnosis may enlarge faulty decision even when the system is fault-free. When doing fault isolation, it needs to compare the normal and every faulty mode to identify the exact fault. It requires heavy computation load.

Finite state automaton (FSA) is a language based analysis method, which qualitatively analyzes the behavior of the discrete event systems by listing all possible transitions and states. When applying automaton on fault diagnosis, the main difficulty is the significant size of state space, which leads to problems of over-complexity, memory and speed of execution for the diagnosis. Petri net is an alternative to automaton on analyzing discrete event systems. Because Petri net focuses on structure modeling of the DES systems, applying Petri net on fault diagnosis may overcome the potential complexity deficiency. In the following section, we will review the Petri net's application on diagnosing stochastic faults.

2.3.2 Literature on Petri net diagnoser design

Following the way on reviewing FSA diagnoser, the research on PN diagnoser were also classified according to the fault character and diagnoser architecture as tabulated in Table 5.

PNs have been applied on many applications such as discrete event simulation and control, manufacturing system planning and scheduling, and manufacturing modeling and evaluation [79]. Recently, PNs have also been used for fault diagnosis applications including electro-mechanical equipment [80], power system [81], discrete event systems [24~39] etc. Maria gave a comprehensive survey of the state of the art of fault diagnosis and identification with the framework of Petri net [24], but most of researches in Maria's survey focused on diagnosing deterministic faults instead of stochastic ones.

Table 5 Review on PN based diagnoser design

Reference	**Method**	**Fault Character**	**Diagnoser Architecture**
Ushio [25]	Extend Sampath's diagnosability theory to unbounded Petri net, Proposed ω -diagnoser and ω -refined diagnoser	Deterministic	Centralized
Chung [26]	Extend Ushio's diagnosability condition on PN, constructed the label propagation function and the range function in a diagnoser, proposed the associated verifier to check PN's diagnosability	Deterministic	Centralized
Lefebvre [27]	Decide which set of places must be observed, defined minimal diagnosers to detect and isolate the firing of fault transitions immediately	Deterministic	Centralized
Ruiz-Beltrán [28]	Use interpreted PN to model the system behavior that includes partially observable events and states. Based on the IPN model derived from an on-line methodology, proposed an on-line diagnosis scheme utilizing a solution of a programming problem	Deterministic	Centralized
Genc [37]	Distributed diagnosis of PN faults	Deterministic	Decentralized
Wen [32]	Proposed an approach to test diagnosability by checking the structure property of T-invariants of the nets under the assumption that a given subset of places are observable	Deterministic	Decentralized
Ru [38]	Transformed the partially observed PN into labeled PN, calculate the belief regarding the occurrence of faults belonging to each type	Stochastic	Centralized
Lefebvre [39]	Proposed a fault diagnosis approach based on timed PN, but the transition period in the timed PN is stochastic,	Stochastic	Centralized

Ushio et al. is the first have developed a Petri net model for discrete event system with faulty behaviors and introduced the diagnosability into PN diagnoser [25]. They assumed some of the places and all transitions were unobservable, ω -diagnoser and ω -refined diagnoser were designed to detect failure transitions using coverability tree based on this assumption. Finally they extended Sampath's diagnosability condition [17] to unbounded Petri net.

Chung et al. [26] extended Ushio's PN fault diagnosis approach. They further assumed only some of the transitions (not all of them) were unobservable, and showed how to construct the label propagation function and the range function in PN diagnoser with newly available information provided by observable transitions, and presented a verifier algorithm as a polynomial check mechanism on PN diagnoser's diagnosability.

Dimitri et al. proposed an approach using partial but unbiased measurement of the places marking to estimate the firing sequence [27]. They modeled PN diagnoser's diagnosability as the undetermined cycles included in the reachability graph, so they could decide which set of places must be observed for the exact estimation of some giving firing sequences, then designed minimal diagnosers to detect and isolate the firing of fault transitions immediately. They also investigated causality relationships and directed paths to characterize the influence and dependence areas of the fault transitions to design a delayed diagnosers.

Ruiz-Beltrán et al. proposed fault detection and localization in DES using interpreted Petri net (IPN) [28]. They constructed a system model and a diagnoser by comparing normal and faulty markings, thus identifying fault places with fault isolation algorithm. An IPN is event-detectable when any pair of transitions can be distinguished from each other by the observation of input/output symbols. However, the IPN approach only identifies one fault and their PN model enters a place sink that leads to being blocked.

In real applications, diagnosis rules had been used to set up the firing rules between places to transitions or transitions to places in Petri net [29]. The diagnoser monitored the running information from the system, calculated the fault probabilities and truth degrees for the precondition and output events. Then it combined these truth degrees with fuzzy set operations to decide the root cause of the faults. Colored timed Petri net (CTPN) had been used on failure modeling and process monitoring for flexible manufacturing systems by Kuo et al. [30], they also studied CTPN's applications on statistic process control, fault diagnosis, and failure model effect analysis (FMEA) . Miguel et al. had applied Petri net on the fault diagnosis and modeling of a liquids packaging process [31]. Wen et al. have used Petri net to analyze and enhance the diagnosability of discrete event systems such as semiconductor fabrication facility [32]. They formulated the diagnosability problem into a binary integer linear programming with feasible solution, and improved the non-diagnosable system to diagnosable by adding extra sensors. Prock proposed a Petri net fault detection method for large systems using dynamic measurement signals [33]. Yao et al. have proposed a Visual Basic (VB) human machine interface (HMI) platform for hybrid PLC and PC control based on Petri net theory for manufacturing systems [34, 35]. Hu et al. used hybrid Petri net to model and detect faults in a hybrid automated manufacturing systems [36]. Genc and Lafortune et al. presented a PN fault diagnosis approach using limited places on the same example of HVAC as Sampath's [37]. In Genc's work, components were modeled without synchronous composition, and a distributed diagnostic algorithm was proposed to identify the faults in an independent way. The approach presented a problem of coupling in places that were executed at the same time, while in the process seeking to be reduced by means of an algorithm. They finished diagnosis simulation with Matlab and Graphviz. However, this technique is too complicated for implementation thus less possible to be applied to medium level of complex industrial processes.

To study the undetermined fault beliefs with PN, Ru et al. studied fault diagnosis in discrete event systems modeled by partially observed Petri nets [38]. They transformed a partially observed Petri net into an equivalently labeled Petri net, a translator was constructed translating the sensor information from place/transition sensors into a sequence of labels in the equivalent labeled Petri net, then calculated the belief regarding the occurrence of faults belonging to each type as the sequence order of observations from place and transition sensors. Then online monitor was built to recursively produce these beliefs by tracking the existence of faulty transitions in execution paths that match the sequence of labels observed so far. Nevertheless, Ru's belief calculation assumed that each event has the equal occurrence opportunity; he did not take the stochastic property into account.

Lefebvre proposed an approach based on stochastic Petri nets (PNs) to design reference and faulty DES models [39]. They used the statistical analysis of the collected alarm sequences on the considered system to design and identify of these models. The model structure is described as a state graph, and the parameters of the probability density functions (pdfs) for transition firing periods are estimated. The reference models, described as timed PNs, are then used for fault detection and isolation issues. Finally, stochastic PNs with normal and exponential pdfs are considered to include a representation of the faulty behaviors.

2.3.3 Summary on DES diagnoser design

Although FSA can build complex system models from individual components models and then combine them by parrell composition in a systematic manner, this process tries to explicitly enumerate all the possible states, and connect them with possible transition events with the system through transition function. Thus the complexities of FSA diagnoser usually increase exponentially when systems get larger and more complex. PN has more structure in their representation of the transition function. State information is embedded among a set of places that captures the key operations of the system, only relevant conditions are captured by the places, and then properly connect these places to transitions. PN's "place-transition" graphic mechanism and mathematical capability may be used to model "IF-THEN" knowledge rules more conveniently.

As identified in Section 1.2, stochastically failing automated assembly systems often have incomplete fault messages and data that render diagnose faults ineffective. Although these messages and experiences are useful, it is not that complete and precise with uncertainties in it. How can we integrate the experience and knowledge about the stochastic faults and address the uncertainty issue to improve the effectiveness of the diagnosis? Comparing with FSA, Petri net may be a better fit to the knowledge integration problem. In order to tackle the uncertainty issue, we are trying to introduce the concept of fuzzy logic into the Petri net construction. Based on that, we are trying to solve them through designing a realtime fuzzy Petri net fault diagnoser for PLC based automated assembly systems. The detailed proposed fault diagnosis method will be presented in the later “real time fuzzy Petri net diagnoser” section.

The difference between the proposed PN diagnoser and the work in [40] lies in that: they assume the DES system is a “black box” and no information about the internal structure of the system is required to design the reference model. Then they collected the alarm sequences about the DES system, and analyze the alarm sequence using statistics techniques. We will use the PN to model the key operation of the DES and design the reference model. Once observed output is different from the reference model, fault isolation algorithm will be started to locate the root causes.

CHAPTER III

PROPOSED SENSOR DEPLOYMENT CONSIDERING HETEROGENEITY AND MULTIPLE-OBJECTIVE OPTIMIZATION

To tackle the question in Section 1.2 on sensor deployment, we proposed the approach as Figure 3. Firstly, failure mode effect analysis (FMEA) on the manufacturing system is conducted to decide system fault mode. The fault information will formulate the values of fault nodes. FMEA also provides some information on how to initially select sensor type to detect certain faults. Secondly, a bipartite fuzzy graph is used to model cause-effect relation between fault nodes and sensor nodes in sensor deployment. In order to handle the uncertain vagueness and trade-offs in the sensor deployment decision, quantitative fuzzy graph [82], which is an extension of crisp graph theory, is applied to represent the detection relationship between sensors and faults. In the fuzzy bipartite graph, fault nodes include the quantitative information such as occurrence rate, severity and detecting rate, which are extracted from FMEA; sensor nodes include sensor characteristics including signal noise ratio (SNR), accuracy etc; and the edges between sensor nodes and fault nodes represent sensors' detectabilities to faults. The detetctabilities can be modeled with sensor property, fault information and sensor fault relation including sensing time and sensing gain. It involves aggregating the heterogeneous sensor-fault information into a single edge element value. Here we can use analytic hierarchy process (AHP) to achieve this. Note that those nodes and edges information quantitatively model heterogeneous physical properties, also include uncertainties on faults, sensors and detecting relations. Fuzzy theory is used to normalize heterogeneous information into comparable quantitative values and handle uncertainties. Finally, we need to optimize the sensor system's cost and reliability under the constraints of detectability. To solve the sensor deployment optimization problem, lexicographical mixed linear integer programming and greedy algorithm are applied respectively to optimize assigning of sensors to faults.

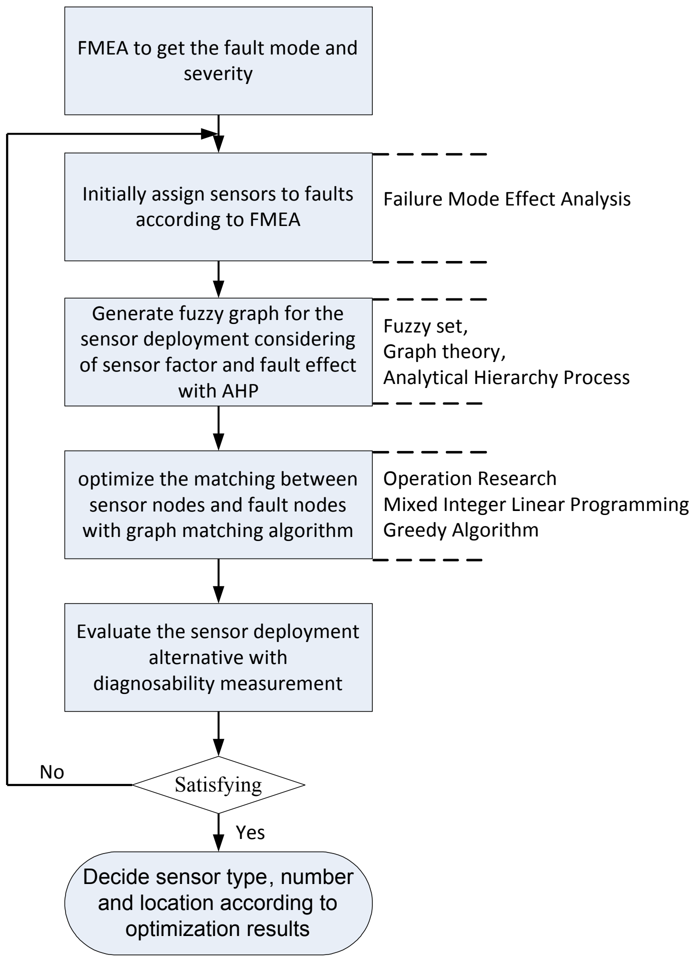

FMEA to get the fault mode and severity
Initially assign sensors to faults according to FMEA
Failure Mode Effect Analysis
Generate fuzzy graph for the sensor deployment considering of sensor factor and fault effect with AHP
Fuzzy set,
Graph theory,
Analytical Hierarchy Process
optimize the matching between sensor nodes and fault nodes with graph matching algorithm
Operation Research
Mixed Integer Linear Programming
Greedy Algorithm
Evaluate the sensor deployment alternative with diagnosability measurement
No
Satisfying
Yes
Decide sensor type, number and location according to optimization results

Figure 3 Sensor deployment strategy

3.1 Failure mode effect analysis on manufacturing system

FMEA helps understand fault modes involved in the manufacturing system being diagnosed. It is based on the combination of functional analysis and hardware analysis to identify the possible failure modes. These effects or consequences of failure modes may provide some guidelines on methods of detecting the identified modes, detection performance evaluation, and possible means of prevention [83]. Followed that is to rank the unique effect of failure one by one. Once being ranked, the systems, components designated as severity in term of failure effect, are provided analysis in a grade fashion. An example on FMEA work sheet is shown in Table 6. The quantitative information extracted from FMEA will be contributed into the fault node values for later sensor deployment modeling and optimization. The detail on extraction of quantitative information will be presented in the later section.

3.2 Fuzzy graph

A fuzzy bipartite graph $\widetilde{G}_F = \left(\widetilde{E}, \widetilde{R} \cup \widetilde{S}\right)$ is used to model the cause-effect relations in sensor deployment problem. The bipartite graph for the sensor deployment is illustrated as Figure 4. The fuzzy graph is composed of root nodes $\widetilde{R}$ and sensor nodes $\widetilde{S}$, as well as edges $\widetilde{E}$ .Root nodes $\widetilde{R}$ include the information of fault severity, occurrence rate and detection rate from FMEA; the FMEA information, which were aggregated and normalized with fuzzy set, are included in the square brackets. They represent system's functional information from diagnosis perspective. Sensor nodes $\widetilde{S}$ include information such as SNR, resolution, accuracy, and sensor uncertainties. The sensor node values are

Table 6 An example on failure mode effect analysis worksheet

Function	Failure Mode	Effects	Severity Rating	Cause	Occurrence Rating	Current controls	Detection rating	Critical characteristic	Risk priority number	Recommended actions
Extend the arm by cylinder	Cylinder can't be retracted or extended	Pallet cannot be stopped	6	Cylinder leakage or rod stuck	2	Regularly Check by operators	5	N	60	Check the pressure and air flow in the pipeline

decided in a similar way with fault nodes, and included in the parenthesis. The arc $\tilde{E}$, representing fault detection relation between, $\tilde{S}$ and $\tilde{R}$, whose values are included in the square bracket linking between sensor nodes $\tilde{S}$ and root node $\tilde{R}$ represents the sensor's detectability to fault root under the consideration of issues such as fault sensing time, sensing gain etc. The smaller the $\tilde{E}$ value is, the weaker detection ability for $\tilde{S}$ to detect $\tilde{R}$ is.

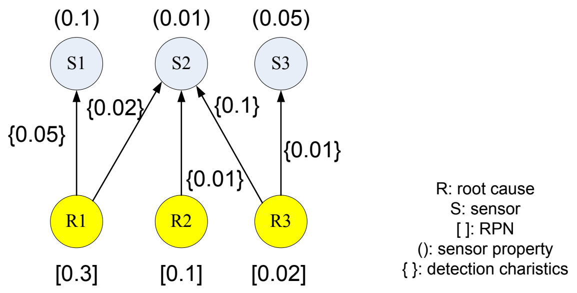


Figure 4 A fuzzy bipartite graph for sensor deployment

Here path connection with fuzzy nodes and fuzzy edges is determined as: considering fuzzy set $\tilde{R}$, $\tilde{S}$ of nodes and $\tilde{E}$ of edge, then the path p_{ij} from R_i to S_j is defined as $p_{ij} = (R_i, S_k, \cdots, S_j)$, so the value of this path connection, which represents the comprehensive detectability of those sensors to that fault is decided as below equation:

$$p_{ij}(R_i, S_k, \cdots, S_j) = \mu_{\tilde{E}}(R_i, S_k) \wedge \mu_{\tilde{E}}(S_k, S_l) \wedge \cdots \wedge \mu_{\tilde{E}}(S_{j-1}, S_j) \wedge \mu_{\tilde{V}}(R_i) \wedge \mu_{\tilde{V}}(S_k) \wedge \cdots \wedge \mu_{\tilde{V}}(S_j) \quad (3\text{-}1)$$

Here we modified the path connection as $p_{ij}(R_i, S_k, \cdots, S_j) = \mu_{\tilde{E}}(R_i, S_k) \wedge \mu_{\tilde{E}}(S_k, S_l) \wedge \cdots \wedge \mu_{\tilde{E}}(S_{j-1}, S_j)$, because the node values will be aggregated into edge values using AHP. In this equation, $\mu_{\tilde{E}}(R_i, S_k)$ means the fuzzy possibility of connecting two nodes R_i and S_k ; "$\wedge$" is the conjunction operation in fuzzy set theory and this equation will calculate the minimum possibility of connecting R_i and S_j. If there are several paths from R_i to S_j, these possible paths form the set of

$$P(R_i, S_j) = \{p_{ij}(R_i, S_j) | p_i(R_i, S_j) = (x_{i_1} = R_i, x_{i_2}, \cdots, x_{i_r} = R_j)\},$$

then the connection strength between R_i and S_j is gathered as

$$l^*(R_i, S_j) = \vee_{P(R_i, S_j)} p_{ij}(R_i, S_k, \cdots, S_j) \qquad (3\text{-}2)$$

Here "$\vee$"is the disjunction operation in fuzzy set theory, which means the maximum intensity connection among those possible paths is the connection strength between fault R_i and sensor S_j. The path connection strength formed the element values in adjacent matrix P which is composed of columns representing fault node, rows representing sensor nodes; and element $\lfloor p_{ij} \rfloor$ representing cause-effect relation on sensor *j*'s detectability to fault *i*.

3.3 Fault node value calculation

As stated in the section 3.2, fault nodes include information on fault occurrence rate (O), severity rate (S) and detection rate (D). Here risk priority number (RPN) [83] is introduced to comprehend these three kinds of information together. After ranking the severity, occurrence and detectability, the RPN value can be easily calculated by multiplying these three numbers as $RPN = S \times O \times D$. The fault modes that have the highest RPN should be given the highest priority for corrective action.

The RPN value need to be normalized into comparable values based on fuzzy membership function [82]. Let *RPN* be a classical set of object, called universe, whose generic elements are denoted as rpn. A fuzzy subset A on *RPN* is a set defined by a membership function μ_A which represents a mapping:

$$\mu_A : RPN \rightarrow [0,1]$$

Here the value of $\mu_A(rpn)$ for the fuzzy set A is called the membership value or the grade of membership of $rpn \in RPN$. The membership value represents the degree of rpn belonging to the fuzzy set A. Figure 5 is a fuzzy membership function to normalize *RPN* value. With fuzzy set, we can map *RPN* value for different types of faults into comparable values between 0 and 1.

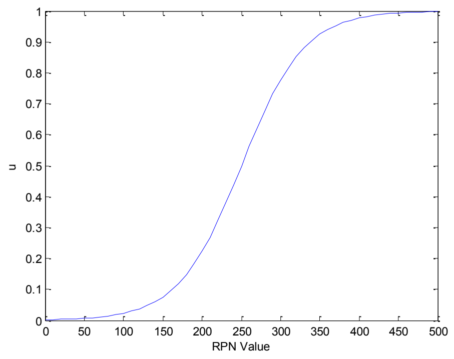


Figure 5 Fuzzy membership function for fault node

3.4 Sensor node value calculation

Sensor nodes include information such as signal noise ratio (*SNR*), sensitivity (*sen*), resolution (*res*), and accuracy (*acc*). Here sensor index (*SI*) is introduced to comprehend the information together. SI can be calculated by multiplying these four factors together as: $SI = SNR \times sen \times res \times (1 - acc)$. Similar to the fuzzy normalization step on deciding fault node value, the SI values can be mapped to [0, 1] for comparable values. Some illustrations on sensor properties are given below:

Noise caused by sensor measurement: here we use SNR as the measurement to quantify how much a signal has been corrupted by noise when sensing signals. It is defined as nature logarithm operation on the ratio of signal power to the noise power corrupting the signal. A higher SNR values means a better detectabity of sensor to faults with other equal factors.

Sensitivity: Sensitivity of a sensor is defined as how much the sensor's output changes when the measured faulty cause quantity changes. Sensors that measure very small changes must have very high sensitivities. The sensor’s fault detectability is positive proportional to its sensitivity.

Accuracy: The accuracy of the sensor is the maximum difference that will exist between the actual value (which must be measured by a primary or good secondary standard) and the indicated value at the output of the sensor. The accuracy can be expressed either as a percentage of full scale.

Resolution: The resolution of a sensor is the smallest change it can detect in the quantity that it is measuring. The resolution is related to the precision with which the measurement is made.

3.5 Sensor-fault value calculation

Sensor-fault values include information such as sensing gain (SG), sensing time (ST). Here sensor-fault index (SFI) is introduced to comprehend the information together. SFI can be calculated as sensing gain dividing by sensing time as: $SFI = SG / ST$. Similar to the fuzzy normalization step on deciding fault node and sensor node values, the SFI values can be mapped to [0, 1] for comparable values. Some illustrations on sensor-fault relation are given below:

Sensing time and **Sensing gain** are defined based on a sensor's step response. The sensing time is defined as the rise time, while sensing gain is defined as the steady state gain [84].The shorter the sensing time and the higher the sensing gain, the better the sensor fault detectability.

3.6 Edge relationship value calculation using AHP

After calculation values on fault nodes and sensor nodes, deciding the values of edge elements in the graph is critical to the success of sensor deployment because the edge values represents sensors' capabilities to observe certain fault signatures under constraints of sensor characteristics. When determining the edge values, it usually involves multiple attributes on faults, sensors and sensor-fault relations. All these factors will affect sensor deployment on fault diagnosis.

In order to integrate these properties into one edge element value in fuzzy graph, analytic hierarchy process (AHP) [85] is used as the mechanism to handle the trade-off between the properties in deployment. The detailed decision hierarchy of sensor deployment is shown as Figure 6.

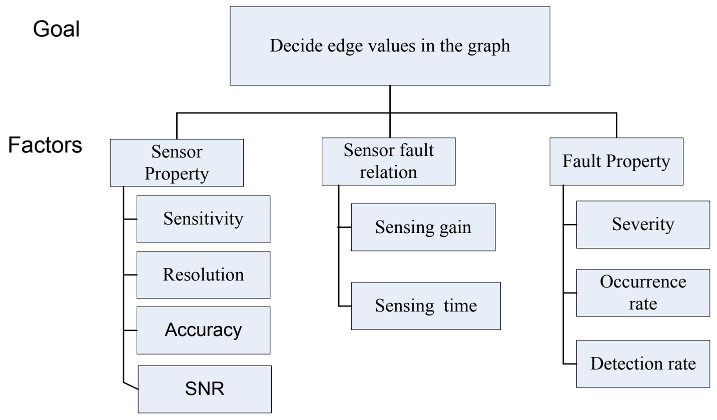


Figure 6 Decision hierarchy for sensor deployment

In Figure 6, the goal is to achieve the edge value between sensor and fault nodes for diagnosis purpose under the consideration of attributes including sensor properties, fault characteristics, and sensor-fault relations. When applying AHP to coalesce these heterogeneous properties, the procedures are as following:

1) Generate comparison matrix. a_{ij} is the pair-wise significance comparison of objective element *i* to objective element *j*, and its value is decided using Satty's scale table [85]. Assuming there are n pieces of criteria, the pair-wise comparison matrix CM_n is obtained as equation (3), here $a_{ij} = 1/a_{ji}$.

$$CM_n = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \qquad (3\text{-}3)$$

2) Calculate the geometric means of each row (m_i) and relative priorities (p_j): $m_i = \left[\prod_{j=1}^{n} a_{ij}\right]^{1/n}$ as well as $p_j = m_j \Big/ \sum_{i=1}^{n} m_i$, then we are going to have principle eigenvector for n criteria $PV = (p_1, p_2, \cdots, p_n)^T$.

3) Calculate the consistency index (CI) to verify the consistency of the result. If we define $PV' = CM_n \times PV$, and the maximum or principal eigen value from PV' as $\lambda_{\max}$, we get $\lambda_{\max} = \frac{\sum_{j=1}^{n}(p_j'/p_j)}{n}$. Then the consistency index (CI) is obtained as: $CI = \frac{(\lambda_{\max}-1)}{(n-1)}$, and we can calculate the consistency ratio (CR) as $CR = \frac{CI}{RI}$, RI is the random index can be gotten from [85]. If CR is smaller or equal than 0.1, the AHP process is acceptable.

4) Once the CR is value is acceptable, which means the decision process is consistent, we can coalesce the multiple attributes into single value with the following equation: $\widetilde{E} = \{att_1, att_2, \cdots, att_n\} \times \{P_1, P_2, P_3, \cdots, P_n\}^T$, here att_i is the value for the ith attribute. Using the AHP iteratively from the lowest decision hierarchy to the highest hierarchy, finally we can aggregate the decision attributes involved in the sensor deployment into the edge element $\widetilde{E}$.

Sensor characteristics including sensitivity, SNR, resolution and accuracy; and sensors observabilities to faults including sensing gain and sensing time were collected for sensor deployment. They along with fault effect will be processed using AHP method as Figure 6. Comparison weight matrix was decided based on AHP priority matrix as shown in Table 7. From the table, if we assume equal importance to the three factors since they are independent from each other, we can see that these factors will have the priority vector (PV) of 33.33% each. With these PV values, the heterogeneous sensor deployment decision attributes can be aggregated into single edge element value as equation (3-4):

$$\widetilde{E} = \{SI, SFI, FI\} \times \{PV_S, PV_{SF}, PV_F\}^T \qquad (3\text{-}4)$$

Table 7 Comparison matrix and AHP results

Criteria	Sensor	Sensor-Fault Relation	Fault	Priority vector
Sensor	1	1	1	33.33%
Sensor-Fault Relation	1	1	1	33.33%
Fault	1	1	1	33.33%
$\lambda_{\max} = 3, CI = 0\%, CR = 0\%$				

3.7 Sensor deployment optimization model

After constructing the fuzzy graph as well as calculating the fuzzy relation value of the connecting edges and the node values, the final step is to match the fault nodes and sensor nodes into groups. This kind of matching should achieve overall minimum unobservability and cost as under the predefined performance requirements such as detectability.

The unobservability with faults on the manufacturing system is defined as the equation (3-5) [9]:

$$\max(U_i) = \max\left(f_i \prod_{j=1}^{n} (\Pr_j)^{d_{ij}\times x_j} \right) \qquad (3\text{-}5)$$

This equation means the maximum probability of fault occurrence at the moment of sensor failure. At that time, the fault is unable to be observed due to sensor failure. Here f_i is fault occurrence probability, $\Pr_j$ is sensor failure probability, x_j is the number for sensor j. D is a binary bipartite matrix, which represents the cause-effect information between faults and sensors. The rows of this matrix correspond to faults, and the columns correspond to sensor nodes. The (i,j)th entry (d_{ij}) of this matrix is 1 if fault *i* affects sensor *j* and is zero otherwise.

$$(d_{ij})_{R\times S} = \begin{cases} 1 & \mathit{sensori\ to\ observe\ fault\ j} \\ 0 & \mathit{otherwise} \end{cases} \qquad (3\text{-}6)$$

To map the relations between sensors and faults, another matrix is generated: detectability connection bipartite matrix (P). In this matrix, the columns are associated with candidate sensor pairs and the rows are corresponding to the faults. p_{ij} is the connection strength between sensor j and fault i. p_{ij} 's value, which has fused sensor nodes' detectability information including sensing time, sensing gain, sensitivity etc, as well as fault nodes' occurring rate and severity etc, was calculated through AHP with Figure 6's decision hierarchy.

$$P = (p_{ij})_{R\times S} = \begin{cases} p_{ij} & \mathit{fault-sensor\ detect\ relation} \\ 0 & \mathit{otherwise} \end{cases} \qquad (3\text{-}7)$$

The objective of sensor deployment is to achieve the minimum unobservability and cost under the constraints of as detectability. Mathematically, this can be expressed as:

$$\begin{aligned}
&Min: \max(U_i) = \max\left(f_i \prod_{j=1}^{n} (\Pr_j)^{d_{ij} \times x_j} \right) \\
&Min: \quad \sum_j (C_j \times x_j) \qquad\qquad (3\text{-}8) \\
&Subjectto: \sum_j p_{ij} x_j \geq M_i^* \\
&\qquad x_j \in Z^+
\end{aligned}$$

In the objective modeling, x_j are the decision variables which mean how many j type sensors need to be placed in order to monitor the system operation; unobservability and cost are the primary and secondary objective function that needs to be minimized under the constraints of detectability on the ith fault (M_i^*) requirement.

We notice that the primary objective is nonlinear function, which involves huge computation complexity on optimization. Thus we transform this nonlinear unobservability equation (3-5) into linear equation with the same objective through logarithmic operation, the unobservability for the system can be expressed as equation (3-9).

$$\max(\log(U_i)) = \max\left(\log\left(f_i \prod_{j=1}^{n} \Pr_j^{d_{ij}} \right)\right) = \max\left(\log(f_i) + \sum_{j=1}^{n} x_j \times d_{ij} \times \log(\Pr_j) \right) \qquad (3\text{-}9)$$

Now we prove the logarithm transformation won't change the optimality.

Proof:

From $U_i(x) = f_i \prod_{j=1}^{n} (\mathrm{Pr}_j)^{d_{ij} \times x_j}$, we can get $\frac{\partial U_i}{\partial x_j} = (d_{ij} \ln(\mathrm{Pr}_j)) \times f_i \times ((\mathrm{Pr}_j)^{d_{ij} \times x_j})$.

We know that for all $j = 1, 2, \cdots, n$, $\frac{\partial U_i}{\partial x_j} = (d_{ij} \ln(\mathrm{Pr}_j)) \times f_i < 0$ for $0 < \mathrm{Pr}_j < 1$ while $((\mathrm{Pr}_j)^{d_{ij} \times x_j}) > 0$, so $U_i(x) = f_i \prod_{j=1}^{n} (\mathrm{Pr}_j)^{d_{ij} \times x_j}$ is monotonically decreasing with its negative derivative. In another word, the more sensors are added; the lower the unobservability for the system it is.

Because the derivative of $\log(x)$ is a positive quantity for positive x, when applying logarithm, $\log(x)$ is a monotonically increasing function for x >0. Then we have for $x_1 > 0$ and $x_2 > 0$,

$$x_1 > x_2 \Rightarrow \log(x_1) > \log(x_2)$$

The unobservability of a fault i, $U_i = f_i \prod_{j=1}^{n} (\mathrm{Pr}_j)^{d_{ij} \times x_j}$ is always nonnegative. Given a set of selected sensors, the fault i for which U_i is maximum will also give the maximum value of $\log(U_i)$. Hence, minimizing the maximum U_i is the same as minimizing the maximum $\log(U_i)$. Therefore, logarithm transformation $\log(U_i)$ is equivalent to U_i on objective solution.

After logarithm transformation, we can get the linear expression for equation (3-8). Mathematically, this can be expressed as (3-10):

$$\begin{aligned}
&Min: \max(\log(U_i)) = \max\left(\log(f_i) + \sum_{j=1}^{n}\left(d_{ij} \times \log(\Pr_j) \times x_j\right)\right) \\
&Min: \quad \sum_j \left(C_j \times \sum_i d_{ij}\right) \\
&Subject to: \sum_j p_{ij} x_j \geq M_i^* \\
&\qquad x_j \in \mathrm{Z}^+
\end{aligned} \tag{3-10}$$

3.8 Optimization approaches

As illustrated in section 3.6, the sensor deployment was formulated as multiple objectives on observability (primary) and cost (secondary) under the constraints of detectability. Given the above defined objective function, now the proble is one of choosing the minimum number of sensors key components that would cover all the root nodes, also satisfy the detectability requirement. This is the well-known ''minimum set covering'' problem [7]. An exhaustive search algorithm, which covers all fault nodes from 1 to n and tests if any combinations of sensor nodes can form a minimal cover set, is a possible solution. However, it was proved that the "set covering" problem is NP-hard; that is, an algorithm to obtain the optimal solution in polynomial-time has not been found [62]. Thus the exhaustive searching on all options may not be computationally efficient. In order to solve the multiple-objective optimization problem, we developed two ways: 1) lexicographical mixed integer linear programming problem (L-MILP), and 2) greedy algorithm.

3.8.1 Lexicographical mixed integer linear programming

In the survey paper [86], authors summarized current multi-objective optimization methods as the below Table 8. It is hard to say which method is better than the other because it really depends on the application. But we need look into whether the selected approach is necessary and sufficient for Pareto optimality. Other considerations include programming complexity (PC), software use complexity (SUC), and computation complexity (CC).

Table 8 Summarization optimization methods based on the priori preference [86]

	Scalar Method	Possible Pareto Opt.	Necessary for Pareto Opt.	Sufficient for Pareto Opt.	PC	SUC	CC
Weighted Global Criterion	√		N/A	√	0	1	1
Weighted Sum	√			√	0	1	0
Lexicographic				√	2	1	2
Weighted Min-Max			√	weak Pareto opt.	1	1	2
Exponential Weighted	√		√	√	0	1	0
Weighted Product	√		N/A	√	0	1	1
Goal Programming		√			1	1	2
Bounded Obj. Function		√			1	1	1
Physical Programming			√	√	3	3	1

For weighted criterion method: $U = \sum_{i=1}^{k} \omega_i [F_i(x)]^P$, it requires a relatively large value of P to capture certain Pareto optimal points especially with non-convex Pareto optimal sets. As P is approaching infinity, this minimization is no longer sufficient for Pareto optimality; but only sufficient for weak Pareto optimality. For other weighted methods, a priori selection of weights does not necessarily guarantee the final solution will be acceptable; the weight must be a function of original objectives instead of constant in order for a weighted sum to mimic a preference function accurately. However, varying the weight weights constantly and continuously may not necessary result in an even distribution of Pareto optimal points and an accurate, complete representation of the Pareto optimal set. With physical programming, the decision-makers need to specify a relatively large amount of information, which can be viewed as a hindrance or as an opportunity. With relatively complex preferences, one must provide more information for physical programming. Then, the more information one provides, the more accurately preferences are represented.

Thus we pick up lexicographical method for our sensor deployment optimization; now let us prove that lexicographical method won't change the optimality on our sensor deployment scenario.

Proof:

Definition: Pareto optimal which is defined as: a point, $x^* \in X$ is Pareto optimal if and only if there does not exist another point, $x \in X$ such that $F(x) \le F(x^*)$, and $F_i(x) \le F_i(x^*)$ for at least one function [87]. In another word: when a system is under Pareto optimal already, given a "Pareto optimal" allocation of resources among a set of individuals, a change to a different allocation that makes at least one individual better off will make any other individuals worse off.

According to lexicographic method's assumption [87], objective functions are first arranged in the order of their importance. After ordering, the most important objective will be firstly optimized subject to the original constraints. If the problem has the only solution at this moment, this optimized solution is the solution to the whole multiple-objectives optimization problem. Otherwise, the second most important objective will be optimized. However, when optimizing the second objective, we need to add a new constraint that guarantees the first objective function preserves its optimal value in addition to the original constraints. If this problem has the unique value on the second round optimization, it is the solution of the original problem. Otherwise, the process has to be iterated as above until going over the whole set of objectives.

Firstly, suppose we had arranged the objective functions in the order from the most important F_1 to the least important F_k, and then the lexicographic problem can be re-written as

$$\begin{aligned} &Min : F_1(x), F_2(x), \cdots, F_k(x) \\ &Subject \quad to : x \in s \end{aligned} \qquad (3\text{-}11)$$

Here $F_1(x), F_2(x), \cdots, F_k(x)$ are the unobservability and cost functions in (3-10). We have previously proved that the unobservability function and cost function are monotonic on sensor numbers. Assume that $x^* \in s$ is a solution to the lexicographical multiple-objective optimization problem but not Pareto optimal, then there should exist another $x \in s$ such that $F_i(x) \leq F_i(x^*)$ for all $i = 1,2,\cdots,k$, and there exists at least one j which strictly has $F_j(x) < F_j(x^*)$.

As aforementioned lexicographic optimization process, there are two possible situations when determining the optimized solution: 1) the unique solution can be found during the optimization process, or 2) optimizations have to be performed for every objective function $F_i(x), i = 1,2,\cdots,k$.

1) If the unique solution x^* to F_i can be found by lexicographic method before examining every objective function. The assumption $F_i(x) \leq F_i(x^*)$ and the fact that objective functions in (3-10) are all monotonic functions imply that $F_i(x) = F_i(x^*)$ is the only option, which is a contradiction with at least one i that has $F_i(x) < F_i(x^*)$. Thus x^* is Pareto optimal, and there does not exist another point, $x \in s$ such that $F_i(x) \leq F_i(x^*)$ for at least one i.
2) If optimizations have to examine every objective function from $F_1(x)$ to $F_k(x)$, when $i = 1$, since lexicographic ordering requires that $F_1(x)$ reaches minimum at x^*, we will have $F_1(x) \leq F_1(x^*)$, we also know $F_1(x)$ is monotonic, at this moment $F_1(x) = F_1(x^*)$ is the only possible situation. Using the similar reasoning we have $F_i(x) = F_i(x^*)$ for every $i = 2, \cdots, k$. This contradicts the assumption that at least one objective function is strictly inequal. Thus x^* is also Pareto optimal.

So for the sensor deployment problem, the lexicographical optimization to optimize the objectives in an ordered sequential way won't change the optimality of the original problem.

Now we know that we can optimize the multiple objectives problem in sensor deployment in an ordered lexicographical MILP manner. There are several ways to solve a MILP problem. One is linear programming (LP) relaxation, which uses linear programming to solve the integer programming problem [88]. It attempts to use the approximate procedure of simply applying the simplex method to the LP relaxation and then rounding the noninteger values to integers as the resulting solution. However, an optimal linear solution is not necessarily feasible after the rounding. Even if an optimal solution for the LP relaxation is rounded successfully, there is still no guarantee that this rounded solution will be the optimal integer solution. Because of these, another common approach called "branch and bound" was developed for integer programming. It systematically enumerates all candidate solutions, then discards large subsets of fruitless candidates, by using upper and lower estimated bounds of the quantity being optimized. For the branch and bound method, when listing a finite number of feasible solutions ensures that the problem is readily solvable, finite numbers usually increase exponentially. The key to the remarkable efficiency of the branch and bound method lies in removing some feasible solution from a linear programming problem which will make it easier to solve. However, this efficiency condition is not always satisfied especially when we try to solve multiple-objective optimization sequentially. Because of these, a better approach for dealing with multiple objective IP problems that are too large to be solved exactly is to use one of the available heuristic algorithms such as greedy search. They tend to be considerably more effective in finding good feasible solutions.

3.8.2 Greedy algorithm

To reduce the optimization load, we also tackled this optimization problem with greedy algorithm, which iteratively adjusts the cover (R, S) until the subgraph $G_{R,S}$ has an optimized matching. The detail of the greedy algorithm is shown as below:

1) Initialize those variables: current solution ($CS = 0$), optimal solution ($OS = 0$), total available cost (TC), cost used ($UC = 0$) and current available cost (AC).

2) Sort the fault nodes in descending order according to the fault nodes' unobseravbility values;
3) Since the primary object is to lower the maximum unobseravbility, pick the fault node with the highest unobservability, because such a selection will lead to the greatest reduction on unobservability. If there exists several fault nodes with the same unobservability, choose the one with the highest RPN value, which means this fault is the most critical;
4) Sort the sensor nodes that have connection strength to the selected fault node;
5) Pick up the sensor node that has the largest connection with the fault node;
6) If more than one sensor node satisfy the requirement, pick up the one with the lowest cost;
7) Update the variables with $CS = CS \cup S_j, UC = UC + C_j$;
8) Recalculate the detectability for the updated system and maximum unobseravlity as equation (3-9) for the sensor remainders;
9) Check the constraints requirement with the updated detectability;
10) If it satisfies all the constraints, stop the search. Otherwise, go back to repeat step 2) to 10) iteratively.
11) The condition that will terminate the procedure is based on the detectability criteria: if the selected sensor set can satisfy the minimum detectability request. We will stop the search on sensor set and the set of sensors in OS is the selected sensor set for the diagnosis purpose.

3.9 Case studies

In order to illustrate the proposed sensor deployment approach on diagnosing manufacturing system, two case studies, one for continuous manufacturing and the other for discrete manufacturing system, were developed respectively.

3.9.1 Application on continuous manufacturing system

3.9.1.1 Graph model on the continuous manufacturing system

A five-tank system (the data is available from Zhang [11]) is employed to demonstrate the proposed sensor deployment approach on continuous manufacturing system. The five tank system is composed of piping, valves, pumps and reservoirs in the facility as shown in Figure 7(a). In Zhang's publication [11], he assumed that all the fault nodes have the same RPN values, so their fault nodes are normalized as [1] in the fuzzy graph the directed graph for the five-tank system is shown as below Figure 7(b). Table 9 and 10 summarized the sensing gain/time, and sensor properties to determine the detectability of sensors to faults. After transform the directed graph into the bipartite graph, the incidence matrix or the connection matrix between the sensors and the faults is shown as Table 11.

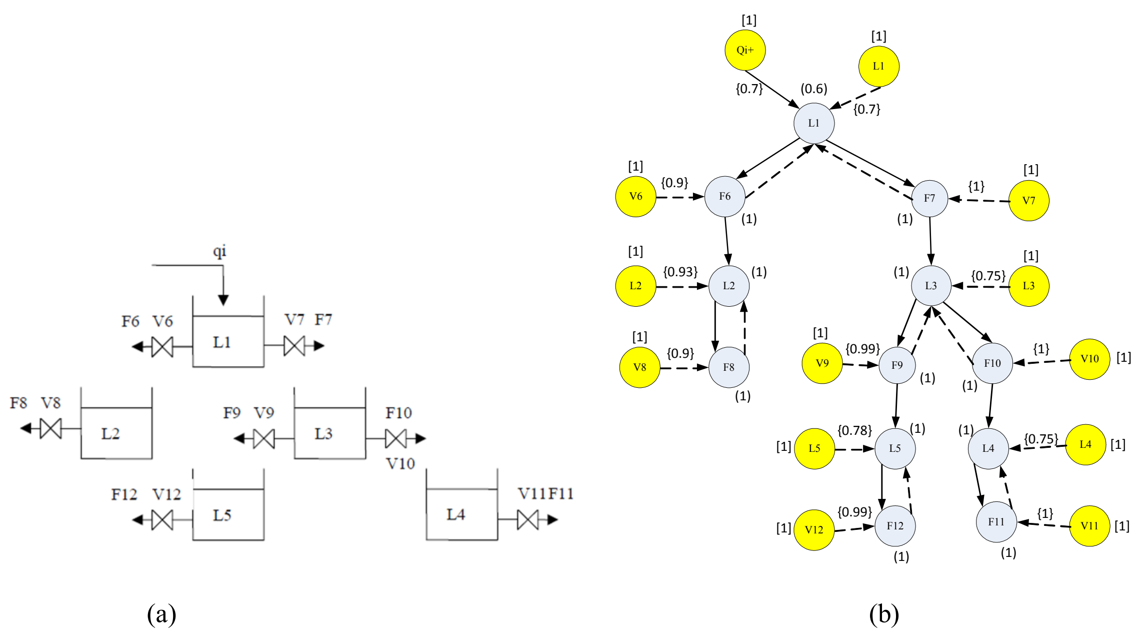


Figure 7 (a) A five tank system, (b) quantitative directed graph between faults and sensors

Table 9 Sensing gain and time to determine the fault detectability [11]

Sensor		Sign	Sens. gain	Sens. time	Sensor		Sign	Sens. gain	Sens. time
From	To				From	To			
F-L1**	L1	-	0.65	1/100	F-L2	L2	-	24.4	4/100
F-L3	L3	-	0.14	1/100	F-L4	L4	-	1.1	4/100
F-L5	L5	-	5.34	4/100	F-V6	F6	-	19.8	0
F-V7	F7	-	99	0	F-V8	F8	-	19.8	0
F-V9	F9	-	44	0	F-V10	F10	-	55	0
F-V11	F11	-	55	0	F-V12	F12	-	44	0
F-Qi+	L1	+	0.34	2/100					
L1	F6	+	5	0/100	F6	L1	-	0.03	2/100
F6	L2	+	0.8	12/100	F7	L3	+	0.01	21/100
L2	F8	+	1.25	0/100	F8	L2	-	0.95	4/100
L1	F7	+	25	0/100	F7	L1	-	0.04	1/100
L3	F9	+	36	0/100	F9	L3	-	0.01	1/100
F9	L5	+	0.2	81/100	F10	L4	+	0.06	20/100
L5	F12	+	5.05	0/100	F12	L5	-	0.23	4/100
L3	F10	+	45	0/100	F10	L3	-	0.01	1/100
L4	F11	+	16.1	0/100	F11	L4	-	0.07	4/100

Table 10 Sensor information about the five tank system [11]

Sensor	Cost	SNR	Pr	Resolution
L1	100	6	0.001	0.01
L2	100	10	0.001	0.01
L3	100	10	0.001	0.01
L4	100	10	0.001	0.01
L5	100	10	0.001	0.01
F6	100	10	0.001	0.01
F7	100	10	0.001	0.01
F8	100	10	0.001	0.01
F9	100	10	0.001	0.01
F10	100	10	0.001	0.01
F11	100	10	0.001	0.01
F12	100	10	0.001	0.01

Table 11 d_{ij} connection matrix for between sensors and faults in five tank system [11]

Sensor Fault	L1	F2	L3	L4	L5	F6	F7	F8	F9	F10	F11	F12
F-L1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
F-L2		-1						-1				
F-L3			-1	-1	-1				-1	-1	-1	-1
F-L4				-1							-1	
F-L5					-1							-1
F-V6	+1	-1	+1	+1	+1	-1	+1	1	+1	+1	+1	+1
F-V7	+1	+1	-1	-1	-1	+1	-1	+1	-1	-1	-1	-1
F-V8		+1						-1				
F-V9			+1	+1	-1				-1	+1	+1	-1
F-V10			+1	-1	+1				+1	-1	-1	+1
F-V11				+1							-1	
F-V12					+1							-1
F-Qi+	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1

With the sensor, fault information, the aggregate them into a single using the aforementioned AHP approach, the connection matrix representing the detecting ability of sensor j to fault$_i$ is shown as Table 12.

Table 12 p_{ij} connection matrix for between faults and sensors in five tank system

	L1	L2	L3	L4	L5	F6	F7	F8	F9	F10	F11	F12	Un	Order
F-L1	0.54	0.67	0.67	0.67		0.68	0.72	0.67	0.67	0.67	0.67		-32	10
F-L2		0.69						0.69					-8	1
F-L3			0.67	0.67	0.67				0.68	0.69	0.67	0.67	-23	9
F-L4				0.67							0.68		-8	2
F-L5					0.67							0.69	-8	3
F-V6	0.53	0.67	0.67	0.67		0.73	0.68	0.67	0.67	0.67	0.67		-32	11
F-V7	0.54	0.67	0.67	0.67		0.70	1.00	0.67	0.67	0.67	0.67		-32	12
F-V8		0.68						0.73					-8	4
F-V9			0.67	0.67	0.67				0.81	0.70	0.67	0.67	-23	7
F-V10			0.67	0.67	0.67				0.70	0.85	0.68	0.67	-23	8
F-V11											0.85	0.67	-8	5
F-V12					0.67							0.81	-8	6
F-Qi+	0.53	0.67	0.67	0.67		0.67	0.68	0.67	0.67	0.67	0.67		-32	13

3.9.1.2 Results on continuous manufacturing system and discussions

Zhang's SDG model cannot include the quantitative sensor-fault information in the model, so we used the fuzzy graph to improve the existing work. In the fuzzy graph based optimization, lexicographic and greedy optimizations were used to optimize the model in equation (3-10) with the quantitative connection matrix information. The outcome performance comparisons between the SDG and fuzzy graph sensor deployment follow the flow chart in Figure 8.

Table 13 summarized the comparisons on performance between SDG and fuzzy graph on sensor deployment. In this table, the second row is the optimization outcome based on signed directed graph (SDG), which was applied by [11] to optimize the sensor deployment on the five-tank system. The third and fourth rows are the results based on fuzzy graph with lexicographical and greedy optimization respectively. We compared the SDG and fuzzy graph's performance on deploying sensor to diagnose faults from the aspects of: total number of selected sensors, total cost, minimum detectability among all faults, and average detectability among all faults. The definition of minimum detectability and average detectability were defined as below equation (3-12) and (3-13) respectively.

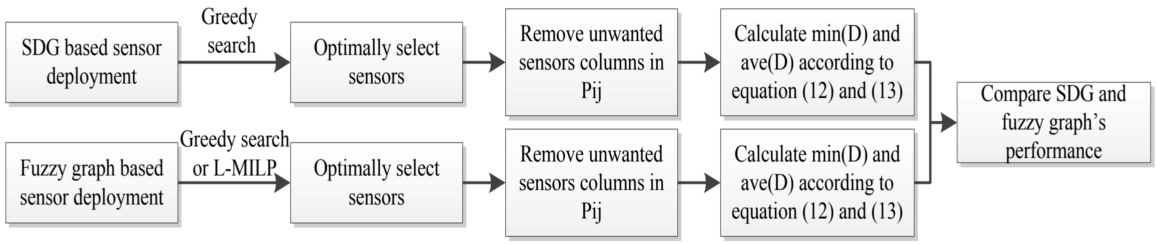

SDG based sensor deployment
Greedy search
Optimally select sensors
Remove unwanted sensors columns in Pij
Calculate min(D) and ave(D) according to equation (12) and (13)
Fuzzy graph based sensor deployment
Greedy search or L-MILP
Optimally select sensors
Remove unwanted sensors columns in Pij
Calculate min(D) and ave(D) according to equation (12) and (13)
Compare SDG and fuzzy graph's performance

Figure 8 Comparisons between SDG and fuzzy graph on sensor deployment

The Minimum detectability among all faults is defined as:

$$\min(D) = \min_{\forall i}\left[\max_{j \in selected\ sensor}\left[p_{ij}\right]\right] \qquad (3\text{-}12)$$

The average detectability among all faults is defined as:

$$ave(D) = average_{\forall i}\left[\max_{j \in selected\ sensor}\left[p_{ij}\right]\right] \qquad (3\text{-}13)$$

From Table 13, we can see that the minimum detectability on the five-tank jumped from SDG's 0 to fuzzy graph's 0.67, and average detestability also increased from SDG's 0.62 to fuzzy graph's 0.70. Thus we conclude that with the inclusion of quantitative information, fuzzy graph based sensor deployment modeling greatly enhanced the diagnosing systems' detectability to faults than signed directed graph approach.

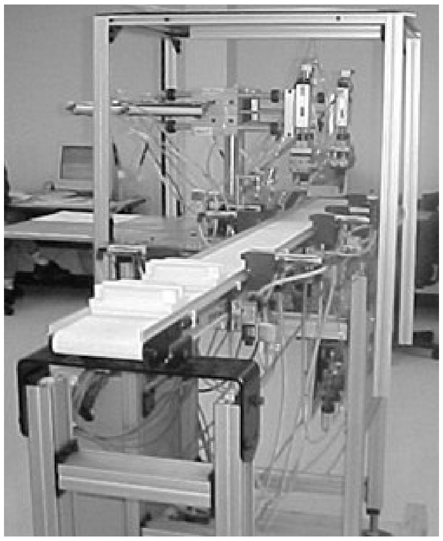
Table 13 Results for the five tank system

Approach	**Sensors selected**	**# of selected sensors**	**Total cost of selected sensors**	**Minimum detectability among all faults**	**Average detectability among All faults**
Signed directed graph [11]	[L2, L4, L5]	3	300	0	0.62
Fuzzy graph with lexicographical	[F8,F11, F12]	3	300	0.67	0.70
Fuzzy graph with greedy	[F8, F11,F12]	3	300	0.67	0.70

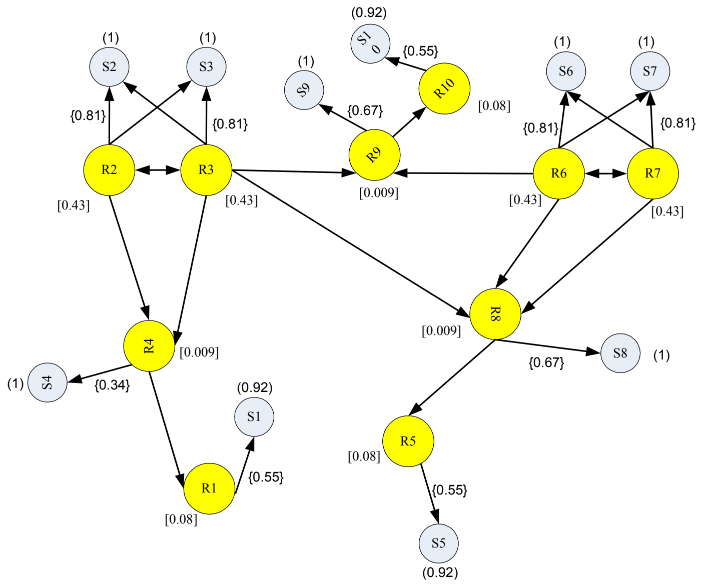
The SDG method only cares the single attribute that whether there is a connection between the fault and sensor, but not considering how the connection it is, for example, the strength of the connection. The fuzzy graph included quantitative information such as sensor properties, fault modes and sensor-fault relations into the connection. These attributes are heterogeneous and not comparable, so we used fuzzy membership function to normalize them into values in [0, 1], then aggregate them into single edge values with AHP method. In the SDG approach, the aforementioned quantitative sensor-fault relation cannot be included in the modeling. Thus when allocating the sensors, it is just the problem of choosing the minimum number of sensors key components that would cover all the root nodes. Thus SDG's optimization always tries to choose the sensor node that has the most edge connections with the faults. After including the quantitative information in fuzzy graph, its optimization algorithm will pick up the sensor that has the largest edge connection with the fault at each time. Here we included the SDG based sensor deployment in the paper appendix.

3.9.2 Application on discrete manufacturing system

The other case study for discrete manufacturing system was developed on a dual robot in Rockwell® Automation System Integration Laboratory at Texas A&M University. The dual robot is as shown in below Figure 9 (a) [89]. It is a robot work cell including conveyor, two 4-axis robot arms, stoppers, parts feeders, computer vision and controller. The robot arm1 is composed of shoulder, elbow, wrist and gripper. The sequential operations for the robot arm are: 1) stopper block the pallet and send signal to initiate arm; 2) open gripper, lower elbow, pick up part from part feeder; 3) raise elbow, extend the shoulder, lower elbow, open gripper; and 4) close gripper, raise elbow, retract shoulder. These actions were controlled by Allen Bradley® programmable logic controller SLC 5/05 with other modules. Due to space limitation in the paper, we only concentrate on deploying sensors to diagnose faults with robot arm1.



(a)



(b)

Figure 9 (a) An automated assembly dual robot (b) initial fuzzy graph for dual robot

3.9.2.1 Graph model on the discrete manufacturing system

Arm1 includes shoulder, elbow and gripper which are mostly composed of pneumatic control devices such as air cylinders, solenoid valves; and Hall sensors. A simplified FMEA on arm1 is shown as Table 14. Literature [90] and [91] illustrated that the most common failures with solenoid valves were caused by either overpowering and eventual overheating of the valves, or wearing out of the valve components; while the failures with cylinder were due to leakage and cylinder stuck. Thus the promising sensor candidates to detect those faults can be pressure sensors and voltage current checking.

Table 14 Faults and sensors in dual robot arm

Component	Fault Node	Possible Fault	Occurrence Rate	Severity Rate	Detection Rate	Sensor Candidates
Shoulder	R1	Cylinder	6	7	2	pressure sensor (S1)
	R2	Hall sensor1	2	1	5	voltage sensor (S2)
	R3	Hall sensor2	2	1	5	voltage sensor (S3)
	R4	Solenoid	6	8	9	voltage sensor (S4)
Elbow	R5	Cylinder	6	7	2	pressure sensor (S5)
	R6	Hall sensor1	2	1	5	voltage sensor (S6)
	R7	Hall sensor2	2	1	5	voltage sensor (S7)
	R8	Solenoid	6	8	9	voltage sensor (S8)
Gripper	R9	Solenoid	6	8	9	voltage sensor (S9)
	R10	Cylinder	6	7	2	pressure sensor (S10)

Table 15 Factors to determine fault detectability

Sensors to detect faults		Sensitivity (full scale)	SNR (dB)	Resolution (bit)	Accuracy (full scale)	sensor fail rate (Pr)	Sens. gain	Sens. time (ms)	Cost ($)
Fault node	**Sensor node**								
R1	**S1**	2.5%	20	12	3%	0.001	1	4.6	30
R2	**S2**	1%	100	16	1%	0.001	1	3	80
R3	**S3**	1%	100	16	1%	0.001	1	3	80
R4	**S4**	1%	100	16	1%	0.001	1	3	80
R5	**S5**	2.5%	20	12	3%	0.001	1	4.6	30
R6	**S6**	1%	100	16	1%	0.001	1	3	80
R7	**S7**	1%	100	16	1%	0.001	1	3	80
R8	**S8**	1%	100	16	1%	0.001	1	3	80
R9	**S9**	2.5%	20	12	3%	0.001	1	4.6	80
R10	**S10**	1%	100	16	1%	0.001	1	3	30

Table 15 listed the factors that will determine the fault detectability. After normalization of the sensor nodes, fault nodes as well as edges, the bipartite graph for the sensor deployment is initially shown as Figure 9(b), and the bipartite connection matrix p_{ij} with the graph are also listed with Table 16.

Table 16 p_{ij} Connection matrix for initial fuzzy graph

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	Unobservability	RPN	Order
R1	0.55										-6	0.083	1
R2		0.81	0.81	0.34				0.67	0.67		-17	0.43	9
R3		0.81	0.81	0.34				0.67	0.67		-17	0.43	10
R4	0.55			0.34							-9	0.009	4
R5					0.55						-6	0.083	2
R6						0.81	0.81	0.67	0.67		-14	0.43	7
R7						0.81	0.81	0.67	0.67		-14	0.43	8
R8					0.55			0.67			-9	0.009	5
R9									0.67	0.55	-9	0.009	6
R10										0.55	-6	0.083	3

3.9.2.2 Results on discrete manufacturing system and discussions

With this bipartite graph and connection matrix p_{ij}, we started to optimize the sensor deployment using lexicographical mixed integer linear programming (L-MILP) and greedy algorithm respectively. The L-MILP and greedy optimization results are listed in Table 17 and 18 respectively. It illustrated how to choose sensor deployment including location, number and type in order to satisfy detectability requirements. In Table 17, when required detectability constraint was 0.1, L-MILP predicted that using 5 sensors with cost of 250 could achieve minimum unobservability of -5. These five selected sensors locate at the end of the graph (S1, S3, S5, S7, S10 in Figure 9 (b)). This is consistent with the fault identification reasoning: these three sensors (S1, S5, and S10) are for monitoring the solenoids, which drive cylinders with the most observable actions for the troubleshooting. Thus in order to maintain the minimum detectability, the sensors to the solenoid shall be considered first. S3 and S7 are for monitoring control input signals (A1XH3 and A1ZH2), so they are also selected for better detectability. Later when required detectability constraints increased to 0.8, L-MILP predicted that it needed 10 sensors with cost of 500 to achieve minimum unobservability of -5. In Table 18, when required detectability constraint was 0.1, greedy predicted that 5 sensors with cost of 250 could achieve minimum unobservability of -5. Later when required detectability constraints increased to 0.8, greedy predicted that it needed 8 sensors with cost of 340 to achieve minimum unobservability of -5.

Table 17 Optimization results using L-MILP

Required Detectability	Sensor selected	Cost utilized	Achieved unobservability	Total Sensor number	Optimization Time (Seconds)
0.1	S1, S3, S5, S7, S10	250	-5	5	0.01
0.2	S1, S3, S5, S7, S10	250	-5	5	0.02
0.3	S1, S3, S5, S7, S10	250	-5	5	0.02
0.4	S1, S2, S5, S6, S10	250	-5	5	0.03
0.5	S1, S2, S5, S6, S10	250	-5	5	0.03
0.6	S1(2), S2, S5, S7(2), S10(2)	390	-5	5	0.02
0.7	S1(2), S2(2), S5(2), S6(2), S10(2)	500	-8	10	0.03
0.8	S1(2), S2(2), S5(2), S6(2), S10(2)	500	-8	10	0.03

Table 18 Optimization results using greedy algorithm

Required Detectability	Sensor selected	Cost utilized	Achieved unobservability	Total Sensor number	Optimization Time (Seconds)
0.1	S1, S2, S5, S6, S10	250	-5	5	0.0050
0.2	S1, S2, S5, S6, S10	250	-5	5	0.0050
0.3	S1, S2, S5, S6, S10	250	-5	5	0.0058
0.4	S1, S2, S5, S6, S10	250	-5	5	0.0049
0.5	S1, S2, S5, S6, S10	250	-5	5	0.0049
0.6	S1, S2, S5, S6, S10	250	-5	5	0.0049
0.7	S1(2), S2, S5(2), S6, S10(2)	340	-5	8	0.0050
0.8	S1(2), S2, S5(2), S6, S10(2)	340	-5	8	0.0049

Comparisons were conducted from the perspectives of cost and detectability on the system. Cost of the sensor system is simply summing all the sensor costs involved in the deployment. The unobservability of the system follows the equation (3-12) and (3-13)'s definition.

Table 19 Results comparison for the dual robot system

Approach	Sensors selected	# of selected sensors	Total cost of selected sensors	Minimum detectability among all faults	Average detectability among all faults
Signed directed graph	[S1, S5, S9, S10]	4	170	0.55	0.61
Fuzzy graph with lexicographical	[S1, S3, S5, S7, S10]	5	250	0.55	0.654
Fuzzy graph with greedy	[S1, S2, S5, S6, S10]	5	250	0.55	0.654

From Table 19, further consideration is that if we pick up any one more sensors in addition to the current SDG optimization result, how the optimization comparison will be looked like? The newly achieved minimum detectability and average detectability for adding one more sensor are summarized in the Table 20. From the table, we can see that the SDG approach still cannot have better performance than the proposed approach even adding an additional sensor to the current optimized sensor selection, because the proposed approach selected the sensor of [S1, S3, S5, S7, S9] and achieved minimum detectability of 0.55 and average detectability of 0.654 (Table 19).

Table 20 Summarization of results on adding any another sensor

Newly added sensor	New sensor configuration	Minimum detectability	Average detectability
S2	**[S1, S2, S5, S9, S10]**	0.55	0.638
S3	**[S1, S3, S5, S9, S10]**	0.55	0.638
S4	**[S1, S4, S5, S9, S10]**	0.55	0.598
S5	**[S1, S5, S6, S9, S10]**	0.55	0.638
S6	**[S1, S5, S7, S9, S10]**	0.55	0.638
S8	**[S1, S5, S8, S9, S10]**	0.55	0.61

Articles have mentioned that including redundant sensors will improve the performance of fault detection [92]. However due to the budget or space installation constraint, it may not be possible to install every redundant sensor. In literature [92], it noted that that in a nuclear plant, adding extra sensor will impact the maintenance, sometimes only with an average variation of 0.35% on the optimization results, it can still make some difference on the efficiency.

We also compared the optimization approaches with the reasoning process and found that the selection of sensor for each fault node is based on the biggest path connection due to the greedy principle. Only one sensor either S2 or S3 is kept in the sensor system. Possible reason for this deletion is that: sensor nodes S2 and S3 monitor fault nodes R2 and R3, which are the hall sensors A1XH2 and A1XH3's abnormalities. Those two Hall sensors are the input signals to activate the actions of shoulder. A1XH1 and A1XH2's on/off states are opposite to each other ($A1XH2 = \overline{A1XH3}$). Thus the faults R3 can be inferred from the signals related on S2. Besides this, R2 and R3 are the least crucial (with the unobservability order of 9 and 10, and unobservability value of -17) in Table 12, so only one S2 kept in the optimized sensor system will still guarantee the detectability. The selection of S6 or S7 can follow the similar reason as selecting S2 and S3.

We compared the offline sensor selection computation cost in terms of the computation time. The greedy search was programmed in Matlab® and MILP in LINGO®. The computation was performed on a 2.7G dual core computer with memory of 6.0G RAM. The computation time is shown as Figure 10. We can see that greedy search is more computation efficient that the lexicographical integer programming. Possible reason may be like this: L-MILP was carried out to solve the optimal objectives one by one; branch and bound is used as the mechanism to find the integer solution. This recursion process hinders the computation efficiency. However, greedy algorithm selects the solutions that target the objectives directly. It may be an efficient approach for dealing with mixed integer linear problem that is too large to be solved exactly, especially when there is some experience knowledge to guide the search.

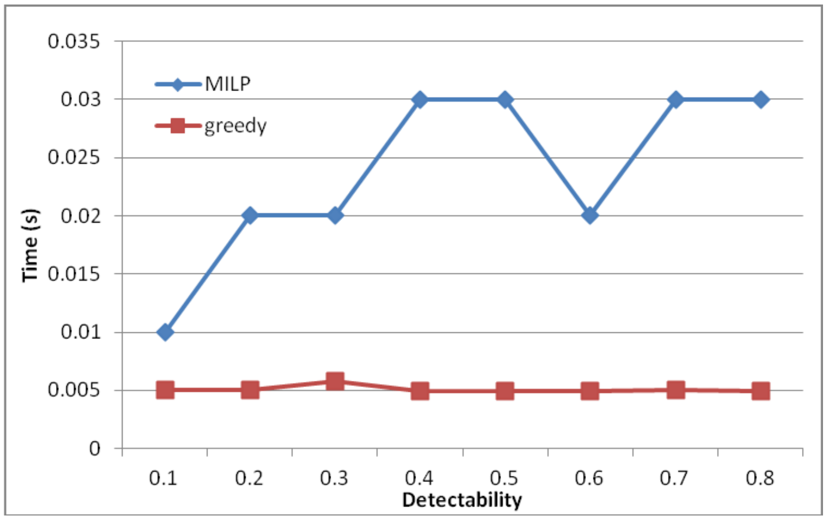


Figure 10 Sensor selection computation comparison

3.10 Conclusion

Optimal sensor deployment is an important research issue for monitoring and detecting faults in that the accurate and efficient collection of fault signal signature is the very first step for diagnostics. The objective of this chapter is to develop a fuzzy quantitative graph based sensor deployment methodology which can process heterogeneous properties and multiple-objective optimization in fault diagnosis. In the proposed methodology, sensor deployment started with studies on failure mode effect analysis (FMEA) on the manufacturing system, which specified the failure modes and their criticality and probability of occurrence. Quantitative fuzzy graph was used for modeling cause effect between faults and sensors. The element values in the fuzzy graph were decided with the considerations on fault characteristics, sensor properties, as well as sensors' detectabilities to faults. To handle the trade-off caused by heterogeneous properties among sensors, AHP was applied to aggregate those properties into a single edge value. With the fuzzy bipartite graph, sensors were optimally assigned to faults under the help of lexicographical mixed integer linear programming and greedy algorithm respectively. Finally case studies on how to deploy sensors to detect faults on continuous and discrete type manufacturing systems were presented to illustrate the proposed design. Various issues discussed in the methodology were demonstrated in the case studies; results from different modeling and optimization approaches were compared and discussed also. The case study validated that the proposed fuzzy graph based methodology can greatly enhance the detectability to faults than SDG based sensor deployment (from SDG's 0.62 to fuzzy graph's 0.70 in the five-tank application and from SDG's 0.61 to fuzzy graph's 0.654 in the dual-robot application). The contribution of our work lies on that the optimized sensor deployment approach can include kinds of heterogeneous quantitative information to direct the multiple-objective decision in sensor deployment, thus achieving the minimum unobservability and cost under the constraints of detectability, resources and uncertainties. The fuzzy graph based sensor deployment is systematic and can be easily integrated into diagnosis architecture to detect faults in complex systems such as flexible manufacturing systems (FMS), HVAC and semiconductor production facilities.

CHAPTER IV

PROPOSED DIAGNOSER DESIGN BASED ON REALTIME FUZZY PETRI NET

To efficiently detect the stochastic faults in automated assembly system, considering integration of industrial experience (useful but with uncertainty) into the diagnoser design, we proposed a realtime fuzzy Petri net (RTFPN) diagnoser architecture as shown in Figure 11. This architecture includes a realtime PN model (Q, M_0), and a fuzzy PN diagnoser (Q_d, M_0). The realtime model keeps checking the input and output of the PN plant, and compares them with the pre-settings. Once a difference is detected, it will start the fuzzy Petri net diagnoser to locate the root cause of the fault. The details of this diagnosis architecture will be specified in the following sections.

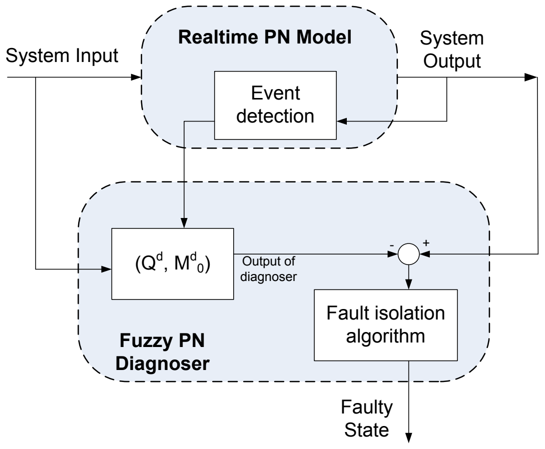


Figure 11 Fault diagnosis based on realtime fuzzy Petri net for automated system

4.1 Realtime Petri net model

The procedure of constructing RTPN model to monitor discrete manufacturing systems is as following [79]:

1) Model the control sequence using PN to obtain the model of the sequence controller.
2) Formulate an input mapping table between the system's input devices such as limit switches, sensors with the places in the PN model. The initial states of the system decide the initial marking of RTPN. Identify the timing information for activities to formulate the later mapping table.
3) Assign output channels to the system's output devices such as solenoids, relays. Also, identify timing information for activities to formulate an output mapping table.
4) Using output mapping table and the actions that are modeled by a transition, assign a number to each transition in a PN based controller. The operations and time delays given in the sequence to be controlled decide firing time function of RTPN.

Following the above procedures, Petri net model is defined as $Q = \{P,T,I,O,W,M_0,X,Y,h\}$, where: $P = \{P_1,P_2,\cdots,P_m\}$ is the finite set of places. $T = \{T_1,T_2,\cdots,T_n\}$ is the finite set of transitions, these transitions are classified as observable transitions (T_o) or unobservable transitions (T_{uo}), so we have $T = T_o \bigcup T_{uo}$ and $T_o \bigcap T_{uo} = \phi$. The observable transitions can be indicated by sensors on whether a transition has been fired or not. In contrast, the unobservable transitions cannot be observed directly given the current sensor configuration. The association between sensors and transitions are linked by the labeling function: $LA: T^* \rightarrow \Sigma^*$, which will be defined later.

In our PN model, faults (T_F) were modeled as a subset of unobservable transitions and partitioned into m independent types, so we have (1) $T_F \subseteq T_{uo}$, (2) $T_F = T_{F1} \bigcup T_{F2} \bigcup \cdots \bigcup T_{Fm}$, and (3) $T_{Fi} \bigcap T_{Fj} = \phi$ if $i, j \in \{1,2,\cdots,m\}$ and $i \neq j$. This model includes both normal and faulty behavior of the system; fault can happen at any state of the normal operation. $I: P \rightarrow T$ is the pre-condition of transition which is the finite set of arcs from places to transitions; $O: T \rightarrow P$ is the post-condition of transition which is the finite set of arcs from transitions to places; M_0 is the finite set of initial token markings; and W is the incidence matrix and $W = O - I$; $X: P \rightarrow \{0,1,2,\cdots k\}$ and $X(p_i) \neq X(p_j), i \neq j$, is the input signal vector; Y is the output signal vector. h is the mapping function between input/output devices and places/transitions. It is noted that:

Input signal vector (X) reads the state of input signals from digital input interface. X associates attributes with every transition. $X_i = h(t_i)$ is the attribute associating with transition t_i.

Output signal vector (Y) intends to send output signals through digital output interface. Y also associates attributes to every place. $Y_i = h(p_i, t_i)$ is the attribute associating with transition t_i and place p_i which represents the number that is to be sent to the digital output interface.

With the realtime PN model structure, we further build an integrated I/O mapping table to describe the relation between the PN model and the input/output devices. This table integrates different combinations of input sensor readings and output status, so it can help locate which place is achieved and which may lead to the faults according to the location of the process. The mapping table is built in this way: let $R(Q,M_0)$ be the reachable set from initial marking, so we will refine the model to remove the states that cannot be reached, thus we have $\forall p_i : p_i \in P \rightarrow M(p_i) \in R(Q,M_0)$. Then the input/output function $h(M_i) := P_i \rightarrow (X_i, Y_i)$ is set up for each possible input/output. h associates each marking in $R(Q,M_0)$ with input/output vector with sensors, the timing information of each I/O are also integrated in the table.

Finally, the state equation for the realtime Petri net can be calculated as:

$$\begin{cases} M_{i+1} = M_i + u_i W \\ (X_i, Y_i) = h(p_i, t_i, M_i) \end{cases} \tag{4-1}$$

Where M_i is the marking of plant; u_i is the firing vector to indicate which transition is currently fired; h is the mapping table between the discrete set of possible inputs/outputs (X_i, Y_i) each marking (p_i, t_i, M_i) in the reachable set $R(Q,M_0)$.

4.2 Diagnoser design

In automated assembly system, faults maybe caused by design errors, manufacturing defects, users or programs that do not follow the protocols, component aging/deterioration etc. Because each faulty event in the manufacturing system has a degree of uncertainty, which can be expressed as the IF-THEN rule. These events will form a possible set of place/transition modes described by the fuzzy Petri net. We can model the faulty events by PN as: 1) a transition is fired, but the tokens are neither added nor removed to the respective input or output places of that transitions; or 2) a transition is fired, but the tokens are moved to the incorrect output places of the transition. In order to detect these two kinds of fault situations, the fuzzy PN diagnoser $Q_d = (P_d, T_d, I, O, \alpha, \beta, l)$, which has the ability to incorporate uncertainties in event detection, is used to determine the fault mode by detecting events signifying a switch in the sequence of operation. The fuzzy Petri net consists of two parts: the structure of Petri net $\{P_d, T_d, I, O\}$ and fuzzy logic reasoning $\{\alpha, \beta, l\}$ associated with each transition. The following sections will present how to model the Petri net structure and fuzzy reasoning.

4.2.1 Petri net structure

The Q_d observes the RTPN mode Q's input/output signals which represent the running status of the manufacturing plant in realtime. In this diagnoser, P_d is the set of places which relates to the fault partitions, some places may contain a token marked with a fuzzy truth value between 0 and 1, and some places may not, which are the same as the ones in ordinary Petri net. T_d is the set of faulty transitions and belongs to the unobservable transitions. $LA: T^* \rightarrow E^*$ is the transition labeling function; E is the event set for transition labeling, which is a set of fault labels which are defined with

$\Delta f = \{f_1, f_2, \cdots, f_m\}$ and normal status is as N. Then all possible set of labels with diagnoser is $E = \{N\} \cup \Delta f$. For every place of diagnoser it has the form (P_{di}, E_i) and the initial place is $(P_0, \{N\})$. The language generated from the Petri net diagnoser is:

$$L(Q_d) := \{l(s) \in E^* : s \in T^* \; with \quad initial \quad state\} \tag{4-2}$$

Suppose that the system starts from normal status with the initial marking, when an observed marking is changed, a new diagnoser state is generated. Let $q \in Q_d$ be a diagnoser state, which in general consists of pairs of markings and a label concerning whether a fault has occurred. In notations, for each diagnoser state $q \in Q_d$,$Q_d \subseteq 2^{R(M_0) \times \Delta}$, it takes the general form of: $q = \{(M_1, l_1), \cdots, (M_n, l_n)\}$, of which all of the markings in the estimate pairs of marking and label look the same and belong to the same equivalent class, $M_i \in R(M_0)$ and each label $l_i \in l$. Then labels are assigned to that failure by fault labeling function. Suppose s is a string that starts from state x_0 and ends in an observable event, it forms a sequence of firing transitions $s = M_0 t_1 M_1 t_2 \cdots t_n M_n$. Labeling function LA propagates the label l over s. Thus we have $LA : R(M_0) \times l \times s \to l$, LA assigns the label l over s starting from P following the state transition of Q_d as:

$$LA(P,l,s) = \begin{cases} \{N\}, & \forall i \left[T_{f_i} \notin s\right] \\ \{f_i\}, & \forall i \left[T_{f_i} \in s\right] \end{cases} \tag{4-3}$$

The fault labeling function helps assign the labels from one diagnoser state to another state over observed sequence s .

4.2.2 Fuzzy logic reasoning with transitions

Once the Petri net structure and labels are ready, the possible event transitions will be decided with fuzzy reasoning on input/output signal features to describe which faulty event will be fired. If the reasoning possibility is greater than or equal to the threshold, then that transition will be fired.

As shown in Figure 12, we first assumed that every fault event has a unique feature from sensor signals and a threshold characterizing by the DES state transitions. Mode identification will firstly be processed by extracting features from I/O sensor signal data. The signal feature data are then classified through fuzzy logic to determine the faulty mode. Features are extracted by a feature extraction module and placed into the diagnostics database. The fuzzy logic reasoning loads the required features from the database and then determines the current operating mode through inference engine and defuzzification algorithms.

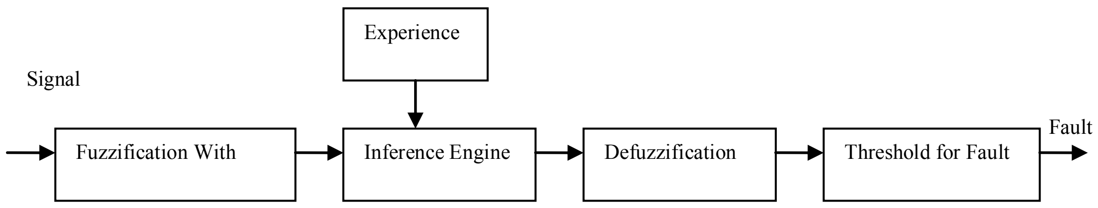


Figure 12 Fuzzy logic reasoning process

In the reasoning process, signal features will be firstly fuzzificated with membership functions in a variety of shapes including triangular, trapezoidal, Gaussian, Sigmoid, bell functions etc. Expertise information about the symptoms of faults is then used to create both the membership functions and rules to detect faults. For each rule, there are two kinds of fuzzy rule implications: either (1) Mamdani model, or (2) Takagi-Sugeno-Kang (TSK) model, be used for rule inference [93]. Examples of rules in Mamdani or TSK type are expressed as below:

Mamdani type:

***Ri**: IF elbow is extended for too long period THEN sensor AXH2 is abnormal*

TSK type:

***Ri**: IF elbow is extended for too long period THEN sensor AXH2 reading AXH2=3+4*t*

The difference between these two kinds of rule models lies in the consequent part: Mamdani model's consequence is a fuzzy set, which is easier to understand and more suitable for capturing imprecise human expertise; While TSK's functional consequent can be used to approximate complex nonlinear model using only a small number of rules. Like that, Mamdani model fits our application better, because how to effectively render knowledge accumulated from the system operations to facilitate diagnosis is the targeting problem we identified in Section 2. Thus we use the Mamdani model as our rule implication in the fuzzy Petri net. After the fuzzy implication has been performed for each rule for a particular fault, these rules need to be unioned through "AND" or "OR" operation upon the resulting membership functions. The typical union operations in fuzzy Petri net are shown as Figure 13.

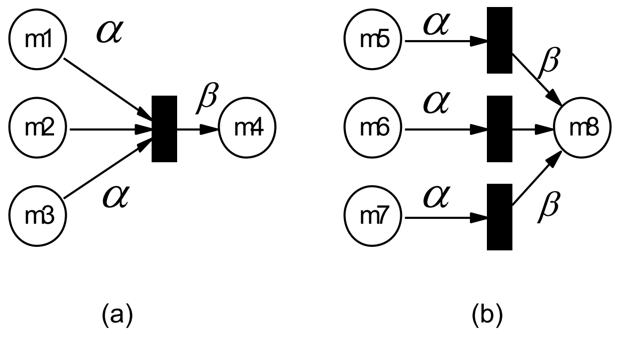


Figure 13 Union operations in fuzzy Petri net (a) "AND" operation, (b) "OR" operation

The mathematic operations on "AND" and "OR" are as below:

"AND" operation:

$$\alpha[P(j+1)] = \left[\underset{j}{\wedge}\alpha(P_j)_{P_j \in I(T_i)}\right] \times \beta(P_j)_{P_j \in O(T_i)} = \left[1 - \prod_{j=1}^{n}\left[1 - \alpha(P_j)_{P_j \in I(T_i)}\right]\right] \times \beta(P_j)_{P_j \in O(T_i)} \quad (4\text{-}4)$$

"OR" operation:

$$\alpha[P(j+1)] = \max_j\left[\alpha(P_j)_{P_j \in I(T_i)} \times \beta(P_j)_{P_j \in O(T_i)}\right] \quad (4\text{-}5)$$

After these operations, de-fuzzification is performed on the union output results based either on (1) Mean of Maximum (MOM) method, or (2) Center of Area (COA) method [92].

For MOM, suppose "y is A" is a fuzzy conclusion to be defuzzied, MOM de-fuzzification can be expressed as:

$$MOM(A) = \frac{\sum_{y^* \in P} y^*}{|P|} \quad (4\text{-}6)$$

Where P is the set of output values y with highest possibility degree in A.

For COA, suppose $\mu_A(y)$serves as the weight for value y, then COA de-fuzzification is:

$$COA(A) = \frac{\sum_y \mu_A(y) \times y}{\sum_y \mu_A(y)} \qquad (4\text{-}7)$$

Between these two de-fuzzification methods, MOM is less complicated and easy to implement. However, a major limitation of MOM de-fuzzification is that it does not consider the overall shape of the possibility distribution. Two fuzzy conclusions with the same peak points, but otherwise different shape, will yield the same de-fuzzification result using the MOM method. So here we pick up COA as our de-fuzzification approach. The de-fuzzification output result will be compared with the threshold. Once the result is greater than the threshold, that faulty event will be fired; otherwise, it will not be changed.

4.3 System developments

4.3.1 Descriptions on the dual robot arm

The proposed approaches were implemented to diagnose a dual robot assembly arm as shown in below Figure 14. It is a robot work cell including a conveyor, two 4-axis robot arms, four stoppers, two parts feeders, a computer vision and a controller. The robot arm is composed of a shoulder, an elbow, a wrist and a gripper. The sequential operations for the robot arm are: 1) the stopper blocks the pallet and sends signal to initiate the arm; 2) open gripper, lower elbow, pick up part from part feeder; 3) raise elbow, extend the shoulder, lower elbow, open gripper; and 4) close gripper, raise elbow, retract shoulder. These actions were controlled by Allen Bradley® programmable logic controller SLC 5/05 with other I/O modules. Due to the space constraint, we concentrated more on station3 and robot arm1 since they are functioning together. The diagnosis on arm2 can follow the similar procedures from arm1. The possible faults and their corresponding symptoms are listed in Table 21. As we assumed in the beginning, the occurrences of these faults were classified as unobservable transitions, so their places/transitions cannot be related to the observable input/output sensor events.

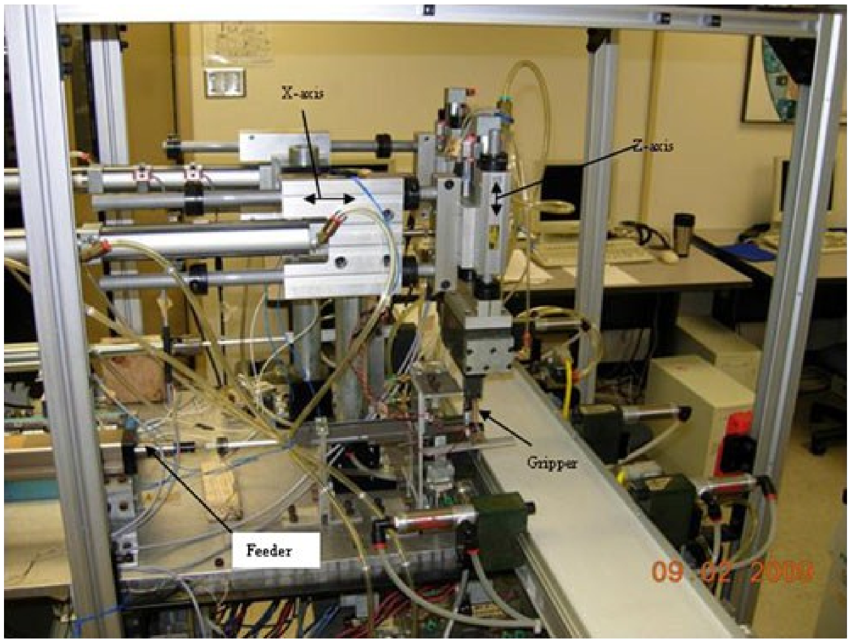
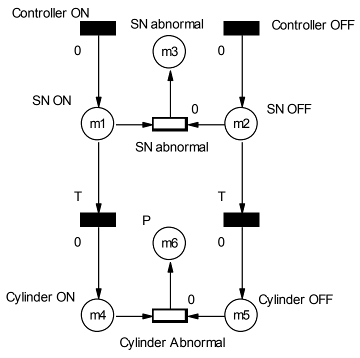


Figure 14 An automated assembly dual robot at Texas A&M University

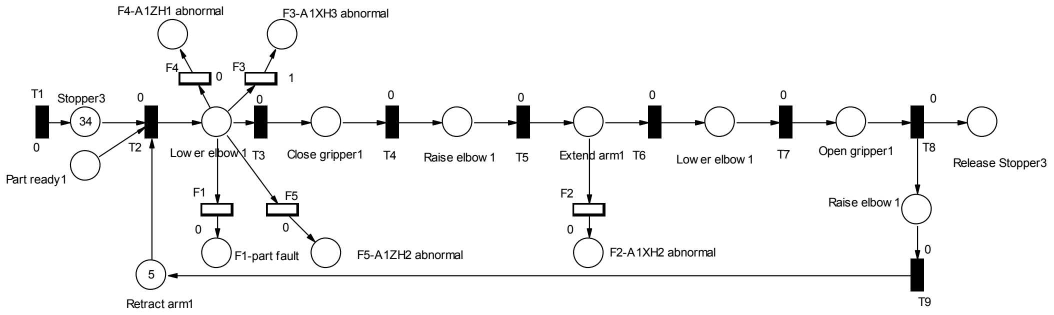
The RTFPN model is shown as Figure 15 (a) and (b). All the actuators used in the dual robot arm1 are pneumatic cylinders, which are driven by the solenoids after receiving commands from PLC. For those pneumatic devices, possible faults are solenoid abnormal and cylinder abnormal. Their corresponding model is as Figure 15 (a). This diagnoser fits the application for all the dual robot arm pneumatic devices including a stopper, a gripper and X, Z actuators. They will be integrated into the diagnoser for the whole system. Figure 15 (b) is the PN model supervising the whole system. In these two plots, solid boxes are the observable transitions while the nonsolid boxes are unobservable transitions. Places described by the possible sensor sets with each state and transition are related with the actions. Since we assumed that faults are unobservable places, no sensor mapping is assigned to faults, but their existences can be inferred from the sensor signal features. With this in mind, we constructed the I/O mapping as Table 22.

Table 21 Possible faults and symptoms

No.	Failure Root Cause	Symptom
F1	Not a correct product	The gripper cannot pick up the part, but the arm1 will move to the carrier position then only move up and down in the z-direction. At this moment, the ZH1 and ZH2 are triggered alternatively, Z is also triggered periodically.
F2	XH2 abnormal	It lowers the arm and grip the part, then extends the arm, but it will open the gripper without lowering the arm at the carrier position. XH2 can get signal when arm1 is retrieved back, but will not be triggered when it is extended, z is triggered only on the part feeder position, but won't be triggered when it arrive on the carrier position.
F3	XH3 abnormal	The arm1 will not be lowered, neither the gripper pick up the part, but arm1 will move to the carrier position, then move up and down in the Z direction.
F4	ZH1 abnormal	It will lower the arm, but it won't grip the part, then raise the arm, then it extend the arm, then stands above the carrier and does nothing.
F5	ZH2 abnormal	It will not lower the arm, neither grip parts. It only extends the arm. The durance of arm is controlled by timer, not A1ZH2.



(a)



(b)

Figure 15 PN model for arm1

4.3.2 Diagnoser design

With the RTPN model in Figure 15, the diagnoser for the dual robot was built as Figure 16. This Figure 16 was built based on the concept of "coverability tree" [5], which represents all possible markings. Because we assume that some transitions and places are unobservable, an efficient way to update the system estimate is to use the observation on the changes in observed markings. In this figure, the first 16 numbers mean the possible markings in each place associating with the sequence of operations. The last letter associates with the possible normal/faulty states. We can see that F2 can be easily identified without any confusion. However, in order to differentiate the faults of (F1, F3, F4, F5), diagnoser need to be delicately designed. When we were running and testing the system, we have accumulated plentiful experiences on its normal and faulty conditions. If we can integrate these experiences into our diagnoser design, it will greatly improve the efficiency and accuracy of diagnostics. Here fuzzy Petri net was used in our diagnoser to isolate the faults.

Table 22 Place, transition and input/output mapping table

Places	Description	Output Devices	Transition	Input events
P1	Stopper3	{C3,PSS3,PS3}	T1	Start
P2	Feeder part ready	{Part, PSS3}	T2	Lower elbow1
P3	Elbow1 lowered	{PSS3, Part Ready, A1ZH2,A1XH3}	T3	Close gripper2
P4	Part1 picked (close gripper1)	{A1ZH1,A1XH3}	T4	Raise elbow2
P5	Elbow1 raised	{T40.TT}	T5	Extend arm1
P6	Arm1 extended	{PSS3, T11}	T6	Lower elbow1
P7	Lower elbow1	{PSS3,A1XH2,A1ZH2}	T7	Open gripper1
P8	Open gripper1	{T55.DN}	T8	Raise elbow1
P9	Raise elbow	{T40.DN}	T9	Retract arm1
P10	Retract arm1	{A1XH2}		
Faulty Places			**Faulty Transitions**	
PF1	Part fault	{A1ZH2, A1XH2, gripper}	F1	
PF2	A1XH2 abnormal	{A1ZH2, A1XH2, gripper}	F2	
PF3	A1XH3 abnormal	{A1ZH2, A1XH2, gripper}	F3	
PF4	A1ZH1 abnormal	{A1ZH2, A1XH2, gripper}	F4	
PF5	A1ZH2 abnormal	{A1ZH2, A1XH2, gripper}	F5	

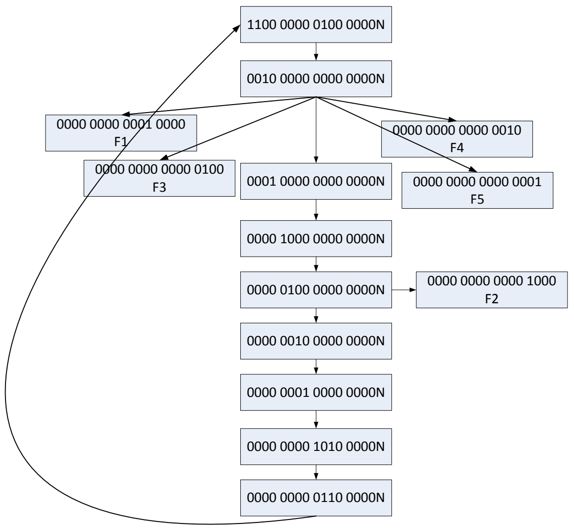
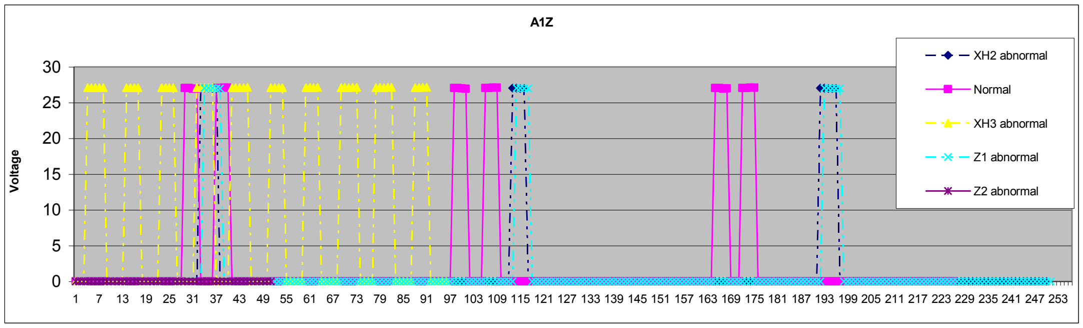


Figure 16 Fault diagnoser design for dual robot

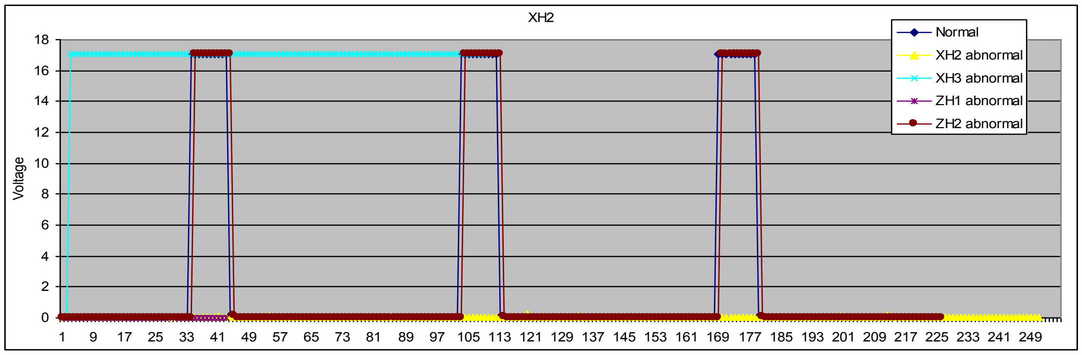
For the control of the robot arm, it involves coordinating actions in the X-axis and Z-axis as well as gripping; we selected the readings from the sensor Z and XH2 as the input features to our fuzzy reasoning process. Their signal features on normal and different faulty states are shown as Figure 17 (a) and (b).

As shown in the figures, those two signals (A1Z and XH2) have the characteristics of event driven. For this type of system, sequence of events and timing intervals of each event are the most useful signal features. Thus they are selected as the inputs to the fuzzy reasoning. In Figure 17 (a), at the normal states (pink square dot line), there are two intervals in each work cycle; and each interval lasts for 1 second. If the intervals appear periodically as yellow dots in the figure, possible fault is XH3 abnormal. When the interval appears only once, possible faults can be either ZH1 abnormal or XH2 abnormal. If the interval never appears, ZH2 abnormality might happen.

In Figure 17 (b), at the normal state, there is only one interval in each working cycle for signal XH2 and the interval lasts for 2 second to cover the duration of Z. If the signal stays at high forever, the possible fault can be XH3 abnormal. If the interval never appears, XH2 abnormal might happen.



(a)



(b)

Figure 17 Input signal features

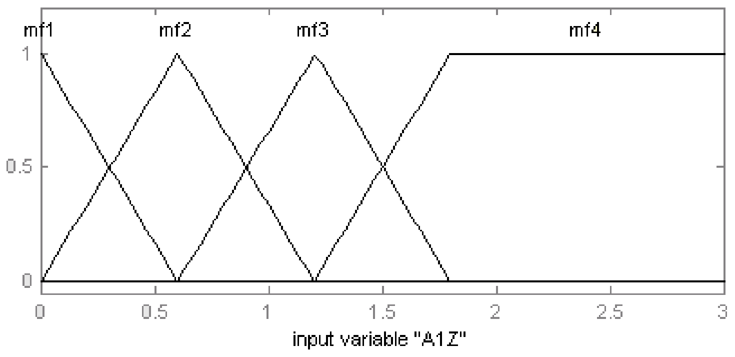
Summary on these, the fuzzy rules were derived as below:

R1: IF *A1Z has two 1 second cycles* AND *each cycle lasts for 1 second*, THEN *the system is normal*

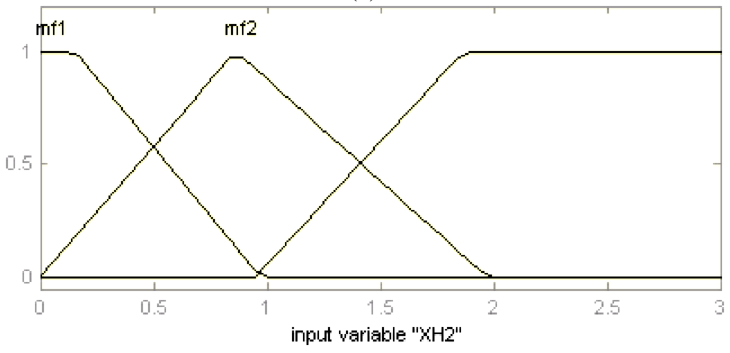
R2: IF A1Z *has periodic cycles*, THEN *XH3 is abnormal (F3)*

R3: IF *A1Z has one 1 second cycle* AND *XH2 also has one 1 second cycle*, THEN *ZH1 is abnormal (F4)*

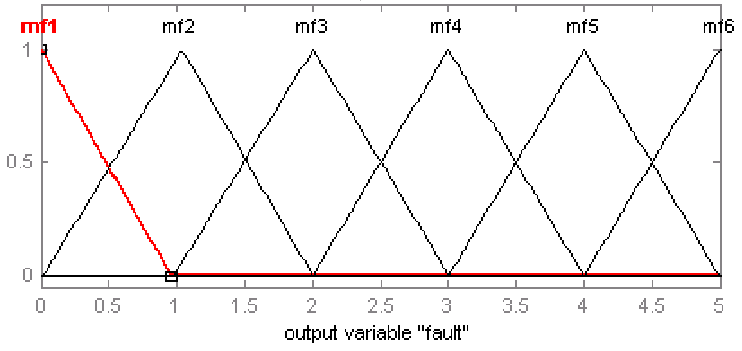
R4: IF *A1Z never appears*, THEN *ZH2 is abnormal (F5)*



(a)



(b)



(c)

Figure 18 Membership functions for A1Z (a), XH2 (b) and faults (c)

The membership functions for A1Z, XH2 and possible faults are illustrated in Figure 18 (a), (b), and (c). Their linguistic terms are shown on these figures respectively also. The identified rules are plotted in Figure 19. Possible faults were encoded as numbers and formed the Z-axis in Figure 19, inputs are the periods in XH2 and Z. Figure 19 illustrated what the possible fault output is as observed signals in XH2 and A1Z vary.

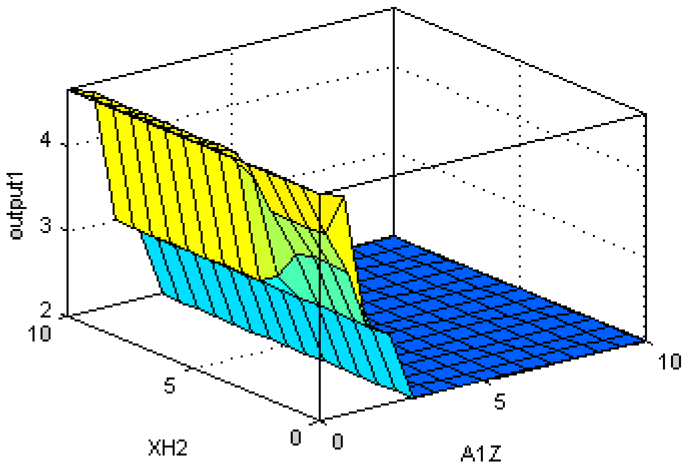


Figure 19 Fuzzy rules for dual robot diagnosis

Finally the fuzzy Petri net to isolate the faults is constructed in Figure 20. With the inclusion of the signal features, the faults can be identified without any confusion.

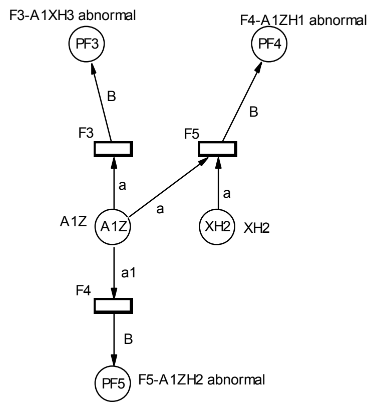


Figure 20 Fuzzy diagnoser to isolate faults

4.3.3 System implementation

In order to illustrate how to apply the proposed approach on diagnosing a real manufacturing system, we implemented the proposed PN model and diagnoser using Visual Basic®. The architecture for the diagnosis platform is shown in Figure 21. This architecture consists of: 1) the main Petri net screen to monitor the whole system; 2) four distributed Petri net forms to monitor each substation respectively; and 3) an additional forcing output form to manually drive the outputs of PLC when diagnosis needs. The implemented diagnoser capitalizes the event-driven and discrete I/O characters of the discrete manufacturing system to monitor and diagnose the system in realtime. To enable the real time property, all the inputs (sensors) and outputs (actuators) of the system are physically connected to the I/O modules of the PLC, and then PLC communicates with the personal computer through the Visual Basic's MSComm object (serial RS232 communication port). The detailed implementation on communication setup, Petri net implementation and fault reasoning will be presented in the following sections.

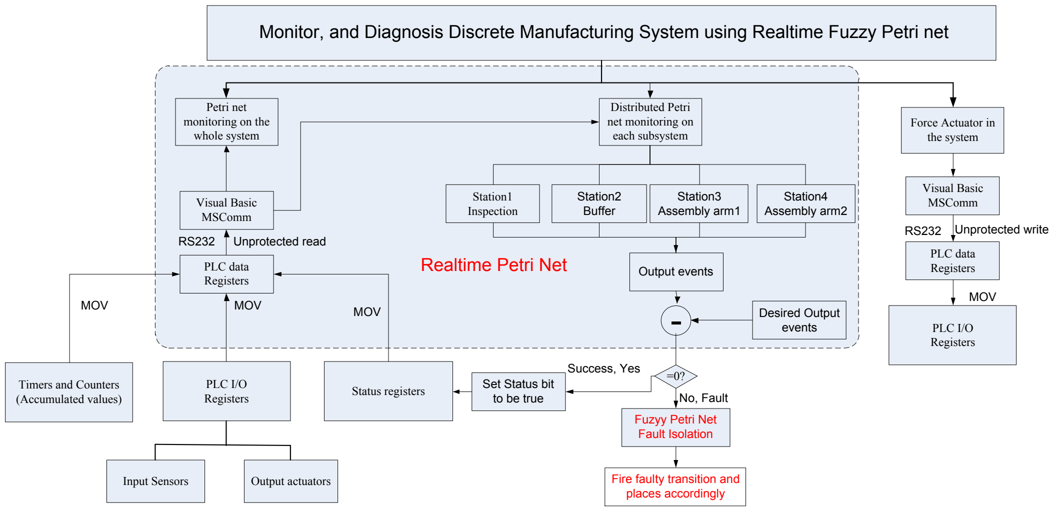


Figure 21 Flow chart for the RTFPN diagnoser on dual robot arm

4.3.3.1 Communication setup to enable realtime control

It is critical to set up the communication between the PC and PLC, so the Petri net can monitor or diagnose the plant in real time. This communication set up involves two aspects: 1) enable the communication protocol between the PC and PLC, and 2) enable PC to read or write the PLC's I/O registers.

To enable PC read/write the PLC register in real-time, we need programming to set up the communication protocol between lower machine and host computer. In this research RS232 serial communication is the protocol, PLC is the lower machine and PC is the host machine. The communication set up on host machine is as follows.

```
Private Sub CommunicationSetup( )
' Set up the Communications Port
   MSComm1.CommPort = 2 'change according to your system configuration
   ' 19200 baud, no parity, 8 data, and 1 stop bit.
   MSComm1.Settings = "19200,N,8,1"
```

```
    ' Tell the control to read entire buffer when Input is used.
    MSComm1.InputLen = 1
'Set the characters to return to 1
MSComm1.RThreshold = 1
    ' Open the port.
     MSComm1.PortOpen = True
   'set up PLC Communication Parameters
    bDST = 1 'destination address
    bSRC = 0 'source address: The computer is address zero
 End Sub
```

In this application, the function of MOV in PLC programming was used for transferring the status of inputs and outputs of PLC into data registers for data interchange with the computer. A typical PLC program for this data transfer is shown in Figure 22. It moves the data from source to destination for either indicating the input/output or forcing the output. In this way the information of PLC and manufacturing operation status can be used in the diagnosis platform.

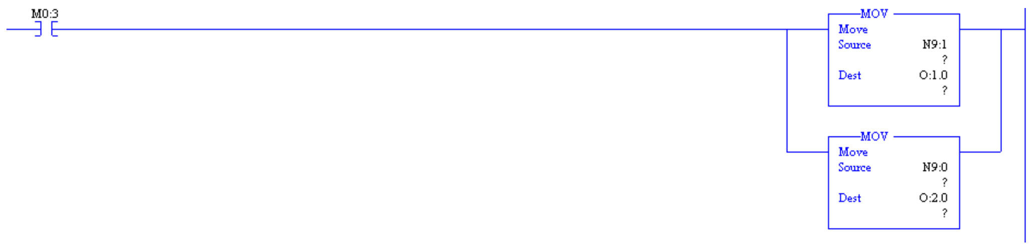


Figure 22 An example PLC rung on read/write Register

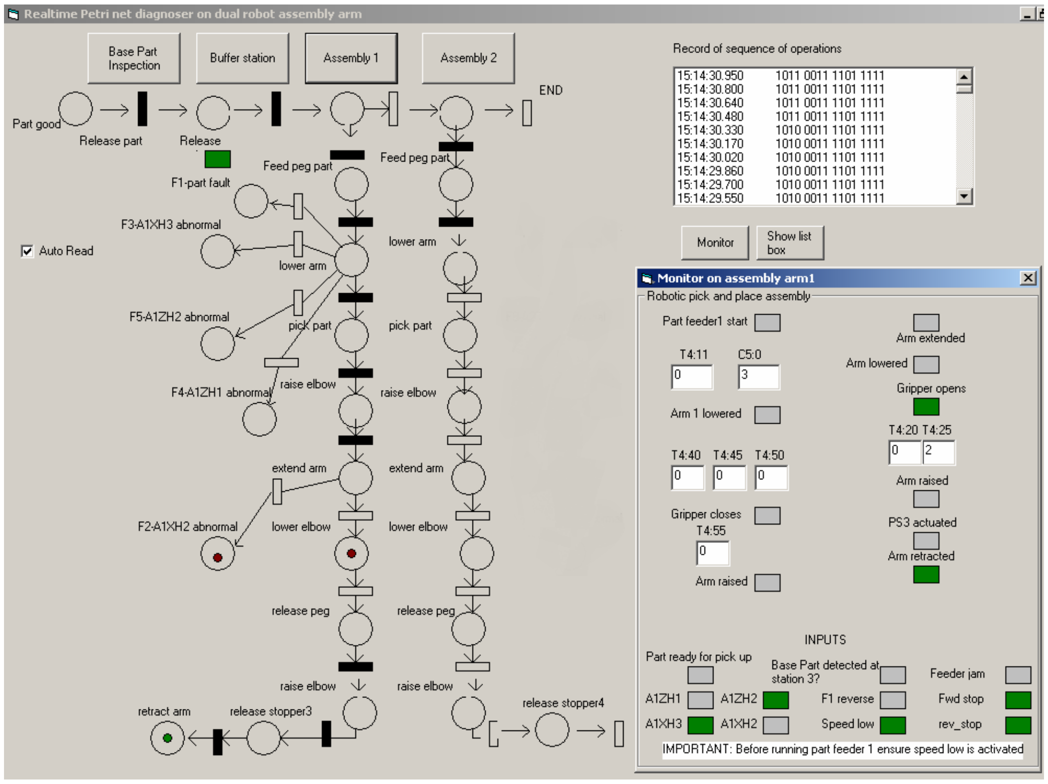
4.3.3.2 Petri net programming

Figure 23 illustrates the main Petri net interface at runtime. The VB application reads data from the PLC's data registers. displays the firings on Petri net and tokens in the main Petri net screen in real time, and it also shows the "Record of sequence of operations” and timer intervals on each station. Each station in the assembly line is represented by a circular place beneath the button for the station. Additionally, there are places corresponding to every event in the sequence of operations at each station. When the base part coming along the conveyor reaches a station, the token goes into that particular place indicating that the operation at that station is fired and active. Once the base part leaves the station and all the outputs corresponding to that station have been de-energized in accordance with the control logic, the token leaves that place and gets ready to the next place.

The user can also go into great detail about the running status on each substation by clicking on either one of the buttons: ‘Base Part Inspection’ or ‘Buffer station’ or ‘Assembly 1’ or ‘Assembly 2’. It will open a more detailed sub-form on the status of all the I/Os at that station. For example, by clicking “Assembly 1”, the detailed view of station 3 (assembly arm1) is shown in the Figure 23. It is possible to see the real time status of all I/Os at that station and the accumulation of timing and/or counting elements through the control logical sequence.

4.3.3.3 Fault localization and isolation based on fuzzy logic

The Petri net model keeps checking the input/output device status. If a difference is detected between the desired I/O value and the actual value, it means a faulty event happens then the fuzzy Petri net will be started to isolate the fault. The signal features to identify a failed event are signal values and timing intervals. An event is considered faulty if it does not occur at the time when it was supposed to occur (non-occurrence) or the occurrence interval is not correct (mis-timed occurrence). The non-occurrence or mis-timed occurrence event is predetermined through simulations and experiments. During the operation, the ladder logic in the PLC verifies whether each event occurs as timed. In case of a faulty event, the Petri net fires the identified fault place on the Petri net interface to indicate a faulty operation.



```
15:14:30.950    1011 0011 1101 1111
15:14:30.800    1011 0011 1101 1111
15:14:30.640    1011 0011 1101 1111
15:14:30.480    1011 0011 1101 1111
15:14:30.330    1010 0011 1101 1111
15:14:30.170    1010 0011 1101 1111
15:14:30.020    1010 0011 1101 1111
15:14:29.860    1010 0011 1101 1111
15:14:29.700    1010 0011 1101 1111
15:14:29.550    1010 0011 1101 1111
```

Part feeder1 start ☐ | Arm extended ☐
T4:11 0 | C5:0 3 | Arm lowered ☐
Gripper opens 🟩
Arm 1 lowered ☐
T4:20 0 | T4:25 2
T4:40 0 | T4:45 0 | T4:50 0
Arm raised ☐
Gripper closes ☐ | PS3 actuated ☐
T4:55 0 | Arm retracted 🟩
Arm raised ☐

INPUTS

Part ready for pick up ☐ | Base Part detected at station 3? ☐ | Feeder jam ☐
A1ZH1 ☐ | A1ZH2 🟩 | F1 reverse ☐ | Fwd stop 🟩
A1XH3 🟩 | A1XH2 ☐ | Speed low 🟩 | rev_stop 🟩

IMPORTANT: Before running part feeder 1 ensure speed low is activated

Figure 23 PN diagnoser with detailed view on assembly-1

In order to illustrate how the Petri net diagnoser diagnoses faults, let us consider the event of "pick part" which corresponds to closing of the gripper. Figure 24 shows the part of the ladder logic for detecting the failure of the event: closing the gripper on robot arm1. T4:9 is the triggered timer when the base part reaches station 3, the first assembly station detected by the part stopper sensor at that station. Within 2 seconds of this timer, the output to close the gripper, "A1_GRIPPER" must be energized for a period of 0.4 seconds. If this is true, the bit "N9:26/4" will be latched indicating that the event is executed successfully. Otherwise, the event is faulty and the fault places in the diagnoser to "pick part" will be fired to denote a failed event. The cause of this failure is the loss of input from the sensor "A1XH2" or "A1ZH1" which senses that the arm's position along the x-axis and z-axis. To locate the exact fault, fuzzy Petri net will read the PLC bits from the I/O register, and feed them to the fuzzy reasoning inference to notify the faulty state. This interface corresponding to this failure is depicted in Figure 25 (a), where the token marking in the first "lower arm" place is on, which means this place has been executed; while the second "lower arm" place never on, which means this place has never been executed, this place/transition sequence corresponded to A1ZH1's failure mode. So the fuzzy isolation algorithm concludes it is A1ZH1 abnormal. From the detailed view of station 3 in Figure 25 (b), it is possible to view the real time status of the input A1ZH1, which always keeps grey and never turns green. It also means that the fault happens with A1ZH1.

4.3.3.4 Forcing outputs

In order to confirm the diagnosis result from the previous methodology, we also included a forcing outputs module in our diagnosis system. Making temporary changes to the process like bypassing certain inputs and forcing outputs on or off is an important part of troubleshooting. The form that can be used to force the actuators manually acting is shown in Figure 26. While the right half of the form displays the current I/O status, the left half of the form is used for forcing the outputs by clicking the label above the output that is desired to be energized. The inputs and the outputs have been placed beside one another in order to be able to see how forcing outputs affect the inputs at all. For example, forcing the robot arm to extend along the x-axis will activate “A1XH2” to sense that the arm is extended. If it does not extend, we can conclude that some faults happen with input or output, then troubleshooters can go into the detailed “forcing output” form to check which device is in abnormal.

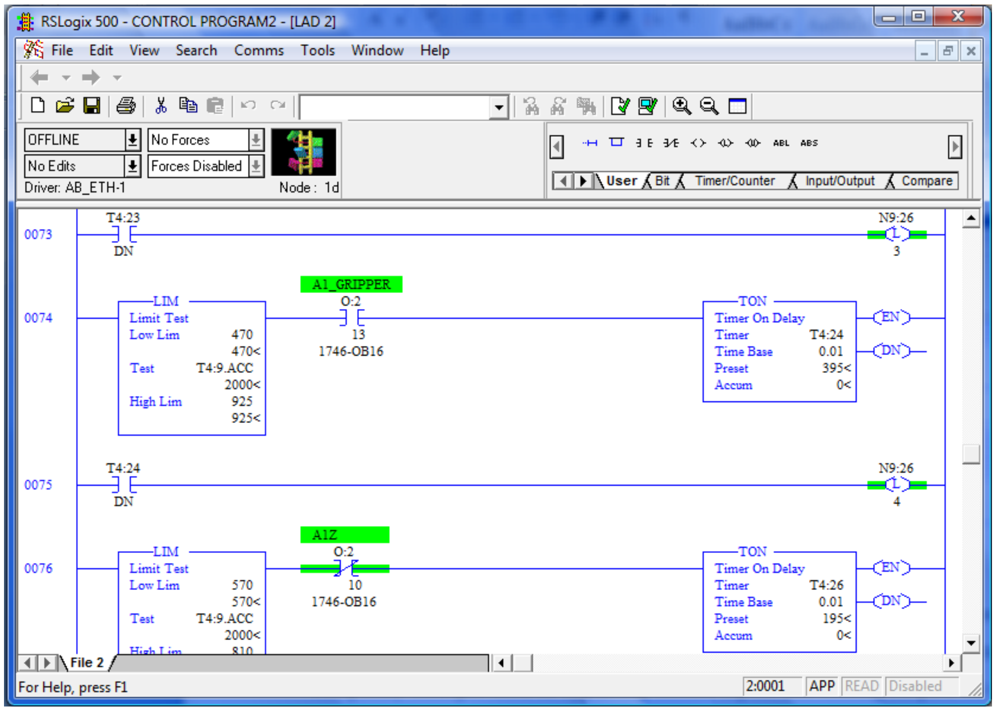
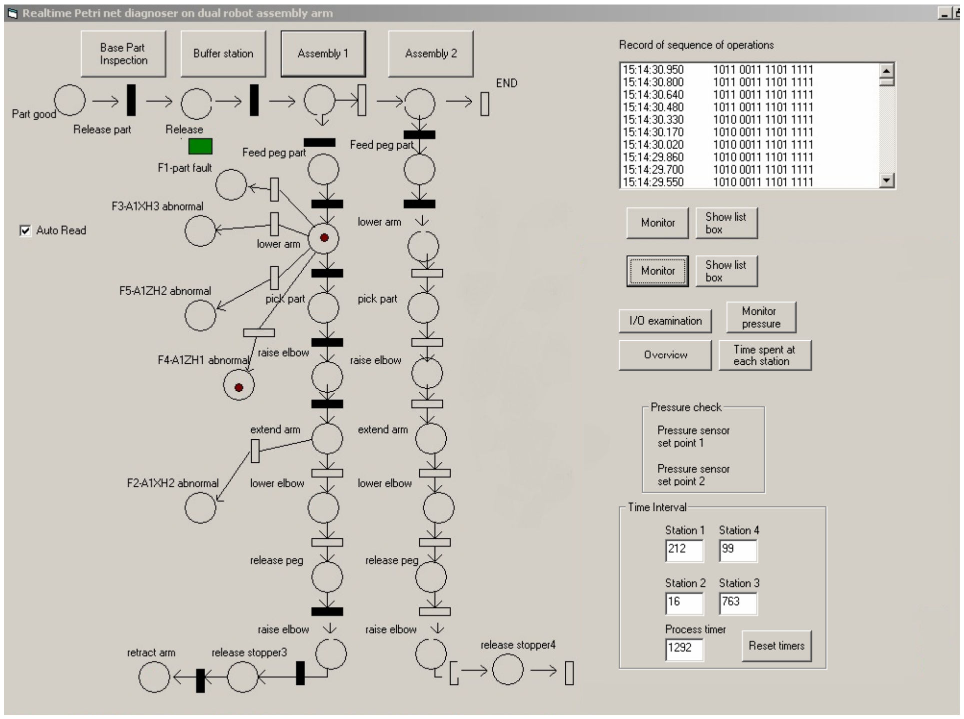


Figure 24 Abnormal event detection logic



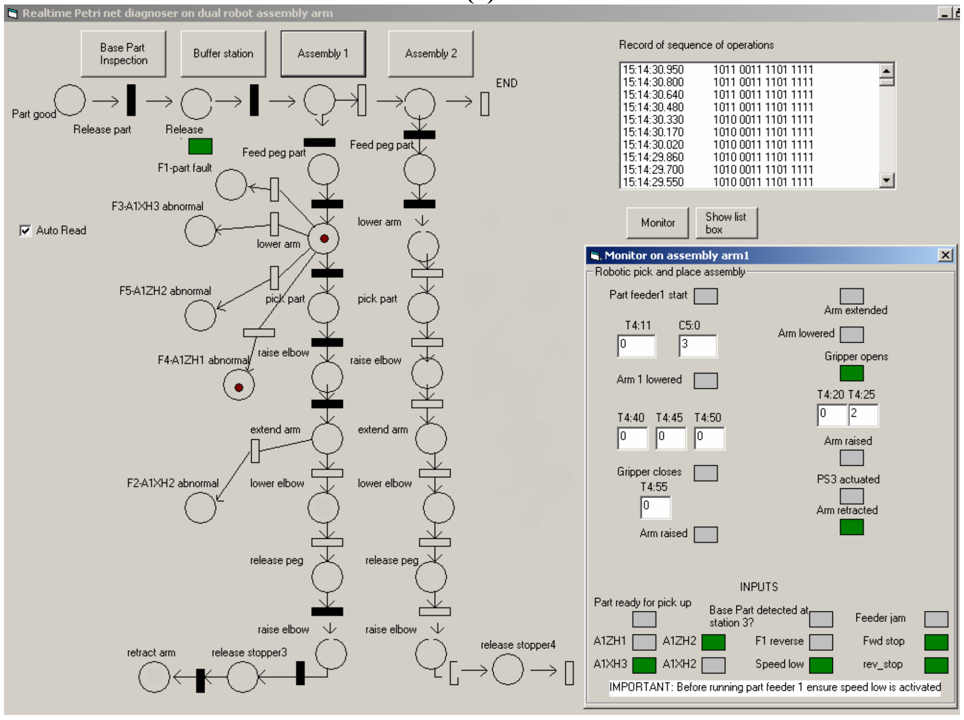
Record of sequence of operations

```
15:14:30.950    1011 0011 1101 1111
15:14:30.800    1011 0011 1101 1111
15:14:30.640    1011 0011 1101 1111
15:14:30.480    1011 0011 1101 1111
15:14:30.330    1010 0011 1101 1111
15:14:30.170    1010 0011 1101 1111
15:14:30.020    1010 0011 1101 1111
15:14:29.860    1010 0011 1101 1111
15:14:29.700    1010 0011 1101 1111
15:14:29.550    1010 0011 1101 1111
```

Time Interval

Station 1: 212	Station 4: 99
Station 2: 16	Station 3: 763
Process timer: 1292	

(a)



Record of sequence of operations

```
15:14:30.950    1011 0011 1101 1111
15:14:30.800    1011 0011 1101 1111
15:14:30.640    1011 0011 1101 1111
15:14:30.480    1011 0011 1101 1111
15:14:30.330    1010 0011 1101 1111
15:14:30.170    1010 0011 1101 1111
15:14:30.020    1010 0011 1101 1111
15:14:29.860    1010 0011 1101 1111
15:14:29.700    1010 0011 1101 1111
15:14:29.550    1010 0011 1101 1111
```

Monitor on assembly arm1 — Robotic pick and place assembly

Part feeder1 start ☐	Arm extended ☐
T4:11: 0	C5:0: 3
Arm lowered ☐	Gripper opens 🟩
Arm 1 lowered ☐	T4:20: 0 T4:25: 2
T4:40: 0 T4:45: 0 T4:50: 0	Arm raised ☐
Gripper closes ☐	PS3 actuated ☐
T4:55: 0	Arm retracted 🟩
Arm raised ☐	

INPUTS

Part ready for pick up ☐	Base Part detected at station 3? ☐	Feeder jam ☐	
A1ZH1 ☐	A1ZH2 🟩	F1 reverse ☐	Fwd stop 🟩
A1XH3 🟩	A1XH2 ☐	Speed low 🟩	rev_stop 🟩

IMPORTANT: Before running part feeder 1 ensure speed low is activated

(b)

Figure 25 PN diagnoser for fault to close gripper

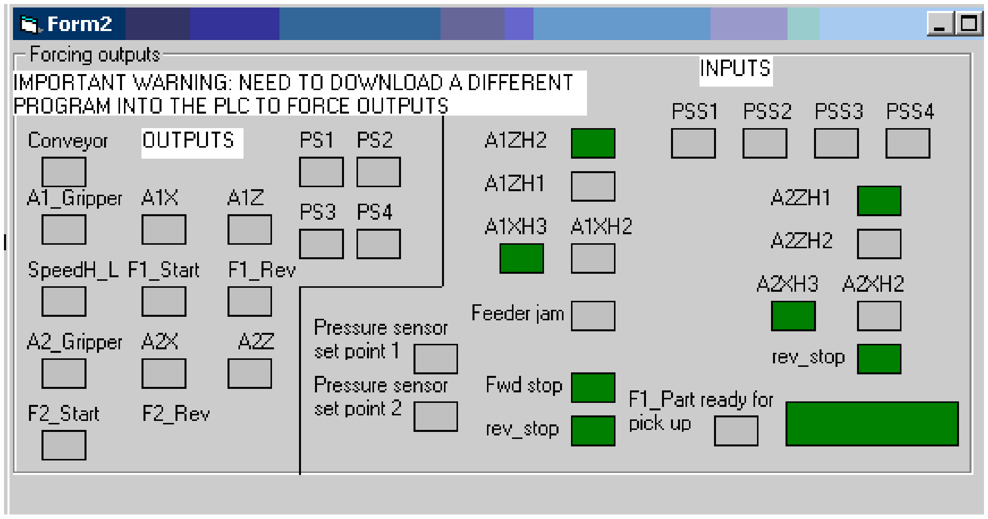


Figure 26 Forcing outputs in VB and examining I/Os

4.4 Experiments

A series of experiments were carried out to validate the proposed approach and system implementation on the diagnosis accuracy, delay etc. Before the experiment, failure mode effect analysis (FMEA) had been conducted to identify the possible faults, their severities and effects on the mission of the system. There are five most common faults F1~F5 involved in the dual robot arm. The FMEA analysis is shown as Table 23.The effectiveness of the diagnoser to these faults was evaluated using the metrics including recognition accuracy, diagnosability.

Table 23 Failure mode effect analysis on the dual robot arm

Function	Failure Mode	Effects	Current controls	Severity Rate	Occurrence Rate	Detection rate	Risk priority number	Recommended actions
Grip	Not correct part (F1)	Gripper can not pick up the part, and the arm1 only move up and down in the z-direction.	Regular check by operator	6	2	5	80	Check the part and reject the wrong part
sense that the arm is extended	XH2 abnormal (F2)	Gripper opens without lowering the arm at the carrier position	Regular check by operator	6	2	5	80	Check the I/O circuit with force I/O
sense that the arm is retracted	XH3 abnormal (F3)	Same effect as not correct part	Regular check by operator	6	2	5	80	Check the I/O circuit with force I/O
sense that the elbow is raised	ZH1 abnormal (F4)	It won't grip the part, then raise the arm, then it extend the arm, then stands above the carrier and does nothing.	Regular check by operator	6	2	5	80	Check the I/O circuit with force I/O
sense that the elbow is lowered	ZH2 abnormal (F5)	It will not lower the arm, neither grip parts. It only extends the arm. The durance of arm is controlled by timer, not A1ZH2.	Regular check by operator	6	2	5	80	Check the I/O circuit with force I/O

4.4.1 Recognition accuracy

The recognition accuracy of fault detection is simply defined as equation (4-8): the ratio of correct fault detections to the total trials. 10 runs of each fault were introduced into the experiment randomly, so total 60 runs of experiment were conducted.

Recognition accuracy= (total number of correct decisions)/(total number of trials) × 100% (4-8)

The results of recognition accuracy are shown as below Table 24. In this table, "Y" means correct detection of fault, and "N" means misdetection of fault. It shows that "F1" is the hardest to detect while "F3" and "F5" are the second. Possible explanation for this can be traced as: F1 is the part fault, which means the fault cannot trigger the Hall sensor to operate the right actions. This fault can be interrupted by many factors such as the position between the part and the sensor, the part itself and the Hall sensor abnormality etc. Thus it complicates the fault isolation on F1. It is hard to detect F3 because F3 has almost the same symptom as F1, so the sequences of firing transitions and places for F1 and F3 are similar to each other. Thus it created some mis-detections. The same situation fits for F5.

Table 24 Correctness measurement

	Runs	1	2	3	4	5	6	7	8	9	10	Recognition accuracy
F1 Part fault	10	Y	Y	Y	Y	N	Y	N	Y	Y	Y	80%
F2 A1XH2 abnormal	10	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100%
F3 A1XH3 abnormal	10	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	90%
F4 A1ZH1 abnormal	10	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100%
F5 A1ZH2 abnormal	10	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	90%
normal	10	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	100%
Total	60											93%

4.4.2 Diagnosability evaluation

Diagnosability checking is for the purposes of: 1) on-line detection and isolation of faults and 2) off-line verification of system whether can be diagnosed or not [17]. The diagnoser should be able to detect fault after a finite step of delay from its occurrence. Suppose σ is the sequence of observable transitions. Therefore, a PN that represents the system model is diagnosable if in a finite number of observable transitions it reaches a fault marked as $M(P_f)$. Only the $M(P_{f_i})$ or with other fault marked as $M(P_{f_k})$ can identify a fault of higher order or a critical fault.

Following the diagnosability definition: the occurrence of faulty event F_i should be detected in at most n_i transitions of the system after the occurrence of an indicator event, a delay diagnosis matrix (DDM) for diagnoser on each fault was built to define diagnosability quantatively [22]. This matrix has a number of columns equal to the number of the diagnosis desired states, while the number of rows is equal to the number of fault partitions or labels. Each matrix element DDM_{ij} indicates if a fault belonging to the fault partition of line i can be diagnosed, then that element $DDM_{ij} = 1$, after the occurrence of one observable transition or the violation of an expected starting from the place j or not, $DDM_{ij} = 0$. Then the diagnosis delay can be calculated as the maximum number of successive zeros in each line plus one. In this way, the delay diagnosis matrix was built to evaluate the diagnosability. The number of elements was observed from the experiment running. In Table 25, we can see that in order to detect F3 it takes eight steps, which is the longest diagnosis delay. So we concluded that the diagnosis delay for this diagnoser on those identified faults was equal to eight occurrences of transitions.

Table 25 Diagnosis delay matrix

	Feeder peg part	Lower arm	Pick up part	Raise elbow	Extend arm	Lower elbow	Release peg	Raise elbow	Release stopper	Retra arm
F1	0	1	0	0	0	1	0	0	0	0
F2	0	0	0	0	0	1	0	0	1	0
F3	0	1	0	0	0	0	0	0	0	1
F4	0	0	1	0	0	0	0	0	1	0
F5	0	1	0	0	1	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0

4.5 Conclusion

We presented a realtime fuzzy Petri net (RTFPN) based approach to detect progressive faults in discrete manufacturing system. This fault diagnosis approach includes: 1) a real time Petri net (RTPN) model aiming at the nonsynchronizing problem in the PN diagnoser, and 2) a fuzzy Petri net (FPN) diganoser with the ability of integrating knowledge and handling uncertainties. The RTPN model replicates the sequence of operations of the plants, set up the mapping between the I/O devices and PN places/transitions thus monitor the manufacturing plant in real-time; while the FPN uses the signal features as inputs to decide the occurrence of possible outputs through fuzzy reasoning inference, so useful but incomplete knowledge can be integrated into diagnosis process to locate and isolate fault root causes upon occurring. To validate the proposed approach, a diagnosis prototyping system to a dual robot arm was developed using Visual Basic. Performance evaluations including correctness, diagnosability and time delay on the diagnoser were analyzed through experiments. It illustrated that the prototyping diagnoser can have a high accuracy rate of 93% and maximum diagnosis delay of eight steps. The proposed system remedies the nonsynchronizing issue and can handle uncertainty thus achieving high correctness. Experiment process showed that this system can perform multiple fault diagnosis on system with fast fault propagation and complex physical phenomena. Also, it allows various dynamic windows of human-machine interface to be created providing visual data and realtime place-transition diagrams for operators, engineers and experts. The proposed approach is systematic and can be easily extended to other complex systems as flexible manufacturing system (FMS), HVAC, and semiconductor manufacturing facilities.

CHAPTER V

BENCHMARK DIAGNOSER BASED ON FINITE STATE AUTOMATON AND SEQUENTIAL FUNCTION CHART

To efficiently detect the stochastic faults and compare the performance between PN and FSA diagnosers, I implemented a diagnoser based on FSA with sequential function chart. The finite state automaton models keep checking the inputs/outputs of the DES plant, and calculating the probabilities of possible normal/abnormal states. Sequential function charts behave as a decision coordinator to locate the exact root cause of the fault. The details of this FSA diagnoser will be specified in the following sections.

5.1 Plant model

The plant model follows the definition and concept in [16]. The manufacturing system G is modeled as an automaton: $G = (X, E, f, x_0)$, and $L = L(G)$ is $G's$ corresponding prefixed closed language. In this model, X is the finite set of states; E is the finite set of events associated with transitions in G. There are two ways to classify the total event set according to whether they are abnormal/functional, or observable/unobservable. Events can be either faulty (E_f)/normal (E_n) events, or observable (E_o)/unobservable (E_{uo}) events. Typical observable events are the commands issued by the controller, the sensor reading changes after the execution of commands, etc. The unobservable events include the faulty events and events that cannot be recorded by the sensors etc. It is obvious that $E := E_f \cup E_n = E_o \cup E_{uo}$. f is the state transition function, $f : E \times X \to X$ and $X(m+1) = f(X(m), E(m+1))$; x_0 is the initial state. Furthermore, plant G is decomposed into subsystems $G_1, G_2, \cdots, G_k$; each subsystem is modeled as its own automaton $G_i = (X_i, E_i, f_i, x_{0i})$, so we have $G = G_1 \parallel G_2 \parallel \cdots \parallel G_k$, $X = X_1 \times X_2 \times \cdots \times X_k$, $E = E_1 \cup E_2 \cup \cdots \cup E_k$, and $x_0 = (x_{01}, x_{02}, \cdots, x_{0k})$.

Then projection is applied on the automaton of each subcomponent. Projection is an operation such that let $P : E \to E_o$:

1. $P(\varepsilon) := \varepsilon$
2. $(\forall \mu\sigma \in E^*) P(\mu\sigma) := \begin{cases} P(\mu)\sigma, & if \sigma \in E_0 \\ P(\mu), & if \sigma \in E_{u0} \end{cases}$ (5-1)

In other words: given that P is the standard projection from E^* to E_o^*, we will have that $P^{-1}(s) := \{t \in E^* : P(t) = s\}$. The projection operation aims at screening the disturbing sequential events such as unobservable events to retain the useful information from the automaton.

5.2 Sequential function chart based control

SFC is a graphical programming language used for processes that can be split into steps in PLC applications [94]. Main components of SFC are: 1) steps with associated actions, 2) transitions with associated logic conditions, and 3) directed links between steps and transitions. The steps in SFC can be either activated or deactivated. Steps are activated when all steps above it are active and the connecting transition is superable (i.e. its associated condition is true). When a transition is passed, all steps above are deactivated at once and after all steps below are activated immediately. In our proposed approach, the PLC control logic is generated into a ladder diagram from SFC; SFC is also used as the basis to composite the action of each component thus modeling the plant and designing the diagnoser.

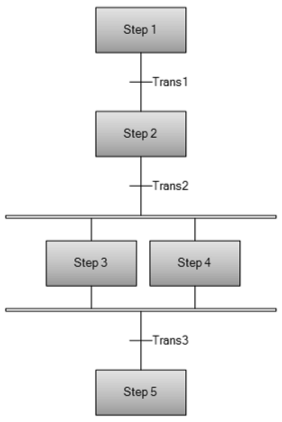


Figure 27 A sequential function chart example

5.3 FSA diagnoser design

When the system is running, the diagnoser keeps monitoring the observable events generated by the plant, and makes conclusion on normal/abnormal status of the system. To construct the stochastic diagnoser, we first classify the set of faulty events as $E_f = E_{F1} \bigcup E_{F2} \bigcup \cdots \bigcup E_{Fm}$ and $E_f \in E_u$, then the possible fault labels are defined as $l = \{N\} \bigcup \{F_1, F_2, \cdots, F_m\}$. We further define the label propagation function under event string s $LP : X_o \times l \times E_o \xrightarrow{f} l$ as

$$LP(x,l,s)=\begin{cases}\{N\} & l=\{N\}\wedge\forall i\lfloor\Sigma_{f_i}\notin s\rfloor\\ \{F_i\} & l=\{F_i\}\vee\Sigma_{f_i}\in s\end{cases}\tag{5-2}$$

Combining observable state X_o and *LP*, we newly generate a set of labeled states Q_d, which have the form of(x_i,l_i). Then we define the automaton for the diagnoser as a five tuples:

$$G_d=(Q_d,E_o,f_d,q_0,h).$$

- Q_d is a set of labeled states generated from the plant. The set of logical elements, Q_d is the subset of labeled states which are reachable and observable from q_o under the transition function f_d. An element $q_d\in Q_d$ has the form of: $q_d=\{(x_1,l_1),\cdots,(x_n,l_n)\}$. The set of all (x_i,l_i)s in q_d have the relationship of$q\in Q_d,x_i\in X_o,l_i\in l$. The number of such components (x_i,l_i)in an element q_d will be denoted by$\|q_d\|$.
- E_o is a set of observable events.
- f_d is the transition function of the diagnoser. It includes the state transition and label transition, and can be defined as $f_d(q,\sigma)=\bigcup_{(x,l)\in q}\bigcup_{s\in L_f(G,x)}\{(f(x,s),LP(x,l,s))\}$. The function

LP shows that a label $F_i$is added whenever the true behavior of the system contains an event$E_F\in\Sigma_{f_i}$. Once this label is appended, it cannot be removed regardless of whether an event in Σ_{f_i} occurs or not in the system behavior following the label.

- q_o is the initial labeled state in the diagnoser, which is$q_o=\{x_0,N\}$.

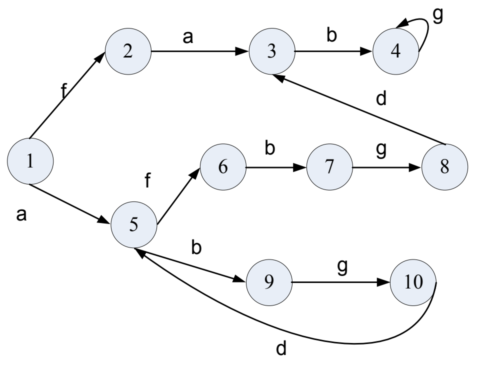
This $(Q_d, \Sigma_o, f_d, q_0)$ is used to provide estimates of the state and information on the possible faulty events. This is the "discrete-event" part of the stochastic diagnoser, which is used to determine the logical element of the diagnoser state.

- h is the sensor mapping between the states and the readings on the sensors. Suppose there are m sets of sensors to monitor the manufacturing system. Sensor mapping is denoted as $h(x) = (h_1(x), h_2(x), \cdots, h_m(x))$.

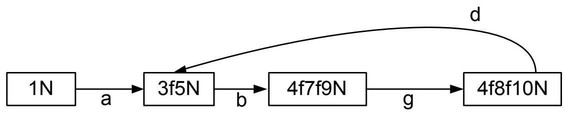
Here let us use a simple example to illustrate the proposed diagnoser design approach. Figure 28 (a) presented an automaton. It has the set of states is $X = \{1, 2, \cdots, 10\}$, and the initial state is $x_0 = 1$. The set of events is $E = \{a, b, g, f\}$, here $E_o = \{a, b, g\}$ and $E_{uo} = \{f\}$. f also belongs to E_f. A transition arc is drawn between two states if the probability of that transition occurring is greater than zero. With the labeled states, we build the diagnoser as Figure 28 (b).

5.4 Fault isolation and detection

The fault isolation reasoning is as follows: for the SFC based control the previous and current machine states and state changes (events) in time sequence are inputs to the fault decision. The current fault state $F(t)$ is determined by the combination of previous step, current step and current input commands. Then we have $F(t) = O(t-1)I(t)O(t)$. Here $F(t) = 0$ means that fault happening with current step. $O(t)$ is also dependent on $O(t-1)$ as well as $I(t)$. Either $O(t-1) = 0$ or $I(t) = 0$ won't make desired $O(t)$ happen. Assume $O(t)$ and $O(t-1)$ are the combined observed output states of all the steps for the t*th* and (t-1)*th* steps respectively. Since the system was decomposed into subsystem $G = G_1 \parallel G_2 \parallel \cdots \parallel G_k$, then we are going to have $O(t) = O_1(t)O_2(t)\cdots O_k(t)$ as well as $O(t-1) = O_1(t-1)O_2(t-1)\cdots O_k(t-1)$. In this equation, $O_i(t)$ means whether the output steps with plant G_i are satisfying or not; $O = 1$ means the current step satisfies and the next step can be started, while $O = 0$ means the current step does not satisfy and desired sequential actions cannot be carried out. Let $I(t)$ be the combined event of input commands in the t*th* step, then $I(t) = I_1(t)I_2(t)\cdots I_k(t)$. Here $I = 1$ means the commands are received, while $I = 0$ means the commands are not received.



(a)



(b)

Figure 28 Finite state automaton plant (a) and its diagnoser(b)

Once abnormality happens on one step, SFC will alarm that step based on $O(t-1), I(t)$ and $O(t)$ information. Then further fault checking will trace into the local sensors to identify the exact fault. In such a way, faults are isolated and detected.

5.5 Case study

5.5.1 System description

The proposed FSA diagnoser design was implemented to diagnose a dual robot system as shown below in Figure 29 [95]. It is a robot work cell including a conveyor, two 4-axis robot arms, four stoppers, two parts feeders, a computer vision and a controller. It mimics the pick and place operation in the assembly process. Here we focus on diagnosing arm1, and the diagnosis on arm2 can follow the similar procedure as on arm1.

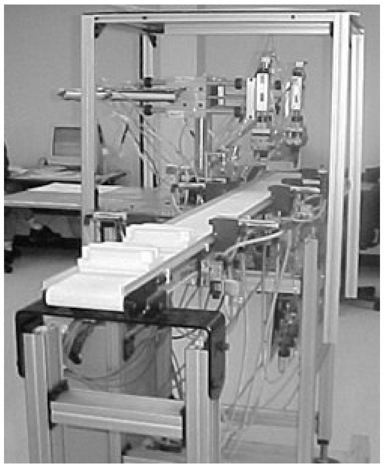


Figure 29 An automated assembly dual robot at Texas A&M University

The robot1 arm is composed of shoulder, elbow, wrist and gripper. The sequential operations for the robot arm are: 1) stopper blocks the pallet and sends signal to initiate arm; 2) open gripper, lower elbow, pick up part from part feeder; 3) raise elbow, extend the shoulder, lower elbow, open gripper; and 4) close gripper, raise elbow, retract shoulder. The control on the arm1 based on sequential function chart (SFC) is show as below in Figure 30. The enable condition and desired outcome in each state is summarized in Table 26.

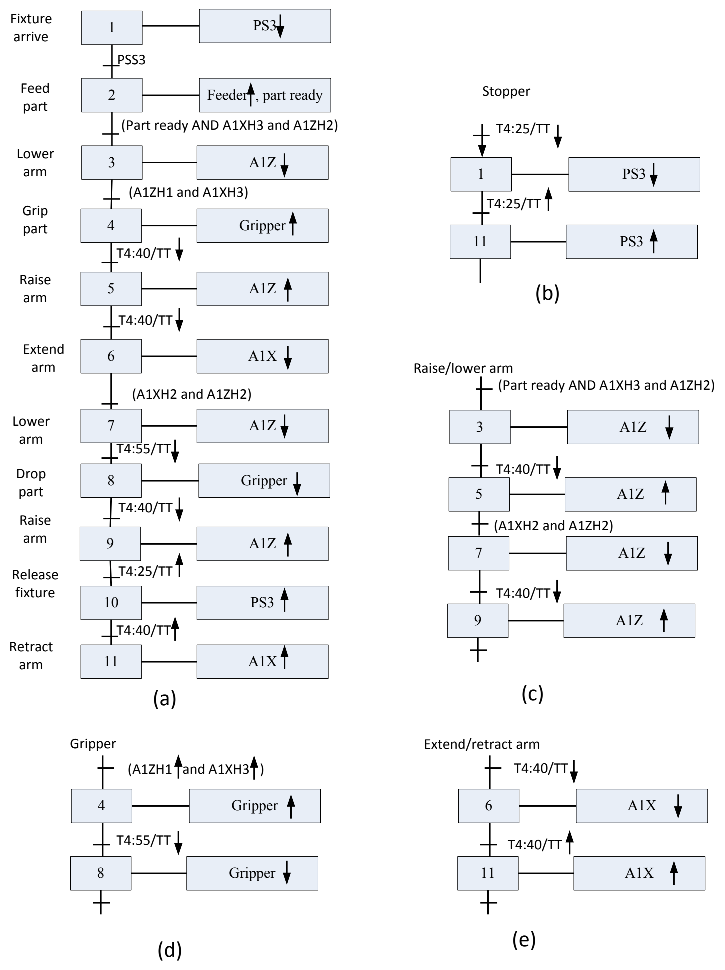
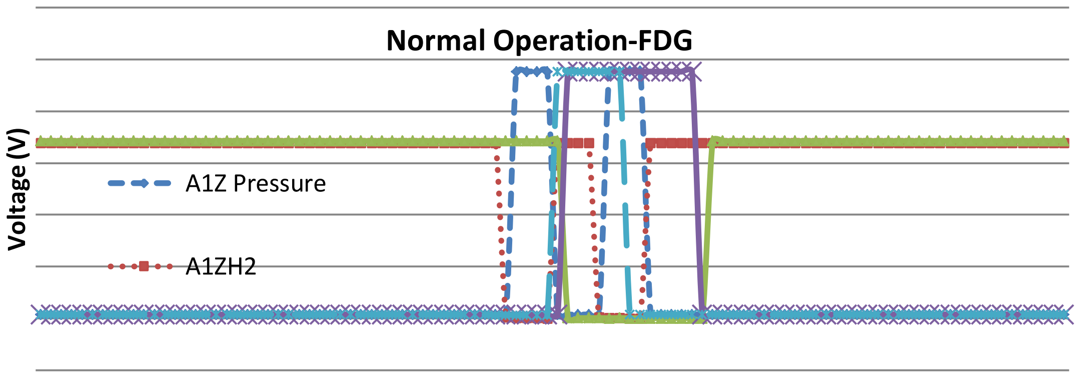


Figure 30 Sequential function chart for dual robot assembly (a) and its components-stopper (b), shoulder (c), gripper (d) and arm (e)

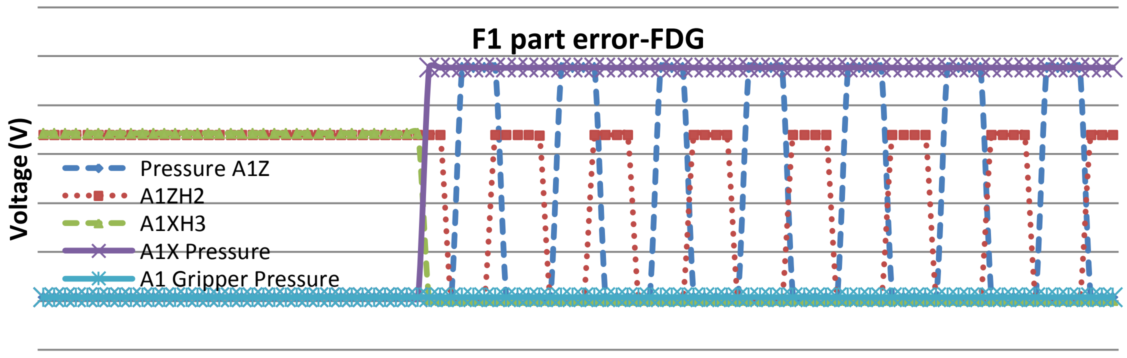
Table 26 Enablement conditions for states in the dual robot arm assembly

	States	**Enabled outcomes**	**Preconditions**
1	Fixture arrive	PSS3 (I:5/10)↑	N/A
2	Part feed	Feeder start (O:1/1)↑	PSS3 (I:5/10)↑
3	Lower arm	A1Z (O:2/10)↑, T4:40/TT↑	PSS3(I:5/10)↑, part(I:4/3)↑, A1XH3(I:5/11)↑, A1ZH2 (I:5/9)↑
4	Grip part	Gripper(O:2/13)↑, T4:55/TT↑	A1ZH1(I:5/13)↑, A1XH3(I:5/11) ↑
5	Raise arm	A1Z(O:2/10)↓	T4:40/TT↓
6	Extend arm	A1X(O:2/9)↓	T4:40/TT↓, A1XH3(I:5/11)↑
7	Lower arm	A1Z(O:2/10)↓, T4:40/TT↑	A1XH2(I:5/7)↑, A1ZH2 (I:5/9)↑
8	Drop part	Gripper(O:2/13)↓	T4:55/TT↓
9	Raise arm	A1Z(O:2/10)↓	T4:40/TT↓
10	Release fixture	PS3(O:1/13)↑, T4:25/TT↑	T4:55/DN↑
11	Retract arm	A1X(O:2/9)↑, T4:11/TT↑	PSS3(I:5/10)↓

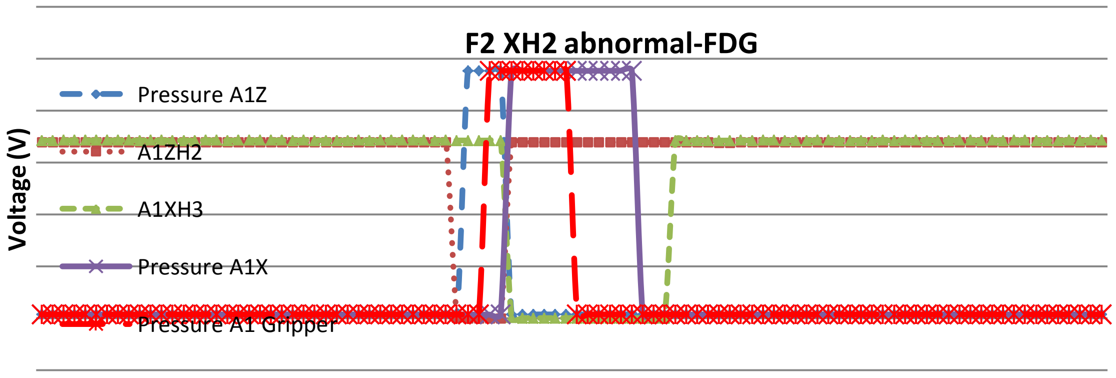
As the analysis from the Chapter III, we used five sensors to monitor the assembly system. These five sensors are [S1, S3, S5, S7, S10]. They measured the signals of: pneumatic pressure on A1X solenoid valve, voltage reading on A1XH3, pneumatic pressure on A1Z solenoid valve, voltage reading on A1ZH2, and pneumatic pressure on gripper solenoid valve. The signal features on these five sensors at different normal/faulty situations are plotted as below Figures 31 (a) ~ (e). We need to note that the pressure values were already calibrated into voltage readings. Because they are discrete signals at either ON/OFF or low/high voltage status, here we use “0” on sensor readings representing OFF and “1” representing ON. The mapping h between the states and sensor readings is as below Table 27.



(a)

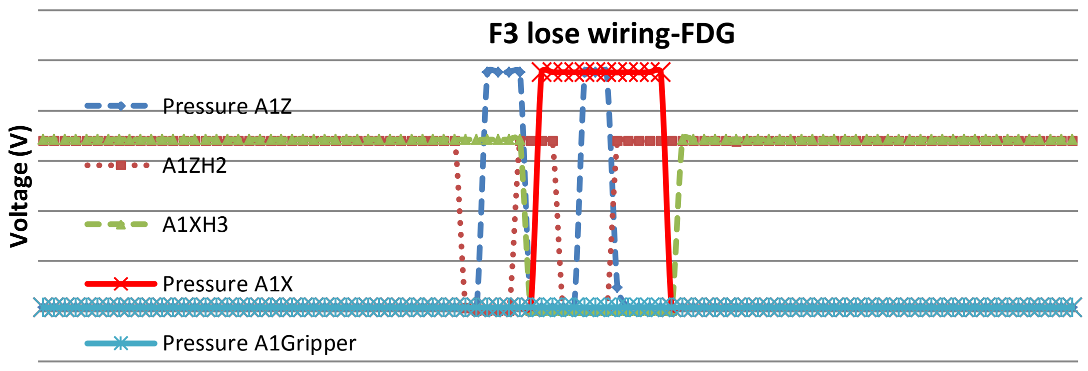


(b)

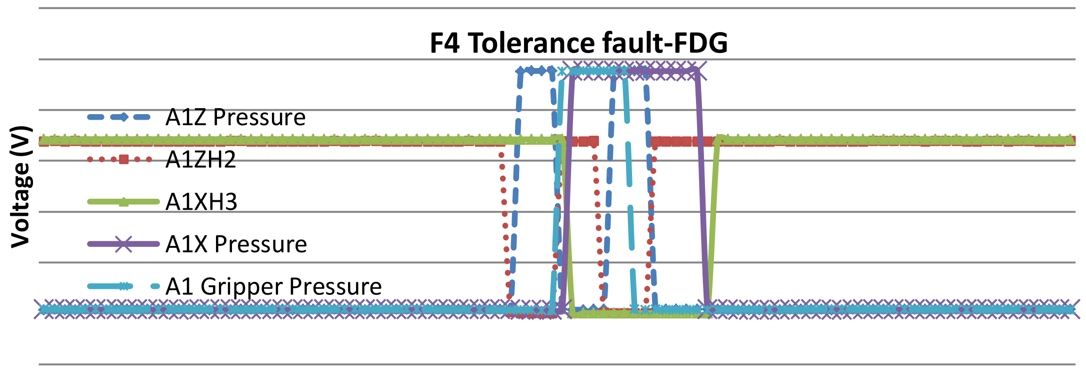


(c)

Figure 31 Signal features for different situations: (a) Normal operation, (b) F1 part error, (c) F2 XH2 abnormal, (d) F3 lose wiring and (e) F4 tolerance fault



(d)



(e)

Figure 31 Continued.

Table 27 Sensor mapping between states and sensors

State	Physical meaning	Readings on S1, S3, S5, S7, S10
1	Fixture arrived	11010
2	Feed part	11010
3	Lower arm	11100
4	Grip part	11101
5	Raise arm	11011
6	Extend arm	00011
7	Lower arm	00101
8	Drop part	00100
9	Raise arm	00010
10	Release fixture	00010
11	Retract arm	11010

Possible faults with the arm1 and their symptoms are tabulated in the Table 28.

Table 28 Possible faults and symptoms with the arm

Faults	**Root causes**	**Symptoms**
Failure to pick up part (F1)	Bad part that cannot be detected by the sensor prior to assembly combined with ladder logic written without accounting for failed inputs.	Robot extends along the X-axis without initially lowering the elbow to grasp the peg. Once extended, the elbow raises and lowers repeatedly. The base part is never released.
Failure to lower arm and pick up parts (F2)	Loss of signal from sensor (A2XH2) that detect the arm is extended in the X direction, caused due to misplacement of the sensor.	Robot does not lower arm and grasps peg, it extends (along X). but raise and lower arm alternatively in the Z-axis. The fixture will not be released.
Failure to close gripper (F3)	Communication failure between the PLC output port and the solenoid valve controlling the gripper caused due to a disconnected wire.	Gripper fingers are jammed open during the entire assembly sequence while the rest of the operations occur normally. Consequently, the peg is never grasped. The final product is output without a peg.
Insertion failure (F4)	Loss of tolerance between the grasping location of the gripper and the location of the parts in the part feeder. The parts are round and are made of rubber. They push each other on the feeder track. Given minor clearance between the parts and the walls of the track, they move sideways resulting in inaccurate placement of the parts prior to grasping by the robotic gripper.	Insertion failure occurs in the form of a non- insertion of the peg into the hole similar to jamming or scratching of the edge of the hole in the base part by the peg before insertion. In the first couple of runs, the pegs under-travel the hole and in the last couple of runs the pegs over-travel the hole.

When we are conducting the experiment, we found that the occurrences of events under different faults are as below:

F1: feed part-F1-extend arm-lower arm-raise arm-lower arm-….;

F2: feed part-lower arm-pick part-raise arm-extend arm-F2-drop part-release fixture-retract arm;

F3: feed part-lower arm-pick part-raise arm-extend arm-lower arm-drop part-raise arm-release fixture-retract arm;

F4: feed part-lower arm-pick part-raise arm-extend arm-lower arm-drop part-raise arm-release part-retract arm.

Combining the normal occurrence and faulty occurrence of the events, we have the automaton for the dual robot arm as below Figure 32.

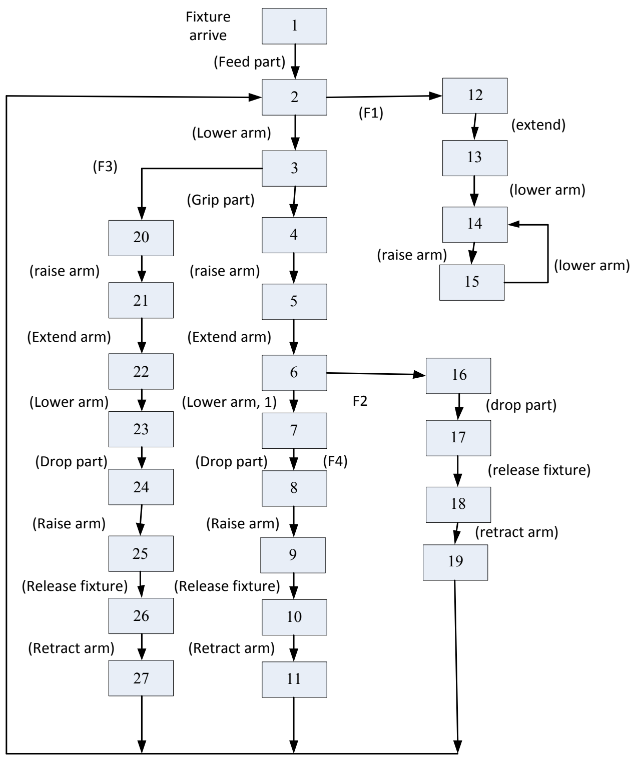


Figure 32 Plant automaton for the dual robot

On this automaton, we first took the projection operator P as follows. Given a timed sequence ρ and a set of events $E' \subseteq E$, $P(\rho, E')$ was the timed sequence obtained by erasing from ρ all events in E' and summing up the delays between successive events in the resulting sequence. This projection is very important in analyzing the manufacturing automaton in that manufacturing process involves multiple parallel actions which may disturb each other and thus confuse the analysis. Such an erasing projection can screen the disturbing sequential events from the analysis, thus making the automaton analysis only focus on the most related events.

Applying the build up process of diagnoser including projection, form labeled state, and sensor mapping, we get the diagnoser as below Figure 33. We will implement this diagnoser in a VB human machine interface to indicate the occurrence of faults.

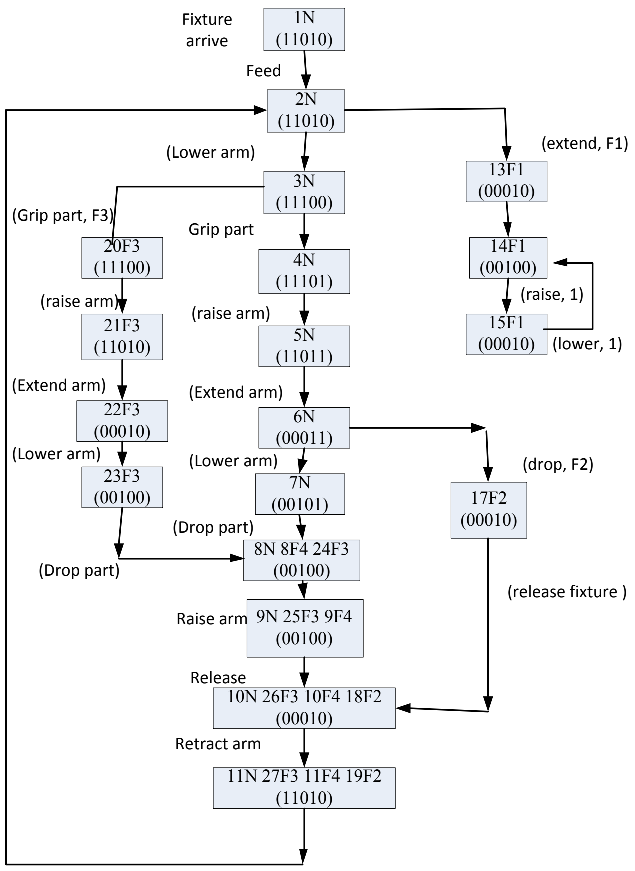


Figure 33 Stochastic diagnoser for the dual robot

5.5.2 System implementation

The above diagnoser structure was implemented into a human machine interface (HMI) as shown in Figure 34. The interface capitalizes the discrete event nature of the automated PLC system. All the inputs and outputs of the dual robot arm system are physically connected to the I/O modules of the PLC and the PLC's communication with the VB interface is through the RS-232 communication port. The serial communications protocol is used to pass commands, information and sensor data reading between the PLC and a personal computer.

"Unprotected read operation" between PLC and PC is applied for PC to read the information on PLC I/O porters. During the unprotected read operation, PLC's MOV commands are used to continually transfer the data contained in the input registers I:5 and I:4 to the registers N9:2 and N9:3 respectively and also the data contained in the output registers O:2 and O:3 to the registers N9:4 and N9:5. This data is used to represent the status of I/Os on the interface. Each register mentioned above is a group of 16 bits or one word. Along with these, several other data registers are used during the 'Read' operation to transfer the accumulated values of all the timers in the process. When an input or output is high, the corresponding bit in the data registers is set, since the contents of the I/O registers are transferred to the data registers by the program. This activates appropriate shape elements (rectangular indicators) under appropriate labels on the interface. For example, if the MSComm detected the bits on "S1, S3, S5, S7, S10" is "11100", we can know from the sensor mapping that the event of "lower arm" had happened, then its rectangular indicator under "lower arm" will turn green to indicate the occurrence of "lower arm".

5.5.3 Normal working status

As shown in Figure 34, when fixture is placing on the conveyor and “Autoread” is checked, the VB HMI starts to read data from the PLC data registers. Along the assembly line, we have four stations: inspection station, buffer station, assembly1 and assembly2. These four stations are respectively represented by a large rectangular indicator beneath the button for the station. Additionally, there are smaller indicators corresponding to every state in the sequence of operations at each station. When the base part moving along the conveyor reaches a station, the large indicator at that particular station turns green, indicating that the process at that station is active. Once the base part leaves the station and all the normal output states corresponding to that station have been successfully carried out in accordance with the control logic, the large indicator turns green. In Figure 34, it illustrates that all the desired operation events/states in sequential function chart for assembly1 and assembly2 were successfully implemented, so all the smaller indicators under the events were turned green.

5.5.4 Abnormality diagnosis

If any of these smaller indicators at the assembly stations turns red, it means that the operation is deviated from the normal status. In conjunction with this, the large indicator for the station on the assembly line containing the failed event turns grey. Then the application will start the fault isolation program. In order to understand how the stage diagram aids diagnosis, let us consider the scenario in which the robot arm fails to lower when extended to insert the peg into the base part. The cause of this fault is the loss of input from the sensor “A1XH2” which senses that the arm is extended along the x-axis.

When an A1XH2 abnormality happens, the event sequence for this abnormality is: Fixture arrive-Feed part-Lower arm-Pick part-Raise arm-Extend arm-F2-Drop part-Release fixture-Retract arm. From this observed sequence of operations under A1XH2 abnormality.

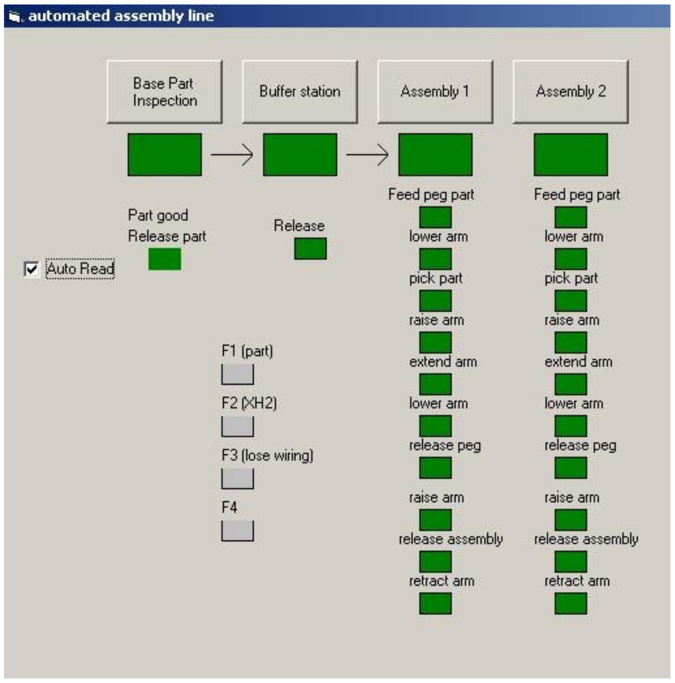
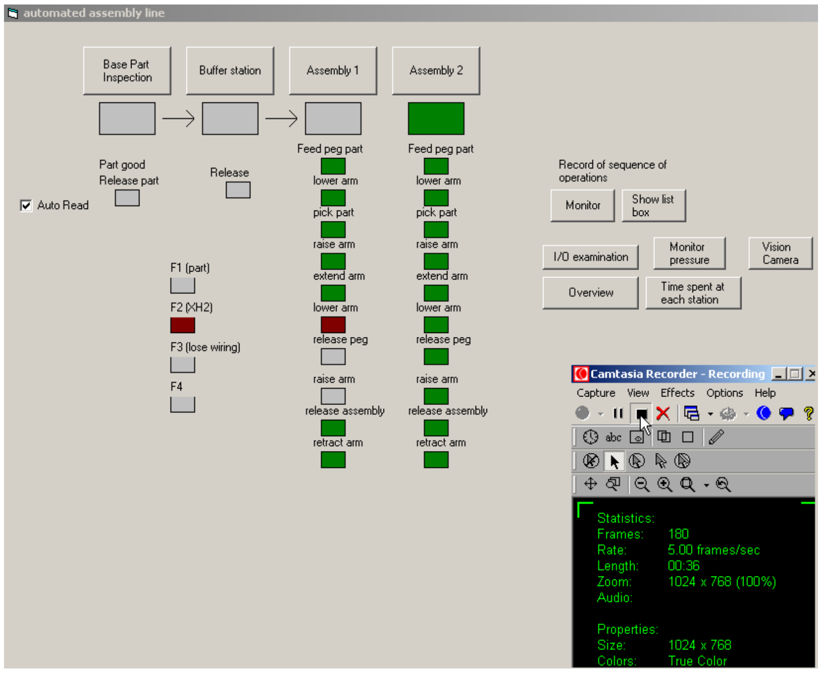


Figure 34 Normal operation on assembly arm1 and 2

We know it skipped step 7). The reason for step 7) not being executed is fault of (A1XH2, or A1ZH2). It will not be A1ZH2's abnormality; otherwise step 3) will not be executed. We even don't need sensor mapping, and we can conclude that it is "A1XH2 fault". The interface corresponding to this failure is illustrated on the stage diagram as shown in Figure 35. The fault indicator "F2 (XH2)" has been turned red. The indicator below the tag "lower arm" has turned red indicating an abnormal event. Also, the indicators for the station ‘Assembly1’, and ‘release peg’ as well as ‘raise arm’ have turned grey indicating an incomplete assembly process on assembly arm1, ‘release peg’ and ‘raise arm’ have not been executed. This helps the diagnoser to isolate the problem to a sensor malfunction which is later confirmed from the time based record of events.



```
Statistics:
Frames:     180
Rate:       5.00 frames/sec
Length:     00:36
Zoom:       1024 x 768 (100%)
Audio:

Properties:
Size:       1024 x 768
Colors:     True Color
```

Figure 35 Operation on XH2 fault

5.6 Experiment results and discussions

Series of experiments on detecting were carried out to validate the proposed approach and system implementation on detecting four kinds of typical faults involved in the dual robot arm. The effectiveness of the diagnoser to these faults was evaluated using the metrics including recognition accuracy, detect delay.

5.6.1 Accuracy evaluation

The recognition accuracy of fault detection is simply defined as Equation (5-3): the ratio of correct fault detections to the total trials. Ten runs of each kind of fault were introduced into the experiment randomly, so the total 40 runs of experiment were conducted.

$$\text{Recognition accuracy}=(\text{total number of correct decisions})/(\text{total number of trials})\times 100\% \quad (5\text{-}3)$$

Table 29 Correctness measurement

Runs	N	F	N	N	F	F	F	N	F	F	Accuracy
F1	F3	F1	N	N	F1	F1	F1	N	F1	F1	90%
F2	F3F2	F2	N	N	F2	F2	F2	N	F2	F2	90%
F3	N	F3	N	N	F3	F3	F3	N	F3	F3	100%
F4	N	N	N	N	N	N	N	N	N	N	40%
Total	40										80%

In Table 29, it summarizes the diagnosis results for faults. For example, if the 'N' columns mean introducing no fault into the system, at this moment, the correct diagnosis outcome is 'N' (normal) only, all the other diagnosis results are wrong. If 'F' columns meet with the F1 row, it means F1 fault were introduced into the experiment at this moment. The correct diagnosis outcome should be 'F1' only. All the other diagnosis results are wrong. The other contents in this table follow the similar explanation.

In the correctness measurement table, we can see that the diagnoser has high accuracy rate over faults F1 (part error), F2 (XH2 abnormal) and F3 (lose wiring) while a poor performance on F4 (tolerance fault). Except the F4, the average accuracy of diagnosing F1~F3 is 93%. This reason is because that F1, F2 and F3 are faults related to equipment error, which is the typical application of DES fault diagnosis. These faults occurrences can be inferred with DES diagnoser design and sensor mapping. While F4 is more related to product quality defect, when F4 occurs, it has the same outcome event sequence as well as sensor mappings with the normal operation. Thus the diagnoser cannot isolate F4 with normal operation. For the product tolerance error, it is usually handled by statistical quality control and there is no report on applying DES approach to model and detect quality defects yet.

5.6.2 Detect delay evaluation

The occurrence of a faulty event F_i should be detected in finite time after its occurrence. If the diagnoser cannot detect a fault, the detect delay on that fault is infinite. A delay table on diagnosing each fault was built to define detection delay quantitatively. In the experiment evaluation, we included the software of Camtasia® (black block shown in Figure 35) to record the indication of events in the VB HMI, as well as the time span between the fault's occurrence and its identification. Table 30 below records the detect delay for the four kinds of faults. This table has a number of columns equal to whether normal or faulty status is included in the experiment, while the number of rows is equal to which kind of fault is included. We can see that the maximum detect delay for the F1, F2, F3 fault is 9 seconds. While the diagnoser cannot detect the tolerance fault, so the detect delay is infinite for fault F4.

Table 30 Detect delay

Runs	N	F	N	N	F	F	F	N	F	F
F1	N/A	9s	N/A	N/A	7s	8s	9s	N/A	9s	8s
F2	N/A	6s	N/A	N/A	7s	6s	6s	N/A	5s	6s
F3	N/A	7s	N/A	N/A	9s	7s	7s	N/A	8s	7s
F4	N/A	∞	N/A	N/A	∞	∞	∞	N/A	∞	∞

5.6.3 Complexity of the DES model

Following [43], we considered the complexity of the DES diagnoser design. Suppose a DES system $G=(Q,\Sigma,\eta,q_0)$ and his diagnoser is $G_d=(Q_d,\Sigma_o\,\eta,q_0)$. For each component of the system, the local diagnosers are $G_{d1}=(Q_{d_1},\Sigma_{o_1}\,\eta,q_{01})$ … $G_{dm}=(Q_{d_m},\Sigma_{o_m}\,\eta,q_{0m})$ codiagnoser $G_T=(Q_T,\Sigma_T,\delta_T,q_0^T)$. And we have that in the most complex situation for the system: $|Q_{\max}|=\max(|Q_d|,|Q_{d_1}|\cdots,|Q_{d_m}|)$, $|\Sigma_{\max}|=\max(|\Sigma_o|,|\Sigma_{o_1}|\cdots,|\Sigma_{o_m}|)$. The illustration of the complexity is shown as Table 31.

Table 31 Complexity of DES model and diagnoser

System	Notation	Number of states	Number of events	Possible number of transitions
$G^{'}$	DES model	$\lvert Q\rvert$	$\lvert\Sigma\rvert$	$\lvert Q\rvert\times\lvert\Sigma\rvert$
$G_D^{'}$	DES Diagnoser	$2^{2\lvert Q_d\rvert}$	$\lvert\Sigma_o\rvert$	$2^{2\lvert Q_d\rvert}\times\lvert\Sigma_o\rvert$
G_d^1	Local DES diagnoser 1	$2^{2\lvert Q_{d1}\rvert}$	$\lvert\Sigma_{o,1}\rvert$	$2^{2\lvert Q_{d1}\rvert}\times\lvert\Sigma_{o,1}\rvert$
…		…		…
G_d^m	Local DES diagnoser m	$2^{2\lvert Q_{dm}\rvert}$	$\lvert\Sigma_{o,m}\rvert$	$2^{2\lvert Q_{dm}\rvert}\times\lvert\Sigma_{o,m}\rvert$
G_T	DES codiagnoser	$2^{2(m+1)\lvert Q_{\max}\rvert}$	$(\lvert\Sigma_{o,\max}\rvert)^{m+1}$	$2^{2(m+1)\lvert Q_{\max}\rvert}\times(\lvert\Sigma_{o,\max}\rvert)^{m+1}$
Complexity of DES approach		$2^{2(m+1)\lvert Q_{\max}\rvert}\times(\lvert\Sigma_{o,\max}\rvert)^{m+1}$		

5.7 Conclusion

In this chapter, we designed and implemented a diagnoser based on finite state automaton and sequential function chart for diagnosing discrete manufacturing systems. Firstly, the system was decomposed into subsystems based on component properties and functionalities. Finite state automaton was used to model the event-state relationship on each subsystem; projection was operated on the automaton to extract the desired information from each subsystem; and the event-state based model was used to infer the occurrence of the faults and locate the possible cause. Finally, a global coordinator based on sequential function chart (SFC) was proposed to fuse the decisions from each local diagnoser to gain a comprehensive view on the whole system. To test the proposed decentralized diagnoser design, a detailed experiment was studied on detecting typical faults on a PLC controlled dual robot system. It illustrated that the proposed diagnoser can detect and isolate most of the DES faults with a high fault detection rate of 93% and maximum fault detection delay of 9 seconds, although it cannot handle the product tolerance fault. The contribution of this work lies on the proposed plant and diagnoser model, which can model observations from complex discrete manufacturing systems, thus detecting stochastically unobservable faults.

CHAPTER VI

COMPREHENSIVE EXPERIMENT EVALUATION ON PROPOSED METHODOLOGIES

6.1 Experiment objective and description

In order to determine how sensor deployment and diagnoser facilitate fault diagnosis in discrete manufacturing systems and to understand the impact of the factors on troubleshooting performance, the objectives were established in the following aspects:

1) To develop a model for evaluating diagnosis performance under alternative combinations of sensor deployments, diagnosers and faults.

2) To study the effect of the sensor deployment strategy on performance with a diagnosis architecture.

3) To study the effect of the diagnoser on performance with a diagnosis architecture.

4) To study the effect of the nature of faults diagnosed on the troubleshooting performance with different diagnostic configurations.

The proposed experiment was carried out on the dual robot system in Rockwell® Automation System Integration Laboratory at Texas A&M University. The dual robot is shown in below Figure 36 [95]. It is a robot work cell including a conveyor, two 4-axis robot arms, four stoppers, two parts feeders, a computer vision and a controller. The actions of those components are controlled by programmable logic controller (PLC). We concentrate more on one stopper (stopper3) and one robot arm (arm1) as they are functioning together, and the diagnosis on stopper4 and arm2 can follow the procedures from arm1.

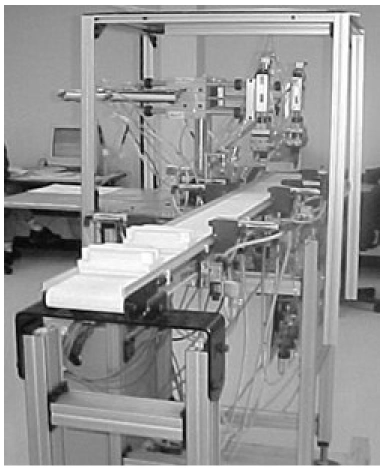


Figure 36 An automated assembly dual robot at Texas A&M University

6.2 Design of experiment

6.2.1 Design table

Factorial design is an experimental methodology which permits researchers to study behavior under conditions in which independent variables vary simultaneously, so the researchers can investigate the joint effect of two or more factors on a dependent variable [96]. The factorial design also facilitates the study of interactions, illuminating the effects of different conditions of the experiment on the identifiable subgroups of subjects participating in the experiment. Specifically, the full factorial design is an experimental design which consists of two or more factors, each with discrete possible values or "levels", and whose experimental units take on all possible combinations of these levels across all such factors.

For this research, there are three variables: Sensor deployment strategy (X1), Diagnoser (X2), and Fault (X3). Detect delay, probability of detection (POD), probability of false alarm (POFA), and accuracy are the response variables. The levels of factors used in the experimental design are listed in Table 32. There are two sensor deployment strategies (2), two diagnosers (2) and four faults (4), so total combinations of experiments are going to be $2 \times 2 \times 4 = 16$. In one combination, I repeated the experiment 10 times. Faults were randomly introduced in experiment as shown in Table 29.

Table 32 Experiment design table

RUN	Sensor deployment strategy	Diagnoser	Fault	Detect delay (Seconds)	POD	POFA	Accuracy
1	1	1	1	8.33	1.00	0.25	0.90
2	1	1	2	6	1.00	0.25	0.90
3	1	1	3	7.5	1.00	0.00	1.00
4	1	1	4	11	0.00	0.00	0.40
5	1	2	1	8	1.00	0.00	1.00
6	1	2	2	3.67	0.92	0.25	0.85
7	1	2	3	6.4	0.83	0.00	0.90
8	1	2	4	11	0.00	0.00	0.40
9	2	1	1	11.00	0.00	1.00	0.00
10	2	1	2	5.33	0.75	1.00	0.45
11	2	1	3	6.5	0.75	0.25	0.75
12	2	1	4	11	0.00	0.00	0.40
13	2	2	1	8.67	0.83	0.50	0.70
14	2	2	2	3	0.75	0.50	0.65
15	2	2	3	6	0.50	0.25	0.60
16	2	2	4	11	0.00	0.00	0.40

The descriptions for the input and output are as following:

Input Variables

X1: sensor deployment strategy. There are two sensor deployment strategies applicable to discrete event systems: (1) optimized sensor deployment results based on fuzzy quantitative directed graph (FQDG, proposed approach) [97]; and (2) optimized sensor deployment results based on sign directed graph (SDG, benchmark approach). They formed the alternatives in X1. Through the SDG methodology, the chosen sensors are (S1, S5, S9, S10), which represent [pressure on A1X valve, pressure on A1Z valve, voltage reading on gripper controller, pressure on gripper valve]. Using the proposed quantitative fuzzy directed graph (FQDG) methodology, the chosen sensors are (S1, S3, S5, S7, S10), which represent [pressure on A1X valve, voltage reading on A1XH3, pressure on A1Z valve, voltage]. The detailed selection process of these two sensor deployment strategies is specified in Appendix II.

X2: Diagnoser. FSA and PN are the two most popular approaches for analyzing discrete event systems. Two kinds of fault diagnosers: 1) diagnoser based on finite state automaton and sequential function chart (FSA+SFC, benchmark), and 2) realtime fuzzy Petri net diagnoser (RTFPN, proposed) were respectively developed to detect faults in discrete manufacturing systems. The construction process can be referred as Chapter IV and V, Appendix II also states how to build up the diagnoser.

X3: Faults. From the observations in Section 2.1, in the robot assembly process there are five typical faults involved: 1) hardware fault, 2) software fault,3) product fault,4) task fault, and 5) tolerance fault. Inserting and grasping a part and keeping hold of it are among the difficult robotic operations and are most susceptible to faults. This is followed by sensing failure and bad parts. These failures would ideally represent typical situations that need to be addressed in the implementation of diagnosis for automated assembly systems. Thus we replicate the faults as Table 33 to represent the typical faults.

Table 33 Fault candidates for experiment design

Faults	Root causes	Symptoms
Failure to pick up part (F1), **represent software fault and task fault**	Bad part that cannot be detected by the sensor prior to assembly combined with ladder logic written without accounting for failed inputs.	Robot extends along the X-axis without initially lowering the elbow to grasp the peg. Once being extended, the elbow raises and lowers repeatedly. The base part is never released.
Failure to lower arm and pick up parts (F2), **represent hardware fault**	Loss of signal from sensor (A1XH2) that detects the arm is extended in the X direction, caused by the misplacement of sensor.	Robot does not lower arm and drop part at the lower position; it extends (along X). but drops the part at the higher position. The fixture will be released finally.
Failure to close gripper (F3), **represent product fault**	Communication failure between the PLC output port and the solenoid valve controlling the gripper caused due to a disconnected wire.	Gripper finger is jammed open during the entire assembly sequence while the rest of the operations occur normally. Consequently, the peg is never grasped. The final product is output without a peg.
Insertion failure (F4), **represent tolerance fault**	Loss of tolerance between the grasping location and the part insertion location in the fixture. The parts are made of rubber. They push each other on the feeder track. Inaccurate placement of the parts prior to grasping by the robotic gripper makes scrape when insertion.	Insertion failure occurs in the form of a non- insertion of the peg into the hole similar to jamming or scratching of the edge of the hole in the base part by the peg before insertion. In the first couple of runs, the pegs under-travel the hole and in the last couple of runs the pegs over-travel the hole.

Output variables

The output variables refer to the measures of fault diagnosis performance. In [98], the authors listed the major performance metrics for diagnostic system include, among others: 1) False positive alarm, 2) False negative miss-detection, 3) Accuracy, and 4) Time delay.

These performance metrics can be calculated using the decision matrix in Table 34. It is based on the hypothesis-testing methodology and represents the possible fault-detection combinations that may occur.

Table 34 Decision matrix for fault-detection evaluation [98]

Outcome	Fault (F1)	No Fault (F0)	Total
Positive (D1) (detected)	a number of detected faults	b number of false alarms	a+b total number of alarms
Negative (D0) (not detected)	c number of missed faults	d number of correct rejection	c+d total number of non-alarms
	a+c total number of faults	b+d total number of fault-free cases	a+b+c+d total number of cases

From Table 34, the decision matrix can be computed readily. The probability of detection (POD) given a fault assesses the detected faults over all potential fault cases:

$$\text{POD}=P(D_1/F_1)=\frac{a}{a+c} \qquad (6\text{-}1)$$

The probability of a false alarm (POFA) considers the proportion of all fault-free cases that trigger a fault detection alarm:

$$\text{POFA}=P(D_1/F_0)=\frac{b}{b+d} \qquad (6\text{-}2)$$

The accuracy metric is used to measure the effectiveness of the diagnostic system in correctly distinguishing between fault-presence and the fault-free condition.

$$\text{Accuracy}= P(D_1/F_1 \;\&\; D_0/F_0)=\frac{a+d}{a+b+c+d} \qquad (6\text{-}3)$$

Time delay is the time span between initiation and the detection of a fault event.

6.2.2 Hypothesis and statistic testing

Statistical analysis of the data involving all the sensor deployment strategies, diagnosers, and faults was performed using the general analysis of variance (ANOVA) in order to test if the different levels of any of the factors are statistically different in terms of the various performance metrics. The datasets collected were assumed to be replicates which allowed the ANOVA analysis to be performed considering the factors-sensor deployments, diagnosers and faults. Three two-way interactions (diagnoser-fault, sensor-fault, sensor-fault) and one three-way interaction (sensor-diagnoser-fault) were conducted. The hypotheses for the ANOVA analysis can be formulated as:

Null hypothesis (H0): There is no significant effect of different levels of a factor on the true average performance.

Alternate hypothesis (H1): At least one level of the factor has significant effect on the true average performance.

The following experimental hypotheses for the main effects are formulated:

1. H0: There is no difference in the diagnosis performance with the SDG and FQDG based sensor deployment strategies.

H1: FQDG based sensor deployment induces better diagnosis performance than SDG based sensor deployment.

2. H0: There is no difference in the diagnosis performance for all the four faults.

H1: Fault-1, 2 and 3 induce better diagnosis performance than fault-4.

3. H0: There is no difference in the diagnosis performance with FSASFC and RTFPN diagnosers.

H1: The RTFPN diagnoser's diagnosis performance is better than FSASFC's.

To compare the performance of sensor deployments and diagnosers in improving the diagnosis performance, the experiment results were examined using two stages of statistical testing: analysis of variance (ANOVA), and Least Significant Difference (LSD) comparison.

First, ANOVA was tested to determine whether there is any significant difference between the alternatives. Significant difference means input variables affect the performance; then paired comparisons between alternatives were conducted. However, if there was no statistically detectable difference, the statistical testing process stopped.

Following, ANOVA was paired comparisons which consider a set of statistical inferences simultaneously. Errors in inference, including confidence intervals that fail to include their corresponding population parameters, or hypothesis tests that incorrectly reject the null hypothesis, are more likely to occur when one considers the family as a whole. As a statistical testing method, LSD is an effective test for detecting differences in means [96]: therefore, it is chosen as a candidate to compare the measured performance mean values of the search methodologies.

Suppose that the data to be analyzed is composed of A groups; a given group is denoted a. The number of observations of the ath group is denoted S_a. If all groups have the same size it is denoted S. The total number of observations is denoted N. The mean of Group a is denoted M_{a+}. From the ANOVA, the mean square of error (i.e., within group) is denoted $MS_{S(A)}$ and the mean square of effect (i.e., between group) is denoted MS_A.

The value of the t statistics evaluating the difference between groups a and a' is equal to

$$t = \frac{M_{a+} - M_{a'+}}{\sqrt{MS_{S(A)}\left(\frac{1}{S_a} + \frac{1}{S_{a'}}\right)}} \qquad (6\text{-}4)\ [96]$$

The ratio t is declared significant at a given α level if the value of t is larger than the critical value for the α level obtained from the t distribution and denoted $t_{\alpha,N-A}$. Rewriting this ratio shows that a difference between the means of Group a and a' will be significant if

$$|M_{a+} - M_{a'+}| > LSD = t_{\alpha,N-A}\sqrt{MS_{S(A)}\left(\frac{1}{S_a} + \frac{1}{S_{a'}}\right)} \qquad (6\text{-}5)\ [96]$$

6.3 Analysis results

6.3.1 Stage I-ANOVA results

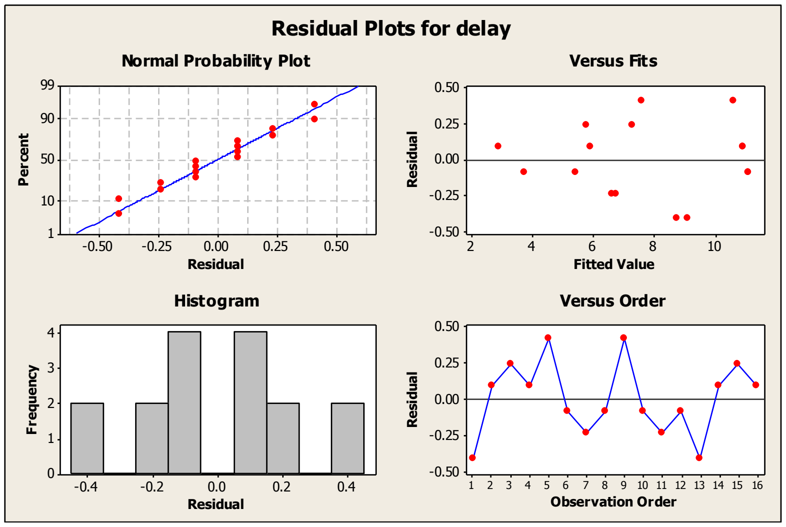
The raw data collected from the experiments were listed in Appendix III and summarized in Table 32. The general linear models on all factors' effects on the diagnosis performance are shown below as Table 35. This table illustrates that sensor deployment strategy and fault are important factors that will influence the diagnosis results, while diagnosers did not show significant impact on the diagnosis performance. Possible reasons for the diagnoser's insignificant difference may be traced to the well-tuned and delicately designed diagnosers . Another conclusion that we can draw from this table is cross-effects (Sensor*Diagnoser, Diagnoser*Fault and Sensor*Fault) are not important factors influencing the diagnosis performance, so we will not perform two-way ANOVA, but only the one-way ANOVA in the later analysis.

Table 35 General linear model

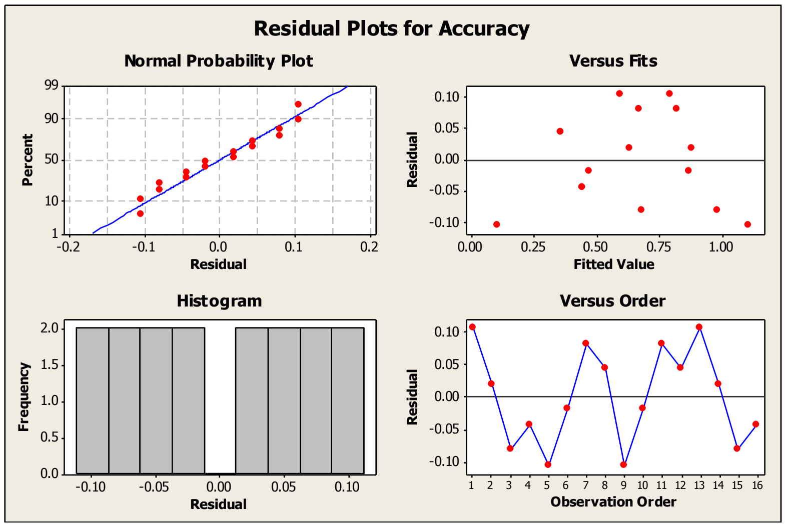
Source of variation (Factors)	Detect delay	POD	POFA	Accuracy
Sensor	F=0.07 P=0.809	F=6.58 P=0.083	F=33 P=0.01	F=12.3 P=0.039
Diagnoser	F=15.42 P=0.029	F=0.16 P=0.719	F=6.82 P=0.08	F=1.49 P=0.31
Fault	F=99.26 P=0.002	F=13.9 P=0.029	F=16.27 P=0.023	F=4.2 P=0.135
Sensor * Diagnoser	F=0.38 P=0.581	F=0.97 P=0.396	F=2.45 P=0.215	F=1.14 P=0.364
Diagnoser* Fault	F=2.96 P=0.198	F=1.58 P=0.357	F=2.45 P=0.24	F=1.63 P=0.349
Sensor * Fault	F=0.83 P=0.15	F=1.31 P=0.414	F=5.36 P=0.101	F=2.3 P=0.256
Error df	3	3	3	3
Total df	15	15	15	15

We need to note the results on the detect delay in Table 32 and Appendix III. Our experiments show that it took 11 seconds for the arm1 to finish the whole assembly process. Theoretically, detect delay is supposed to be infinite if the faults cannot be isolated. When fault occurs, if the fixture is not blocked in the assembly process (for example, when fault 3 or 4 occurs), the detect delay should be smaller than 11 seconds. Otherwise, the fault cannot be detected. In other words, if we cannot detect the faults in 11 seconds after its occurrence, then we cannot isolate them from other faults, so we take

11 seconds as the detect delay for the faults that cannot be detected by our proposed sensor deployment and diagnoser design.

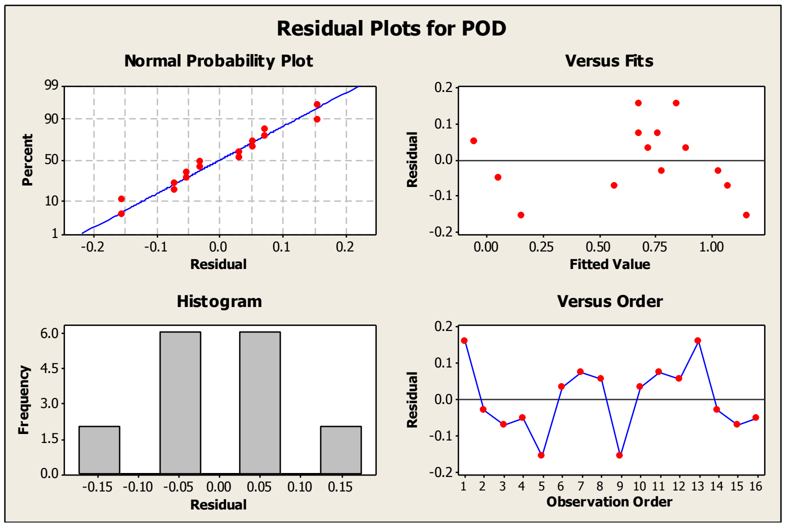


(a)

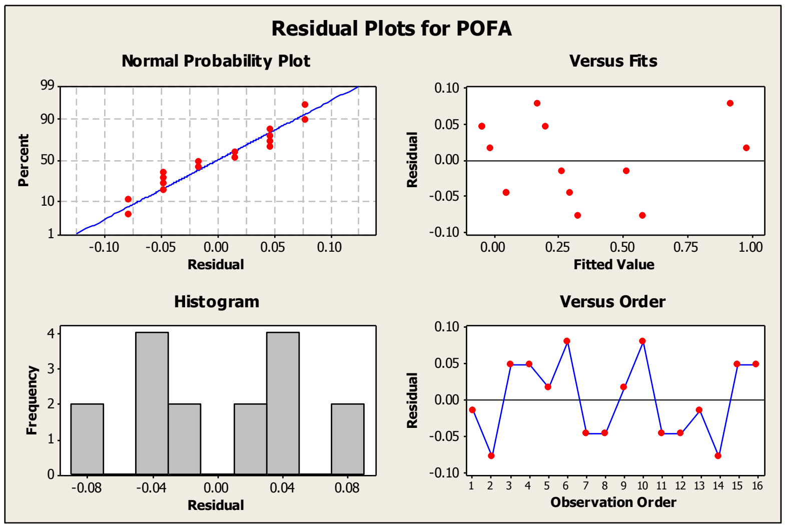


(b)

Figure 37 Residual plots for detect delay (a), accuracy (b), positive detection (c) and false alarm (d)



(c)



(d)

Figure 37 Continued.

To check the normality assumptions, we draw residual plots (Figure 37) for the detect delay, accuracy, POD, and POFA. Normality assumptions usually are checked with normal probability plots or histogram plots. The histogram can be made to check the normality. However, with small samples, considerable fluctuation often occurs, so the appearance of a moderate departure from normality does not necessarily imply a serious violation of the assumption. Gross deviations from normality are potentially serious and require further analysis [96]. Another useful procedure is to construct a normal probability plot of residuals. If the underlying error distribution is normal, this plot will resemble a straight line. In Figure 37, the residuals distributed closely along a straight line in the normal probability plots, with no abnormalities observed in the plot. Thus, the experiments fit a normal distribution. This implies that the experiments are valid with ANOVA to analyze the results. To identify the hypothesis we raised before, we will do the paired comparisons.

1) One-way ANOVA on Accuracy
One-way ANOVA: Accuracy versus Diagnoser

```
Source     DF      SS      MS     F      P
diagnoser   1  0.0400  0.0400  0.51  0.487
Error      14  1.0975  0.0784
Total      15  1.1375

S = 0.2800   R-Sq = 3.52%   R-Sq(adj) = 0.00%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level  N    Mean   StDev  -----+---------+---------+---------+----
1      8  0.5875  0.3281  (-------------*-------------)
2      8  0.6875  0.2216         (-------------*-------------)
                          -----+---------+---------+---------+----
                             0.45      0.60      0.75      0.90

Pooled StDev = 0.2800
```

One-way ANOVA: Accuracy versus Sensor

```
Source  DF      SS      MS     F      P
Sensor   1  0.3306  0.3306  5.74  0.031
Error   14  0.8069  0.0576
Total   15  1.1375

S = 0.2401   R-Sq = 29.07%   R-Sq(adj) = 24.00%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
```

```
Level  N    Mean   StDev  ----+---------+---------+---------+-----
1      8  0.7813  0.2390              (--------*--------)
2      8  0.4937  0.2412  (--------*--------)
                          ----+---------+---------+---------+-----
                            0.40      0.60      0.80      1.00

Pooled StDev = 0.2401
```

One-way ANOVA: Accuracy versus Fault

```
Source  DF      SS      MS     F      P
fault    3  0.3387  0.1129  1.70  0.221
Error   12  0.7987  0.0666
Total   15  1.1375

S = 0.2580   R-Sq = 29.78%   R-Sq(adj) = 12.23%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level  N    Mean   StDev  -----+---------+---------+---------+----
1      4  0.6500  0.4509            (----------*----------)
2      4  0.7125  0.2056              (-----------*----------)
3      4  0.7875  0.1436                 (-----------*----------)
4      4  0.4000  0.0000  (----------*----------)
                          -----+---------+---------+---------+----
                             0.25      0.50      0.75      1.00

Pooled StDev = 0.2580
```

From the ANOVA, significant differences were observed for sensors' detection accuracy at the significance level of 0.05 among different alternatives. However, there is no significant difference on different diagnosers' and faults' effects on the diagnosis accuracy. Further comparison will be conducted on accuracy for the sensors.

2) One-way ANOVA on POD
One-way ANOVA: POD versus Sensor

```
Source  DF     SS     MS     F      P
Sensor   1  0.293  0.293  1.69  0.214
Error   14  2.429  0.173
Total   15  2.722

S = 0.4165   R-Sq = 10.78%   R-Sq(adj) = 4.41%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level  N    Mean   StDev  -----+---------+---------+---------+----
1      8  0.7188  0.4475             (------------*-----------)
```

```
2      8  0.4479  0.3830  (------------*------------)
                          -----+---------+---------+---------+----
                             0.25      0.50      0.75      1.00

Pooled StDev = 0.4165
```

One-way ANOVA: POD versus Diagnoser

```
Source      DF     SS     MS     F      P
diagnoser    1  0.007  0.007  0.04  0.853
Error       14  2.715  0.194
Total       15  2.722

S = 0.4404   R-Sq = 0.26%   R-Sq(adj) = 0.00%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level  N    Mean   StDev  ---------+---------+---------+---------+
1      8  0.5625  0.4772  (----------------*----------------)
2      8  0.6042  0.4003     (---------------*----------------)
                          ---------+---------+---------+---------+
                                 0.40      0.60      0.80      1.00

Pooled StDev = 0.4404
```

One-way ANOVA: POD versus Fault

```
Source  DF      SS      MS     F      P
fault    3  1.8576  0.6192  8.59  0.003
Error   12  0.8646  0.0720
Total   15  2.7222

S = 0.2684   R-Sq = 68.24%   R-Sq(adj) = 60.30%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level  N    Mean   StDev  -------+---------+---------+---------+--
1      4  0.7083  0.4787                   (-------*------)
2      4  0.8542  0.1250                       (------*-------)
3      4  0.7708  0.2083                     (------*-------)
4      4  0.0000  0.0000  (------*------)
                          -------+---------+---------+---------+--
                               0.00      0.40      0.80      1.20

Pooled StDev = 0.2684
```

From the ANOVA, we find that at the significance level of 0.05 there is no significant difference on different sensor deployment strategies' effects and diagnosers' effects on the POD. However significant differences were observed among faults. Further LSD comparison will be conducted on POD for the faults.

3) One-way ANOVA on POFA
One-way ANOVA: POFA versus Sensor

```
Source  DF      SS      MS     F      P
Sensor   1  0.4727  0.4727  5.46  0.035
Error   14  1.2109  0.0865
Total   15  1.6836

S = 0.2941   R-Sq = 28.07%   R-Sq(adj) = 22.94%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level  N    Mean   StDev  ------+---------+---------+---------+---
1      8  0.0938  0.1294  (----------*----------)
2      8  0.4375  0.3953                   (----------*----------)
                          ------+---------+---------+---------+---
                              0.00      0.20      0.40      0.60

Pooled StDev = 0.2941
```

One-way ANOVA: POFA versus diagnoser

```
Source     DF     SS     MS     F      P
diagnoser   1  0.098  0.098  0.86  0.369
Error      14  1.586  0.113
Total      15  1.684

S = 0.3366   R-Sq = 5.80%   R-Sq(adj) = 0.00%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level  N    Mean   StDev  ---+---------+---------+---------+------
1      8  0.3438  0.4213         (------------*------------)
2      8  0.1875  0.2216  (-----------*------------)
                          ---+---------+---------+---------+------
                           0.00      0.20      0.40      0.60

Pooled StDev = 0.3366
```

One-way ANOVA: POFA versus fault

```
Source  DF      SS      MS     F      P
fault    3  0.6992  0.2331  2.84  0.083
Error   12  0.9844  0.0820
Total   15  1.6836

S = 0.2864   R-Sq = 41.53%   R-Sq(adj) = 26.91%
```

```
                           Individual 95% CIs For Mean Based on Pooled StDev
Level  N    Mean   StDev     +---------+---------+---------+---------
1      4  0.4375  0.4270                 (----------*---------)
2      4  0.5000  0.3536                   (----------*---------)
3      4  0.1250  0.1443         (---------*----------)
4      4  0.0000  0.0000     (---------*---------)
                             +---------+---------+---------+---------
                          -0.30      0.00      0.30      0.60

Pooled StDev = 0.2864
```

From the ANOVA, we find significant differences on different sensor deployment strategies' effects ($\alpha = 0.05$) and faults' effects ($\alpha = 0.1$) on the diagnosis POFA. However no significant difference was observed among diagnosers. Further LSD comparison will be conducted on the POFA for the sensors and faults.

4) One-way ANOVA on Delay
One-way ANOVA: delay versus Sensor

```
Source  DF      SS    MS     F      P
Sensor   1    0.02  0.02  0.00  0.958
Error   14  108.66  7.76
Total   15  108.69

S = 2.786   R-Sq = 0.02%   R-Sq(adj) = 0.00%

                         Individual 95% CIs For Mean Based on
                         Pooled StDev
Level  N   Mean  StDev  ---+---------+---------+---------+------
1      8  7.737  2.482  (----------------*-----------------)
2      8  7.813  3.060  (-----------------*-----------------)
                        ---+---------+---------+---------+------
                         6.0       7.2       8.4       9.6

Pooled StDev = 2.786
```

One-way ANOVA: Delay versus Diagnoser

```
Source     DF      SS    MS     F      P
diagnoser   1    4.97  4.97  0.67  0.426
Error      14  103.71  7.41
Total      15  108.69

S = 2.722   R-Sq = 4.58%   R-Sq(adj) = 0.00%


                         Individual 95% CIs For Mean Based on
                         Pooled StDev
Level  N   Mean  StDev  ------+---------+---------+---------+---
```

```
1      8  8.332  2.386          (-------------*------------)
2      8  7.218  3.020  (-------------*-------------)
                        ------+---------+---------+---------+---
                            6.0       7.5       9.0      10.5

Pooled StDev = 2.722
```

One-way ANOVA: Delay versus Fault

```
Source  DF      SS     MS      F      P
fault    3   96.03  32.01  30.35  0.000
Error   12   12.66   1.05
Total   15  108.69

S = 1.027   R-Sq = 88.36%   R-Sq(adj) = 85.44%


                         Individual 95% CIs For Mean Based on
                         Pooled StDev
Level  N    Mean  StDev  ------+---------+---------+---------+---
1      4   9.000  1.361                  (---*---)
2      4   4.500  1.400  (---*---)
3      4   6.600  0.638          (---*----)
4      4  11.000  0.000                          (---*---)
                         ------+---------+---------+---------+---
                             5.0       7.5      10.0      12.5

Pooled StDev = 1.027
```

From the ANOVA, we find no significant difference on different sensor deployment strategies' effects on detect delay at the significance level of 0.05. Neither significant difference on the diagnosers' effects was found. However, significant difference was observed among different faults ($\alpha = 0.05$). Further LSD comparison will be conducted on the detect delay for the faults.

6.3.2 Stage II-LSD comparison results

1) Least Significant Difference Analysis on accuracy for Sensor ($\alpha = 0.05$)

Table 36 LSD of pairwise comparisons in accuracy with sensors

Experimental Group		
	FQDG	SDG
FQDG	0.00	0.3
SDG		0.00

The LSD for accuracy is calculated as below

$$LSD_{Acc} = t_{\alpha,N-a}\sqrt{MS_E\left(\frac{1}{S_a}+\frac{1}{S_{a'}}\right)} = 1.761\sqrt{0.0576\left(\frac{1}{8}+\frac{1}{8}\right)} = 0.211$$

From Table 36 and LSD for accuracy with sensors, we conclude that FQDG is better than SDG on accuracy at the significance level of 0.05.

2) Least Significant Difference Analysis on POFA for Sensor ($\alpha = 0.05$)

Table 37 LSD of pairwise comparisons in POFA with sensors

	Experimental Group	
	FQDG	SDG
FQDG	0.00	-0.34
SDG		0.00

The LSD for POFA is calculated as below

$$LSD_{POFA} = t_{\alpha,N-a}\sqrt{MS_E\left(\frac{1}{S_a}+\frac{1}{S_{a'}}\right)} = 1.761\sqrt{0.0865\left(\frac{1}{8}+\frac{1}{8}\right)} = 0.259$$

From Table 37 and LSD for POFA with sensors, we conclude that FQDG is better than SDG on accuracy at the significance level of 0.05.

3) Least Significant Difference Analysis on delay for faults ($\alpha = 0.05$)

Table 38 LSD of pairwise comparisons in detect delay with faults

	Experimental Group			
	Fault1	Fault2	Fault3	Fault4
Fault 1	0.00	4.5	2.4	-2
Fault 2		0.00	-2.1	-6.5
Fault 3			0.00	-4.4
Fault 4				0.00

The LSD for delay is calculated as below

$$LSD_{delay} = t_{\alpha,N-a}\sqrt{MS_E\left(\frac{1}{S_a}+\frac{1}{S_{a'}}\right)} = 1.782\sqrt{1.05\left(\frac{1}{4}+\frac{1}{4}\right)} = 1.29$$

From Table 38 and LSD for detect delay with faults, we concluded that it takes the least amount of time to detect fault 2, then follows are fault 3 and fault 1 sequentially. Fault 4 takes the longest time to be detected.

4) Least Significant Difference Analysis on POFA for Faults ($\alpha = 0.1$)

Table 39 LSD of pairwise comparisons in POFA with faults

	Experimental Group			
	Fault1	Fault2	Fault3	Fault4
Fault 1	0.00	-0.0625	0.3125	0.4375
Fault 2		0.00	0.375	0.5
Fault 3			0.00	0.125
Fault 4				0.00

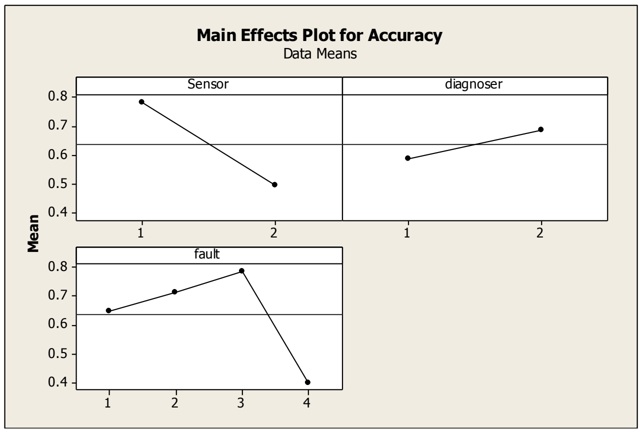
The LSD for POFA is calculated as below

$$LSD_{POFA} = t_{\alpha,N-a}\sqrt{MS_E\left(\frac{1}{S_a}+\frac{1}{S_{a'}}\right)} = 1.356\sqrt{0.082\left(\frac{1}{4}+\frac{1}{4}\right)} = 0.275$$

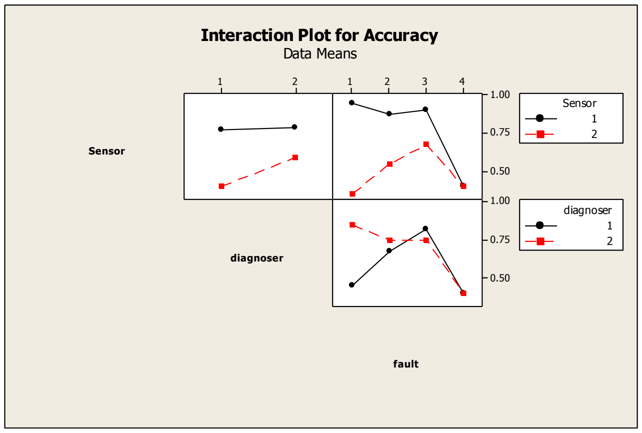
From Table 39 and LSD for POFA with faults, we concluded that fault 1 and 2 have better POFA performance than fault 3 and 4.

6.3.3 Main effects and interaction analysis

On the main effect plots on accuracy, the sensor strategy 1 (FQDG method based sensor deployment) and diagnoser 2 (RTFPN diagnoser) show better performance than the other alternatives. On interactive plots, the combination of strategy 1 and diagnoser 2 also show the same trends that the combination of FQDG and RTFPN diagnoser has the best accuracy performance.



(a)

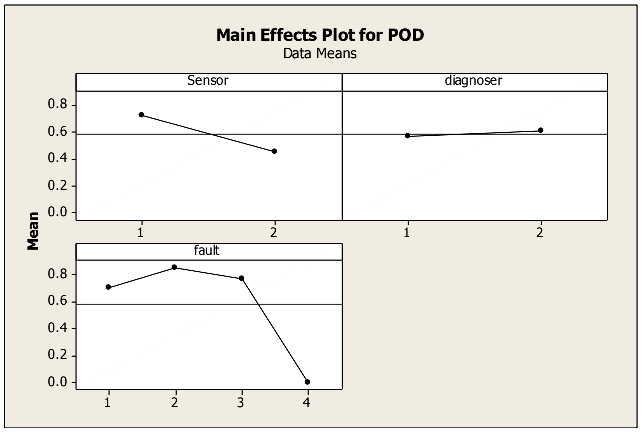


(b)

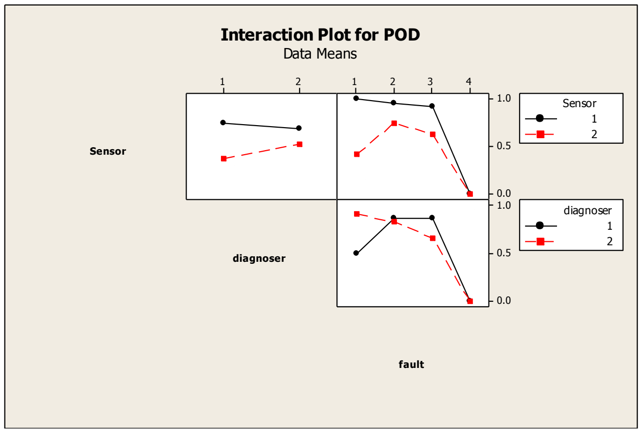
Figure 38 Main effects plots (a) and interaction plots (b) for accuracy

On the main effect plots on accuracy (Figure 38), the sensor strategy 1 (FQDG method based sensor deployment) and diagnoser 2 (RTFPN diagnoser) show better performance than the other alternatives. On interactive plots, the combination of strategy

1 and diagnoser 2 also show the same trends that the combination of FQDG and RTFPN diagnoser has the best accuracy performance.



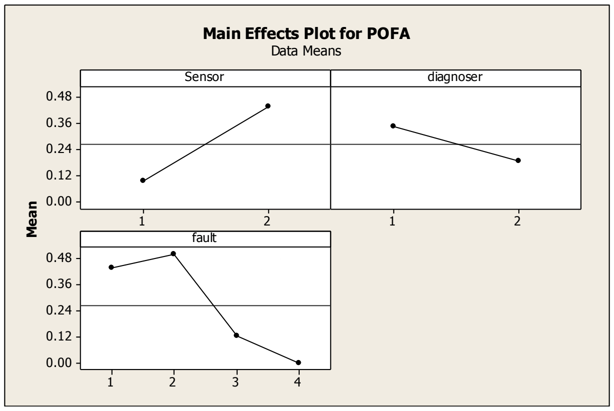
(a)



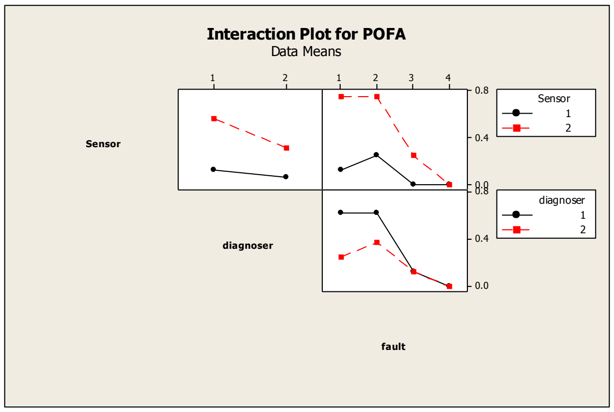
(b)

Figure 39 Main effects plots (a) and interaction plots (b) for POD

On the main effect plots on POD (Figure 39), the sensor strategy 1 (FQDG method based sensor deployment) and diagnoser 2 (RTFPN diagnoser) show better than the other alternatives. On interactive plots, the combination of strategy 1 and diagnoser 2 also show the same trends that the combination of FQDG and RTFPN diagnoser has the best POFA performance.



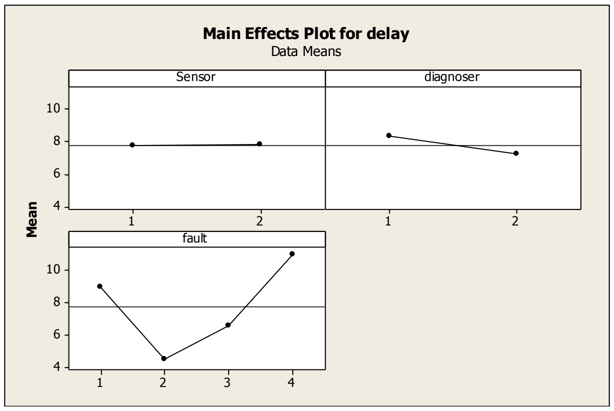
(a)



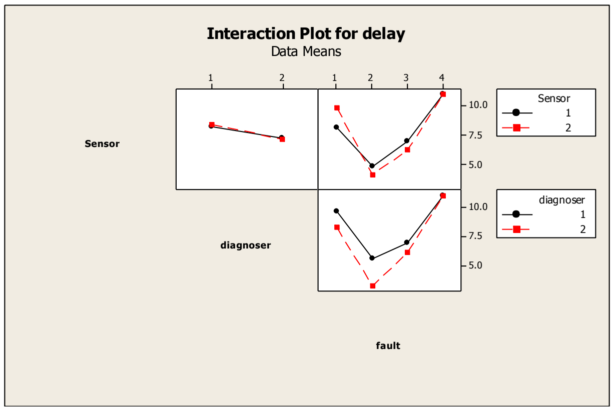
(b)

Figure 40 Main effects plots (a) and interaction plots (b) for POFA

On the main effect plots on POFA (Figure 40), the sensor strategy 1 (FQDG method based sensor selection) and diagnoser 2 (RTFPN diagnoser) show better than the other alternatives. On interactive plots, the combination of strategy 1 and diagnoser 2 also show the same trends that the combination of FQDG and RTFPN has the best POFA performance.



(a)



(b)

Figure 41 Main effects plots (a) and interaction plots (b) for detect delay

On the main effect plots on accuracy (Figure 41), the sensor strategy 1 (FQDG method based sensor deployment) and diagnoser 2 (RTFPN diagnoser) show better than the other alternatives. On interactive plots, the combination of strategy 1 and diagnoser 2 also show the same trends that the combination of FQDG and RTFPN has the best detectdelay performance.

6.4 Conclusion

This chapter presented a comprehensive evaluation of proposed approaches including sensor deployment strategies, and diagnosers on detecting representative faults in a PLC based automated assembly system. The diagnosis performance evaluation metrics are accuracy, false alarm (POFA), positive detection rate (POD) and detect delay, which are the responsive variables in the experiment. The experiment combinations were conducted using design of experiment techniques. Two-stage statistical analysis, including analysis of variance (ANOVA) and least significant difference (LSD), was conducted on the collected responsive variable data. Linear, main, and interaction effects of the diagnosis system variables on diagnosis performance were also determined. It was observed that that when considering the quantitative information from sensors and faults into sensor deployment, fuzzy quantitative directed graph (FQDG) provides better diagnosis performance on accuracy and false alarm than without quantitative information such as SDG. On the diagnoser design, it also shows that the proposed real time fuzzy Petri net diagnoser (RTFPN) has better performance than the diagnoser design based on finite state automaton and sequential function chart.

CHAPTER VII

CONCLUSIONS AND FUTURE DIRECTIONS

In this chapter, we highlight the key contributions of this dissertation and identify potential research directions.

7.1 Significant contributions

The significant contributions of this research are two folds.

On sensor deployment:

1) A systematic strategy to direct the sensor deployment. It started from failure mode effect analysis (FMEA) to obtain the fault information on the system and initially choose sensors. Then quantitative fuzzy graph was used to model sensor deployment. The nodes of the graph were constituted of fault nodes and sensor nodes. The fault nodes contained fault information from FMEA, while sensor nodes contained sensor properties such as reliability, sensitivity, and sensor failure rate. The edges between sensor nodes and fault nodes represented the sensors' fault detectability to certain faults. Finally, lexicographical integer linear programming or greedy algorithms were respectively conducted to optimally assign the sensor nodes to fault nodes, thus optimizing the sensor deployment. The proposed strategy covers all the aspects in sensor deployment for fault diagnosis and can share some light with other applications such as optimal control.
2) A novel sensor deployment approach considering heterogeneous sensor characteristics. Previous sensor deployment approaches only considered whether

there is a connection between the fault node and the sensor node, without considering how the connection forms such as the strength of connections between them. It is a single attribute decision problem. Our proposed sensor deployment approach based on FQDG includes multiple attributes into the sensor deployment. Fault information, sensor characteristics and fault-sensor relationship are included into the cause-effect modeling between faults and sensors. In this approach, the fuzzy graph was used to model the cause-effect relationship between sensors and faults. The heterogeneous information in the sensor deployment was normalized using fuzzy membership function. Through fuzzy normalization, different sensor/fault characteristics were successfully transformed into the comparable values for later decision. In order to handle the trade-off on multiple attributes, the analytical hierarchy process (AHP) was applied to aggregate multiple sensor attributes into single edge values. In this way, the relation between sensor and fault was quantitatively decided to help deploying sensors.

3) Practical ways to optimize the sensor-fault graph targeting multiple objectives. With the fuzzy quantitative graph, we first transformed the nonlinear programming problem into linear optimization, and then we proved that this kind of transformation will not change the optimality of the original problem. To solve the multiple-objective optimization in deployment, two approaches based on lexicographical integer linear programming and greedy algorithms were respectively applied to optimally assign sensors to faults. The effects of these approaches on the optimization results were also discussed for choosing the proper optimization

approach. A case study on a five tank system showed that compared to the signed directed graph, the proposed fuzzy graph based methodology can greatly enhance the detectability to faults (from SDG's 0.62 to fuzzy graph's 0.70). The sensor number was reduced from unoptimized 13 to optimized 3, which greatly reduced the cost. The other case study on a dual robot arm showed how the sensor selection proceeds when undetectability requirements vary. The system's detectability improved from signed directed graph approach's 0.61 to fuzzy quantitative graph's 0.65. It illustrated that the optimization approach can assign sensors to faults in a reliable and cost effective way. It also illustrated that the proposed methodology is applicable to both the continuous time and discrete event type systems.

On real time fuzzy Petri net diagnoser design:

1) A systematic design approach on modeling and diagnosing manufacturing systems with the Petri net. A realtime PN model, aiming at the nonsynchronization problem, was set up to monitor the manufacturing system. The mapping between I/O devices with the PN places avoided the aimlessness of monitoring tasks. In order to integrate the knowledge and handle uncertainties for diagnosis, a fuzzy Petri net diagnoser was proposed to locate the faults.
2) Solid implementation work of the proposed approach on diagnosing a dual robot arm. The proposed diagnoser was implemented into a PN human machine interface using Visual Basic. The troubleshooter can monitor the running status of the whole system or check the very detailed running status of each station conveniently. It illustrated that the implementation of the system is relatively easy. It avoided the complexity

problem in the FSA diagnoser, and it is easy for implementation with distributed forms.

3) A novel evaluation experiment on the accuracy and diagnosability on the PN diagnoser. Sixty runs of experiments on fault detection were randomly carried out on the accuracy of the diagnosis approach; it proved that the proposed system has a high accuracy rate of 93%. Then the detect delay matrix was applied to evaluate to the diagnosability of the diagnoser. It proved that the diagnoser can detect faults within the maximum delay of eight steps. The proposed methodology can remedy the nonsynchronizing between the plant and the diagnoser, and it can handle uncertainties in knowledge integration.

7.2 Future work

Potential extensions for this dissertation research can be explored in the following areas:

1) Automate the system configuration process in the diagnostic/prognostic architecture, including feature selection, system initial configuration and reconfiguration. To enhance the flexibility and agility of manufacturing industries, reconfigurable manufacturing systems (RMS) are promising solutions that can quickly adjust the production capacity and functionality within a part family in response to swift market changes or intrinsic system changes. How to reconfigure the diagnosis system as the manufacturing system's configuration changes will be an interesting research topic. As identified in Chapter I, how to select and fuse signal features also

affects diagnosis performance significantly. Some researchers have explored the feature selection approach through the Taguchi method [2]: the features were selected manually through many experimental observations. Potential work can be carried out in the area of automated feature selection algorithms.

2) Although much research progression has been accomplished in the areas of fault diagnosis and supervisory control of complex discrete event systems, these two problems were tackled separately, and there is no integrated architecture in which control and diagnosis can be addressed simultaneously. When integrating different techniques together, their underlying modeling frameworks and assumptions pose problems to integration. It is important to have a delicately designed architecture, so that such tasks can be accomplished while at the same time existing techniques can be exploited to the greatest possible extent. The work in this dissertation has individually addressed some of these tasks in the diagnosis area. It has the potential to be integrated and used as the basis for a unified architecture.
3) The proposed diagnoser can be extended to handle multiple simultaneous faults. Multiple faults may mask or compensate each other's effects thus hindering the fault isolation. Additionally, it is challenging in that the number of candidates grows exponentially with the number of faults. Our proposed diagnoser is based on the single fault assumption; it can lead to incorrect or failed diagnoses when multiple faults happen. We did not see much research involving multiple faults diagnosis in discrete event systems. How to isolate multiple faults can be a future direction for our research.

4) Study the optimal configuration of sensor system under the life cycle operation cost for the automated manufacturing system. Sometimes increasing more sensors in the diagnosis system may not be cost efficient when the automated system is subjected to the maintenance scheduling, although it can enhance detectability. How to include the sensor cost into the life cycle operation of automated system thus optimizing its maintenance schedule will be an interesting topic in the future.

REFERENCES

1. Pradhan D (1996) Fault tolerant computer system design. Prentice-Hall, New Jersey
2. Al-Habaibeh A, Zorriassatine F, Gindy N (2002) Comprehensive experimental evaluation of a systematic approach for cost effective and rapid design of condition monitoring systems using Taguchi's method. Journal of Materials Processing Technology 124 (3): 372-383
3. Sekar R, Hsieh SJ, Wu Z (2011) Remote diagnosis design for a PLC-based automated system: 1-implementation of three levels of architectures. International Journal of Advanced Manufacturing Technology 57:683-700
4. Sekar R, Hsieh SJ, Wu Z (2011) Remote diagnosis design for a PLC-based automated system: 2-factors affecting remote troubleshooting performance, International Journal of Advanced Manufacturing Technology, under review
5. Cassandras CG, Lafortune S (2008) Introduction to discrete event system. 2nd edition, Springer, New York
6. Lewis F (2007) Intelligent fault diagnosis and prognosis, Automation & Robotics Research Institute, ARRI, http://arri.uta.edu/acs/FL%20talks/Metrocon%2007%20intel%20diag%20and%20prog-%20Lewis.pdf. Accessed on 11 January 2012
7. Raghuraj R, Bhushan M, Rengaswamy R (1999) Locating sensors in complex chemical plants based on fault diagnostic observability criteria. AICHE Journal 45(2): 310-322

8. Bhushan M, Rengaswamy R (2000) Design of sensor network based on the signed directed graph of the process for efficient fault diagnosis. Ind. Eng. Chem. Res., 39 (4): 999–1019
9. Bhushan M, Rengaswamy R (2002) Comprehensive design of a sensor network for chemical plants based on various diagnosability and reliability criteria. 1. Framework. Ind. Eng. Chem. Res. 41: 1826-1839
10. Bhushan M, Rengaswamy R (2002) Comprehensive design of a sensor network for chemical plants based on various diagnosability and reliability criteria. 2. applications. Ind. Eng. Chem. Res. 41: 1840-1860
11. Zhang G (2005) Optimum sensor localization/selection in a diagnostic/prognostic architecture. Ph.D. dissertation, Georgia Tech
12. Zhang G, Vachtsevanos G (2007) A methodology for optimum sensor localization/selection in fault diagnosis. In: Proceedings of IEEE Aerospace Conference, 2007, pp 1-8
13. Ding Y, Elsayed EA, Kumara S, Lu JC, Niu F, Shi J (2006) Distributed Sensing for Quality and Productivity Improvements. IEEE Transactions on Automation Science and Engineering 3(4): 344-359
14. Sherif YS (1982) Optimal maintenance schedules of systems subject to stochastic failure. Microelectronics and Reliability, 22 (1): 15-29
15. Darwiche A, Provan G (1996) Exploiting system structure in model based diagnosis of discrete-event systems. In: Proceedings of 7th International Workshop on Principles of Diagnosis, 1996, pp 95-105

16. Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis D (1996) Failure diagnosis using discrete-event models. IEEE Transactions-Control Systems Technology 4(2): 105-124

17. Sampath M, Sengupta R, Lafortune S (1995) Diagnosability of discrete event systems. IEEE Transactions on Automatic Control 40(9): 1555-1575

18. Zad SH, Kwong RH, Wonham WM (2003) Fault diagnosis in discrete-event systems: framework and model reduction. IEEE Transactions on Automatic Control 48(7): 1199-1212

19. Holloway L, Chand S (1994) Fault monitoring in manufacturing systems using concurrent discrete-event observations. AAAI Technical Report, SS-94-04

20. Holloway, LE, Chand S (1996) Distributed fault monitoring in manufacturing systems using concurrent discrete-event observations. Integrated Computer-Aided Engineering 3(4): 244-54

21. Fade HK, Holloway, LE (1999) Using SPC and template monitoring method for fault detection and prediction in discrete event manufacturing systems. International Symposium on Intelligent Control/Intelligent Systems and Semiotics, 1999, pp 150-155

22. Sayed-Mouchaweh M, Philippot A, Carre-Menetrier V (2008) Decentralized diagnosis based on Boolean discrete discrete event models: application on manufacturing systems. International Journal of Production Research 46(19): 5469-5490

23. Cabasino MP (2009) Diagnosis and identification of discrete event systems using Petri nets. Ph.D. dissertation, University of Cagliari

24. Fanti MP, Seatzu C (2008) Fault diagnosis and identification of discrete event systems using Petri nets. In: Proceedings of the 9th International Workshop on Discrete Event Systems, 2008, pp 432-435

25. Ushio T, Onishi I, Okuda K (1998) Fault detection based on Petri net models with faulty behaviors. IEEE International Conference on System, Man and Cybernetics, vol. 1, pp 113-118

26. Chung SL (2005) Diagnosing PN-based models with partial observable transitions. Int J Comput Integr Manuf 18:158–169

27. Lefebvre D, Delherm C (2007) Diagnosis of DES with Petri net models. IEEE Transactions on Automation Science and Engineering 4(1): 114-118

28. Ruiz-Beltrµan A, Rivera-Rangel I, Lopez-Mellado E (2007) Online fault diagnosis of discrete event systems: a Petri net based approach. IEEE Transactions on Automation Science and Engineering, 4(1):31–39

29. Looney CG (1988) Fuzzy Petri nets for rule-based decision making. IEEE Transactions on Systems, Man and Cybernetics 18(1): 178-183

30. Kuo CH, Huang HP (2000) Failure modeling and process monitoring for flexible manufacturing systems using colored timed Petri nets. IEEE Transactions on Robotics and Automation 16(3):301-312

31. Angel M, Martínez T, Moreno EG (2008) Fault diagnosis and modeling of the liquids packaging process-a research based on Petri nets. 10th Intl. Conf. on Control, Automation, Robotics and Vision, 2008, pp 1620-1624

32. Wen YL, Chung SL, Jeng LD, Jeng MD (2007) Intelligent design of diagnosable systems: a case study of semiconductor manufacturing machines. Knowledge-Based Intelligent Information and Engineering Systems, Part II, LNAI 4693, pp 877–884

33. Prock J (1991) A new technique for fault detection using Petri nets. Automatica 27(2): 239–245

34. Yao WL, Liao HT, Chi SC, Peng SS (2005), A Petri net based offline simulation and online diagnostic platform for manufacturing system. Journal of the Chinese Institute of Industrial Engineers 22(1): 64-75

35. Yao WL (2005) Design and implementation of Web-based diagnosis and management system for an FMS. Int J Adv Manuf Technol 26:1379–1387

36. Hu H, Huang D, Hu H, Sun G (2004) Combining modeling and fault detection in automated manufacturing systems based on hybrid Petri net. In: Proceedings of the 2004 International Conference on Intelligent Mechatronics and Automation, 2004, pp 728-732

37. Genc S (2006) Distributed diagnosis of places-boundered Petri nets. Ph.D. dissertation, University of Michigan

38. Ru Y, Hadjicostis CN (2009) Fault diagnosis in discrete event systems modeled by partially observed Petri nets. Discrete Event Dynamic System 19:551–575

39. Lefebvre D, Leclercq E (2011) Stochastic Petri net identification for the fault detection and isolation of discrete event systems. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 41(2):213-224

40. Lunze J, Schroder J (2001) State observation and diagnosis of discrete-event systems described by stochastic automata. Discrete Event Dynamic Systems: Theory and Applications, 11(4): 319–369

41. Lunze J, Supavatanakul P (2004) Diagnosis of timed automata with an application to industrial actuators. Integrated Computer-Aided Engineering 11: 25-36

42. Thorsley D (2006) Applications of stochastic techniques to partially observed discrete event systems. Ph.D. dissertation, University of Michigan

43. Liu F, Qiu D, Xing H, Fan Z (2008) Decentralized diagnosis of stochastic discrete event systems. IEEE Transactions on Automatic Control 53(2): 535-546

44. Chang SJ, DiCesare F, Goldbogen G (1991) Failure propagation trees for diagnosis in manufacturing systems. IEEE Transactions on Systems, Man and Cybernetics 21(4): 767-776

45. Stanton MJ (1999) A fault monitoring architecture for the diagnosis of hardware and software faults in manufacturing systems. In: Proceedings of 7th IEEE International Conference on Emerging Technologies and Factory Automation, 1999, vol. 1, pp 693-701

46. Holloway LE, Krogh BH (1990) Fault detection and diagnosis in manufacturing systems: A behavioral model approach. In: Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing, 1990, pp 252-259

47. Sekar R (2010) Analysis of remote diagnosis architecture for a PLC based automated assembly aystem. Master's thesis, Texas A&M University, College Station TX

48. Ming C, Jianzhi Z, Wenhan Q (1997) Fault diagnosis system for automated assembly line. In: Proceedings of IEEE International Conference on Intelligent Processing Systems, 1997, vol.2, pp 1478-1482

49. Hardy N, Barnes D, Lee M (1989) Automatic diagnosis of task faults in flexible manufacturing systems. Robotica 7: 25-35

50. Althoefer K, Lara B, Zweiri YH, Seneviratne LD (2008) Automated failure classification for assembly with self-tapping threaded fastenings using artificial neural networks. Proceeding Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 222(6): 1081-1095

51. Linderstam C, Soderquist B (1996) Monitoring the generic assembly operation for impact from gripping to finished insertion. In: Proceedings of IEEE International Conference on Robotics and Automation, 1996, vol.4, pp 3330-3335

52. Ali Y., Narasimhan S. (1993) Sensor network design for maximizing reliability of linear processes. AICHE Journal 39(5): 820-828

53. Ali Y., Narasimhan S. (1995) Redundant sensor network design for linear processes. AICHE Journal 1(10): 2237-2249

54. Ali Y., Narasimhan S. (1996) Sensor network design for maximizing reliability of bilinear processes. AICHE Journal 42(9): 2563-2575

55. Ru Y (2010) State estimation and sensor selection in discrete event systems modeled by Petri nets. Ph.D. dissertation, the University of Illinois at Urbana-Champaign

56. Jiang S, Kumar R, Garcia H.E. (2003) Optimal sensor selection for discrete-event systems with partial observation. IEEE Transactions on Automatic Control 48(3): 369–381

57. Park Y (1996) Model based monitoring of discrete event system. Ph.D. dissertation, Purdue University

58. Liu Q (2004) Optimal coordinator sensor placement for estimating mean and variance component and variance component of variation sources. Master's thesis, Texas A&M University

59. Liu Q, Ding Y, Chen Y (2005) Optimal coordinate sensor placements for estimating mean and variance components of variation sources. IIE Transaction 37: 877-889

60. Ding Y, Kim P, Ceglarek D, Jin J (2003) Optimal sensor distribution for variation diagnosis in multi-station assembly processes. IEEE Transactions on Robotics and Automation 19(4): 543-556

61. Edan Y, Nof SY (2000) Sensor economy principles and selection procedures. IIE Transactions on Design and Manufacturing 32 (3): 195-203

62. Li J, Jin J (2010) Optimal sensor allocation by integrating causal models and set-covering algorithms. IIE Transactions 42(8): 564-576

63. Khan A, Ceglarek D (2000) Sensor optimization for fault diagnosis in multi-fixture assembly systems with distributed sensing. Transaction of ASME, Journal of Manufacturing Science and Engineering 122 (1): 215-226

64. Kannan AA, Mao G, Vucetic B (2006) Simulated annealing based wireless sensor network localization with flip ambiguity mitigation. In: Proceedings of IEEE 63rd Vehicular Technology Conference, 2006, pp 1022-1026

65. Kincaid RK, Padula SL (2002) D-optimal designs for sensor and actuator locations. Computers & Operations Research 29: 701–713

66. Swann C, Chattopadhyay A (2006) Optimization of piezoelectric sensor location for delamination detection in composite laminates. Engineering Optimization 38(5): 511-528

67. Sen S, Narasimhan S, Deb K (1998) Sensor network design of linear process using genetic algorithm. Computers Chemical Engineering 22(3): 385-390

68. Osais Y, St-Hilaire MF, Yu R (2008) The minimum cost sensor placement problem for directional wireless sensor networks. In: Proceedings of IEEE 68th Vehicular Technology Conference, 2008, 1-5

69. Bagajewicz MJ (1997) Design and retrofit of sensor networks in process plants. AICHE Journal 43(9): 2300-2306

70. Yassine AA, Ploix S, Flaus JM (2008) A method for sensor placement taking into account diagnosability criteria. International Journal Applied Mathmatics Computer Science 18(4): 497–512

71. Fijany A, Vatan F (2008) A new efficient method for system structural analysis and generating analytical redundancy relations. In: Proceedings of IEEE Aerospace Conference, 2008, pp 1-12

72. Frank P (1990) Fault diagnosis in dynamic systems using analytical and knowledge based redundancy-a survey and some new results. Automatica, vol. 26:459-474

73. Vries RD (1990) An automated methodology for generating a fault tree. IEEE Transactions on Reliability Engineering 39(1):76–86

74. Milde H, Holtz L (2000) Facing diagnosis reality-model based fault tree generation in industrial applications. In: Proceedings of 11th International Workshop on Principles of Diagnosis (DX'00), 2000, pp 147–154

75. Rich S, Venkatsubramanian V (1987) Model-based reasoning in diagnostic expert systems for chemical process plants. Comput. Chem. Eng., 11:111–122

76. Vemuri AT, Polycarpou MM, Diakourtis SA (1998) Neural network based fault detection in robotic manipulators. IEEE Transactions on Robotics and Automation, 14(2): 342-348

77. Inagaki S, Suzuki T, Saito M, Aoki T (2007) Local/global fault diagnosis of event-driven systems based on Bayesian network and timed Markov mode. In: Proceedings of SICE Annual Conference, 2007, pp 540-545

78. Athanasopoulou E, Li L, Hadjicostis N (2010) Maximum likelihood failure diagnosis in finite state machines under unreliable observations. IEEE Transactions on Automatic Control 55(3): 579-593

79. Zhou, MC, Venkatesh K (1998) Modeling, simulation and control of flexible manufacturing systems: a Petri net approach. World Scientific, Singapore

80. Li Q, Zhu L, Xu Z (2007) Fuzzy Petri-Nets based fault diagnosis for mechanical-electric equipment. In: Proceedings of IEEE International Conference on Control and Automation, 2007, pp 2539-2543

81. Sun J, Qin SY, Song YH (2004) Fault diagnosis of electric power systems based on fuzzy Petri Nets. IEEE Transactions on Power Systems, 19(4): 2053-2059

82. Lee KH, First course on fuzzy theory and applications. Springer 2005, pp. 91-126

83. Otto K, Wood K (2001) Product design-techniques in reverse engineering and new product development. Prentice Hall, New Jersey

84. Franklin GF, Powell JD, Emami-Naeini A (2005) Feedback control of dynamic system. 5th edition, Prentice Hall, New Jersey

85. Satty TL (1990) The analytic hierarchy process: planning, priority setting, resource allocation. RWS Publication: Pittsburg

86. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidisc Optim 26: 369–395

87. Miettinen K (1999) Nonlinear multiobjective optimization. Kluwer Academic Publishers, Boston

88. Hillier FS, Lieberman GJ (2005) Introduction to operations research. 8th edition, McGraw-Hill, New York

89. Wu Z, Hsieh SJ (2010) Application of hidden Markov model and sequential function chart to detect fault in discrete manufacturing system. 2010 International Symposium on Flexible Automation, 2010, pp 1-6

90. Angadi SV, Jackson RL, Choe SY, Flowers GT, Suhling JC, Chang YK, Ham JK, Bae J (2009) Reliability and life study of hydraulic solenoid valve. Part 2: experimental study. Engineering Failure Analysis 16: 944–963

91. Nakutis Z, Kaskonas P (2007) Pneumatic cylinder diagnostics using classification methods. In: Proceedings of Instrumentation and Measurement Technology Conference-IMTC, 2007, pp 1-4

92. Li F (2011) Dynamic modeling, sensor placement design, and fault diagnosis of nuclear desalination systems, Ph.D. dissertation, University of Tennessee

93. Yen J, Langari R (1999) Fuzzy logic: intelligence, information, and control. Prentice-Hall, New Jersy

94. Hugh J (2005) Automating manufacturing systems with PLCs, http://engineeronadisk.com/V2/book_PLC/engineeronadisk.html. Accessed on 12 January 2012

95. Hsieh S (2003) Reconfigurable modules, programmable logic controller, and vision system for dual robot assembly workcell design. Journal of Advanced Manufacturing Systems 2(2): 201-228

96. Montgomery DC (2005) Design and analysis of experiments 6th edition. Wiley, INC, New York

97. Wu Z, Hsieh SJ, Li J, (2011) Optimal sensor deployment strategy considering multiple-attributes and heterogeneous sensors for fault diagnosis, submitted to Robotics and Computer Integrated Manufacturing, under review

98. Vachtsevanos G, Lewis FL, Romer M, Hess A, Wu B (2006) Intelligent fault diagnosis and prognosis for engineering systems. Wiley, John & Sons, New Jersey

APPENDIX I

I/O MAPPING IN THE EXPERIMENT

TableA1. Input/output mapping on the PLC I/O module

BReaddata	2^0	I:5/0 not used	BReaddata2	2^0	I:5/8 PSS4
N9:2	2^1	I:5/1 not used	N9:2	2^1	I:5/9 A1ZH2
	2^2	I:5/2 not used		2^2	I:5/10 PSS3
	2^3	I:5/3 not used		2^3	I:5/11 A1XH3
	2^4	I:5/4 not used		2^4	I:5/12 PSS2
	2^5	I:5/5 not used		2^5	I:5/13 A1ZH1
	2^6	I:5/6 not used		2^6	I:5/14 PSS1
	2^7	I:5/7 A1XH2		2^7	I:5/15 not used
BReaddata3	2^0	I:4/0 not used	BReaddata4	2^0	I:4/8 not used
N9:3	2^1	I:4/1 not used	N9:3	2^1	I:4/9 Pressure1
	2^2	I:4/2 not used		2^2	I:4/10 A2XH2
	2^3	I:4/3 part ready		2^3	I:4/11 load cell
	2^4	I:4/4 feeder2 reverse stop		2^4	I:4/12 A2ZH2
	2^5	I:4/5 vision input		2^5	I:4/13 feeder1 forward stop
	2^6	I:4/6 A2XH3		2^6	I:4/14 A2ZH1
	2^7	I:4/7 pressure 3		2^7	I:4/15 feeder1 reverse stop
BReaddata5	2^0	O:1/0 speedH_L	BReaddata6	2^0	O:1/8 feeder2_reverse
N9:5	2^1	O:1/1 feeder1_start	N9:5	2^1	O:1/9 feeder1_start
	2^2	O:1/2 feeder1_reverse		2^2	O:1/10 feeder1_reverse
	2^3	O:1/3 not used		2^3	O:1/11 not used
	2^4	O:1/4 vision output		2^4	O:1/12 PS4
	2^5	O:1/5 not used		2^5	O:1/13 PS3
	2^6	O:1/6 feeder2_start		2^6	O:1/14 PS2
	2^7	O:1/7 not used		2^7	O:1/15 PS1
BReaddata7	2^0	O:2/0 conveyor	BReaddata8	2^0	O:2/8 A2Z
N9:4	2^1	O:2/1 not used	N9:4	2^1	O:2/9 A1X
	2^2	O:2/2 not used		2^2	O:2/10 A1Z
	2^3	O:2/3 not used		2^3	O:2/11 not used
	2^4	O:2/4 not used		2^4	O:2/12 not used
	2^5	O:2/5 not used		2^5	O:2/13 A1 gripper
	2^6	O:2/6 not used		2^6	O:2/14 A2 gripper
	2^7	O:2/7 A2X		2^7	O:2/15 not used

TableA2. Timer and counter mapping on the PLC program

BReaddata9+ BReaddata10: T4:5 (N9:6)	BReaddata11+ BReaddata12: T4:10 (N9:7)
BReaddata13+ BReaddata14: T4:15 (N9:8)	BReaddata15+ BReaddata16: T4:20 (N9:9)
BReaddata17+ BReaddata18: T4:25 (N9:10)	BReaddata19+ BReaddata20: T4:40 (N9:11)
BReaddata21+ BReaddata22: T4:45 (N9:12)	BReaddata23+ BReaddata24: T4:50 (N9:13)
BReaddata25+ BReaddata26: T4:55 (N9:14)	BReaddata27+ BReaddata28: T4:6 (N9:15)
BReaddata29+ BReaddata30: T4:7 (N9:16)	BReaddata31+ BReaddata32: T4:1 (N9:17)
BReaddata33+ BReaddata34: T4:2 (N9:18)	BReaddata35+ BReaddata36: T4:3 (N9:19)
BReaddata37+ BReaddata38: T4:4 (N9:20)	BReaddata39+ BReaddata40: T4:11 (N9:21)
BReaddata41+ BReaddata42: T4:14 (N9:22)	
BReaddata43+ BReaddata44: C5:0 (N9:23)	BReaddata45+ BReaddata46: C5:1 (N9:24)

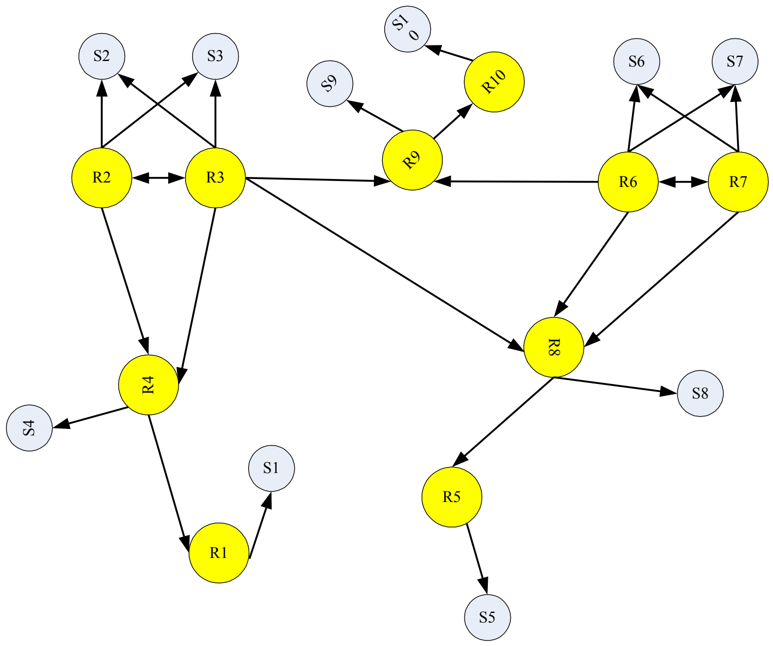
APPENDIX II

NUMERIC EXAMPLE ON THE SENSOR DEPLOYMENT AND DIAGNOSER

A2.1 Sensor deployment

There are two sensor deployments: 1) benchmark method: signed directed graph (SDG), and 2) proposed method: quantitative fuzzy directed graph (FDG).

How to select sensors based on SDG?

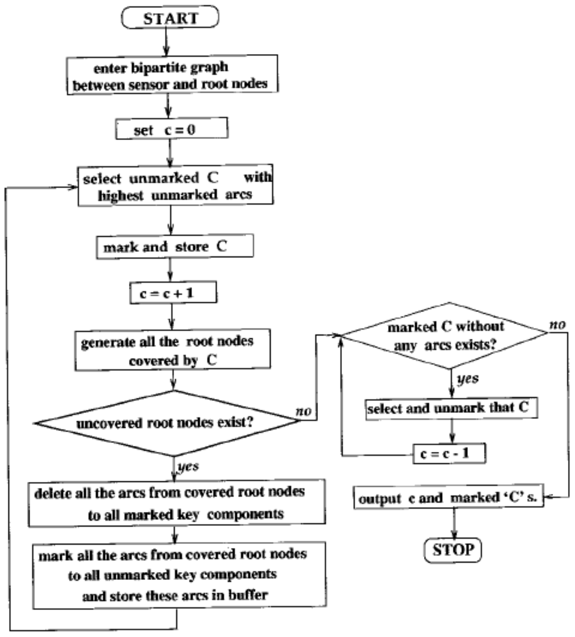


FigureA1. SDG model for sensor deployment on dual robot

TableA3. SDG model for sensor deployment on dual robot

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
R1	1									
R2		1	1	1				1	1	
R3		1	1	1				1	1	
R4	1			1						
R5					1					
R6						1	1	1	1	
R7						1	1	1	1	
R8					1			1		
R9									1	1
R10										1

The formulation of the SDG based cause-effect relationship between faults and sensors is based on the failure mode effect analysis: whether the fault will affect the sensor measurement. The optimization of this SDG based sensor deployment model is according to the greedy algorithm proposed in [21]. The flow chart for the greedy algorithm is as FigureA2.


START
enter bipartite graph between sensor and root nodes
set c = 0
select unmarked C with highest unmarked arcs
mark and store C
c = c + 1
generate all the root nodes covered by C
uncovered root nodes exist?
yes
no
delete all the arcs from covered root nodes to all marked key components
mark all the arcs from covered root nodes to all unmarked key components and store these arcs in buffer
marked C without any arcs exists?
yes
no
select and unmark that C
c = c - 1
output c and marked 'C's.
STOP

FigureA2. Greedy algorithm for SDG based sensor deployment model [21**]**

When applying the algorithm in FigureA2 to optimize the graph in FigureA1, we have the optimization process as below.

Step1): select S8 or S9, because they have the highest unmarked arcs. Now we select S9, mark and store it by C.

Step2): generate all the root nodes covered by C; those root nodes are [R2, R3, R6, R7, R9], release the connection between [R2, R3, R6, R7] with their original sensor node arc connections.

Step 3): identify the uncovered root nodes that are not are [R1, R4, R5, R8, R10]. Repeat step 1), identify the sensor nodes among [S1, S4, S5, S8, S10] with the highest unmarked arcs. It was identified as either S1 or S5.

Step 4): select S1, then the root nodes covered by S1 are [R1, R4]. Release the connection between R4 and S4. Store S1 in C.

Step 5): select S5, then the root nodes covered by S5 are [R5, R8]. Release the connection between R8 and S8. Store S5 in C.

Step 6): uncovered root nodes are R10, select S10 and store it in C.

Finally, the selected sensors based on SDG are [S1, S5, S9, S10]

How to select sensors based on FDG?
Step1): Form the fuzzy graph based sensor deployment model

As mentioned in Chapter III, the FDG based sensor deployment starts from failure mode effect analysis (FMEA), we first collected the fault information about the system and used them to form the fault nodes in the FDG. These fault information is Table A4.

Table A4. Fault nodes in FDG based sensor deployment

Component	Fault Node	Possible Fault	Occurrence Rate	Severity Rate	Detection Rate	RPN	normalized RPN
Shoulder	R1	Cylinder	6	7	2	84	0.083
	R2	Hall sensor1	6	8	9	432	0.431
	R3	Hall sensor2	6	8	9	432	0.431
	R4	Solenoid	2	1	5	10	0.009
Elbow	R5	Cylinder	6	7	2	84	0.083
	R6	Hall sensor1	6	8	9	432	0.431
	R7	Hall sensor2	6	8	9	432	0.431
	R8	Solenoid	2	1	5	10	0.009
Gripper	R9	Solenoid	2	1	5	10	0.009
	R10	Cylinder	6	7	2	84	0.083

FMEA also provided some information on how to treat the faults, so we know what kind of sensor that needs to be selected. The sensor nodes information is in Table A5.

Table A5. Sensor nodes in FDG based sensor deployment

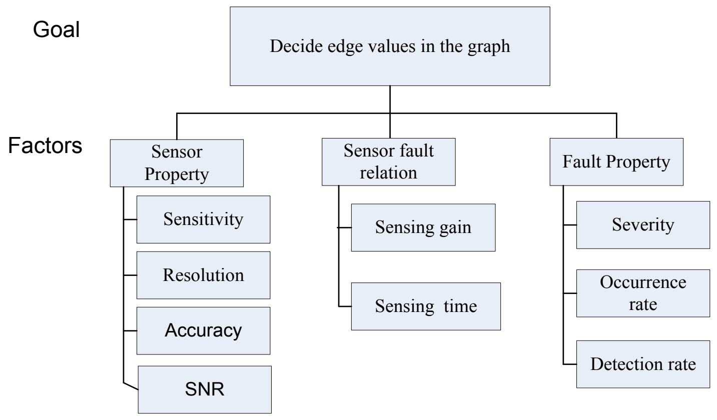
Sensors to detect faults		Sensitivity (full scale)	SNR (dB)	Resolution (bit)	Accuracy (full scale)	1-acc	sensor fail rate (Pr)	Cost ($)	SI 15.84	Normalized SI
Fault node	**Sensor node**									
R1	**S1**	1%	100	15	3%	97%	0.001	30	14.55	0.91
R2	**S2**	1%	100	16	1%	99%	0.001	80	15.84	1
R3	**S3**	1%	100	16	1%	99%	0.001	80	15.84	1
R4	**S4**	1%	100	16	1%	99%	0.001	80	15.84	1
R5	**S5**	1%	100	15	3%	97%	0.001	30	14.55	0.91
R6	**S6**	1%	100	16	1%	99%	0.001	80	15.84	1
R7	**S7**	1%	100	16	1%	99%	0.001	80	15.84	1
R8	**S8**	1%	100	16	1%	99%	0.001	80	15.84	1
R9	**S9**	1%	100	16	1%	99%	0.001	80	15.84	1
R10	**S10**	1%	100	15	3%	97%	0.001	30	14.55	0.91

The sensor fault relation is based on the sensing gain and sensing time of the sensor's to the fault. The raw data for sensors' sensing gain and time is in Table A6.

Table A6. Sensor nodes in FDG based sensor deployment

	Sens. gain	**Sens. time (ms)**	g/t
S1	1	4.6	0.217391
S2	1	3	0.333333
S3	1	3	0.333333
S4	1	3	0.333333
S5	1	4.6	0.217391
S6	1	3	0.333333
S7	1	3	0.333333
S8	1	3	0.333333
S9	1	3	0.333333
S10	1	4.6	0.217391

In order to aggregate sensor nodes, fault nodes and senor-fault relation into single edge value, I used the analytic hierarchy process (AHP). The decision hierarchy is as FigureA3. The edge value is aggregated as $\widetilde{E} = \{SI, SFI, FI\} \times \{PV_S, PV_{SF}, PV_F\}^T$.

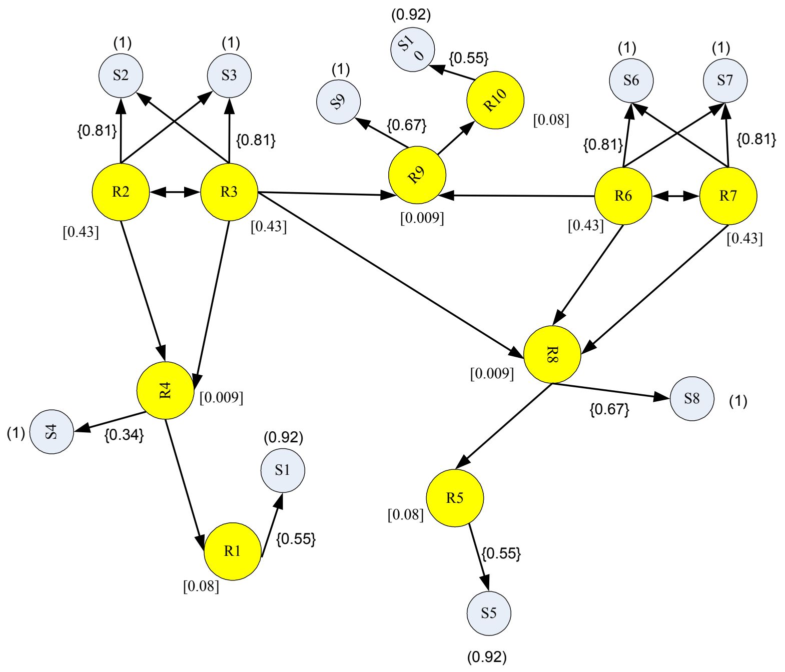


FigureA3. Decision hierarchy in the sensor deployment

We assume equal importance among these three attributes: $\{PV_S, PV_{SF}, PV_F\} = \{0.333 \quad 0.333 \quad 0.333\}$ .Combining the information, we have the edge values in TableA4, and the fuzzy graph model is presented in Figure A7.

Table A7. Connection edge values in FDG based sensor deployment

		0.918	1	1	1	0.918	1	1	1	1	0.918
		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
0.083	R1	0.55									
0.44	R2		0.81	0.81	0.34				0.67	0.67	
0.44	R3		0.81	0.81	0.34				0.67	0.67	
0.009	R4	0.55			0.34						
0.083	R5					0.55					
0.44	R6						0.81	0.81	0.67	0.67	
0.44	R7						0.80	0.81	0.67	0.67	
0.009	R8					0.55			0.67		
0.009	R9									0.67	0.55
0.083	R10										0.55



FigureA4. FDG model for sensor deployment on dual robot

Step2): Optimize the above fuzzy graph based sensor deployment model

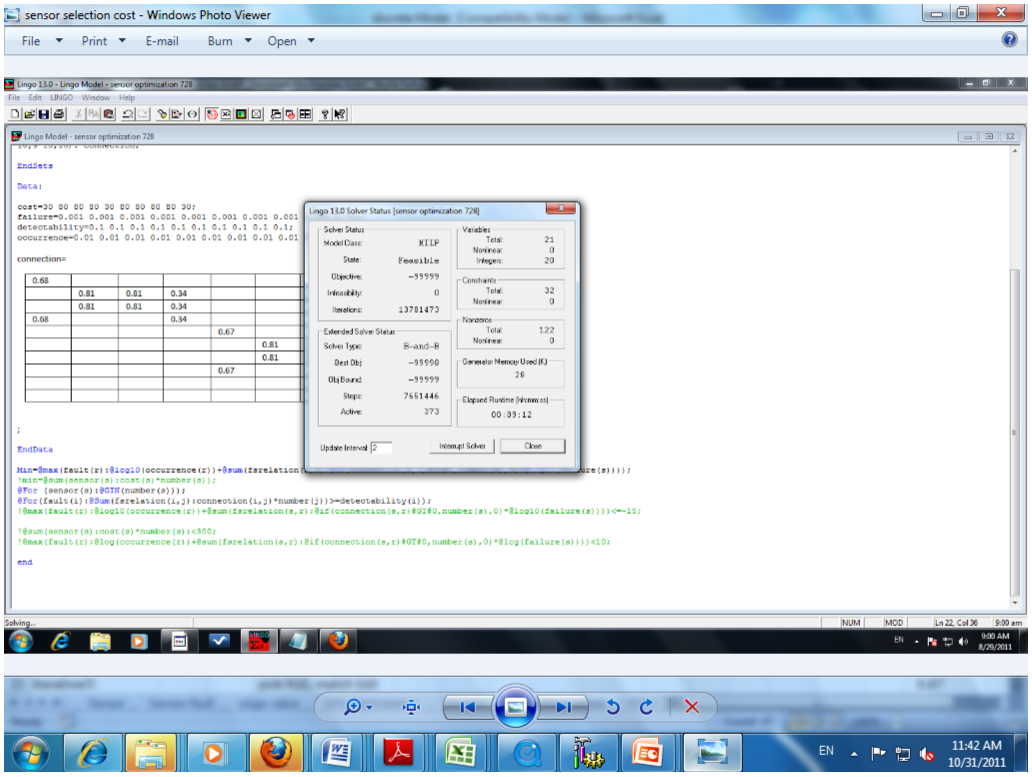
1) From the conjunction matrix in Table A7, if we use the greedy algorithm proposed in Chapter III to optimize this graph, we have the iterative process as Table A8.

Table A8. Iterative process on greedy optimization in FDG based sensor deployment

Iterative1:			pick R1, match S1 and remove S4
Iterative2:			pick R5, match S5 and remove S8
Iterative3:			pick R10, match S10, remove S9
Iterative4:			pick R4, remove S4 since it was covered by S1
Iterative5:			pick R8, Remove S8, since it was covered by S5
Iterative6:			pick R9, remove S9,since it was covered by S10
Iterative7:			pick R6, match S6, remove S7
Iterative8			pick R6, remove S7,since it was covered by S6
Iterative9:			pick R2, match S2, remove S3
Iterative10:			pick R3, remove S3,since it was covered by S2

So the selected sensors are: **S1, S5, S10,S2, S6**

2) Use mixed integer linear programming to optimize this graph, we have: when detectability requirement is 0.1, the selected sensors are: **S1, S3, S5, S7, S10.** When detectability requirement is 0.8, the selected sensors are: **S1, S2, S5, S6, S10.** This is the same with the results from greedy algorithm.



```
EndSets

Data:

cost=30 80 80 80 30 80 80 80 80 30;
failure=0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001
detectability=0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1;
occurrence=0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01

connection=
```

0.68						
	0.81	0.81	0.34			
	0.81	0.81	0.34			
0.68			0.34			
				0.67		
					0.81	
					0.81	
				0.67		

```
;

EndData

Min=@max(fault(r):@log10(occurrence(r))+@sum(fsrelation(                                    ure(s))));
!min=@sum(sensor(s):cost(s)*number(s));
@For (sensor(s):@GIN(number(s)));
@For(fault(i):@Sum(fsrelation(i,j):connection(i,j)*number(j))>=detectability(i));
!@max(fault(r):@log10(occurrence(r))+@sum(fsrelation(s,r):@if(connection(s,r)#GT#0,number(s),0)*@log10(failure(s))))<=-15;

!@sum(sensor(s):cost(s)*number(s))<300;
!@max(fault(r):@log(occurrence(r))+@sum(fsrelation(s,r):@if(connection(s,r)#GT#0,number(s),0)*@log(failure(s))))<10;

end
```

```
Lingo 13.0 Solver Status [sensor optimization 728]

Solver Status                      Variables
  Model Class:        MILP           Total:        21
  State:          Feasible           Nonlinear:     0
  Objective:        -99999           Integers:     20
  Infeasibility:         0         Constraints
  Iterations:     13781473           Total:        32
                                     Nonlinear:     0
Extended Solver Status             Nonzeros
  Solver Type:       B-and-B         Total:       122
  Best Obj:         -99998           Nonlinear:     0
  Obj Bound:        -99999         Generator Memory Used (K)
  Steps:           7651446           28
  Active:              373         Elapsed Runtime (hh:mm:ss)
                                     00:09:12

Update Interval: 2
```

sensor selection cost - Windows Photo Viewer
File
Print
E-mail
Burn
Open
Lingo 13.0 - Lingo Model - sensor optimization 728
File Edit LINGO Window Help
Lingo Model - sensor optimization 728
Interrupt Solver
Close
Solving...
NUM
MOD
Ln 22, Col 36
9:00 am
9:00 AM
8/29/2011
EN
11:42 AM
10/31/2011

FigureA5. LINGO model for FDG based sensor deployment on dual robot

Summarizations on the sensor deployment

There are two sensor deployments: 1) bench mark: SDG deployment: the selected sensors are [S1, S5, S9, S10], which represents [pressure on A1X solenoid, pressure on A1Z solenoid, voltage reading on A1Z controller, pressure on gripper solenoid]; 2) proposed sensor deployment: the selected sensors are [S1, S3, S5, S7, S10], which represent [pressure on A1X solenoid, voltage reading on A1XH3, pressure on A1Z solenoid, voltage reading on A1ZH2, pressure on gripper solenoid].

A2.2 Diagnoser design

There are two diagnoser approaches: 1) bench mark: FSA based diagnoser design; 2) proposed diagnoser design: realtime fuzzy Petri net diagnoser.

1) SDG senor deployment+FSA diagnoser

Step1: set up the plant model and under this configuration is as shown in FigureA6.

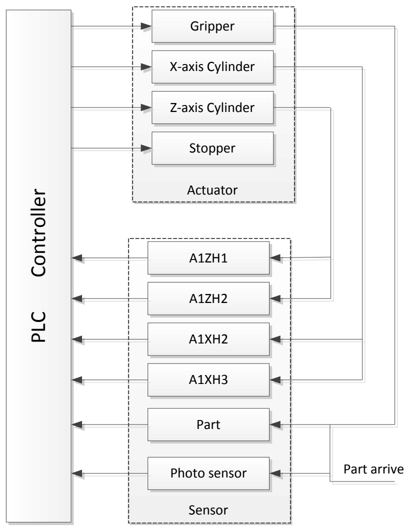


Figure A6. Actuators and sensors of the pick and place station

Plant model

The plant's control model based on sequential function chart is as FigureA7. The enable condition and desired outcome at each step is summarized in Table A9.

Table A9. Enablement conditions for states on the dual robot arm

	Event	Enabled outcome	Precondition
1	Fixture arrive	N/A	N/A
2	Part feed	Feeder start (O:1/1)↑	PSS3 (I:5/10)↑
3	Lower arm	A1Z (O:2/10)↑ ,T4:40/TT↑	PSS3(I:5/10)↑, part (I:4/3)↑, A1XH3(I:5/11)↑, A1ZH2 (I:5/9)↑
4	Grip part	Gripper(O:2/13)↑, T4:55/TT↑	A1ZH1(I:5/13)↑, A1XH3(I:5/11)↑
5	Raise arm	A1Z(O:2/10)↓	T4:40/TT↓ (last 2s using timer)
6	Extend arm	A1X(O:2/9)↓	T4:40/TT↓, A1XH3(I:5/11)↑
7	Lower arm	A1Z(O:2/10)↓ ,T4:40/TT↑	A1XH2(I:5/7)↑, A1ZH2 (I:5/9)↑
8	Drop part	Gripper(O:2/13)↓	T4:55/TT↓(last 4s using timer)
9	Raise arm	A1Z(O:2/10)↓	T4:40/TT↓(last 2s using timer)
10	Release fixture	PS3(O:1/13)↑, T4:25/TT↑	A1ZH1(I:5/13)↑, A1XH3(I:5/11)↑ , T4:55/DN↑
11	Retract arm	A1X(O:2/9)↑ ,T4:11/TT↑	PSS3(I:5/10)↓

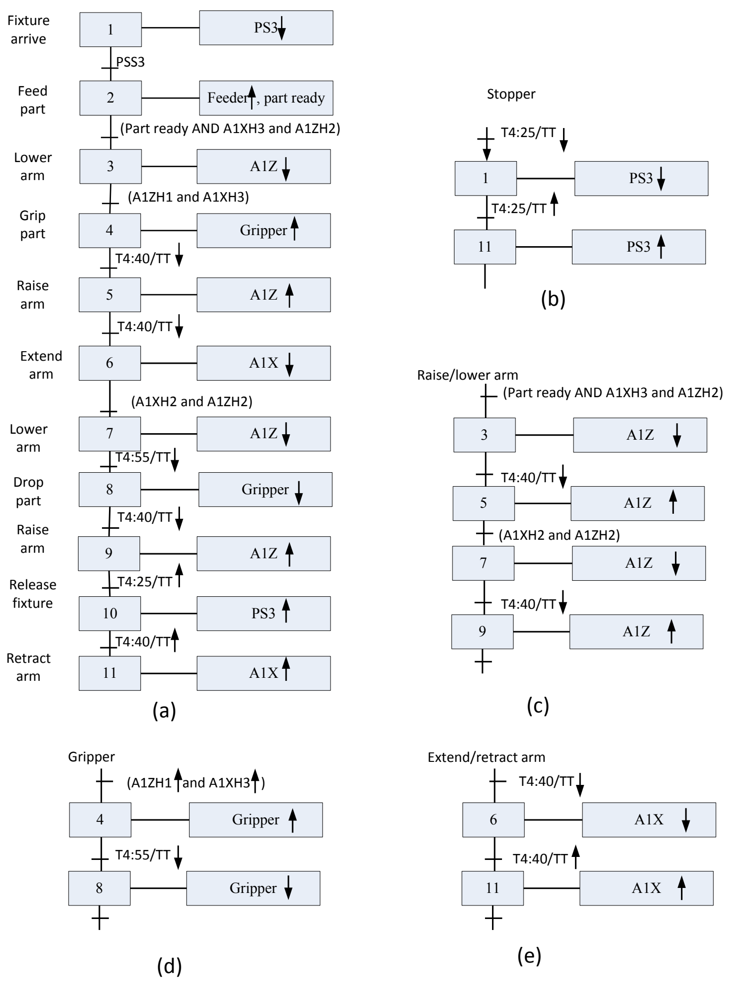


Figure A7 Sequential function chart for dual robot assembly (a) and its components-stopper (b), shoulder (c), gripper (d) and arm (e)

The FSA plant model for the dual robot arm is as Figure A8.

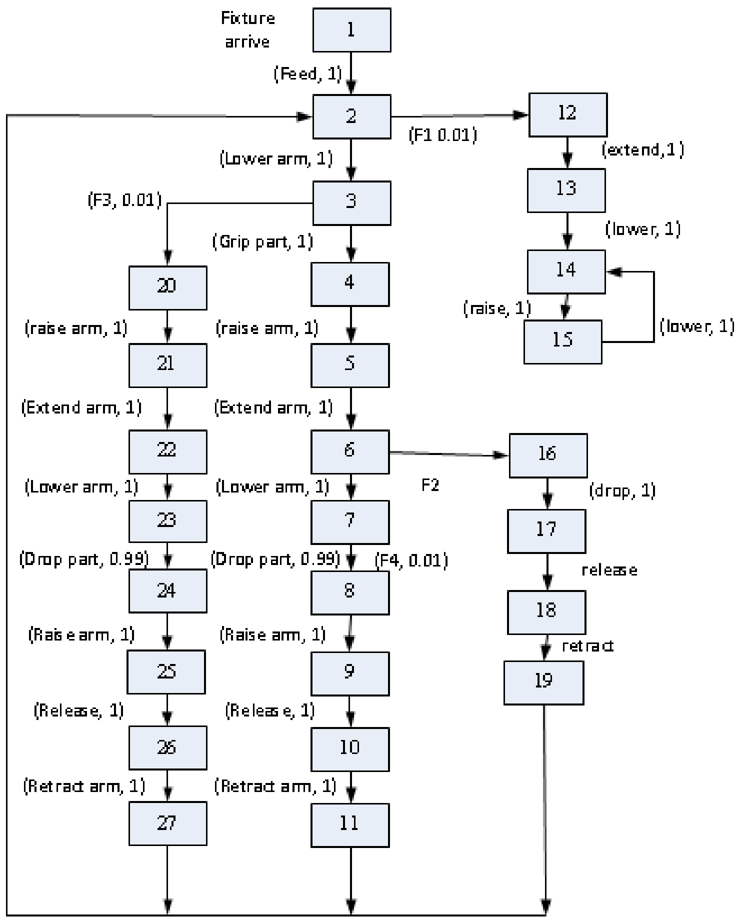


Figure A8. FSA plant model for the dual robot arm

Step 2: Sensor mapping

Table A10. Sensor mapping for the FSA plant under SDG sensor configuration

State	**Physical meaning**	**S1, S5, S9, S10**
1	Fixture arrived	1,0,0,0
2	Feed part	1,0,0,0
3	Lower arm	1,1,0,0
4	Grip part	1,1,1,1
5	Raise arm	1,0,1,1
6	Extend arm	0,0,1,1
7	Lower arm	0,1,1,1
8	Drop part	0,1,0,0
9	Raise arm	0,0,0,0
10	Release fixture	0,0,0,0
11	Retract arm	1,0,0,0
12	F1 occurrence	
13	Extend arm	0,0,0,0
14	Lower arm	0,1,0,0
15	Raise arm	0,0,0,0
16	F2 occurrence	
17	Drop part	0,1,0,0
18	Release fixture	0,0,0,0
19	Retract arm	1,0,0,0
20	F3 occurrence	
21	Raise arm	1,0,1,0
22	Extend arm	0,0,1,0
23	Lower arm	0,1,1,0
24	Drop part	0,1,0,0
25	Raise arm	0,0,0,0
26	Release fixture	0,0,0,0
27	Retract arm	1,0,0,0

Combining the sensor mapping with projection, we get the diagnoser as Figure A9 for the dual robot arm.

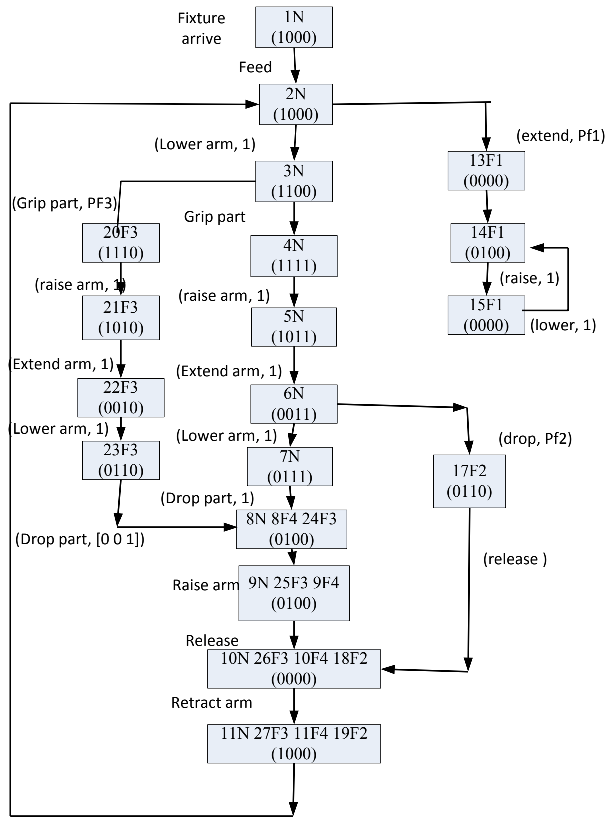


Figure A9. FSA diagnoser model for the dual robot arm with SDG sensor deployment

2) FDG senor deployment+FSA diagnoser

Step 1: The plant and under this configuration is as shown in Figure A10.

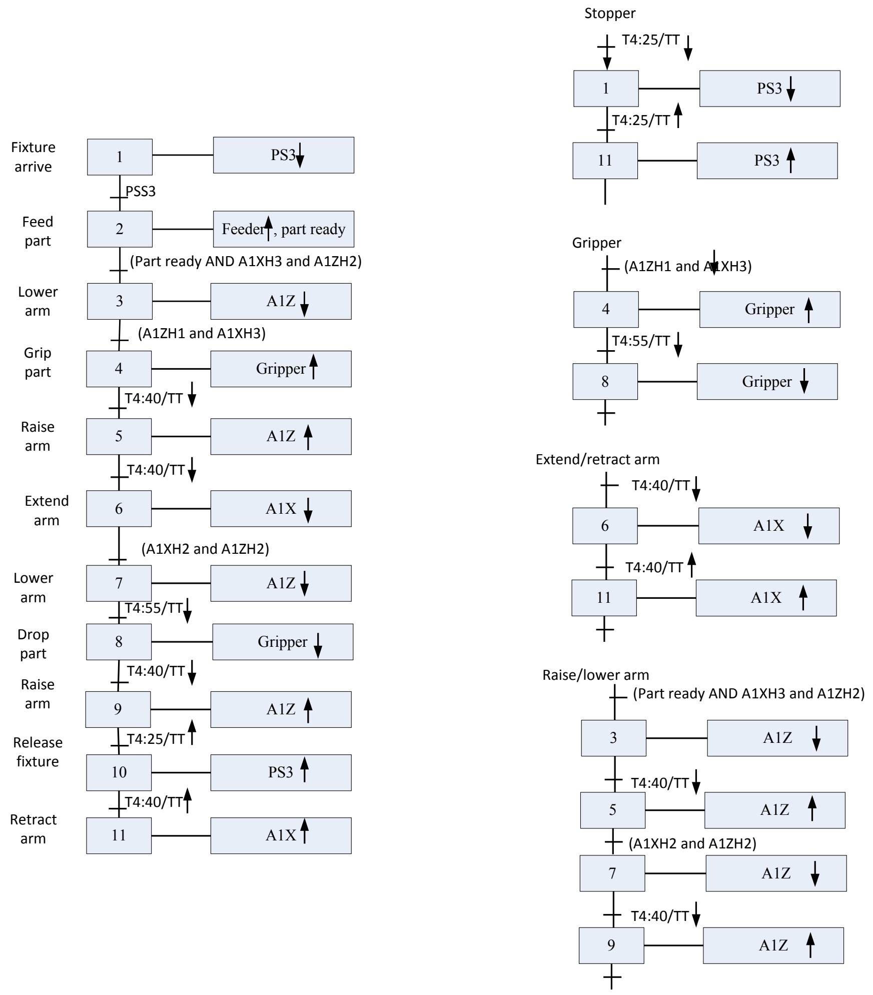


Figure A10. Sequential function chart for dual robot assembly (a) and its components-stopper (b), shoulder (c), gripper (d) and arm (e)

Step 2: Sensor mapping

Table A11. Sensor mapping for the FSA plant under FQDG sensor configuration

State	Physical meaning	S1, S3, S5, S7, S10
1	Fixture arrived	1,1,0,1,0
2	Feed part	1,1,0,1,0
3	Lower arm	1,1,1,0,0
4	Grip part	1,1,1,0,1
5	Raise arm	1,1,0,1,1
6	Extend arm	0,0,0,1,1
7	Lower arm	0,0,1,0,1
8	Drop part	0,0,1,0,0
9	Raise arm	0,0,0,1,0
10	Release fixture	0,0,0,1,0
11	Retract arm	1,1,0,1,0
12	F1 occurrence	
13	Extend arm	0,0,0,1,0
14	Lower arm	0,0,0,0,0
15	Raise arm	0,0,0,1,0
16	F2 occurrence	
17	Drop part	0,0,0,1,0
18	Release fixture	0,0,0,1,0
19	Retract arm	1,1,0,1,0
20	F3 occurrence	
21	Raise arm	1,1,0,1,0
22	Extend arm	0,0,0,1,0
23	Lower arm	0,0,1,0,0
24	Drop part	0,0,1,0,0
25	Raise arm	0,0,0,1,0
26	Release fixture	0,0,0,1,0
27	Retract arm	1,1,0,1,0

Build these events into the plant and use the projection; we can get the diagnoser as Figue A11.

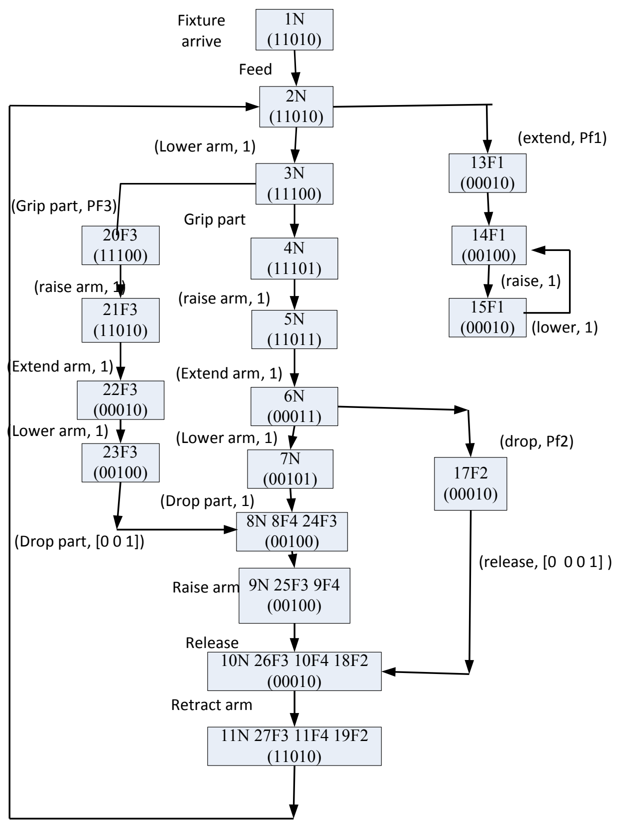


Figure A11. FSA diagnoser model for the dual robot arm with FQDG sensor deployment

The idea of Petri net diagnoser is different from FSA. It uses the signal feature to isolate faults, while the FSA are using the events observations.

Step 3: fault isolation

The behavior which does not correspond to a normal one is considered as faulty. Thus, a fault can occur starting from any state of the desired behavior. This fault occurrence is unobservable and it leads the system to a faulty state. Each one of these faulty states must be reached within a finite delay for all the event sequences that can lead to this state starting from any other one of the desired behavior states.

When part failure occurs, the observed the sequence of events on this:

a) Part fault

The observed event sequence is: feed part-extend arm-lower arm-raise arm-lower arm-raise arm ….

The reason for the arm cannot be lowered can because of failure of (part, or A1XH3, or A1ZH2), the arm was extended is because of failure of (T4:40/TT or A1XH3), arm can be lowered means (A1XH2 and A1ZH2) are normal. Arm cannot be retracted is because of (A1ZH1, A1XH3) problem. It will not be the problem of A1ZH1, because even A1ZH1 has problem, it still can achieve the work of lower arm in the first time. Now from the observed sequence, we cannot isolate the fault with (part, or A1XH3), with help of sensor mapping, we are trying to isolate them.

With SDG based sensor deployment, we cannot tell whether it is part fault or A1XH3 fault, because there is no information about them. However, with FDG based sensor deployment, we can isolate part fault and A1XH3 fault because if it is A1XH3 fault, S3 will always be 0. If S3 initially is not 0 but 1, and the observed event sequence is still like: feed part-extend arm-lower arm-raise arm-lower arm-raise arm .., we conclude that "part fault" occurred.

b) A1XH2 fault

The observed event sequence is: feed part-lower arm-pick part-raise arm-extend arm-drop part-release fixture-retract arm; we know it skipped step 7). The reason for step 7) not being executed is fault of (A1XH2, or A1ZH2). It will not be A1ZH2's abnormality; otherwise step 3) will not be executed. We don't need sensor mapping, and we can conclude that it is "A1XH2 fault".

c) Lose wire between gripper solenoid and controller

The observed event sequence is: feed part-lower arm-pick part-raise arm-extend arm-lower arm-drop part-raise arm-release fixture-retract arm; it skips steps 4), 7) and 8). If step 4) is skipped, it may be because of (A1ZH1, or A1XH3) faults, but it won't be A1XH3 fault, otherwise step 3) won't happen. Step 7) was skipped may be because of (A1XH2, or A1ZH2) faults. However, it won't be A1ZH2 fault, otherwise step 3) won't happen. Step 8) was skipped may be because of (A1ZH1, or A1XH3) fault, but we already remove the possibility of A1XH3. Now let us decide whether it is (A1ZH1 or A1XH2)'s problem? It will not be A1XH2's problem. Because event A1XH2 is faulty, it will still grip the part. Now we cannot decide whether is because A1ZH1 or some other faults by observed event sequence. Now we use the sensor mapping.

By SDG, we observed at step 7), S9 has a reading change, while S10 no change. So we conclude that it is not the fault of A1ZH1, instead it is the wiring between controller and solenoid.

By FDG, we observed that at step 10) fixture is released, so we conclude that it is not the fault of A1ZH1, instead it is the wiring between controller and solenoid.

d) Tolerance fault

The proposed sensor deployment and fault diagnoser design cannot identify this kind of fault with normal operation.

3) SDG senor deployment+PN diagnoser

Step 1: Plant model

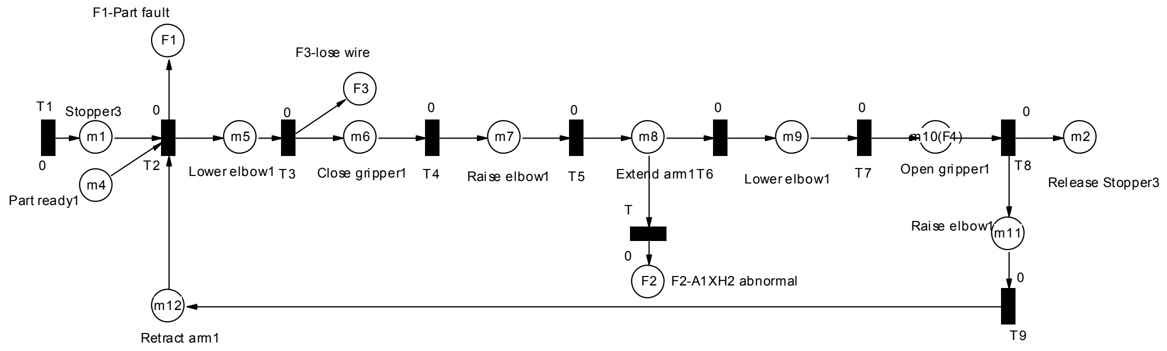


Figure A12. PN plant model for the dual robot arm

Step 2: Sensor mapping with SDG sensor deployment

Table A12. Sensor mapping in PN diagnoser with SDG sensor deployment

Places	**Description**	**Output Devices**	**Transition**	**Input events**
P1	Stopper3	{C3,PSS3,PS3}	T1	Start
P2	Feeder part ready	{Part, PSS3}	T2	Lower elbow1
P3	Elbow1 lowered	{PSS3, Part Ready, A1ZH2,A1XH3}	T3	Close gripper2
P4	Part1 picked (close gripper1)	{A1ZH1,A1XH3}	T4	Raise elbow2
P5	Elbow1 raised	{T40.TT}	T5	Extend arm1
P6	Arm1 extended	{PSS3, T11}	T6	Lower elbow1
P7	Lower elbow1	{PSS3,A1XH2,A1ZH 2}	T7	Open gripper1
P8	Open gripper1	{T55.DN}	T8	Raise elbow1
P9	Raise elbow	{T40.DN}	T9	Retract arm1
P10	Retract arm1	{A1XH2}		
Faulty Places			**Faulty Transitions**	
PF1	Part fault	{ A1ZH2, A1XH2, gripper}	F1	
PF2	A1XH2 abnormal	{ A1ZH2, A1XH2, gripper}	F2	
PF3	A1XH3 abnormal	{ A1ZH2, A1XH2, gripper}	F3	
PF4	A1ZH1 abnormal	{ A1ZH2, A1XH2, gripper}	F4	
PF5	A1ZH2 abnormal	{ A1ZH2, A1XH2, gripper}	F5	

Step 3: Diagnoser

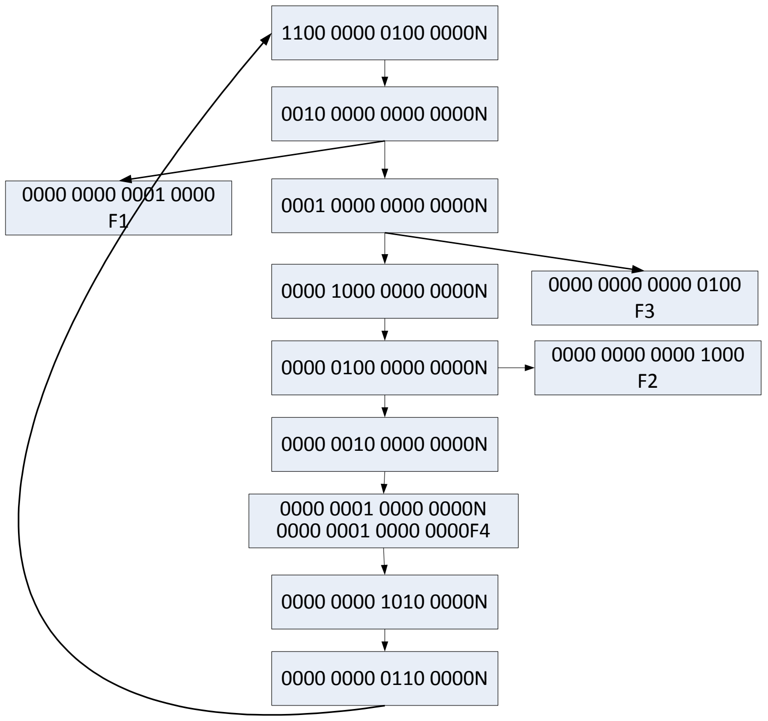


Figure A13. Coverability tree model for the dual robot arm

Step 4: Fuzzy fault isolation rules

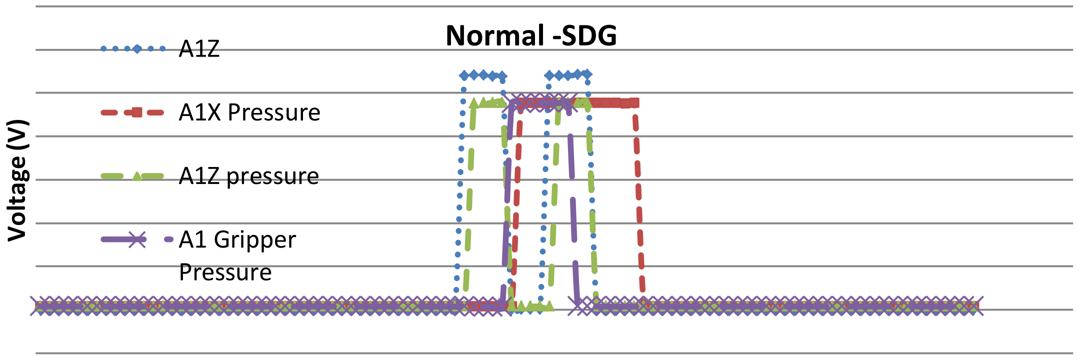


Figure A14. Signal features for normal operation on SDG based sensor deployment

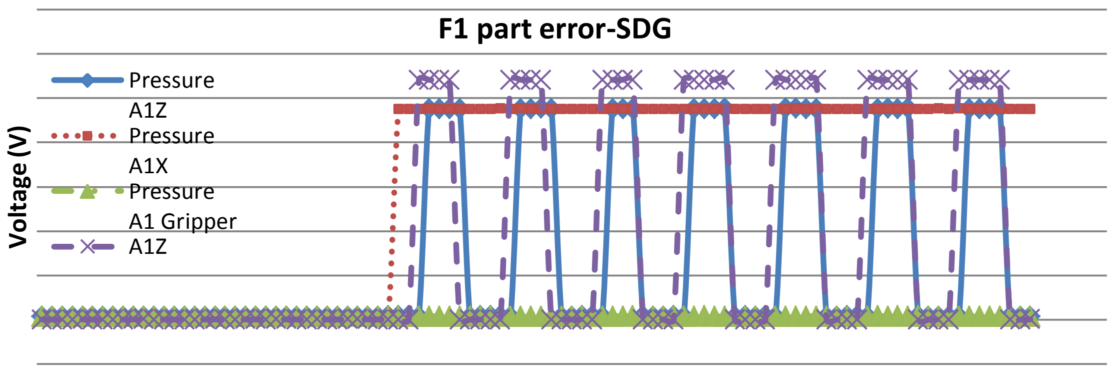


Figure A15. Signal features for F1-part error on SDG based sensor deployment

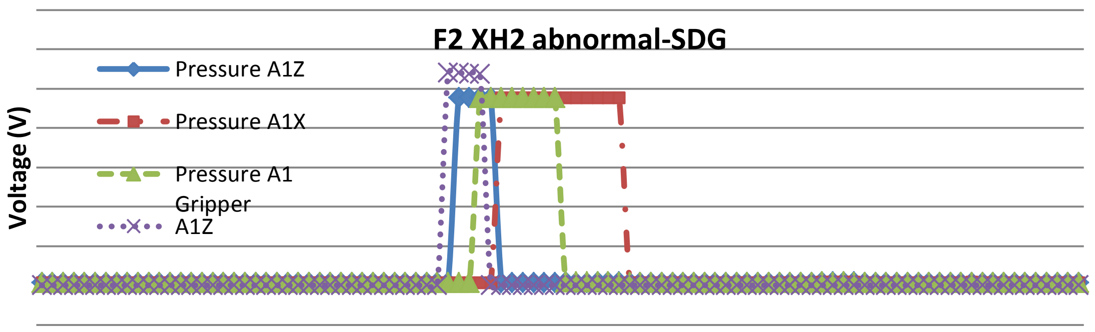


Figure A16. Signal features for F2-XH2 error on SDG based sensor deployment

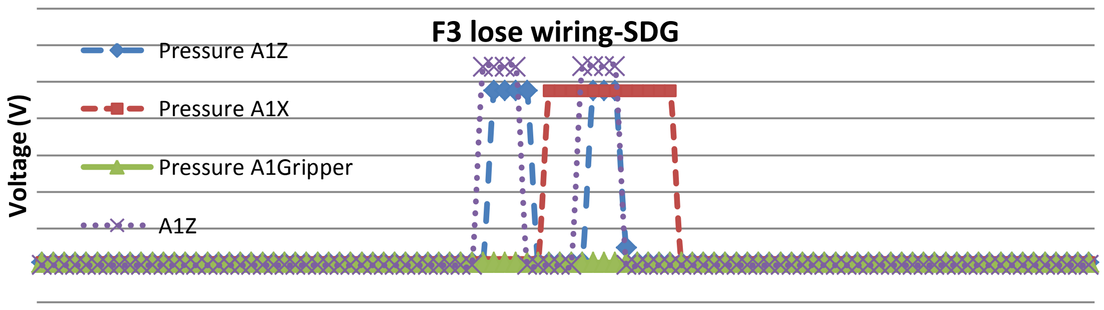


Figure A17. Signal features for F3-lose wiring on SDG based sensor deployment

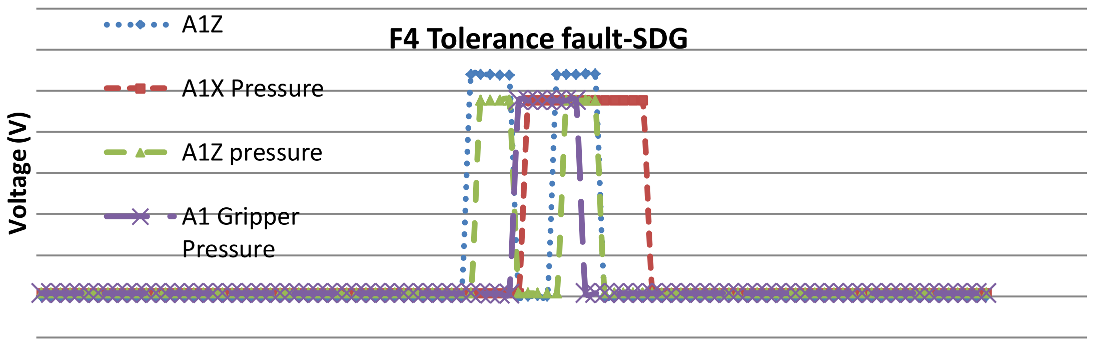


Figure A18. Signal features for F4-tolerance error on SDG based sensor deployment

For the SDG based sensor deployment, the fault isolation rules are as below:

Normal: At the normal operation the A1Z should two periodic cycles of "ON" and each lasts for 2 seconds; AND Gripper pressure should have one periodic cycle of "ON"

R1: IF *A1Z has periodic cycles on low-high signal* AND *XH3 is always off*, THEN *the system is in part fault.*

R2: IF *A1Z has only one cycle on low-high signal change*, THEN *the system is in XH2 fault.*

R3: IF *Gripper has turned "ON"* AND *Gripper pressure keeps always zero*, THEN *the system is lose wiring fault*.

R4: the proposed sensor diagnoser and sensor deployment cannot isolate the tolerance fault.

4) FQDG senor deployment+PN diagnoser

Step 1: Plant model

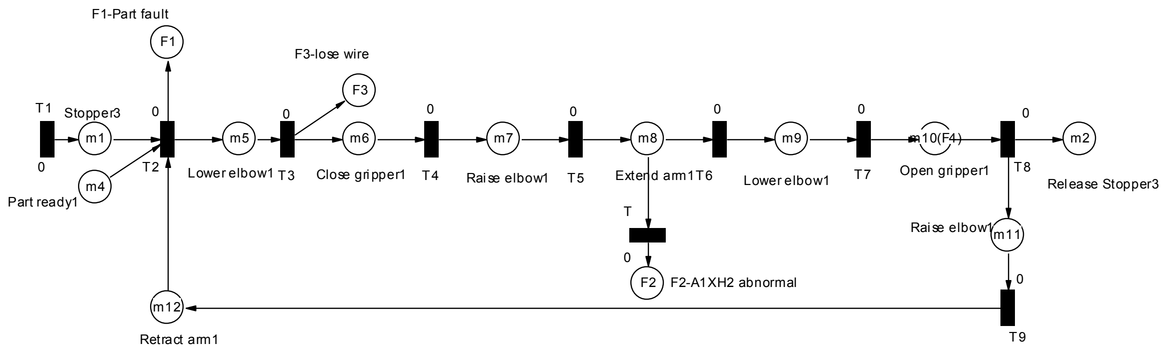


Figure A19. PN plant model for the dual robot arm

Step 2: Sensor mapping

Table A13. Sensor mapping in PN diagnoser with FQDG sensor deployment

Place	**Description**	**Output Devices**	**Transition**	**Input events**
P1	Stopper3	{C3,PSS3,PS3}	T1	Start
P2	Feeder part ready	{Part, PSS3}	T2	Lower elbow1
P3	Elbow1 lowered	{PSS3, Part Ready, A1ZH2,A1XH3}	T3	Close gripper2
P4	Part1 picked (close gripper1)	{A1ZH1,A1XH3}	T4	Raise elbow2
P5	Elbow1 raised	{T40.TT}	T5	Extend arm1
P6	Arm1 extended	{PSS3, T11}	T6	Lower elbow1
P7	Lower elbow1	{PSS3,A1XH2,A1ZH2}	T7	Open gripper1
P8	Open gripper1	{T55.DN}	T8	Raise elbow1
P9	Raise elbow	{T40.DN}	T9	Retract arm1
P10	Retract arm1	{A1XH2}		
Faulty Place			**Faulty Transition**	
PF1	Part fault	{ A1ZH2, A1XH2, gripper}	F1	
PF2	A1XH2 abnormal	{ A1ZH2, A1XH2, gripper}	F2	
PF3	A1XH3 abnormal	{ A1ZH2, A1XH2, gripper}	F3	
PF4	A1ZH1 abnormal	{ A1ZH2, A1XH2, gripper}	F4	
PF5	A1ZH2 abnormal	{ A1ZH2, A1XH2, gripper}	F5	

Step 3: Diagnoser

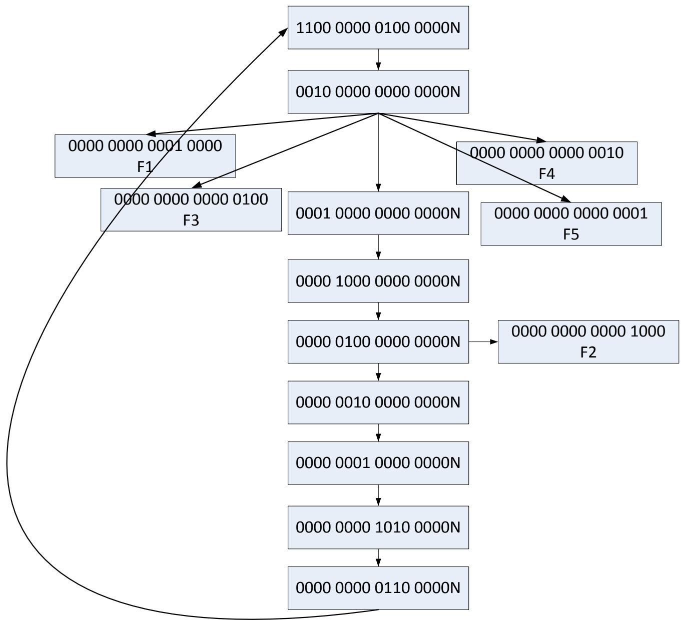


Figure A20. Coverability tree model for the dual robot arm

Step 4: Fuzzy fault isolation rules

For the FDG based sensor deployment, the fault isolation rules are as below:

Normal: At the normal operation the A1ZH1 should have two periodic cycles of "ON" and each lasts for 2 seconds; AND Gripper pressure should have one periodic cycle of "ON".

R1: IF *A1ZH1 has periodic cycles on low-high signal* AND *XH3 is always off*, THEN *the system is in part fault.*

R2: IF *A1ZH1 has only one cycle on low-high signal change*, THEN *the system is in XH2 fault.*

R3: IF *A1ZH1 has the on two periodic cycles of "ON"* AND *Gripper pressure keeps always zero*, THEN *the system is lose wiring fault*.

R4: the proposed sensor diagnoser and sensor deployment cannot isolate the tolerance fault.

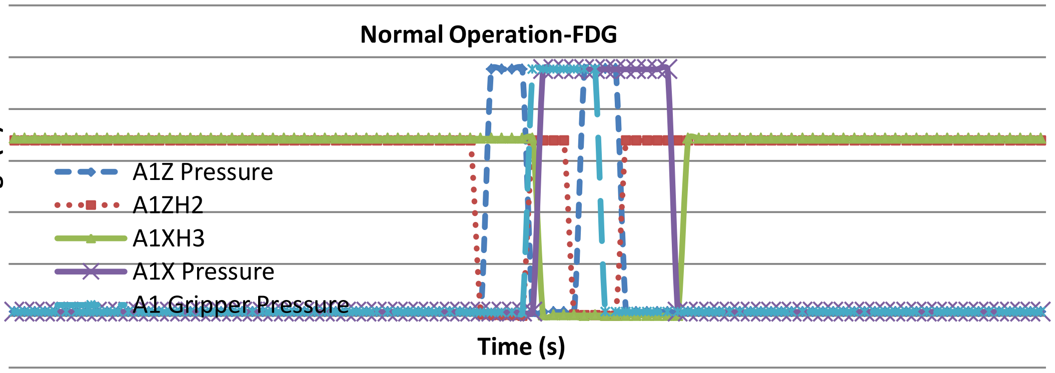


Figure A21. Signal features for normal operation on FDG based sensor deployment

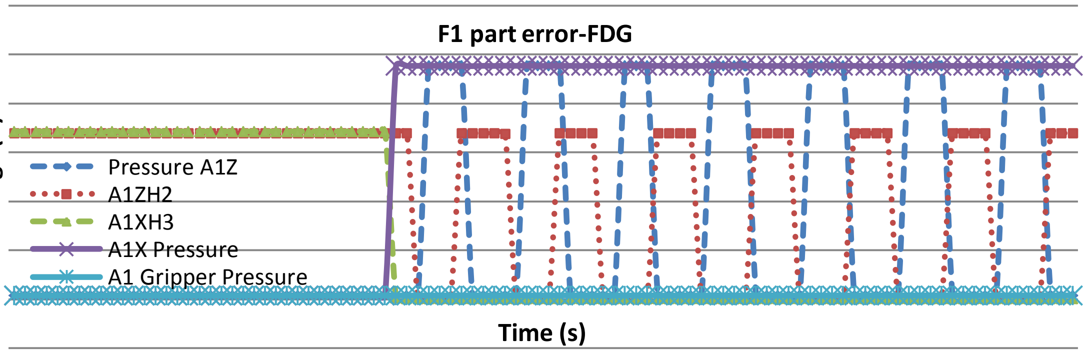


Figure A22. Signal features or F1-part error on FDG based sensor deployment

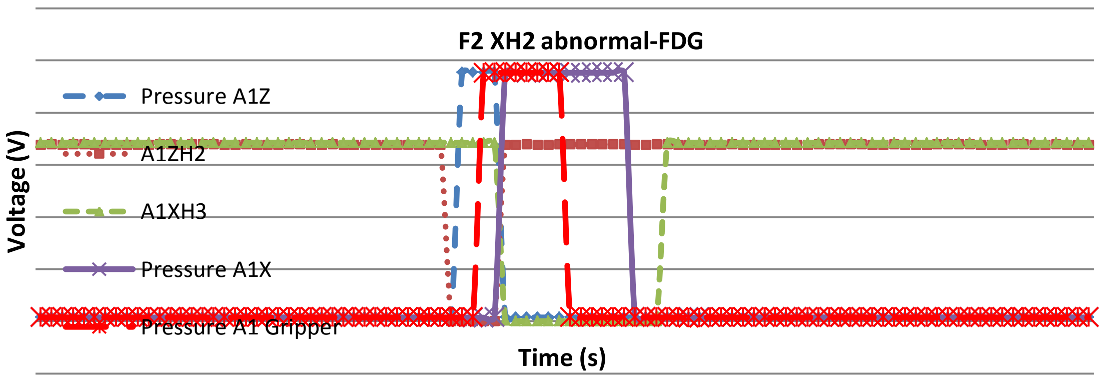


Figure A23. Signal features or F2-XH2 error on FDG based sensor deployment

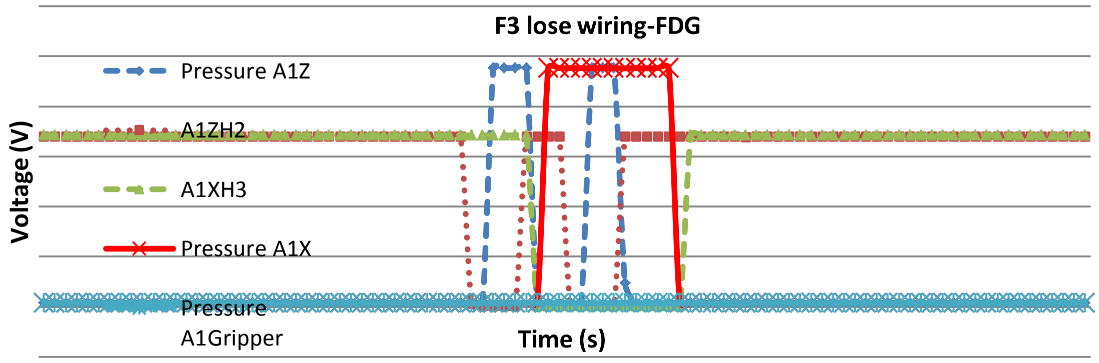


Figure A24. Signal features or F3-lose wiring error on FDG based sensor deployment

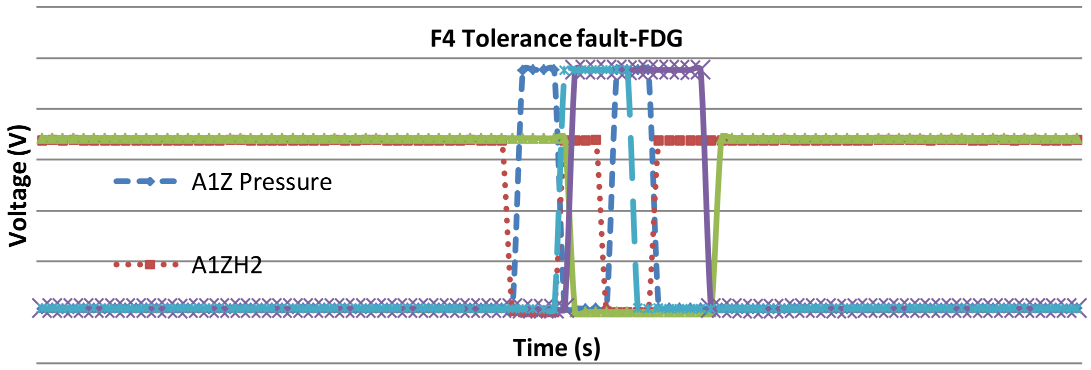


Figure A25. Signal features or F4-tolerance error on FDG based sensor deployment

APPENDIX III

SIGNED DIRECTED GRAPH BASED SENSOR DEPLOYMENT

Signed Directed Graph (SDG) [7, 8] is a graphical representation of the cause effect relation system that utilizes nodes and branches. When applying it to sensor deployment problem, the nodes correspond to the sensor that monitor process state variables and fault that represent malfunctions, and the branches represent the causal relationships between the fault nodes and sensor nodes. The branches are marked with signs according to the relationship between the sensor variables and fault nodes. This representation helps in defining a pattern of observed symptoms that the particular fault will influence on the process variables. We need to mention that not all the variables of a manufacturing process can be measured due to technical or economical infeasibilities. Thus the pattern defined is always partially observed as the ‘‘partial pattern’’. This partial pattern from these sensors helps in detecting symptoms of every fault thus obtaining the cause effect model on fault propagation. The arcs in the SDG represent a ‘‘can cause’’ (be careful, not “will cause”) relationship, that is, an arc from node A to node B only implies that A can cause B, instead of A will cause B. The faults’ positive (high) or negative (low) influences on the states of the variables respectively are assigned ‘+’ or ‘-‘signs to the branches, and normal influences are marked with 0. A nonzero node sign signifies the presence of a failure in the process, and a set of nonzero signs in the SDG represents a pattern of fault symptoms. The SDG graph on sensor deployment can be generated from process graph or failure mode effect analysis.

When the SDG is ready, the problem of sensor allocation becomes to identify the fault root nodes and place the minimum number of sensors on the measurable valid nodes in the cause effect graph. Here, we need ensure that every fault defined for the process has to be observed by at least one sensor. This would ensure that no fault goes unobserved when a given set of sensors is located on the SDG. This is referred to as the ‘‘observability condition’’. Such an optimization procedure is shown as below:

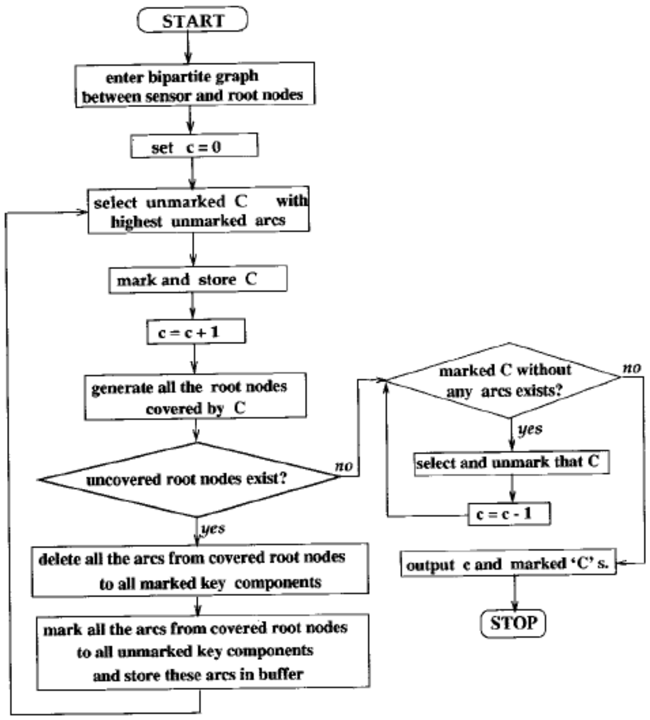

START
enter bipartite graph between sensor and root nodes
set c = 0
select unmarked C with highest unmarked arcs
mark and store C
c = c + 1
generate all the root nodes covered by C
uncovered root nodes exist?
yes
no
delete all the arcs from covered root nodes to all marked key components
mark all the arcs from covered root nodes to all unmarked key components and store these arcs in buffer
marked C without any arcs exists?
yes
no
select and unmark that C
c = c - 1
output c and marked 'C' s.
STOP

Figure A26. Flow chart for sensor selection using greedy search in SDG based sensor deployment [7]

APPENDIX IV

EXPERIMENT DATA ON THE COMPREHENSIVE EVALUATION

RUN	Sensor deployment strategy	Diagnoser	Fault	0		1		0		0		1		1		1		0		1		1	
				dc	dl	dc	dl	dc	dl	dc	dl	dc	dl	dc	dl	dc	dl	dc	dl	dc	dl	dc	dl
1	1	1	1	F3		F1	9	N		N		F1	7	F1	8	F1	9	N		F1	9	F1	8
2	1	1	2	F3, F2		F2	8	N		N		F2	7	F2	6	F2	6	N		F2	5	F2	6
3	1	1	3	N		F3	7	N		N		F3	9	F3	7	F3	7	N		F3	8	F3	7
4	1	1	4	N		N	11	N		N		N	11	N	11	N	11	N		N	11	N	11
5	1	2	1	N		F1	7	N		N		F1	8	F1	9	F1	8	N		F1	8	F1	7
6	1	2	2	F3		F2, F3	8	N		N		F2	5	F2	4	F2	4	N		F2	3	F2	4
7	1	2	3	N		F3	7	N		N		F3	7	F3	6	N		N		F3	6	F3	6
8	1	2	4	N		N	11	N		N		N	11	N	11	N	11	N		N	11	N	11
9	2	1	1	F3		N	11	F3		F2		N	11	N	11	F2	11	F3		N	11	F2	11
10	2	1	2	F2		F2, F3	3	F3		F3		F2	6	F2, F3	6	F2	6	F3		F2		F2, F3	7
11	2	1	3	N		F3	6	N		F2, F3		F2, F3	7	F3	7	N		N		N		F2	3
12	2	1	4	N		N	11	N		N		N	11	N	11	N	11	N		N	11	N	11
13	2	2	1	N		F1, F2	9	F3		N		F1	9	F1	7	F1, F2	9	F2, F3		F1	9	F1	8
14	2	2	2	F2, F3		F2, F3	2	N		N		F2	4	F2, F3	3	F2	3	F2, F3		N		F2	3
15	2	2	3	N		N		F2		N		F2,F3	7	F3	7	N		N		F3	6	F2, F3	6
16	2	2	4	N		N	11	N		N		N	11	N	11	N	11	N		N	11	N	11

In the "sensor deployment strategy" column: 1 means FDG based sensor deployment, 2 means SDG based sensor deployment. In the diagnoser column: 1 means "FSA+SFC" diagnoser, 2 means "RTFPN" diagnoser.

VITA

Zhenhua Wu received his Bachelor of Engineering degree in Mechatronics Engineering from Hefei University of Technology in 2002. He entered the Mechanical Engineering program at Texas A&M University in August 2007 and he received his Doctor of Philosophy degree in May 2012. His research interests include control and diagnosis on automated manufacturing system, sustainable manufacturing, and nano manufacturing process etc.

Zhenhua can be reached through Dr. Sheng-jen Hsieh, Engineering Technology and Industrial Distribution Department, Texas A&M University, College Station, TX 77845. Zhenhua's email is wuzhenhua34@gmail.com.