

## MASTER

### Sequential Testing for the Diagnosis of High-tech Systems

Beurskens, Joep C.A.

*Award date:*  
2022

[Link to publication](#)

#### **Disclaimer**

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

#### **General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain



1CM96

Master Thesis

---

# Sequential Testing for the Diagnosis of High-tech Systems

5th June 2022

---

<b>Author:</b>	J.C.A. (Joep) Beurskens	1498819
<b>Mentor:</b>	Dr. A.E. (Alp) Akçay	TU/e
<b>2<sup>nd</sup> assessor:</b>	Dr. C. (Claudia) Fecarotti	TU/e
<b>3<sup>rd</sup> assessor:</b>	Dr. M. (Melvin) Drent	TU/e
<b>Company Supervisor:</b>	Ir. M.J.A.M. (Emile) Van Gerwen	ESI (TNO)

*Word count:*

17241

TU/e University of Technology Eindhoven, School of Industrial Engineering  
Series Master Theses Operations Management and Logistics

**Keywords:** Bayesian Network, Influence Diagram, Diagnosis, High-tech, Myopic, Limited Memory.

# Abstract

Machines develop to being more complicated and intelligent and are subject to growingly demanding operational conditions. The traditional method of doing maintenance and diagnostics, which involved evaluating the entire system, is no longer appropriate for the job. Though there has been considerable progress in offering effective ways to utilize existing AI technology, it does not perform well in practice in this field. The process through which service engineers are faced with a diagnostic task may be viewed as a sequential decision process. Their goal is to determine the root cause using various actions such as observing and measuring. Every action has a cost associated with it, and the ideal diagnostic strategy is the one that identifies the root cause with the least cost. As a means of addressing the problems of inadequate data, incorporating the least-cost strategy, and guaranteeing computational feasibility, the construct of Bayesian networks extended to influence diagrams is a very appealing modeling scheme for a diagnostic reasoning system. From this, a methodology to generate a reasoning engine for decisions on large scales to cover the best diagnostic action (i.e., test) is proposed. The technique combines the Limited Memory Influence Diagram framework to represent the subset of potentially observable variables relevant to each decision with a Myopic approximation to derive the next best test. The suggestion and the time to suggest the next best test are of acceptable quality. The implementation of a Myopic approach with a Limited Memory Influence Diagram is not seen before in literature. The construction of such a solution framework is fully generic and independent of the underlying Bayesian network. The framework is compatible with the software used by ESI (TNO) and can be modified and operated through Python.

# Executive summary

Today's high-tech machinery is developing towards being more complicated and intelligent and is subject to growingly demanding operational conditions in a highly competitive industry. An essential criterion for this growingly demanding operational condition is keeping the system available with high reliability and minimizing unscheduled downtime caused by system faults. One of the possible uncertainties with respect to downtime is the time required to diagnose the issue. Increasing system complexity puts more pressure on a fast diagnosis of the underlying root cause of an issue. In recent years, there has been an increasing interest in fault detection and diagnosis approaches to cope with such issues. The most interest is in data-driven approaches since these methods play an important role in modern systems since they can reduce time and cost because they do not require the development of models (Alzghoul et al., 2014). Despite the time and cost savings that data-driven techniques may provide, their development and implementation are hindered by operational and functional challenges. These include: (1) a lack of data on individual machine failures (Zhao et al., 2019); (2) the need for a preprocessing step to extract useful information from data with a high computational cost (Tidriri et al., 2016); and (3) it can be difficult to gain a thorough understanding of the inner workings of many of these data-driven approaches. While theoretical research on system failures is growing, many of the studies depend on simulation data for both normal and malfunctioning situations. Because real-world systems typically lack the historical record of malfunctioning data that theoretical research necessitates, the suggested scheme's applicability remains in doubt.

Among some methods for diagnosis, the Bayesian network framework is the only one that integrates uncertainties and allows for the use of both data and expert knowledge. The semantics of the Bayesian network makes it possible to understand the causal mechanism linking a symptom to its root cause. Nonetheless, the Bayesian network demands more resources than any other approach. Literature also suggests that Bayesian networks augmented with utilities (e.g., costs and benefits) and decisions, called influence diagrams, can assure a least-cost strategy. However, the process of building an influence diagram requires a deep understanding of the problem and is therefore still considered an art (Clemen, 1996; Bielza et al., 2011). It is also known that the modeling process is not yet automated and that the majority of the literature has paid little attention to it. Thus, very little feedback from analysts and experts on their experiences using influence diagrams to build decision-making models is present. Compared to the theoretical research on data-driven methods, research on the application of influence diagrams to fault diagnosis of high-tech systems is slim to none despite its introduction that dates back to the early 1980s (Shachter, 1986). As a result, there is a deficiency of verification of the theoretical framework's applicability and reliability in real situations in this domain.

## Research Question and Focus

A lack of understanding of the primary issues encountered by those who attempt to (automatically) model complex decision-making problems on large scales using influence diagrams has been found in the current literature. Also, the applicability of such a method to these systems is currently unexplored. As a result, the purpose of this project is to develop an implementable method to aid in the decision-making process of fast and complex diagnosis of the underlying root cause to minimize unexpected downtime caused by system faults. This project's objective, or main research question, is therefore stated as follows:

*"How to develop a decision support method based on an influence diagram for assisting service engineers in their diagnostics task by advising an efficient diagnostic action?"*

The focus of this research project is specifically on the supporting material for a service engineer. The intention is to provide a tool, regardless of experience, to increase their efficiency. The proposed influence diagram-based decision support model was developed within Bayes Server Limited (2022) Software and is complemented by the Python programming language. These standards for modeling are acceptable within the ESI (TNO) project team since they also develop frameworks in these supporting tools.

## Methodology

The methodology that is proposed consists of two parts, namely, constructing and executing the method. Figure 0.1 shows the flowchart of an influence diagram-based method for diagnosis. In the constructing part (PART 1), an influence diagram may be generated automatically given a Bayesian network and a mapping between tests and network nodes. A mapping of this type is built using the system designer's and the Bayesian network constructor's knowledge. The input, such as component prior failure probability derived from operational data, reliability numbers, or expert judgment, has already been initialized in the Bayesian network.

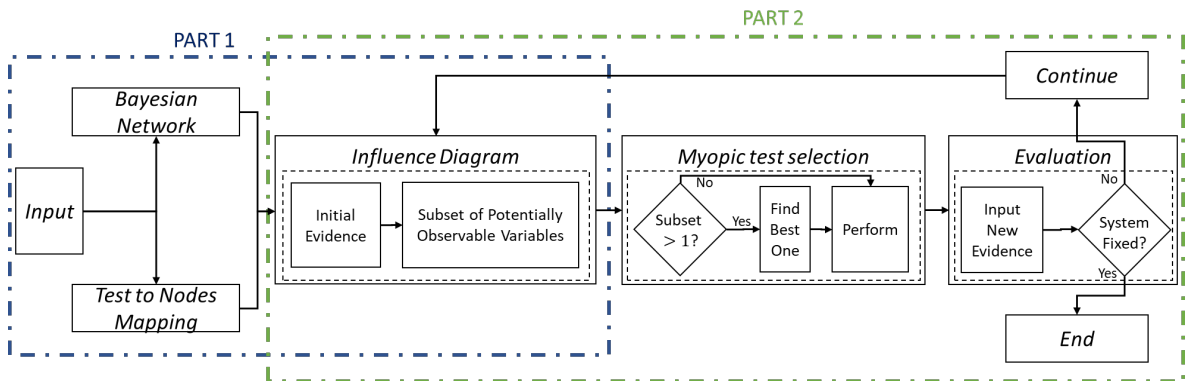


Figure 0.1: Flowchart of approach

The executing part (PART 2) consists of finding a solution to the diagnosis problem. The technique combines the solution of the Limited Memory Influence Diagram framework to represent the subset of potentially observable variables relevant to each test decision with

---

a Myopic approximation to derive the next best test out of the subset of variables. The potentially observable variable that has the highest expected value is the test that will be performed. The evidence obtained after performing that test updates the belief on all the variables in the network. If the belief of a potential root cause is high enough, the network will stop representing a subset of potentially observable variables relevant to each test decision and instead present one or more variables relevant to each repair/root cause decision.

## **Numerical Investigations**

To evaluate the proposed method, another representation of an influence diagram is presented, called the discrete influence diagram. This representation represents an exact solution to the problem, whereas the limited memory influence diagram plus myopic is an approximation. Some representative examples are used for the feasibility study and are of sufficient detail to be comparable with examples from practice. It was found that assuming a single fault, the solutions to the exact and approximate methods are equal. Relaxing this assumption leads to a small decrease in performance for the approximate one. The computation time is very advantageous for the approximation method, especially when the single fault assumption is relaxed. The exact method is only available to derive a solution for the problem with up to 8 decision nodes and is not further considered because of the memory limitation.

## **Conclusion and Future Work**

This work extends existing literature on the use of modeling and solving large and complex test sequencing problems with influence diagrams. The approach uses a novel implementation of a Limited Memory Influence diagram accompanied by a Myopic approach to suggest the next best test. The solution framework is constructed to be fully generic and independent of the Bayesian network since it only depends on a mapping between nodes in the network and the set of possible tests. Its solution is equal with the exact solution of the decision problem under the single fault assumption. Relaxing this assumption does lead to a performance decrease but is considered comparable since its normalized distance to the exact method is, in all cases, relatively close. The suggested approach outperforms the exact solution for larger networks because of the computational restrictions arising in the exact method. In these larger networks, the computation time to generate the next best test is very appealing when the single fault assumption is relaxed since it suggests the next test within a short period of time. Testing different topologies reinforces our belief that the approach is also suitable for different network configurations.

Further research could be to explore approximate or faster and more memory-efficient inference algorithms when the full decision sequence (in the discrete influence diagram) is specified and to explore the possibility of not having to go through all the possible test sequences. Some important modeling issues can also be tackled in further research. An important distinction should be made between probing and intervening actions. Because of the assumption that one should always perform a test before a corrective maintenance action if both are suggested, one does not consider corrective maintenance actions in the set of possible next actions when both are suggested. A research opportunity is, therefore, to include corrective maintenance actions in the set of possible next actions.

# Preface

This master's thesis report marks the end of my studies in Operations Management and Logistics at the Eindhoven University of Technology (TU/e). The project has been supervised by Dr. A.E. Akçay and Dr. C. Fecarotti from the TU/e and Ir. M.J.A.M. Van Gerwen from Embedded Systems Innovation (ESI) of Nederlandse Organisatie voor Toegepast-natuurwetenschappelijk onderzoek (TNO).

First of all, I would like to express my gratitude to my mentor and first supervisor, Dr. A.E. Akçay. The extensive guidance, critical feedback sessions, and discussions on the subject kept me motivated throughout the project. Through these sessions, new insights were gained that helped to take the research to the next level. Furthermore, I would like to express gratitude to my company supervisor, Ir. M.J.A.M. Van Gerwen, for the opportunity to perform my research within ESI (TNO) and for the extensive guidance throughout the entire project. His guidance style and critical questions helped to get relevant outcomes for ESI (TNO). The extensive guidance from both is highly appreciated!

Besides, I would like to thank my colleagues from the CareFree team for the conversations and support throughout the project. Although mostly from home, I had a great time working within the team.

Finally, I would like to thank my family and friends for their support and the fun activities over the past few months that helped to refuel my energy levels. Without all of you, finalizing this report would have been more of a hassle.

*Joep Beurskens*

*5th June 2022*



# Table of Contents

List of Tables	ix
List of Figures	x
List of Acronyms	xii
Glossary	xiii
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Literature Review . . . . .	2
1.3 Research Questions . . . . .	5
1.4 Structure of this Document . . . . .	6
<b>2 Problem Description</b>	<b>7</b>
2.1 Context . . . . .	7
2.2 Scope . . . . .	9
2.3 Specification . . . . .	10
<b>3 Theory</b>	<b>12</b>
3.1 Characteristics Probabilistic Networks . . . . .	12
3.2 Bayesian Network . . . . .	13
3.2.1 Construction of Bayesian Networks . . . . .	14
3.3 Influence Diagram . . . . .	15
3.4 Representation of Influence Diagrams . . . . .	18
3.4.1 The diagnosis problem . . . . .	18
3.4.2 Sequential Influence Diagram . . . . .	19
3.4.3 Limited Memory Influence Diagram . . . . .	19
<b>4 Running Examples</b>	<b>21</b>
4.1 Model 1: Single Light . . . . .	21
4.2 Model 2: Quadruple Light . . . . .	23
<b>5 Methodology</b>	<b>27</b>
5.1 Influence Diagram-based Method for Diagnosis . . . . .	27
5.1.1 Structure of the Model . . . . .	28
5.1.2 Test Modeling . . . . .	30

## Table of Contents

---

5.1.3	Solution LIMID . . . . .	34
5.1.4	Solution Discrete ID . . . . .	36
<b>6</b>	<b>Numerical Investigations</b>	<b>38</b>
6.1	Software and Models . . . . .	38
6.2	Results . . . . .	41
6.3	Scalability (Computational Time) . . . . .	44
6.4	Policy Evaluation . . . . .	45
6.4.1	Sensitivity analysis . . . . .	46
6.4.2	Comparison . . . . .	48
6.5	Chapter Conclusion . . . . .	51
<b>7</b>	<b>Conclusion and Future Work</b>	<b>52</b>
7.1	Conclusion . . . . .	52
7.2	Limitation . . . . .	53
7.3	Further Work . . . . .	54
	<b>Bibliography</b>	<b>56</b>
	<b>Appendices</b>	<b>59</b>
A	Model 1: Single Light visualization in software (BayesServer) . . . . .	59
B	Link specification algorithms . . . . .	61
C	Next best test algorithm . . . . .	62

# List of Tables

1.1	Data-driven diagnosis methods analysis (Thierno M. L. et al., 2018) . . . . .	3
2.1	Some impact factors of corrective maintenance phases (ESI (TNO), 2022) . .	8
3.1	The variables' symbols . . . . .	12
3.2	Example probability table . . . . .	13
3.3	Example utility function $U_1$ of $X_1$ with domain $\{0, 1\}$ and $D_1$ with domain $\{0, 1\}$	17
5.1	Notation used throughout this chapter . . . . .	27
5.2	Utility function corrective maintenance . . . . .	30
5.3	CPT without state <i>not_done</i> . . . . .	31
5.4	CPT with state <i>not_done</i> . . . . .	31
5.5	Calculation on the expected utility for the LIMID + Myopic Approximation .	36
5.6	Finding the best sequence and the maximum expected utility for the Discrete ID	37
6.1	Overview evaluated IDs . . . . .	38
6.2	Translation table theory to software . . . . .	39
6.3	The parameters for the numerical examples . . . . .	42
6.4	Comparison of total expected utility and run time for the solution in the discrete ID and LIMID + Myopic methods for the single light model . . . . .	42
6.5	Comparison of total expected utility and run time for the solution in the discrete ID and LIMID + Myopic methods for the quadruple light model . . . .	43
6.6	Sensitivity parameter for the numerical examples . . . . .	46

# List of Figures

0.1	Flowchart of approach . . . . .	iv
1.1	Taxonomy of terms in tree-like model . . . . .	4
2.1	Phases of corrective maintenance (ESI (TNO), 2022) . . . . .	8
2.2	Alternative representations of an assembly model of a fan for the Bayesian network construction (ESI (TNO), 2022) . . . . .	9
3.1	A directed acyclic graph . . . . .	13
3.2	A influence diagram representation of Figure 3.1 . . . . .	16
3.3	Definition arcs in influence diagram . . . . .	17
3.4	A sequential influence diagram representation of the diagnosis problem . . . .	19
3.5	A limited memory influence diagram representation of the diagnosis problem	20
4.1	Example models for a single light circuit . . . . .	21
4.2	Bayesian network of a single light circuit . . . . .	22
4.3	Example of an electrical drawing of a quadruple light circuit . . . . .	23
4.4	Example of a structural description of a quadruple light circuit . . . . .	24
4.5	Bayesian network of a quadruple light circuit including all nodes . . . . .	25
4.6	Bayesian network of a quadruple light circuit excluding redundant nodes . . .	26
5.1	Flowchart of approach . . . . .	28
5.2	Transforming the Model 1: Single Light Bayesian network to an influence diagram for diagnostic output . . . . .	29
5.3	Automatically determined $D_n$ threshold values . . . . .	30
5.4	Influence diagram of Model 1: Single Light for diagnostic output, as well as isolated tests . . . . .	31
5.5	SLM influence diagram for diagnostic output plus possible additional tests . .	32
5.6	QLM influence diagram for diagnostic output plus possible additional tests .	33
6.1	Single light influence diagrams depicted in the software . . . . .	40
6.2	Influence diagram of the quadruple light circuit depicted in the software . . .	41
6.3	Abstracted influence diagram for scalability comparison . . . . .	44
6.4	Inference time comparison of single sequence of the discrete ID and average time to come up with the next best test in the LIMID + Myopic . . . . .	45
6.5	Policy change for different costs and prior probabilities . . . . .	47
6.6	Normalized distance between the discrete ID and LIMID + Myopic in the range of the discrete ID and testing all components per probability specification . .	49

## List of Figures

---

6.7	Normalized distance between the discrete ID and LIMID + Myopic in the range of the discrete ID and testing all components per ratio binned . . . . .	50
A.1	Posterior network after observation on $I_{wire1}$ . . . . .	59
A.2	Posterior network after observation on $I_{wire2}$ . . . . .	60

# List of Acronyms

**API** Application Programming Interface

**BN** Bayesian network

**CAD** Computer-aided design

**DAG** Directed Acyclic Graph

**ESI** Embedded Systems Innovation

**ID** Influence Diagram

**JVM** Java Virtual Machine

**LIMID** Limited Memory Influence Diagram

**MAU** Multi-Attribute Utility function

**PCA** Principal Component Analysis

**SFA** Single Fault Assumption

**SID** Sequential Influence diagram

**SPU** Single Policy Updating

**TNO** Nederlandse Organisatie voor Toegepast-natuurwetenschappelijk onderzoek

**TU/e** Eindhoven University of Technology

# Glossary

**Black box model**

a model which produces useful information without revealing any information about its internal workings

**CareFree**

team within ESI for the diagnostics to prevent and reduce machine downtime

**Decision policy**

the action for the decision-maker for all possible observations made prior to making the decision

**Failure**

a permanent interruption of a system's ability to perform a required function under specified operating conditions

**Fault**

an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable condition

**Heap space size**

area of memory used to store objects instantiated by applications running on the Java Virtual Machine

**No-forgetting constraint**

the network has perfect recall of all observations and decisions made in the past

**Regularity constraint**

there must be a total order on all the decision variables in the network

# Chapter 1

## Introduction

### 1.1 Background

In modern industry, machines develop towards being more complicated and intelligent and are subject to growingly demanding operational conditions. An essential criterion for this condition is keeping the system available with high reliability and minimizing unscheduled downtime caused by system faults. One of the possible uncertainties with respect to downtime is the time required to diagnose the issue. The traditional method of doing maintenance and diagnostics, which involved evaluating the entire system, is no longer appropriate for the job. Increasing system complexity puts more pressure on a fast diagnosis of the underlying root cause of an issue and involves a certain least-cost strategy.

The goal of diagnosis is to identify the underlying causes of the detected fault. This task encompasses the fault isolation and identification steps that enable us to characterize the type of fault, its size, and its profile. The occurrence of a fault in the system can generate a failure. The term “fault” is defined by Isermann (2006) as an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable condition. On the other hand, failure is a permanent interruption of a system’s ability to perform a required function under specified operating conditions. This study assumes that the system is not able to perform a required function under specified operating conditions since the aim is to minimize unscheduled downtime caused by system faults. Diagnosing efficiency is becoming important because of several trends:

- as product variability increases, it becomes more difficult to gain experience on specific systems;
- increasing product complexity makes diagnosis more difficult;
- service is delegated to dealer organizations, effectively reducing engineer experience.

Current high-tech industry service organizations tackle complex diagnostics based on extensive service manuals. The walk-through of these service manuals is complex and often requires the knowledge and involvement of scarce system designers. In recent years, there has been an increasing interest in fault detection and diagnosis approaches to cope with such issues. Among these approaches, one can distinguish between data-driven approaches, model-based approaches, and expert knowledge ones (Venkatasubramanian et al., 2003; Cai et al., 2017).



Model-based approaches rely on an analytical model of the system, whereas data-driven methods rely on statistical models built from accessible process data. Expert knowledge methods understandings are usually expressed in terms of mathematical functional relationships between the inputs and outputs of the system. The advantage of model-based and expert knowledge ones over data-driven methods is that they are more reliable in describing the process's dynamics with a piece of physical knowledge. However, several factors such as system complexity, high dimensionality, process nonlinearity and/or lack of good data often render it very difficult even impractical, to develop an accurate (mathematical) model for the system. This, of course, limits the usefulness of this approach in real industrial processes (Atoui and Cohen, 2021). Therefore, there is a lot of interest in data-driven approaches since these methods play an important role in modern systems, especially for large-scale industrial applications since they can reduce time and cost since they do not require the development of models (Alzghoul et al., 2014).

Despite the time and cost savings that data-driven techniques may provide, their development and implementation are hindered by operational and functional challenges. Among the major challenges identified are:

- insufficient data concerning individual machine failure (Zhao et al., 2019);
- their application necessitates a preprocessing step to extract useful information from data with a high computational cost (Tidiri et al., 2016);
- it can be difficult to gain a thorough understanding of the inner workings of many of these data-driven approaches.

The inner workings of many of these data-driven approaches are typically black box models. Since explaining these inner workings will not only enable the acceptance of the diagnostic reasoning system but will also help domain specialists with varying levels of expertise. While theoretical research on system failures is growing, many of the studies depend on simulation data for both normal and malfunctioning situations. Because real-world systems typically lack the historical record of malfunctioning data that theoretical research necessitates, the applicability of solely a data-driven method remains in doubt. It is critical to note that this field is still in its early stages and there is still a lack of a trustworthy, affordable, and scalable solution, as well as real-time implementations for diagnosing high-tech systems. Therefore, a hybrid approach is proposed that could both use data as well as expert knowledge to construct a model to overcome the aforementioned challenges.

## 1.2 Literature Review

Some data-driven diagnostic approaches to generate a reasoning engine exist in literature (Namburu et al., 2006; Tidiri et al., 2016; Mirnaghi and Haghighat, 2020). In general, among the most popular data-driven methods, one can cite Neural Networks (Mohd Amiruddin et al., 2020), Bayesian networks (Cai et al., 2017), Control Charts, Principal Component Analysis (PCA), and Partial Least Squares (Tidiri et al., 2016). The study of Thierno M. L. et al. (2018) compared some of these methods to choose the framework that they utilized to model their diagnostic approach. Four comparative criteria that also suit the goal of this study were used:

- dealing with uncertainty (failure probabilities);
- ability to use data as well as expert knowledge;
- user's ability to understand the existing causal relationships between the various parameters (semantic);
- resources required to implement the method (e.g., the time required for modeling tasks, the amount of memory consumed, the necessary computing time).

Table 1.1 displays the comparative analysis of these methods. Among the methods investigated, the Bayesian network framework is the only one that integrates uncertainties and allows for the use of both data and expert knowledge. The semantics of the Bayesian network makes it possible to understand the causal mechanism linking a symptom to its root cause. Nonetheless, the Bayesian network demands more resources than any other approach.

Table 1.1: Data-driven diagnosis methods analysis (Thierno M. L. et al., 2018)

Methods	Consideration of uncertainties	Incorporation of expert know- ledge	Semantics	Resources needed
Neural Network	—	+	—	— —
Bayesian network	+	+	+	— — —
PCA	—	—	+	— —
Clustering	—	—	—	—

“—” = NO or NEGATIVE

“+” = YES or POSITIVE

The constructs of Bayesian networks are very appealing modeling schemas for diagnostic reasoning systems. Since it handles the issue of inadequate data and assures computational feasibility in modeling and solving sequential decision-making problems. Literature suggests that Bayesian networks augmented with utilities (e.g., costs and benefits) and decisions, called influence diagrams, can assure a least-cost strategy, which is also an important criterion. However, the process of building an influence diagram requires a deep understanding of the problem and is therefore still considered an art (Clemen, 1996; Bielza et al., 2011). It is also known that the modeling process is not yet automated and that the majority of the literature has paid little attention to it. Thus, very little feedback from analysts and experts on their experiences using influence diagrams to build decision-making models is present.

An extensive literature study was performed that studies the building blocks and methods that have been proposed to design Influence Diagram (ID)s while keeping an eye on the fault detection and diagnosis strategy for these systems. The terms found in the articles are structured in a tree-like model as depicted in Figure 1.1. Not all fault diagnosis procedures with influence diagrams are described. The ones falling into the scope of the project are represented and each bold leaf node is discussed in this report.

The study found the educational intended books of Clemen (1996), Jensen (2001), Kjærulff and Madsen (2008), and Koller and Friedman (2009). Their primary audience is practitioners rather than students. It is therefore considered to use the books as the fundamental basis to get a more thorough understanding of the techniques used when building influence diagrams. The most important techniques are mentioned in a tree-like model (Figure 1.1). This model is distinguished into four categories applicable to this report. These books only briefly touch upon techniques for modeling complex decision-making problems and do not discuss real-world applications. Since the ultimate goal is to use theory in practice, real-world problems will be considered. Gómez (2004) provides an overview of prototypes and real applications where influence diagrams play a central role. Both in this overview and the more recent articles, it is noted that diagnostic reasoning in the field of medical applications is much more explored than system failure diagnostics. It is interesting to see if the same principles can be applied for the domain of system failures.

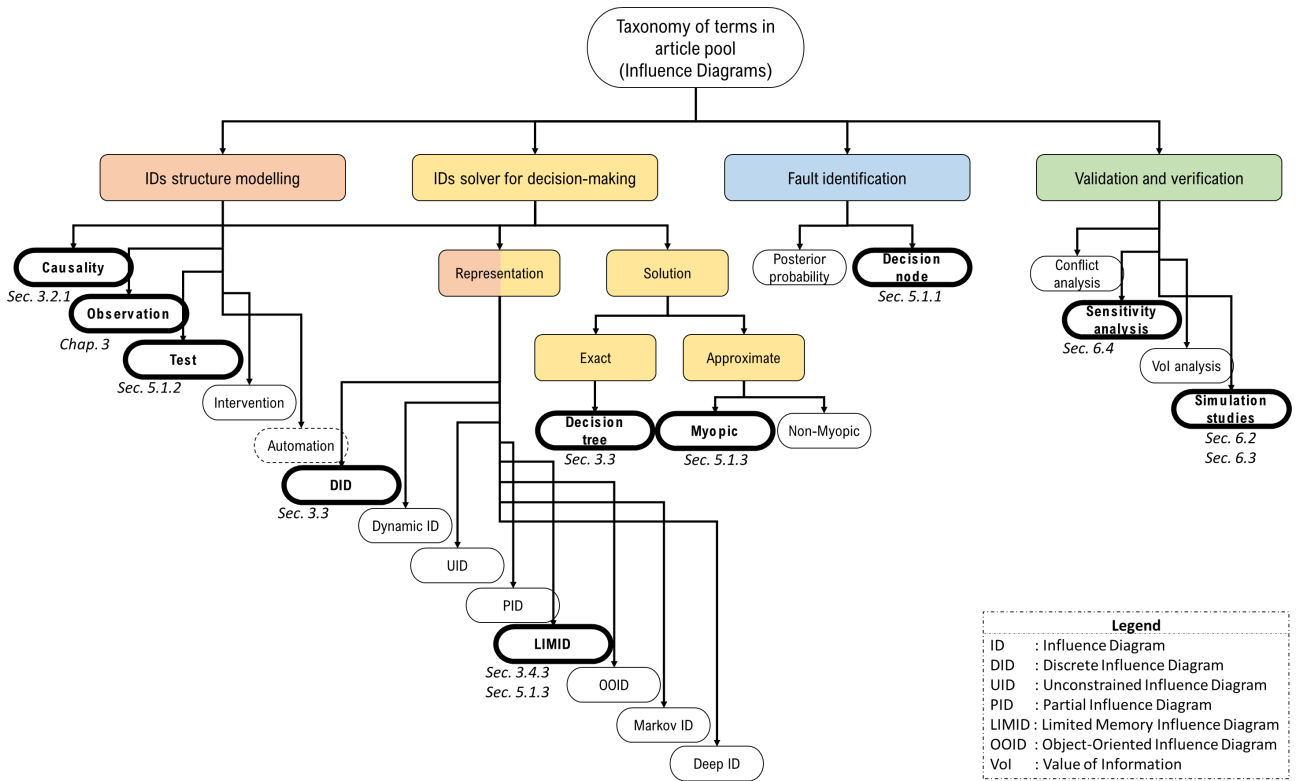


Figure 1.1: Taxonomy of terms in tree-like model

The literature study found some criteria necessary for “IDs structure modeling.” It is important to model causal relations since models with explicit representation of decisions (influence diagrams) must follow the causal relations. Tests can be modelled as these decisions where a distinction is made between observations (probing) and interventions. Probing is a passive observation of the state of a variable, and intervention is an active action that forces a variable to be in a certain state. The latter received little attention when talking about their modeling in influence diagrams. The final criteria in this is that the automated generation of such models has not yet been investigated in current literature.

Also, despite the extensive educational work, most representations and applications still face some widely recognized problems that hinder their application to real-life scenarios. The article of Bielza et al. (2011) reviewed the representation issues and modeling challenges associated with influence diagrams. They discuss various representations accompanied by their strengths and limitations. Some representations to deal with diagnosis problems are unconstrained, sequential, partial, and limited memory influence diagrams as depicted in the 'Representation' branch for the 'IDs solver for decision-making' in Figure 1.1. State-of-the-art representations are presented in the articles of Diez et al. (2017) and Cooper et al. (2019). The latter two discuss Markov and deep influence diagrams. It involves a dynamic network and a differentiable generator network, which require a lot of resources. Thus, they lack the aforementioned challenges of insufficient data, computational cost, and thorough understanding.

In a complex system with several components, component-level diagnostics necessitates the use of a memory-efficient model. When compared to other influence diagrams, the Limited Memory Influence Diagram (LIMID) introduced by Lauritzen and Nilsson (2001) relaxed two of the major constraints: complete order for decision variables and no-forgetting. The latter leads to significant computational demands. By relaxing this assumption, its solution is computationally tractable, whereas the same decision problem could be intractable if the no-forgetting assumption is assumed. However, the optimal solution of a LIMID may not be optimal for the same decision problem if the no-forgetting condition were to be assumed. This gives a trade-off between the tractability of the solution and optimality. According to Cobb (2021), after their debut, Mauá et al. (2011) and Khaled et al. (2013) have released several new computational strategies to improve the solution algorithm for LIMIDs. Nielsen and Sørensen (2010) and Hovgaard and Brincker (2016) have employed LIMIDs for applications in the maintenance of offshore wind turbines and structural damage detection, respectively.

Compared to data-driven methods, research on the application of influence diagrams to fault diagnosis of high-tech systems is scarce despite its introduction that dates back to the early 1980s (Shachter, 1986). Research has developed methods for solving large decision problems using LIMID models. However, applications in offshore wind turbine maintenance and structural damage detection do not fit the current application since they already have a pre-specified sequence of decisions and, in this case, the sequence has yet to be determined. As a result, there is a deficiency of verification of the theoretical framework's applicability and reliability in real situations in this domain.

### 1.3 Research Questions

A lack of understanding of the primary issues encountered by those who attempt to (automatically) model complex decision-making problems on large scales using influence diagrams has been found in the current literature. Also, the applicability of such a method to these systems is currently unexplored. As a result, the purpose of this project is to develop an implementable method to aid in the decision-making process of fast and complex diagnosis of the underlying root cause to minimize unexpected downtime caused by system faults. This project's objective, or main research question, is stated as follows:

*“How to develop a decision support method based on an influence diagram for assisting service engineers in their diagnostics task by advising an efficient diagnostic action?”.*

The following sub-questions have to be answered before the main objective can be achieved:

1. *How can an influence diagram be applied in this domain?*
2. *How to build such a model in a scalable way?*

After answering these sub-questions, an attempt can be made to answer the main research question regarding the induce of a methodology to generate a reasoning engine for decisions on a large scale to cover the best diagnostic action.

## 1.4 Structure of this Document

The remainder of this document is structured as follows: Chapter 2 provides a problem description. Chapter 3 is an introduction to the notation and theorem behind the fundamentals of the methods. The next chapter (4) presents some examples derived from practice. Chapter 5 provides a detailed description of the method for representing and solving the decision support method. In Chapter 6, the proposed methodology is applied and evaluated based on its diagnostic task. Finally, Chapter 7 concludes this thesis and discusses some possible future research lines.

## Chapter 2

# Problem Description

The problem description consists of the formulation of the problem in a general context. The first section presents a broad overview of the problem, accompanied by the current practice. This overview will then be further scoped in the second section to set the boundaries of this research. The final section describes some system/problem-specific issues.

### 2.1 Context

The project is conducted in cooperation with ESI (TNO) and a large professional printer manufacturer. The project's team is working to improve the service process to decrease service costs and system downtime. Their objective is to start with corrective maintenance and work their way up to predictive maintenance. Corrective maintenance is only performed after system failures or breakdowns are reported. It is the technical activity carried out after a failure has occurred, and its purpose is to repair an asset to a condition in which it can perform its intended function. Preventive maintenance measures are planned and performed on a system with the purpose of ensuring that failures do not occur and to lessen the consequences of breakdowns. Preventive maintenance has the considerable advantage of being plannable. This is advantageous to the customer since the unexpected system downtime will be reduced. Predictive maintenance is maintenance that monitors the performance and condition of the system during normal operation to reduce the likelihood of failure. It predicts when equipment failures might occur and tries to prevent the occurrence of the failure by performing preventive maintenance (Jolandie Konig, 2021). This work solely covers hard system down circumstances in which the system is completely inoperable.

For high-tech systems, the time required for corrective maintenance can be broken down into several parts. Figure 2.1 indicates the phases of the time required for corrective maintenance. These phases are further elaborated in Table 2.1 in which some impact factors are depicted.

When looking at Table 2.1, the focus is on the diagnosis phase and specifically the supporting material (in bold). The intention is to provide a tool for the service engineer, regardless of experience, to increase their efficiency. The proposed methodology here must also aid in assessing and improving diagnosability, as creating a technical service manual that captures the detect-and-repair tactic is a difficult undertaking in the current way of working. The difficulty comes from the need to manage multiple and unobservable faults, model uncertainties, noise,

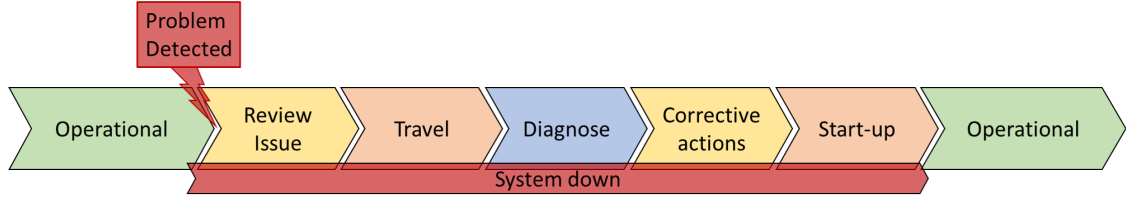


Figure 2.1: Phases of corrective maintenance (ESI (TNO), 2022)

Table 2.1: Some impact factors of corrective maintenance phases (ESI (TNO), 2022)

Review Issue	Travel	Diagnose	Corrective action	Start-up
<ul style="list-style-type: none"> <li>• Customer trying to re-start/repair</li> <li>• Service desk contacting</li> </ul>	<ul style="list-style-type: none"> <li>◦ Travel to customer</li> <li>◦ Make service engineer available</li> </ul>	<ul style="list-style-type: none"> <li>• Diagnosability (e.g. unique error code)</li> <li>• Engineer experience</li> <li>• <b>Supporting material (e.g. service manual)</b></li> </ul>	<ul style="list-style-type: none"> <li>◦ Spare part availability</li> <li>◦ Engineer competence</li> <li>◦ Serviceability (e.g. accessibility to component)</li> </ul>	<ul style="list-style-type: none"> <li>• Rebooting</li> <li>• Stabilizing</li> <li>• Re-calibration</li> <li>• Acceptance test</li> </ul>

and unknown disturbances. This makes the capturing of the process dynamics with a piece of physical knowledge of experts difficult for complex systems. Also, increasing service engineer system understanding is important. An explainable model is beneficial since it will not only enable the acceptance of the diagnostic reasoning system but also help domain specialists with varying levels of expertise.

ESI (TNO) (2022) therefore suggest making use of a diagnostic model-based approach since the aforementioned challenges in data-driven methods (Section 1.1) are more severe than the model-based challenges (Section 1.1 and paragraph above). They proposed a three-step approach in order to construct a proper diagnostic model:

1. From existing technical information, such as design models or technical drawings, create a first model version. Depending on the source of information, this can be partially automated.
2. Manually augments a model into a full diagnostic model.
3. Assess necessary variables, e.g., component prior failure probabilities, from operational data, reliability figures, or expert judgment.

The model's goal is to diagnose a system with  $N$  ( $N \geq 100$ ) components and a set of  $M$  (possible) tests at the component level. If the initial set of observations is too small to reliably identify the root cause, service engineers can improve their diagnosis reliability through various means, such as:

- increasing the number of observations (probing);
- actively changing the system operation (intervention).

If this does not provide a conclusive answer, the process repeats, so the diagnostics task can be regarded as a sequential decision process. In the current work, devising a diagnostic strategy is not easy because multiple failure modes map to the same error code, and a single failure mode can result in multiple error codes. From the technical service manual, with an error code as a starting point, means are suggested to find the root cause. However, deriving such a manual is a difficult undertaking.

## 2.2 Scope

To scope the project, the second step in the diagnostic model construction is the common theme running through this report. There will still be references to the first and last items, but only suggestions will be made. To address the decision-making process in this work, a model of the problem is essential. According to ESI (TNO) (2022), a Bayesian network (BN) model can be (partially) automatically derived from existing technical information. Further details on BNs are provided in Chapter 3. The model of a BN can be derived via one of two approaches, namely the structural or functional approach. See Figure 2.2 for a representation of an assembly from which a BN model can be derived. The model derived using the structural approach uses a formal description of the physical layout, such as a standardized electrical schema or a Computer-aided design (CAD) drawing, to construct a BN (Figure 2.2a). The functional model requires a functional decomposition accompanied by additional human input to come up with a complete BN (Figure 2.2b). ESI (TNO) (2022) suggests using a structural approach where possible and a functional approach when needed.

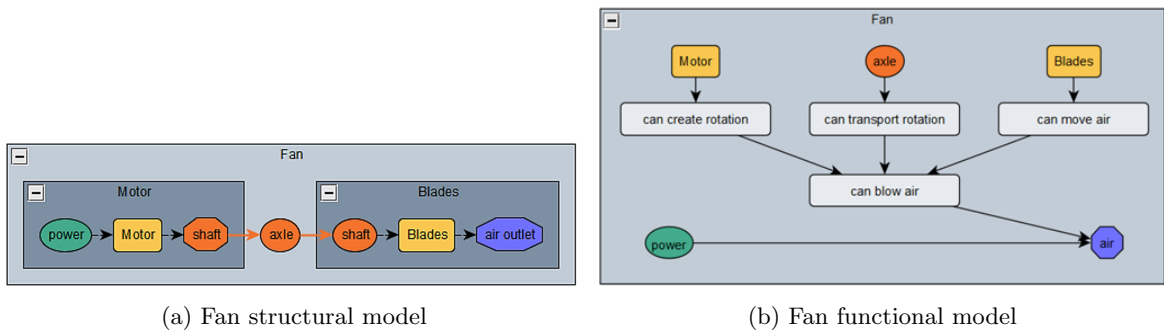


Figure 2.2: Alternative representations of an assembly model of a fan for the Bayesian network construction (ESI (TNO), 2022)

The currently proposed strategy for the decision-making process to improve the service engineers' diagnosis reliability is as follows: Given a BN that is derived from a functional model



and considering all the initial evidence, a calculation of the posterior probability for every state of every node in the network is done. The nodes in the network that represent components and that have a failure probability greater than 0.2 are considered potential failures. For every available test, a calculation of the entropy of the set of potential failures is performed, given the tests. The so-called Shannon's entropy of a test (variable) is the amount of information contained in the variable. It measures the expected (i.e., average) amount of information conveyed by identifying the outcome of a random trial. The top  $m$ -tests that have the lowest entropy (that gives the most information among the set of potential failures) are recommended as tests. This is repeated until the failure probability of a component is larger than a specified threshold or the service engineers' belief is high enough to identify a potential root cause.

The current implementation does not account for the cost of the diagnostic actions nor support the use of the structural model. As the probing and intervening actions also have a resource cost (e.g., time and materials) associated with them, the best sequence of the suggested means to find the root cause. It is also assumed that there is a one-to-one relationship between functions and tests in the functional model. However, there exist functions for which there are no tests or tests that are available but cannot be mapped to a function in the model. It is also remarked that the functional descriptions used in the functional model are typically not precise enough to be easily converted into a model for diagnostics and therefore require additional effort from an expert. It is therefore suggested to continue with the use of structural models.

Two representative examples derived from current practice are depicted in the chapter titled Running Examples of this report. These examples are used for the feasibility study and are of sufficient detail to be comparable with examples from practice. Studying the approach for these examples will demonstrate the suitability for the actual system, but since the research is still in its early stages, full implementation in a complex system is not yet applicable because scalability is still an important topic to address.

## 2.3 Specification

An algorithm or extension of the current approach is necessary to help a decision-maker that tries to decide which information to acquire (which test to do) to reduce the uncertainty over a model, considering costs. The goal is, therefore, to develop an improved solution approach to suggest the next best test. This solution approach has to take some of the cost of used resources into account. The strategy to acquire additional information consequently improves the quality of the diagnosis at hand.

As a part of the research questions, the applicability of influence diagrams is firstly discovered because they seem to fit best with the current approach. Because the current approach uses a Bayesian network and can be augmented with decision and utilities to create an influence diagram its application seems most promising.

Evaluation of the proposed method is important. The currently proposed strategy for decision-making cannot be seen as a benchmark for the newly suggested approach. Since the current

situation does not include cost and uses the unit of bits (“shannons”), which cannot be compared with the value of information when cost is introduced. Therefore, the question of an appropriate comparison material between some models is required for proper evaluation.

# Chapter 3

## Theory

This section will discuss two type of probabilistic networks. The diagnostic tool's basic reasoning engine, a Bayesian network extended to an influence diagram, is a probabilistic graphical model. In the first section, an introduction to the characteristics of the networks used in the subsequent sections is given. The essential mechanisms of a Bayesian network and influence diagram are presented in the second and third sections, respectively. The final section describes two special representations of an influence diagram.

### 3.1 Characteristics Probabilistic Networks

Probabilistic network models are networks in which there are three main classes of vertices, namely, vertices representing chance variables, vertices representing decision variables, and vertices representing utility functions. The edges in the network reflect various types of relationships among the vertices. Chance and decision variables represent an exhaustive collection of mutually exclusive events, known as the variable's domain. These events are also often called "states." The utility function associates a utility value with each configuration of its domain variables. The domain of these vertices might be discrete or continuous; discrete domains are always finite. The vertices representing decision variables and vertices representing utility functions are added when studying influence diagrams (Section 3.3). The variable taxonomy of discrete variables, including the variables in influence diagrams, is summarized in Table 3.1, which was obtained from Kjærulff and Madsen (2008).

Table 3.1: The variables' symbols

(a) The variable taxonomy (Kjærulff and Madsen, 2008)			(b) The variable symbols		
Category	Kind	Subtype*	Category	Kind	Symbol
Chance	Discrete	Labeled	Chance	Discrete	○
Decision	Continuous	Boolean	Decision	Discrete	□
Utility		Numbered	Utility	Discrete	◇
		Interval			

\*Subtype only applies for discrete variables

Throughout this work, circles represent discrete chance variables, rectangles represent discrete decision variables, and diamonds represent discrete utility functions. For ease of explanation, only discrete variables are considered.

### 3.2 Bayesian Network

A Bayesian network is a type of probabilistic network that uses Bayesian inference to update the probability for a hypothesis as more evidence or information becomes available. These models are sometimes referred to as normative expert systems because they give model-based domain descriptions in which the model reflects the problem domain attributes and probability calculus is employed as the uncertainty calculus (Kjærulff and Madsen, 2008). By representing edges between vertices in an Directed Acyclic Graph (DAG), Bayesian networks aim to model conditional dependence. In mathematical terms, a DAG is represented as  $\mathcal{G} = (V, E)$ , where  $V$  is a finite set of distinct vertices (or nodes) and  $E \subseteq V \times V$  is a set of edges. In  $E$ , an ordered pair  $(u, v)$  denotes a directed edge from vertex  $u$  to vertex  $v$ , with  $u$  being a parent of  $v$  and  $v$  being a child of  $u$ . The set of a vertex's ( $v$ ) parents and children are denoted by  $pa(v)$  and  $ch(v)$ , respectively.

Bayesian networks,  $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$ , extend this notation by introducing a set of variables  $\mathcal{X}$  and a set of conditional probability distributions  $\mathcal{P}$ . Each node  $v$  in  $\mathcal{G}$  corresponds to a discrete random variable  $X_v \in \mathcal{X}$  with a finite set of mutually exclusive states. The variables, together with the directed edges  $E$  form a Directed Acyclic Graph. For each variable  $X_v \in \mathcal{X}$ , there is a conditional probability distribution  $P(X_v | X_{pa(v)}) \in \mathcal{P}$ .

**Definition 3.1** (Bayesian network by Jensen (2001)). A Bayesian network consists of the following:

- A DAG  $\mathcal{G} = (V, E)$  with nodes  $V = \{v_1, \dots, v_n\}$  and directed links  $E$ .
- A set of variables  $\mathcal{X}$  represented by the nodes of  $\mathcal{G}$ .
- A set of conditional probability distributions  $\mathcal{P}$  with one distribution  $P(X_v | X_{pa(v)})$ , for each random variable  $X_v \in \mathcal{X}$ .

Note that if  $X_v$  has no parents, then the table reduces to unconditional probabilities  $P(X_v)$ . For the DAG in Figure 3.1, the prior probabilities of  $P(X_1)$  must be specified. The sum of the probabilities in a probability distribution must always equal 1.

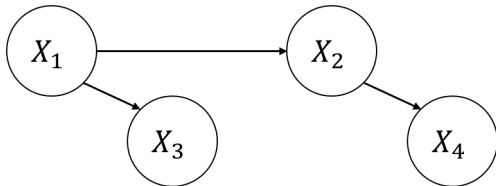


Figure 3.1: A directed acyclic graph

Table 3.2: Example probability table

Distribution $X_1$ with domain $\{0, 1\}$	
$X_1 = 0$	$X_1 = 1$
0.8	0.2

Every Bayesian network demands a particular factorization of a joint probability distribution over a set of random variables,  $\mathcal{X}$ , of a problem domain. The factorization of a joint probability distribution over the variables is given by the directed edges of the DAG. The set of conditional probability distributions,  $\mathcal{P}$ , specifies a multiplicative factorization of the joint probability distribution over  $\mathcal{X}$  as represented by the chain rule of Bayesian networks:

**Theorem 3.1** (Bayesian network chain rule).

$$P(\mathcal{X}) = \prod_{v \in V} P(X_v | X_{pa(v)}) \quad (3.1)$$

where  $X_{pa(v)}$  represents the set of parent variables of variable  $X_v$  for each node  $v \in V$ . The factorization in Equation 3.1 expresses the local Markov property, which states that a node is conditionally independent of its non-descendants given its parents. It is the existence of such an independence assumption that makes it possible to specify the conditional probabilities and to perform inference efficiently in larger Bayesian networks, since, generally, most nodes will have a small set of parents relative to the overall size of the network.

A scenario typically contains some evidence, i.e., evidence or information on some variables have been instantiated, and something about the probability distribution of some other variables needs to be inferred. The conditional probability distribution across the variables of interest is calculated analytically in exact inference. When the Bayesian networks represent a causal relationship between  $X \rightarrow Y$ , where  $X$  is a cause of  $Y$  and where  $Y$  takes the role of an observable effect of  $X$ , assuming that  $X$  cannot be observed itself. Given the observation  $Y = y$  and using the prior distribution  $P(X)$  and the conditional probability distribution  $P(Y|X)$  specified in the model, the posterior probability distribution  $P(X|Y = y)$  can be computed. This computation employs the Bayes' rule, as described in Equation 3.2.

**Theorem 3.2** (Bayes' rule).

$$P(X|Y = y) = \frac{P(Y = y|X)P(X)}{P(Y = y)} \quad (3.2)$$

where  $P(Y = y) = \sum_x P(Y = y|X = x)P(X = x)$ . This theorem plays a central role in statistical inference because the probability of a cause can be inferred when its effect has been observed. Many inference algorithms have been developed, both exact and approximate. One of the more efficient methods of exact inference is through variable elimination (Zhang and Poole, 1994), which takes advantage of the fact that each factor only involves a small number of variables. Jensen et al. (1990) developed inference methods for Bayesian networks based on message passing in a tree structure (junction tree) derived from the structure of the Bayesian network. Both approaches are frequently utilized inference method in current probabilistic network software packages.

### 3.2.1 Construction of Bayesian Networks

As described above, a Bayesian network can be described in terms of a qualitative component, consisting of a DAG, and a quantitative component, consisting of a joint probability distribution that factorizes into a set of conditional probability distributions governed by the structure of the DAG. The construction of a Bayesian network thus runs in two phases.

First, given the problem at hand, one identifies the relevant variables and the (causal) relations among them. The second phase is that the resulting DAG specifies a set of dependence and independence assumptions that will be enforced on the joint probability distribution by constructing the variables priors and conditional probability tables.

Causality plays an important role in the process of constructing probabilistic network models. Although it is not strictly necessary to follow a causal interpretation in a Bayesian network model, models with explicit representation of decisions must follow the causal relations. This is discussed when we talk about influence diagrams in Section 3.3. Causal networks have the additional advantage that they make the model much more intuitive, ease the process of getting the dependence and independence relations right, and significantly ease the process of eliciting the conditional probabilities of the model (Pearl, 2000).

A Bayesian network can be constructed manually, (semi-)automatically from data, or through a combination of a manual and a data-driven process (Jensen, 2001). Manual construction of a Bayesian network can be a labor-intensive task, requiring a great deal of skill and creativity as well as close communication with problem-domain experts. Once constructed, the parameters of a Bayesian network may be continuously updated as new information arrives. Thus, a model for which rough guesses on the parameter values are provided initially will gradually, after updating, improve as it gets presented with more and more cases.

### 3.3 Influence Diagram

Most often, the outputs of interest of a Bayesian network are the posterior probabilities of the variables representing the problem that is reasoned about. These probabilities are often combined with the costs and benefits (utilities) of performing one or more actions to solve the problem. That is, from the posterior probabilities and the utilities, expected utilities for each possible decision option (e.g., different treatment alternatives) can be computed. The decision option with the highest expected utility should then be selected to improve the quality of the decision. A Bayesian network can be augmented with decision variables, representing decision options, and utility functions, representing preferences, that may depend on both chance variables and decision variables. Networks so augmented are called influence diagrams. Nowadays, they have become a popular and standard modeling tool (Pearl, 2005; Bielza et al., 2011).

It is found that influence diagram outputs are remarkably valuable. Given a specific configuration of variables, an influence diagram yields the best course of action. But influence diagram responses are not limited to providing optimal strategies for the decision-making problem. Inferred posterior distributions may be employed to generate diagnosis outputs (probability of a cause). Influence diagrams may also generate explanations of their proposals as a way to justify their reasoning by exploring alternative what-if scenarios (Fernández del Pozo et al., 2005; Bielza et al., 2008).

Influence diagrams,  $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$ , extend the mathematical notation of Bayesian networks by introducing an additional set of decision variables  $\mathcal{X}_D$  in  $\mathcal{X}$  and a set of utility functions  $\mathcal{U}$ . The DAG  $\mathcal{G} = (V, E)$ , has nodes ( $V$ ) which include chance nodes ( $V_C$ ), decision nodes

( $V_D$ ), and utility nodes ( $V_U$ ). These nodes represent random variables, decision variables, and utility functions, respectively. Each decision variable,  $D$ , represents a specific point in time under the model of the problem domain where the decision maker has to make a decision. The states ( $d_1, \dots, d_n$ ) of  $D$  are the decision options or alternatives. The local utility functions associated with  $D$  or one of its descendants in  $\mathcal{G}$  are used to assess the usefulness of each decision option. Each local utility function  $u(X_{pa(v)}) \in \mathcal{U}$ , where  $v \in V_U$  is a utility node, adds to the total utility function  $u(\mathcal{X})$  in  $\mathcal{X}$ . As a result, the total utility function is the sum of all utility functions in the influence diagram,  $u(\mathcal{X}) = \sum_{v \in V_U} u(X_{pa(v)})$ .

**Definition 3.2** (Discrete Influence diagram by Kjærulff and Madsen (2008)). A discrete influence diagram consists of the following:

- A DAG  $\mathcal{G} = (V, E)$  with nodes,  $V$ , and directed links,  $E$ , encoding dependency relations and information precedence, as well as a total order of decisions.
- A set of random variables,  $\mathcal{X}_C$ , and decision variables,  $\mathcal{X}_D$ , such that  $\mathcal{X} = \mathcal{X}_C \cup \mathcal{X}_D$  represented by nodes of  $\mathcal{G}$ .
- A set of conditional probability distributions,  $\mathcal{P}$ , with one distribution,  $P(X_v | X_{pa(v)})$ , for each discrete random variable  $X_v$ .
- A set of utility functions  $\mathcal{U}$ , with one utility function  $u(X_{pa(v)})$ , for each node  $v$  in the utility node subset  $V_U \subset V$ .

A discrete influence diagram supports the representation and solution of sequential decision problems with multiple local utility functions under the so-called no-forgetting constraint (Clemen, 1996), assuming perfect recall of all observations and decisions made in the past. There must also be a total order on the decision variables  $\mathcal{X}_D = \{D_1, \dots, D_n\} \subseteq \mathcal{X}$ . The total order requirement is referred to as the regularity constraint. To specify a total order of ( $D_1, \dots, D_n$ ) on  $\mathcal{X}_D = \{D_1, \dots, D_n\}$ , informational links are present. There needs only to be a directed path from one decision variable to the next one in the decision sequence to enforce a total order on the decisions. In Figure 3.2b the sequence is enforced by modeling a link between  $D_1$  and  $D_2$ . The total order of decisions in this network, therefore, specifies that decision  $D_1$  is a predecessor of  $D_2$ .

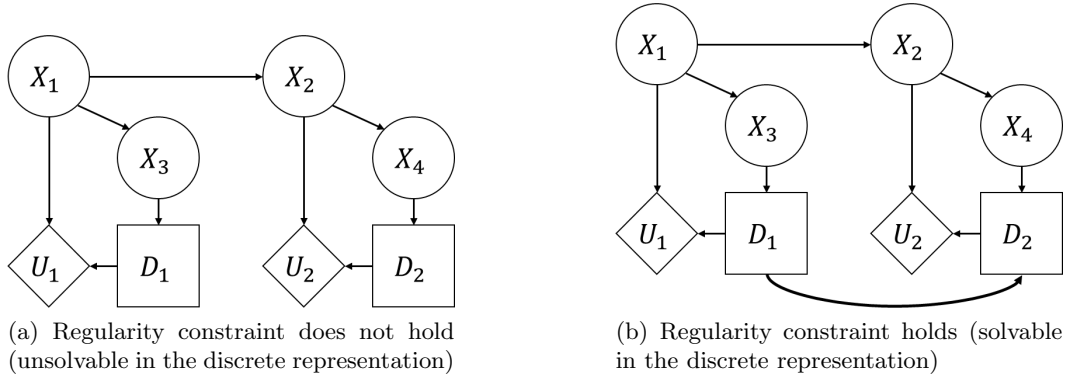


Figure 3.2: A influence diagram representation of Figure 3.1

The interaction between the various nodes assembled into a graph (Figure 3.2b) represents either relevance or sequence, with the meaning indicated by the context of the arrow. Their definition is specified in Figure 3.3. The links into a node representing a random variable,  $X_1$  into  $X_3$ , indicate that the uncertainty at the node is probabilistically conditioned on the preceding nodes and are called conditional arcs (Figure 3.3a). While the links from a node representing a variable,  $X$ , into a node representing a decision variable,  $X_3$  into  $D_1$ , denotes that the state of  $X_3$  is known when decision  $D_1$  is to be made. Everything prior to the decision has to be resolved before it is finalized. Links ending in decision nodes representing (temporal) sequencing are called informational arcs (Figure 3.3b). The preceding nodes can be random as well as other decision variables. For random variables, certainty on the distribution of the variables' potential values is not possible. However, the range of uncertainty can be reduced through prior information-gathering decisions. For prior decision variables, a choice has been made that offers the greatest likelihood of achieving the targeted objectives given the range of subsequent decision alternatives. The final set of links represent functional arcs and end in utility nodes (Figure 3.3c). The functional dependence of  $U_1$  on  $X_v \in \mathcal{X}$  is denoted by a node representing a local utility function,  $U_1$ .

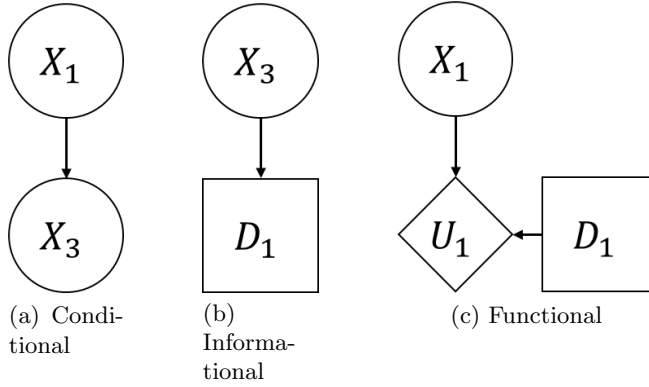


Table 3.3: Example utility function  $U_1$  of  $X_1$  with domain  $\{0, 1\}$  and  $D_1$  with domain  $\{0, 1\}$

$X_1$	$D_1 = 0$	$D_1 = 1$
0	Some cost	some cost
1	Some cost	some cost

Figure 3.3: Definition arcs in influence diagram

As in every Bayesian network, an Influence diagram demands a particular factorization of a joint probability distribution over a set of variables,  $\mathcal{X}$ , of a problem domain. An influence diagram is a compact representation of a joint expected utility ( $EU$ ) function over  $\mathcal{X}$  as represented by the chain rule:

**Theorem 3.3** (The chain rule for influence diagrams).

$$EU(\mathcal{X}) = \prod_{X_v \in \mathcal{X}_C} P(X_v | X_{pa(v)}) \sum_{w \in V_U} u(X_{pa(w)}) \quad (3.3)$$

where  $w$  denotes a vertex from a directed edge between  $w$  to vertex  $v$ . This chain rule can be used to solve  $\mathcal{N}$  by computing the expected utility  $EU(\hat{\Delta})$  of the optimal strategy  $\Delta$ . In order to compute the expected value of the optimal strategy, one must first identify that optimal strategy. A strategy,  $\Delta$ , is an ordered set of decision policies,  $\Delta = (\delta_1, \dots, \delta_n)$  including one decision policy for each decision  $D_n \in \mathcal{X}_D$ . The decision policy outlines how to make choices in order to act in alignment with the purposes and goals. A policy for decision  $D$  specifies



the (optimal) action for the decision maker for all possible observations made prior to making decision  $D$ . For Figure 3.2b, the set of decision policies will look like something like:

$$\delta_{D_1}(X_3) = \begin{cases} 0 & X_3 = 0 \\ 1 & X_3 = 1 \end{cases} \quad \delta_{D_2}(X_4, D_1) = \begin{cases} 0 & X_4 = 0, D_1 = 0 \\ 0 & X_4 = 1, D_1 = 0 \\ 0 & X_4 = 0, D_1 = 1 \\ 1 & X_4 = 1, D_1 = 1 \end{cases}$$

An optimal strategy is defined as  $\hat{\Delta} = (\hat{\delta}_1, \dots, \hat{\delta}_n)$ , which maximizes the expected utility over all possible strategies, that is, it satisfies

$$EU(\hat{\Delta}) \geq EU(\Delta) \tag{3.4}$$

for all strategies using  $\Delta$  (Kjærulff and Madsen, 2008). Later, the introduced discrete influence diagram will be used to evaluate the sequencing problem.

### 3.4 Representation of Influence Diagrams

There are many different representations of influence diagrams, which differ in the way information is handled. Often, the influence diagram is based on the no-forgetting constraint: when making a decision, all previous decisions, as well as previous observations, are known. This requires that there be a (temporal) ordering of the decisions. The no-forgetting constraint leads to significant computational demands. For this reason, two influence diagrams that relax the no-forgetting constraint are discussed. By relaxing this constraint, an unoptimal use of the knowledge is made and, therefore, the decision is not optimal. Both models are shown after a diagnosis problem description is displayed. Before diagnosis problems for system diagnostics are discussed, a medical example will be presented to explain both models.

#### 3.4.1 The diagnosis problem

A doctor is trying to decide on a policy for treating patients who, after an initial examination of their symptoms ( $S$ ) are suspected to suffer from an infection ( $I$ ). This infection has two measurables,  $X_1$  and  $X_2$ , that are observable in e.g. a blood test and urine test. Before deciding on whether or not to treat the infection ( $D_3$ ), the doctor can decide to perform a blood test ( $D_1$ ) and/or a urine test ( $D_2$ ) which will produce the test results. After the doctor has observed the test results (if any), he has to decide whether to treat the patient for the infection. Observe that in this decision problem, the sequence in which the tests are decided upon is unspecified, and that the test result of, e.g., the blood test is only available if the doctor actually decides to perform the test; similarly for the result of the urine test.

To represent this problem with a discrete influence diagram, the unspecified ordering of the  $D_1$  and  $D_2$  has to be represented in a strict ordering of the decisions. Unfortunately, the structure of the decision problem is not apparent from the description, and, for large decision problems, this modeling technique will be prohibitive as all  $N$  possible scenarios ( $N$  possible orderings of  $D$ ), would lead to  $N = |D|!$  different encodings of the model (Jensen et al., 2006).

### 3.4.2 Sequential Influence Diagram

Instead of making the possible decision sequences explicit in the model Sequential Influence diagram (SID) postpone it to the solution phase by allowing the temporal ordering to be unspecified; note that this also implies that when solving the SID we not only look for an optimal strategy for the decisions but also for an optimal conditional ordering of the decisions. The diagnosis problem can therefore be represented by a SID (Figure 3.4) (Jensen et al., 2006).  $D_1$  and  $D_2$  represent two different tests that can be performed to gather information on  $X_1$  and  $X_2$  that take the role of an observable effect of infection,  $I$ . This information can only be observed when the state of  $D_1$  and/or  $D_2$  is *yes* (dash-dot arc). The structural arc emanating from the cluster (dotted circle) indicates that all decisions made within the cluster are followed by the decision  $D_3$ . The arc from symptoms  $S$  indicates that  $S$  is known prior to make decisions on  $D_1$  and  $D_2$ . SIDs are designed for representation of the general class of decision problems, not just diagnosis or troubleshooting problems. When Bayesian networks are augmented to SIDs, fewer additional nodes are required since some links specify the decision outcomes and not the nodes because usually dummy variables are introduced to represent these outcomes. However, the main disadvantage of this representation is that when the number of possible tests is large, solving SIDs may be intractable (Bielza et al., 2011), because every possible scenario in each cluster has to be evaluated.

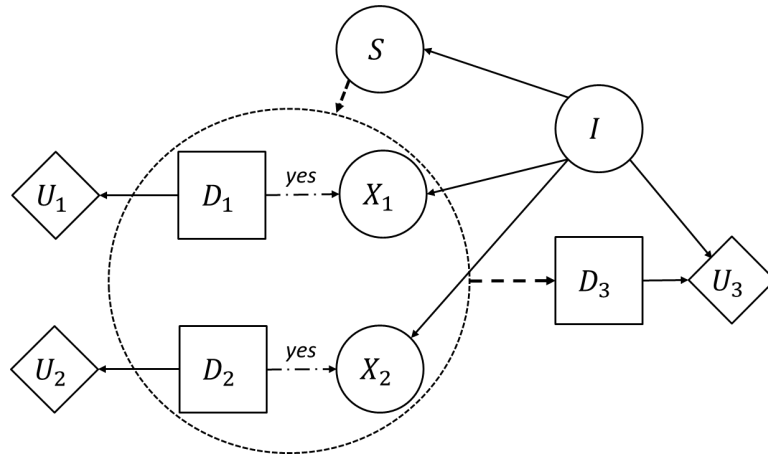


Figure 3.4: A sequential influence diagram representation of the diagnosis problem

### 3.4.3 Limited Memory Influence Diagram

When there may not even be a total order specified among the decision nodes, another representation comes to life. Lauritzen and Nilsson (2001) introduced the notion of Limited Memory Influence Diagram (LIMID) as a model for multistage decision problems in which both conditions dealt with: total order for decision variables and no-forgetting. The decision-making sequence is only defined by its compatibility with the graph's partial order (Figure 3.5), i.e.,  $D_1$  and/or  $D_2$  before  $D_3$ . The parents of a decision node  $D$  represent exactly the variables whose values are known and taken into consideration when a decision at  $D$  has to be made. (Figure 3.5) shows that  $S$  is known before the decisions at  $D_1$  and  $D_2$ , and  $TR_{X_1}$  and  $TR_{X_2}$  are known before the decision at  $D_3$ . Due to the asymmetric nature of the information constraints, this technique will require either dummy variables or dummy states due to the

dummy variables.  $TR_{X_1}$ , for example, is a dummy variable with the values *test\_positive*, *test\_negative*, and *test\_not\_done*. Where *test\_not\_done* indicates that the test has not been performed and therefore no additional costs are incurred. It does mean that that particular test has not been performed because the cost of that test does not outweigh the value of the information it could provide.

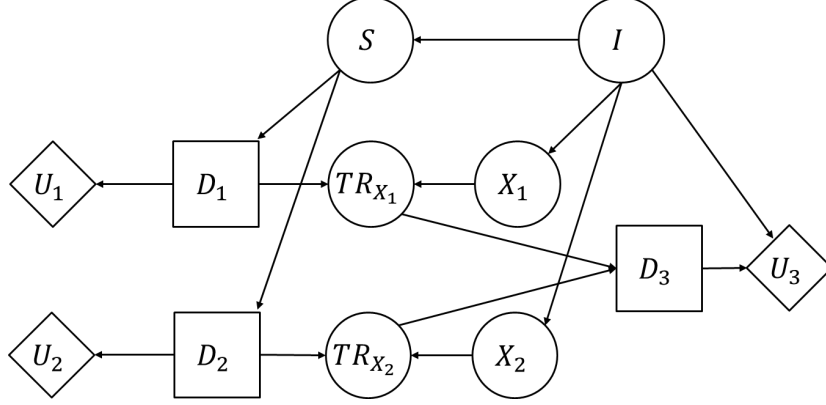


Figure 3.5: A limited memory influence diagram representation of the diagnosis problem

It is also noted that the links point from the measurables ( $X_1$  and  $X_2$ ) to the indicators of the test results ( $TR_{X_1}$  and  $TR_{X_2}$ ). It may appear counter-intuitive at first because the flow of information is from test results to measurables. However, it is the condition of the measurable that causally affects the test result, and not vice-versa. LIMIDs allow for multiple agents of a decision-maker (who may not be able to communicate with each other) or for forgetful decision-makers. The main motivation for LIMIDs is that by limiting the number of information arcs, their solutions are computationally tractable, whereas the same decision problem could be intractable if the no-forgetting condition is assumed. The disadvantage is that an optimal solution of a LIMID may not be optimal for the same decision problem if the no-forgetting condition were to be assumed. This gives a trade-off between tractability of solution and optimality.

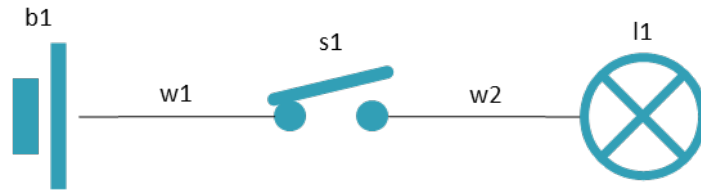
## Chapter 4

# Running Examples

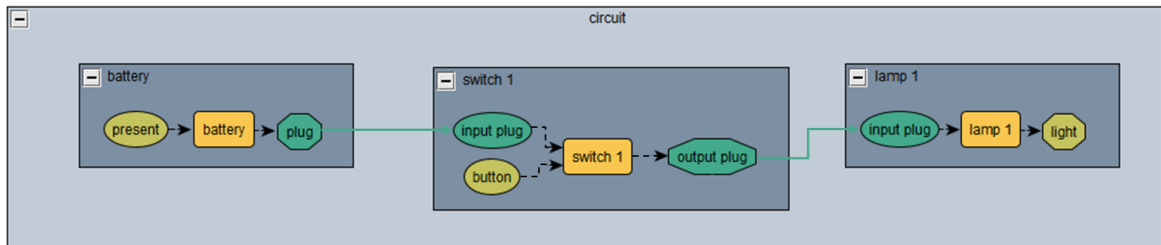
In this chapter, two representative examples are presented to illustrate some test sequence problems. These examples will also be used to introduce some of the proposed methodology's features in Chapter Methodology. In each example, a service engineer must decide to do a certain diagnostic task when the initial set of observations is too small to reliably identify the root cause. Information can be gathered before a repair action can be decided. But how much information should be gathered optimally? Since information does not come for free.

### 4.1 Model 1: Single Light

The single light problem describes an electrical circuit with four different components (see Figure 4.1a): (1) a battery (b1) (2) a switch (s1); (3) a light (l1); (4a) wire (w1) connecting the battery and switch; and (4b) wire (w2) connecting the switch and light.



(a) Electrical circuit model



(b) Structural model

Figure 4.1: Example models for a single light circuit

The system is derived from a structural description of the electrical as depicted in Figure 4.1b. The corresponding structural Bayesian network (BN) with 12 chance nodes is depicted in

Figure 4.2a. The BN is generated by creating a chance node for each component  $I$  that represents its normal behavior and a corresponding node that represents the components' health  $H$ , both in a causal relationship. The BN also includes nodes containing  $P$  of **present**, which show the presence, e.g.,  $P_{bat}$  there is a battery present or  $P_{trig}$  the button is fired. Since these statuses of presence are additional observations and are known upfront, e.g., the trigger is on and there is a battery present, these nodes are redundant and therefore removed from the network (see Figure 4.2b). In this example, the  $I$  represents the presence of power; for example, is there power in the battery ( $P(I_{bat})$ ) given the condition ( $H(I_{bat})$ ) and the presence ( $P(I_{bat})$ ) of the battery? This network can also be generated by the functional model (Section 2.2). This can create a different BN, but a method is proposed that should become independent of the BN structure.

Each components' **Health** node in this network has a corresponding failure distribution of these components attached to it. These distributions consist out of two possible states, for  $H_{bat}$  either *Ok* or *Empty*, the  $H_{wire}$  either *Ok* or *Disconnected*, and for the  $H_{trig}$  and  $H_{lamp}$  either *Ok* or *Broken*. All the states for the nodes  $I$  represent whether there is power thus their states are either *Yes* or *No*.

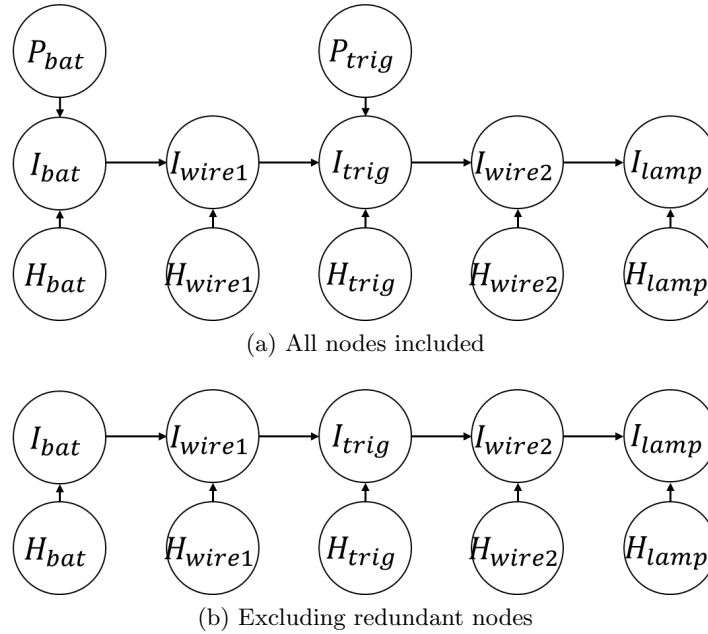


Figure 4.2: Bayesian network of a single light circuit

The problem for these network is: If it is observed that the light is not functioning while the trigger is on and there is a battery present, what is the best action to diagnose why the light is not functioning?

It is useful to recap that due to evidence gathering and Bayesian inference, the probabilities are getting updated. A range of possible tests can be performed to diagnose the problem. These include: (1) observing whether the light is on; (2) checking the status of the light bulb; (3) checking the battery; and (4) measuring the voltage at each wire end. Each has an associated cost of performance. Before deciding which component is likely to be failed,

the decision-maker can decide to perform all or any of these tests. After observing the test results (if any), a decision has to be made on whether the root cause of the problem can be reliably diagnosed. Note that in this decision problem, the sequence in which the tests are decided upon is unspecified and that the test result, e.g., the check light bulb status, is only available if the decision-maker decides to perform the test; similarly for the other test results. It is however assumed that the decision-maker first has the option to gather additional observations or measurements and then decides whether to perform a corrective maintenance action on the components. This makes the decision-maker think about whether it is profitable to perform a test before simply replacing a component.

In the proposed framework, the BN representation is adapted to an ID representation by introducing test results, decisions, and utility nodes. The additional nodes are applied in a general way in which it is easier to possibly automate the adaptation from BNs to IDs in this domain (more details in Chapter 5). This example is introduced to demonstrate the core concept of transforming a BN to an ID. The building blocks derived from this example are applicable to larger networks. The adaptation is not dependent on the BN of the structural model; any BN should fit the adaptation to a ID as long as there is a correct mapping between the possible tests and nodes in the network.

## 4.2 Model 2: Quadruple Light

A larger example of the network described in Section 4.1 is depicted in this example. Here, the electrical circuit problem describes a circuit with 23 components (see Figure 4.3).

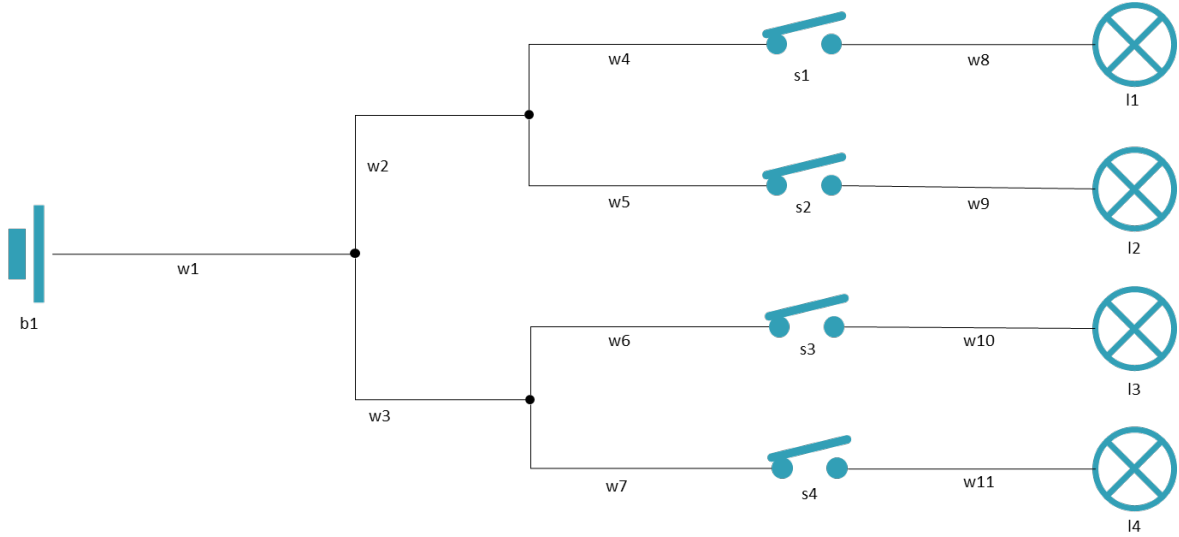


Figure 4.3: Example of an electrical drawing of a quadruple light circuit

The system is depicted in a BN with 54 nodes (see Figure 4.5) and is again derived from a structural description displayed in Figure 4.4. This problem is approached the same way as in Section 4.1. In the BN of Figure 4.6, the redundant nodes are removed. These nodes include again the trigger statuses and the battery presence, but also the joint 1, 2, and 3 health nodes

since their probability of failure is set to zero. Therefore, the same tests or actions could be performed as in the previous section. However, this system is more complex. Figure 4.3 displays an electrical circuit where four lights work in parallel in which additional nodes like joints and switches are added. As a result, the possible number of tests increases due to the increase in the number of components that could be tested. This example is beneficial to explore the possibilities when having a larger and parallel structure in the system. Also, different error scenarios could occur (e.g., one light working, the other not), which gives the possibility of mapping multiple errors into the model.

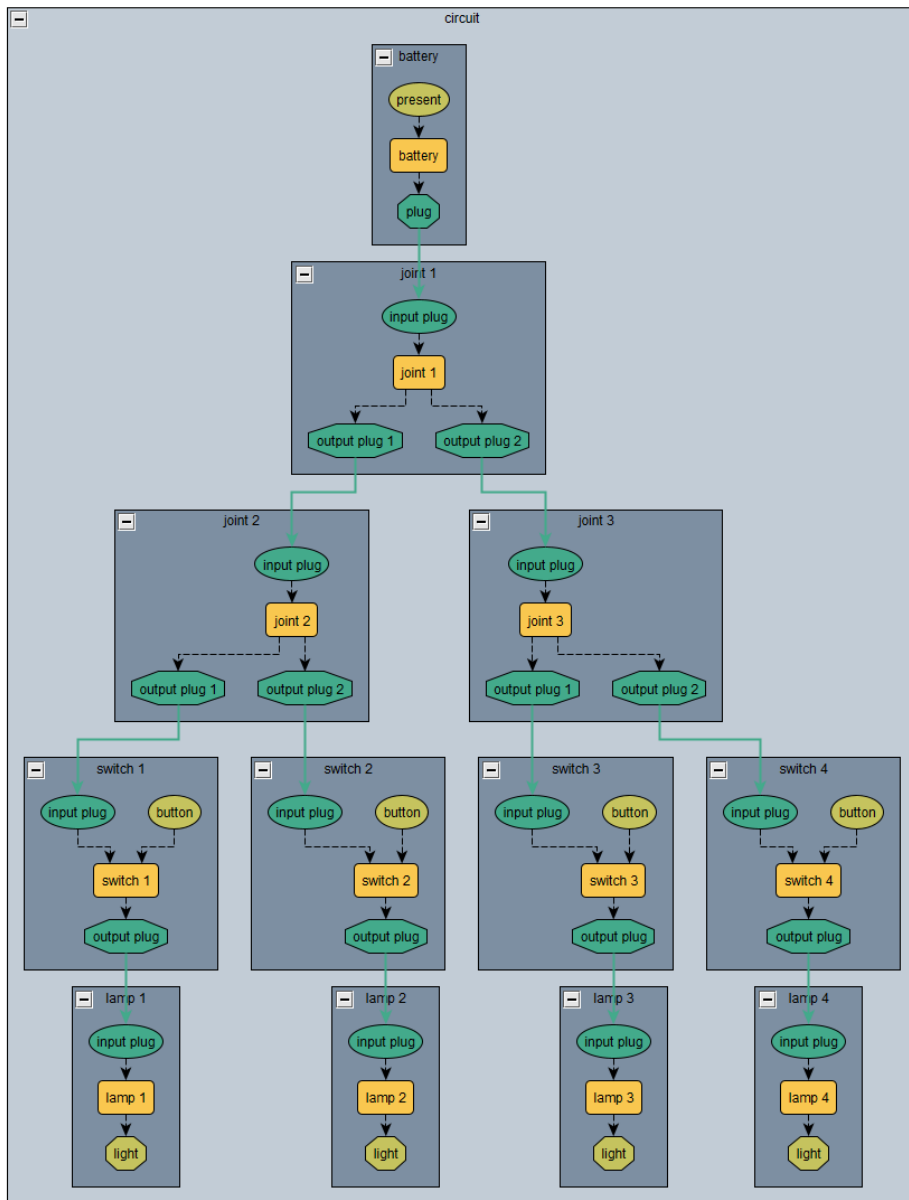


Figure 4.4: Example of a structural description of a quadruple light circuit

Similar to the previous model, a range of possible tests or actions can be performed to diagnose the problem. The association of tests with the functional model is relatively easy in a BN representation because typically a test tests a particular function. However, in these structural models, a clear mapping between the tests and nodes needs to be present. A comprehensible understanding of what each test actually tests is of significant importance. Again, each with a corresponding cost. As mentioned earlier, there are also tests that test components in isolation. Section 5.1.2 addresses this topic by modeling tests in different ways based on what they are doing. The results are shown in Section 6.2 of Chapter 6, which distinguishes between tests for components in isolation and tests for functions.

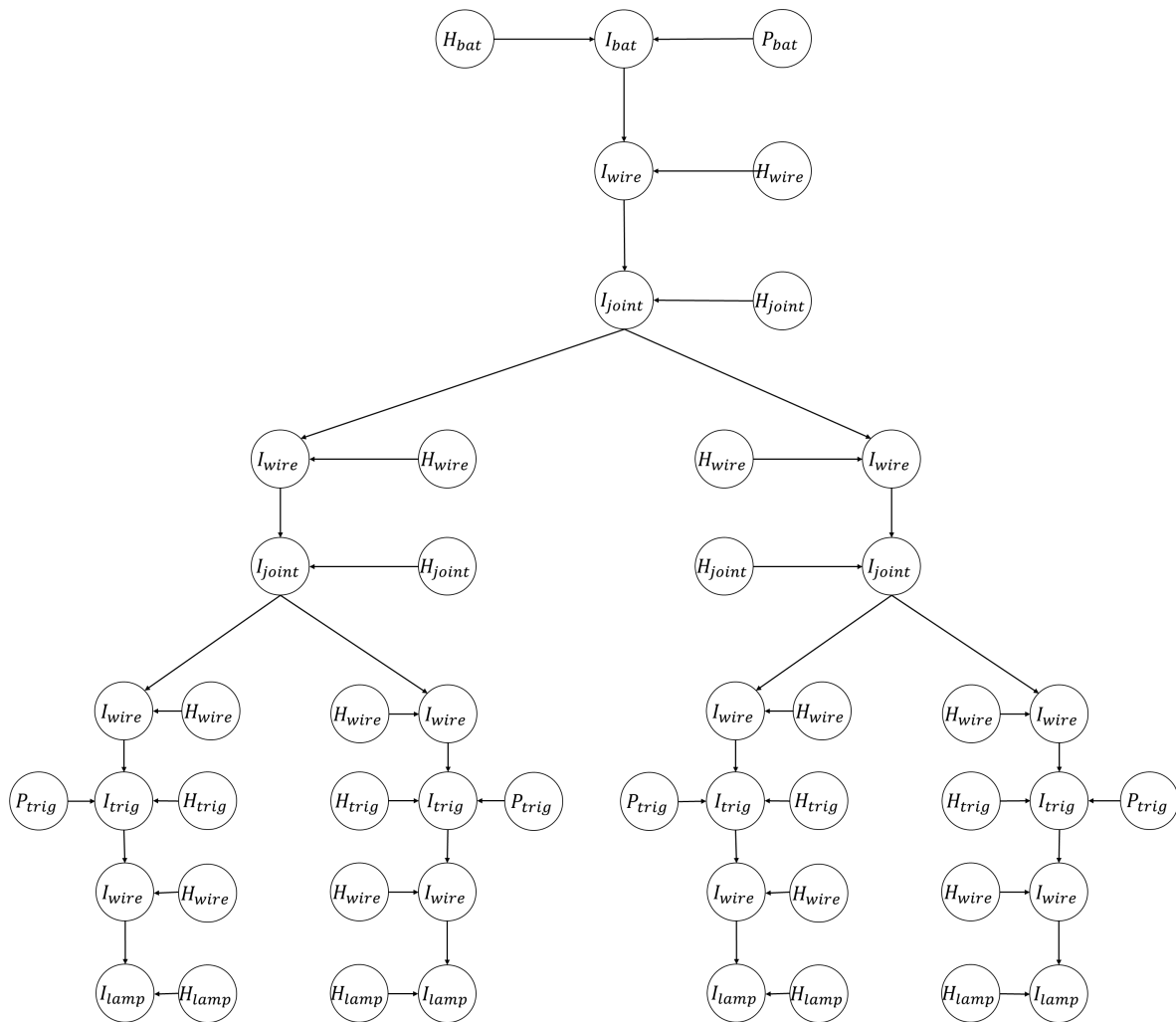


Figure 4.5: Bayesian network of a quadruple light circuit including all nodes



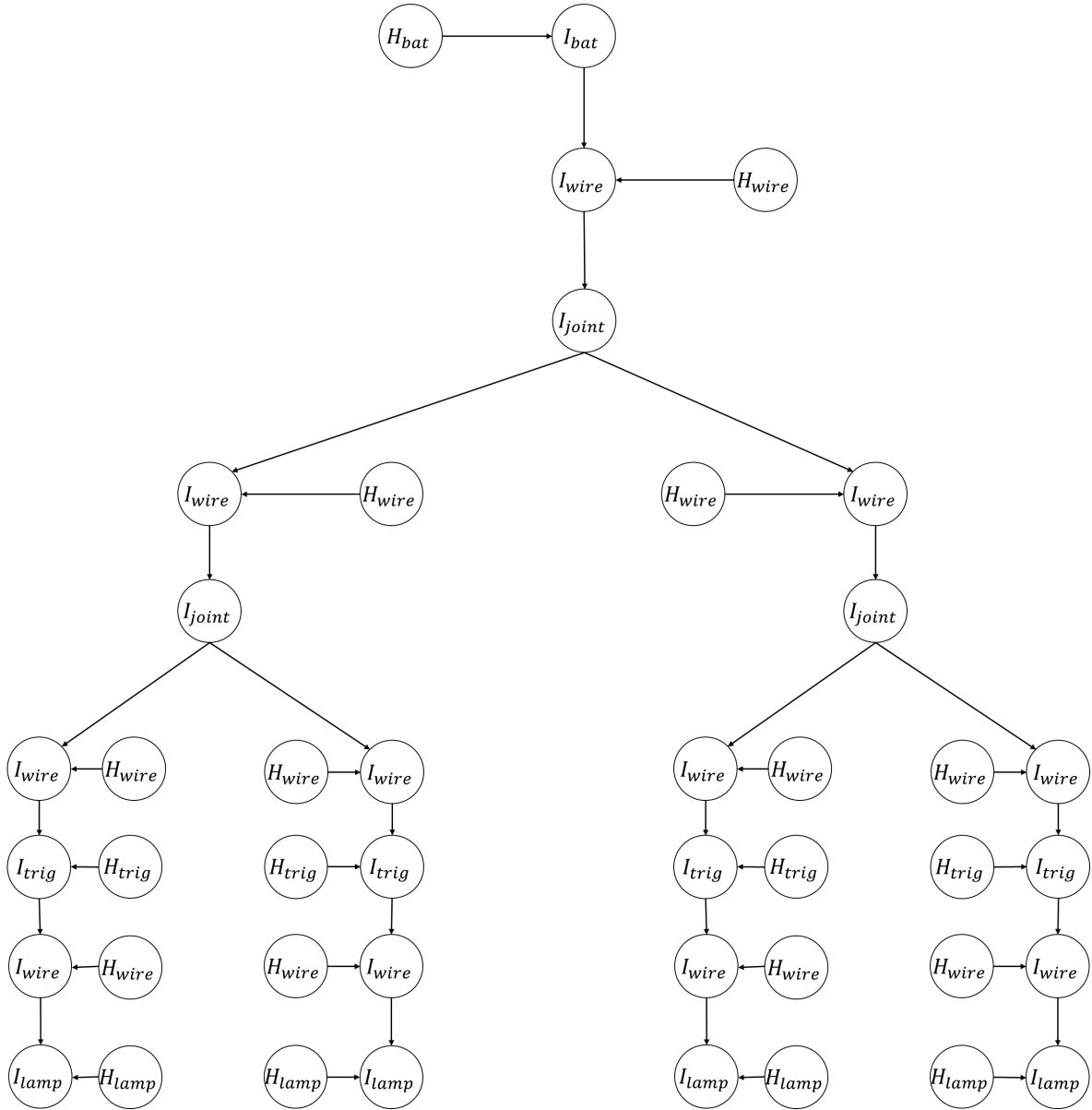


Figure 4.6: Bayesian network of a quadruple light circuit excluding redundant nodes

## Chapter 5

# Methodology

A diagnostic model is proposed to aid in the decision-making process of fast and complex diagnosis of the underlying root cause. This chapter provides details of both the discrete influence diagram and the limited memory influence diagram chosen for the use of modeling and solving the decision-making problem. Table 5.1 provides a summary of the notation used throughout this chapter for convenient reference.

Table 5.1: Notation used throughout this chapter

Notation	Description
$c_1$	Cost of carrying out a test
$c_2$	Replacement cost
$c_3$	Incorrect replacement penalty
$c_4$	Failure-to-repair cost [ $c_4 \gg c_2 + c_3$ ]

### 5.1 Influence Diagram-based Method for Diagnosis

An influence diagram is a valuable tool for developing expert systems that include probabilistic features, including decision-making based on utility values. However, there are several types of influence diagrams that differ in how information is handled. The influence diagram is typically built on the no-forgetting assumption: while faced with a decision, all prior decisions and observations are known. This type is called the Discrete Influence Diagram as described in Section 3.3. This necessitates that the decisions be ordered on time. The notion of no-forgetting imposes enormous computing loads on the system. As a result, the Sequential Influence diagram (SID) and Limited Memory Influence Diagram (LIMID) were developed, which establish an explicit relationship between the nodes known before making the choice and the decision node. Only the parents of a decision node are known in the LIMID at the moment of the decision. This limits the number of nodes considered for the decision, reduces the policy domain, and makes it easier to find the (sub-)optimal strategy with the highest expected utility. In conclusion, the LIMID framework is a viable choice for modeling large and complex domains with an appropriate assumption of decision-maker forgetfulness. The previously introduced SID is hindered by its solving intractability for large networks.

The LIMID is therefore utilized for the influence diagram-based approach for diagnosis. The utilized LIMID is later compared to the discrete influence diagram.

### 5.1.1 Structure of the Model

Following the procedure shown in the flowcharts in Figure 5.1, an influence diagram-based method for diagnosis can be abstracted using the Discrete ID and LIMID. Both these flowcharts also shows the corresponding sections where these procedures are discussed. An influence diagram may be generated automatically given a Bayesian network and a mapping between tests and network nodes. A mapping of this type is built using the system designer's and the Bayesian network constructor's knowledge. The input, such as component prior failure probability derived from operational data, reliability numbers, or expert judgment, has already been initialized in the Bayesian network. The construction is kept generic and is therefore the same for both methods.

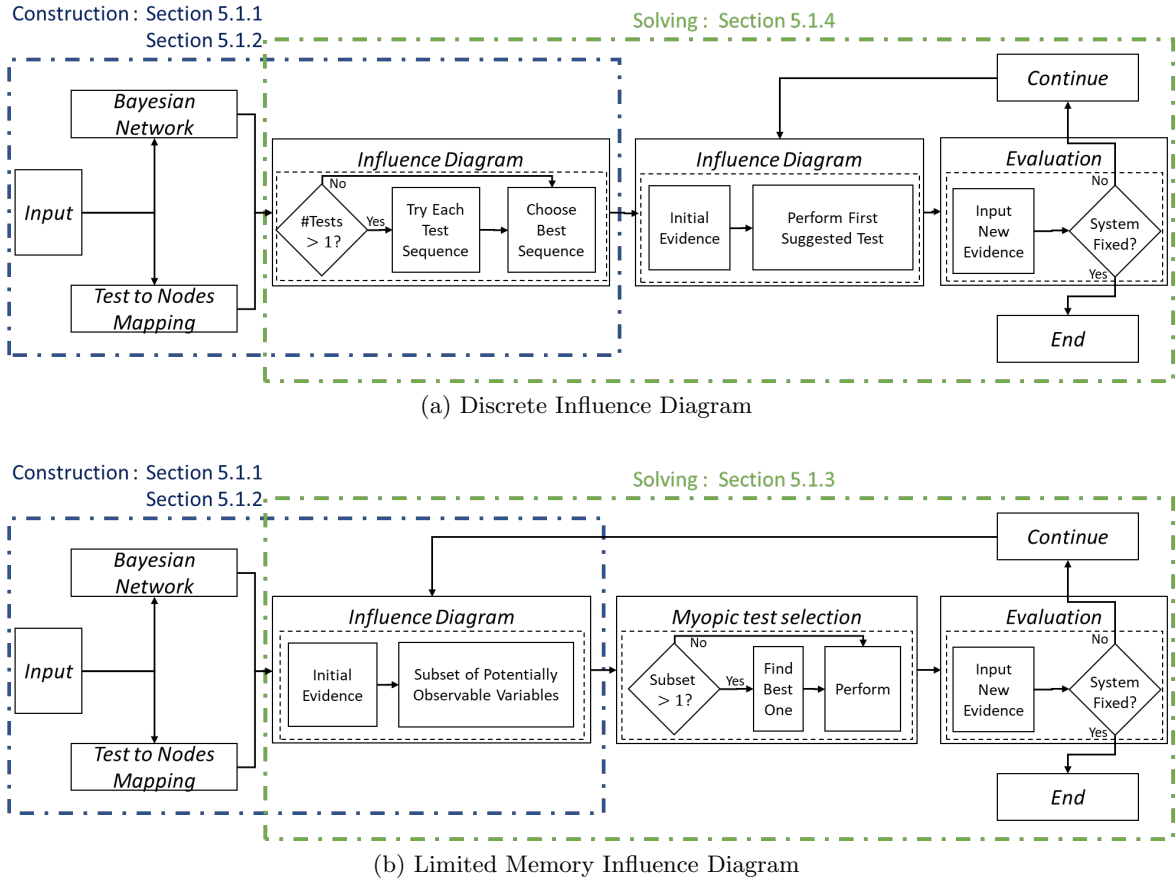


Figure 5.1: Flowchart of approach

There are some initial assumptions for the construction and solving of the network. For the Bayesian network; (1) component failures are independent of each other. For the influence diagram; (2) test costs are independent, (3) tests are perfect, (4) no intermittent faults, and (5) only probing tests are considered.

## Diagnosing

First, the construction of an influence diagram for diagnosis can be derived from models used in medical applications. Lacave et al. (2007) and Luque et al. (2016) both propose models in which they represent the possible tests, their costs, and their outcomes. The found educational books referred to in Section 1.2, assent to this way of modeling. However, an adaption of these models is proposed since all of these models only suggest one possible action (treatment) after diagnosing. This is in line with the single fault assumption. The diagnosis of a high-tech system usually consists of  $n \geq 1000$  components, this increases the probability of having more than one fault. Since the analysis is on the component level, this assumption is relaxed by giving each component its own action (treatment) node. As mentioned in Section 3.3, inferred posterior distributions may be employed to generate diagnosis outputs. Figure 5.2a's model is augmented with utility and decision nodes in Figure 5.2b. This figure uses the **Health** states of the components to generate these diagnosis outcomes.

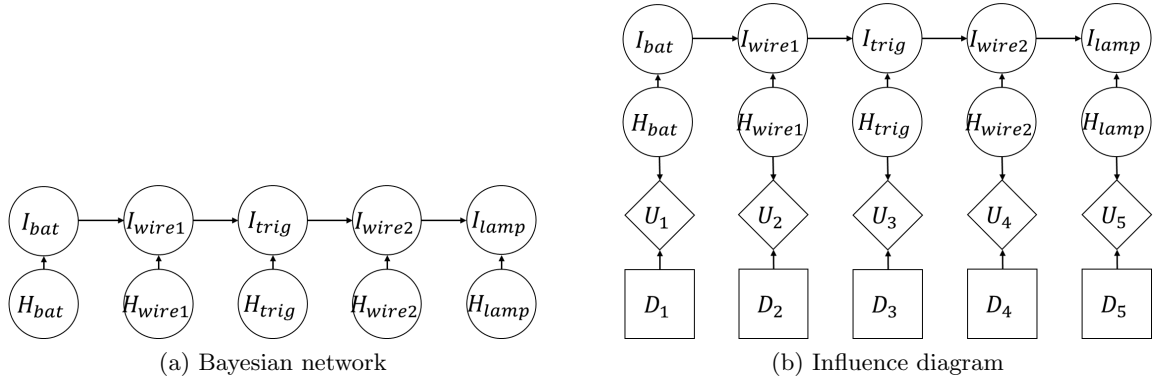


Figure 5.2: Transforming the Model 1: Single Light Bayesian network to an influence diagram for diagnostic output

As mentioned in Section 2.2, the current Bayesian network approach uses rules that give threshold values for inferred posterior distributions, such as potential failure if  $P(H_{bat} = Empty) \geq 20\%$  or replacing the battery if  $P(H_{bat} = Empty) \geq 80\%$ . When modeling as in Figure 5.2b, threshold values are automatically specified based on their local utility function. These functions,  $U_n \in \mathcal{U}$ , denote the functional dependence of  $U_n$  on  $H_n$  and  $D_n$ . Corrective maintenance refers to the decision,  $D_n$ , to perform an action on component  $n$ . It contains the decision options  $(D_n = 0, D_n = 1) = (Do\_nothing, Replace\_n)$ . The threshold is based on the specification of Table 5.2 derived from the ratio after solving the equation:

$$\frac{P(H_n = 0) - (c_2 + c_3)}{P(H_n = 1) \quad c_2 - c_4} = 1 \quad (5.1)$$

where  $P(H_n = 0)$  denotes the probability of a component in good condition and  $P(H_n = 1)$  denotes the probability of a component with a defect or flaw. This opens the opportunity to automatically generate threshold values based on the cost of the part and replacement time. Meaning, more expensive parts need a higher certainty of  $P(H_n = 1)$  before getting a corrective action, which is intuitive. For example, when Equation 5.1 returns  $\frac{P(H_n=0)}{P(H_n=1)} \frac{1}{1} = 1$ ,

the threshold value is depicted as 50% in Figure 5.3,  $D_1$ . This figure also shows different threshold values for the ratios  $\frac{1}{3} = 25\%$  and  $\frac{4}{1} = 80\%$  for  $D_2$  and  $D_3$ , respectively. This shows, the importance of the parameters  $c_3$  and  $c_4$ .

Table 5.2: Utility function corrective maintenance

$H_n$	$D_n = 0$	$D_n = 1$
$H_n = 0$	0	$c_2 + c_3$
$H_n = 1$	$c_4$	$c_2$

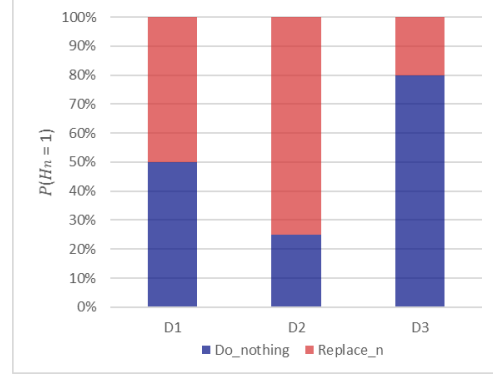


Figure 5.3: Automatically determined  $D_n$  threshold values

Decisions made in the absence of complete information are inevitably uncertain; there is a probability that the decision will be incorrect, with consequences in terms of the payoffs associated with the decision. According to the theory in Section 3.3, the optimal option for a risk-neutral decision-maker is to pick the alternative with the highest expected value, regardless of uncertainty. Nonetheless, decision uncertainty is of importance in determining the value of gathering additional knowledge to better inform future decisions (Meltzer, 2001). A decision-maker may be confronted with the possibility of conducting a test to gather additional information. The “Test to Node Mapping” in Figure 5.1 depicts the collection of available tests. The extension of the model in Figure 5.2b to include test nodes to gather additional information is shown in Figure 5.4. The example assumes that all components are tested individually (in isolation). This does not have to be the case; for example, there could be a test to see if there is power at  $I_{wire1}$ . If the test result observes that there is power, the belief of the antecedents,  $\{H_{wire1}, I_{bat}, H_{bat}\}$ , also has power given Bayesian inference. This will be further discussed in Section 5.1.2.

### 5.1.2 Test Modeling

Performing a test yields a test result that is modeled as a dummy random variable,  $Test$ , with states matching the various test results. The influence diagram includes a decision variable,  $T$ , with states ( $t_1 = no, t_2 = yes$ ), denoting whether or not the test is suggested, in addition to the random variable representing the test result. If the test is carried out, the outcome is made available to the decision-maker (link  $Test_n$  to  $D_n$ ). If, on the other hand, the test is not done, no test results are available. The test establishes the value of information according to the extent to which it might reduce the expected costs of uncertainty by reducing uncertainty in the evidence base. This entails evaluating the expected value of a decision made with and without additional information. When payoffs are represented in monetary terms, an explicit monetary valuation of the expected value of information is provided, which may then

be directly contrasted against the expected cost of information to evaluate if it is valuable. The influence diagram also includes a utility function,  $U \in \mathcal{U}$ , that specifies the cost of the test.

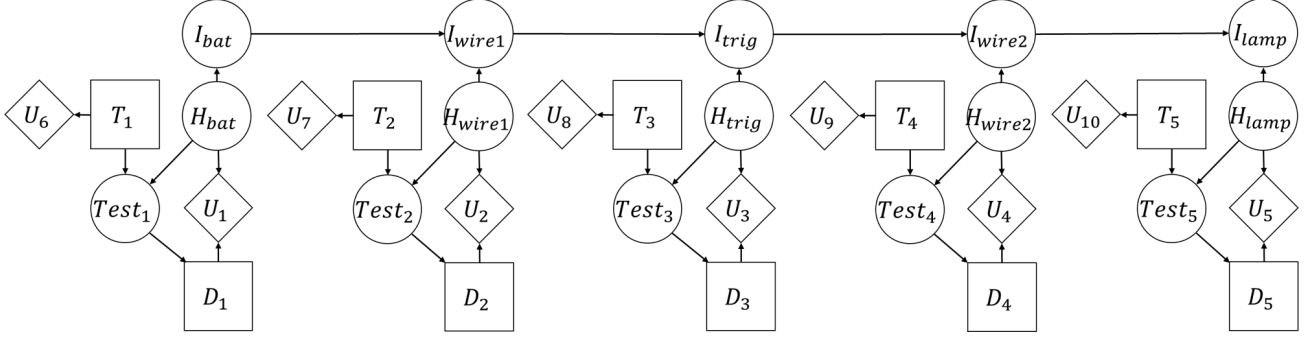


Figure 5.4: Influence diagram of Model 1: Single Light for diagnostic output, as well as isolated tests

The link from  $Test_n$  to  $D_n$  specifies that the value of  $Test_n$  is known when making the corrective maintenance decision  $D_n$ . This cannot, however, be the case when the test is not performed. According to Kjærulff and Madsen (2008), there are two alternative options to correctly model the test result variable to behave as expected when the test is not performed. Both options consider the specification of the conditional probability distribution (CPT)  $P(Test_n|H_n, T_n)$  in different ways. Table 5.3 handles this by specifying  $P(Test_n|H_n, T_n = no)$  as a uniform distribution. Table 5.4 introduced an additional *not\_done* state, such that not performing the test instantiates that state.

Table 5.3: CPT without state *not\_done*

$H_n$	$T_n$	$Test_n = pos$	$Test_n = neg$
0	<i>no</i>	0.5	0.5
0	<i>yes</i>	1	0
1	<i>no</i>	0.5	0.5
1	<i>yes</i>	0	1

Table 5.4: CPT with state *not\_done*

$H_n$	$T_n$	$Test_n = nd$	$Test_n = pos$	$Test_n = neg$
0	<i>no</i>	1	0	0
0	<i>yes</i>	0	1	0
1	<i>no</i>	1	0	0
1	<i>yes</i>	0	0	1

The second option is more semantically clear than the first one because it is self-evident that if the test is not performed,  $Test_n$  should be instantiated to *not\_done*. However, by introducing the additional *not\_done* state in the  $Test_n$  variable, the model's complexity increases. For the sake of convenience, it is chosen to continue with the latter option. Note that there is no uncertainty when performing a test, i.e.,  $T_n = yes \rightarrow P(Test_n = pos) = 1 \vee P(Test_n = neg) = 1$ . The assessment of the value of information is called the “expected value of perfect information” (EVPI) because it quantifies the value of acquiring perfect information about all aspects of the decision (eliminating all uncertainty). This does not have to be the case that all the uncertainty of a variable is eliminated. However, it is assumed that all tests are perfect.

Up till now the set of possible tests exists out of tests that test components in isolation, as seen in Figure 5.4. However, such a test may not exist for some triggers, such as  $T_3$ . An

additional observation can be made by measuring the voltage at the wire ends. This test is likely cheaper than checking the health of each component individually, and it might improve the reliability of the model the most. The network will look like Figure 5.5 if the measurement tests ( $T_6$ ,  $T_7$ ) are included but not the trigger test ( $T_3$ ). Measuring a voltage at the output of  $I_{wire1}$  indicates that it is likely that the trigger, wire2, or lamp is broken. While measuring no voltage indicates there is probability a defect in the battery and/or wire1, plus an additional chance (prior probability) that trigger, wire2, and lamp are also broken. See Appendix A for a visualization in the software of these scenarios.

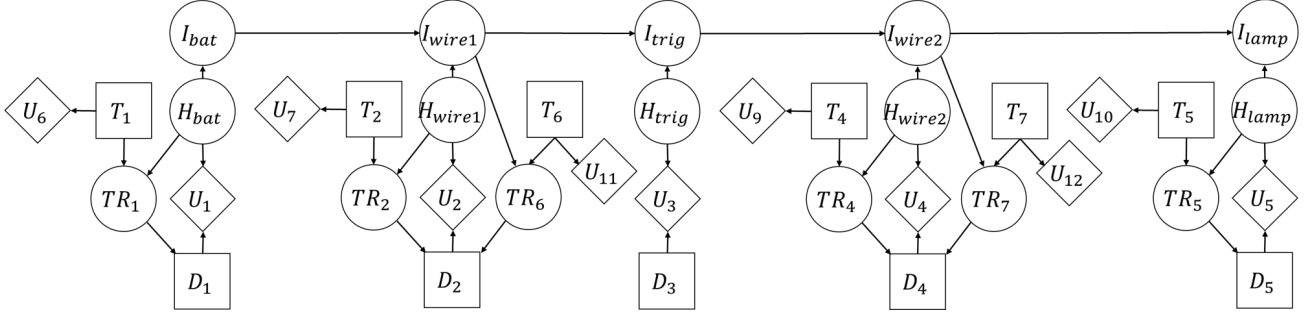


Figure 5.5: SLM influence diagram for diagnostic output plus possible additional tests

The informational links from  $Test_6$  to  $D_2$  and from  $Test_7$  to  $D_4$  in Figure 5.5 indicate that the information is only available prior to making a decision on  $D_2$  and  $D_4$ . However, this information is also available for all decisions. But limiting the number of information links is the main motivation of LIMID. Its solution is computationally tractable, whereas the same decision problem could be intractable if the no-forgetting condition is assumed. It is not always obvious which informational links to include in a LIMID without affecting the policies and the expected utility of the computed policies.

These are considered the building blocks for modeling the diagnosis network and are applicable to Model 2: Quadruple Light. Figure 5.6 depicts the augmented Bayesian network to an influence diagram of the network in Figure 4.6. Hence, it is assumed that every component can be tested in isolation, except for the connecting wires. These are modeled similarly to  $T_6$  and  $T_7$  (Figure 5.5), where the voltage at each wire end can be measured.





### 5.1.3 Solution LIMID

In the ID, decisions are made based on the information available when making those decisions. In the LIMID, these are the nodes with links pointing to the decision node. A policy consists of a set of rules that define which decision to take as a function of the available information. The more information is used for making a decision, the larger the policy domain and, consequently, the computational demand. A set of policies for all the decision nodes in the ID is called a strategy. Lauritzen and Nilsson describes a single policy updating algorithm for finding a (locally) optimal solution to a LIMID.

The LIMID is solved by Single Policy Updating (SPU) via message passing in a junction tree. The SPU algorithm proceeds by passing messages toward a root node containing a decision variable and its parents from all other nodes in the junction tree. As each root node for each decision variable is encountered, the strategy is updated. A forward and backward pass through all the root nodes containing decisions constitutes one iteration of SPU. Iterations of SPU are continued until the maximum expected utility is unchanged. This is the optimal solution to the LIMID, and the current strategy set is optimal.

Mathematically (Lauritzen and Nilsson, 2001), let  $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$  be a LIMID model. The probabilistic policy function is a key encoding in the process of solving a LIMID.

$$\delta'_i(d_i | \mathcal{J}(D_i) = j) = \begin{cases} 1 & \text{if } d_i = \delta_{D_i}(j), \\ 0 & \text{otherwise.} \end{cases} \quad (5.2)$$

of a decision policy  $\delta_{D_i}(\mathcal{J}(D_i))$ . Where  $\mathcal{J}$  is a disjoint information set of random variables relative to the decision variables specifying the precedence order. The information set  $\mathcal{J}$  is the set of variables observed after decision  $D_i$  but before decision  $D_{i+1}$ . And  $\delta_i$  can be thought of as a function that maps relevant observations from  $\mathcal{J}(D_i)$  to the entire decision domain  $\text{dom}(D_i)$ .

The aim of solving  $\mathcal{N}$  is to identify a strategy,  $\Delta$ , that maximizes the expected utility. The SPU algorithm starts with some initial strategy and iteratively updates a single policy until convergence has occurred. Convergence has occurred when no single policy modification can increase the expected utility of the strategy. Assume  $\Delta$  is the current strategy and  $D_i$  is the next decision to be considered for a policy update, then SPU proceeds by performing the following steps according to Kjærulff and Madsen (2008):

- |         |   |
|---------|---|
| Retract | Retract the policy $\delta'_i$ from $\Delta$ to obtain $\Delta_{-i} = \Delta \setminus \{\delta'_i\}$ (i.e., $\Delta_{-i}$ is a strategy for all decisions except $D_i$ ) |
| Update  | Identify a new policy $\hat{\delta}'_i$ for $D_i$ by computing;   |

$$\hat{\delta}'_i = \arg \max_{\delta'_i} EU(\Delta_{-i} \cup \{\delta'_i\}). \quad (5.3)$$

- |         |   |
|---------|---|
| Replace | Set $\Delta = \Delta_{-i} \cup \{\hat{\delta}'_i\}$ |
|---------|---|

The strategy  $\hat{\Delta} = \{\hat{\delta}_{D_i}\}$  is the optimal solution to the algorithm after convergence (when no single policy modification can increase the expected utility of the strategy). The solution

represents the optimal set of decisions given the current evidence of the network. When evidence is added or changed, the solution to the algorithm may change. Each policy is only a function of the most recent test results, if any. This implies that previous results and decisions are ignored. The set of test decision variables suggested to be executed in the influence diagram represents the subset of potentially observable variables relevant to each network state. It is viable that the subset of potentially observable variables relevant to each decision in the optimal strategy  $\hat{\Delta}$  of the LIMID representation suggests several possible tests to execute. This creates a new problem.

### LIMID + Myopic Approximation

When it is allowed to only perform one test and the network suggests several possible tests to execute, a calculation of the expected value of each test may be performed, but it is not sure that the best choice is the one with the highest expected value. A proper analysis of the choices should consist of an analysis of all possible sequences of tests (including the empty sequence). To avoid such an intractable analysis, the so-called myopic approximation is often used. Jensen (2001) calls this *hypothesis-driven*: since usually the distribution of a hypothesis variable  $H$  is the target of the analysis. In the current situation, the myopic approximation is used to come up with the next best test according to the following steps:

Retract	Retract the (locally) optimal solution, $EU(\hat{\Delta}) = EU(\{\hat{\delta}_{D_i}\})$ , to the LIMID
Compute	Calculate the expected value;

$$EV(T) = \sum_{t \in T} EU(\hat{\Delta}|t) * P(t) \quad \forall T \in \hat{\delta}_{D_i} \quad (5.4)$$

Suggest	Suggest the next best decision by computing;
---------	--

$$\hat{D}_i = \begin{cases} \arg \max_{\hat{\delta}_{D_i}} EV(\hat{\delta}_{D_i}) & \text{if } \max_{\hat{\delta}_{D_i}} EV(\hat{\delta}_{D_i}) \geq EU(\hat{\Delta}), \\ \emptyset & \text{otherwise.} \end{cases} \quad (5.5)$$

When the subset of potentially observable variables (test decisions) relevant to each network state is larger as 1, the expected value of each test is calculated. This expected value was obtained by extrapolating one step forward and computing the expected utility based on the test result of one of the potentially observable variables ( $EU(\hat{\Delta}|t)$ ). The expected utility of the network knowing the test result is multiplied by the probability of occurrence of that test result ( $P(t)$ ) because the outcome of the test is unknown (yet). This is done for each test result of the test. And finally, all the multiplied expected utilities of a test are added up ( $\sum_{t \in T} EU(\hat{\Delta}|t) * P(t)$ ). This results in the expected value of a test  $EV(T)$ . This process is repeated for each suggested test. Then, if you are allowed to perform at most one test, the one with the highest expected value is chosen, if it is larger than the initial expected utility. An algorithm that finds the next best test using the described procedure is depicted in Appendix C.

### Solving LIMID + Myopic Approximation

The proposed LIMID + Myopic Approximation only looks one timestep ahead to suggest the next best test. To get the expected utility of the proposed method over the full solution, some simulations have to be executed. Each possible failure scenario (e.g., the battery is broken, the rest is OK) needs to be resolved. This is done by testing and replacing components until *End* is reached (in Figure 5.1). The amount it costs to solve the problem is then multiplied by the probability of that scenario occurring. And the final expected utility is then the sum of all the utilities multiplied by the probabilities. Table 5.5 shows how these expected utilities are derived for the example displayed in Figure 5.4.

Table 5.5: Calculation on the expected utility for the LIMID + Myopic Approximation

$H_{bat}$	Scenarios				Utility*	$P(scenario)^*$	Weighted Utility
	$H_{wire1}$	$H_{trig}$	$H_{wire2}$	$H_{lamp}$			
1	0	0	0	0	-12	0.315	-3.780
0	1	0	0	0	-18	0.016	-0.288
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	1	1	1	-58	$9.870 * 10^{-11}$	$-5.725 * 10^{-9}$
<i>Sum:</i>						1.0	Expected Utility

\*Example values are depicted

#### 5.1.4 Solution Discrete ID

The discrete ID described Section 3.3 is used to compare the method described in Section 5.1.3. The constructed influence diagrams (in Section 5.1.2) are not directly applicable to be solved by a discrete ID. A discrete model requires a total order of the decision variables and ensures perfect recall of all previous observations and decisions. An adaptation to the network is therefore necessitated. All the links specifying total order and no-forgetting need to be depicted in the network. An algorithm that specifies these links is provided in Appendix B. Since a prespecified order of decision variables needs to be present, each order has to be evaluated. Only the order of the test decision nodes needs to be looked at because, by modeling in this way, the corrective maintenance decision nodes always come after the test decision nodes. How discrete IDs come up with the expected utility is already described in Section 3.3. However, since one has to consider all the different sequences of tests when solving the discrete ID, the expected utility of each possible sequence is kept. Hence, the network is initialized with the sequence that has the maximum expected utility over the set of possible sequences. Table 5.6 shows how the maximum expected utility is derived for the example displayed in Figure 5.4.

Table 5.6: Finding the best sequence and the maximum expected utility for the Discrete ID

Prespecified Decision Sequence										Expected Utility*	Best Sequence
$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$D_1$	$D_2$	$D_3$	$D_4$	$D_5$	-18.5	
$T_2$	$T_1$	$T_3$	$T_4$	$T_5$	$D_1$	$D_2$	$D_3$	$D_4$	$D_5$	-17.8	$\leftarrow$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$T_5$	$T_4$	$T_3$	$T_2$	$T_1$	$D_1$	$D_2$	$D_3$	$D_4$	$D_5$	-19.2	

\*Example values are depicted

## Chapter 6

# Numerical Investigations

This chapter begins by explaining the software and models used for the investigation. Then, the numerical examples presented in Chapter 4 and extended to influence diagrams in Chapter 5 will be used in the second section to demonstrate the results obtained. The methods used for solving these networks are the discrete ID and the proposed LIMID + Myopic one. Thereafter, the scalability of the methods is investigated. And this chapter ends with a sensitivity and performance analysis of the discrete ID and the proposed LIMID + Myopic approach. The assumption stated in Section 5.1.1 remains valid.

### 6.1 Software and Models


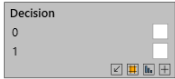
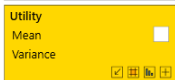
Three influence diagrams of Model 1: Single Light and Model 2: Quadruple Light were modeled and solved using the Application Programming Interface (API) of the Bayes Server Limited (2022) software. An overview of the models is depicted in Table 6.1. As seen the influence diagrams of Model 1 has two alternatives, both alternatives are used for evaluation.

Table 6.1: Overview evaluated IDs

Model		Figure	Software
Model 1: Single Light	(1a)	5.4	6.1a
	(1b)	5.5	6.1b
Model 2: Quadruple Light	(2)	5.6	6.2

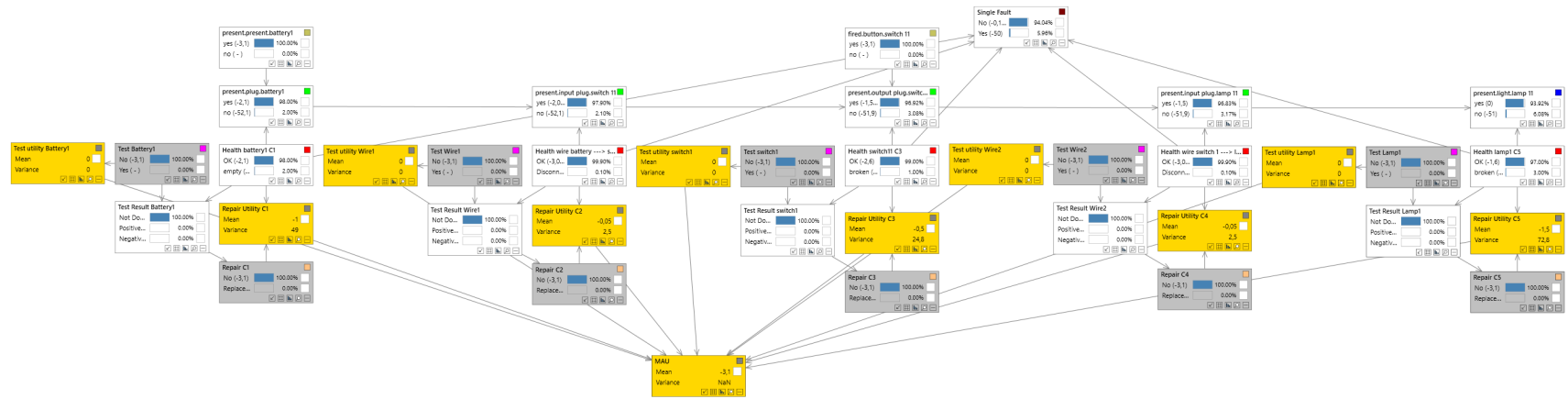
Figure 6.1a and 6.1b show the used alternatives of Model 1 and displays their modelling in the software. The influence diagram of Model 2 is shown Figure 6.2. In these figures it is shown that, not circles but white rectangles represent discrete chance variables, gray rectangles represent discrete decision variables, and not diamonds but yellow rectangles represent discrete utility functions, summarized in Table 6.2. The purpose of these figures is not to demonstrate their readability, but rather to demonstrate how large the network can be. Due to their size, modeling it by hand tends to be very error-prone, and the automated generation of influence diagrams is therefore highly appreciable, if not required.

Table 6.2: Translation table theory to software

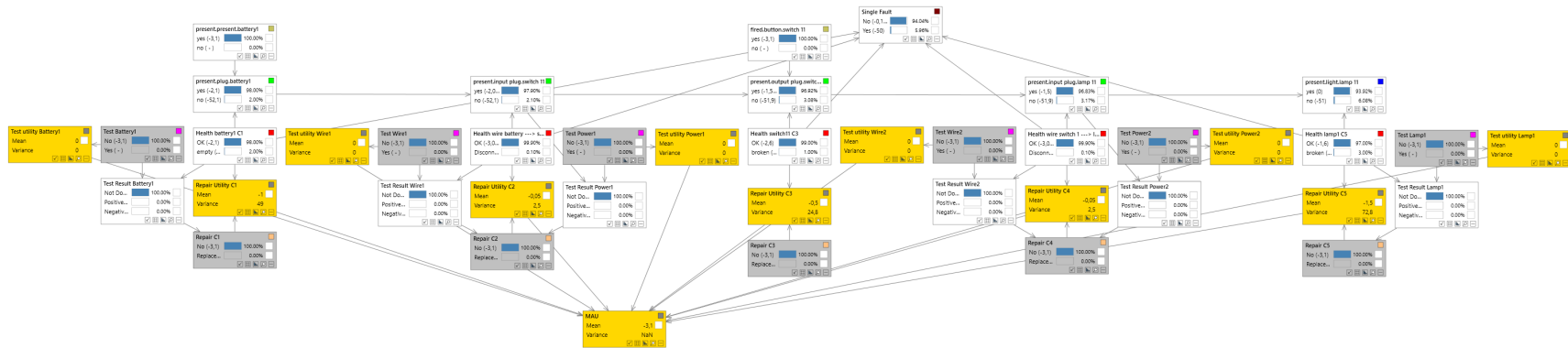
Variable	Theory	Software
Chance	○ →	
Decision	□ →	
Utility	◇ →	

Also, in the upper right-hand corner of each variable in the network in the software, the group to which this node belongs is specified. And two additional nodes are added: **MAU** and **Single Fault**, which represent the sum of all the utility nodes (Multi-Attribute Utility function (MAU)) and the single fault assumption, respectively. The **MAU** node is introduced because the software requires that for influence diagrams with multiple utility nodes, each utility node must have a common descendant that is also a utility node. And the **Single Fault** node is added to initialize the model when the single fault assumption is applied.

The discrete ID described Section 3.3 is used to compare the method described in Section 5.1.3. As mentioned in Section 5.1.4, the constructed influence diagrams (overview in Table 6.1) are not directly applicable to be solved by a discrete ID. A discrete model requires total order on the decision variables and ensures perfect recall of all previous observations and decisions. An adaptation to the network is therefore necessitated. All the links specifying total order and no-forgetting need to be depicted in the network. An algorithm that specifies these links using the API of Bayes Server Limited (2022) is provided in Appendix B. The LIMID + Myopic approach does not necessitate additional tweaking to the network. There is an additional algorithm on top that finds the next best test. This algorithm is depicted in Appendix C. The software runs on a laptop with a 1.70 GHz processor and 16 GB of random access memory. Additional algorithms are run in Python 3.9. The API runs on a Java Virtual Machine (JVM) with a heap space size of 12 GB.



(a) Model 1: Isolated tests



(b) Model 1: Additional tests

Figure 6.1: Single light influence diagrams depicted in the software

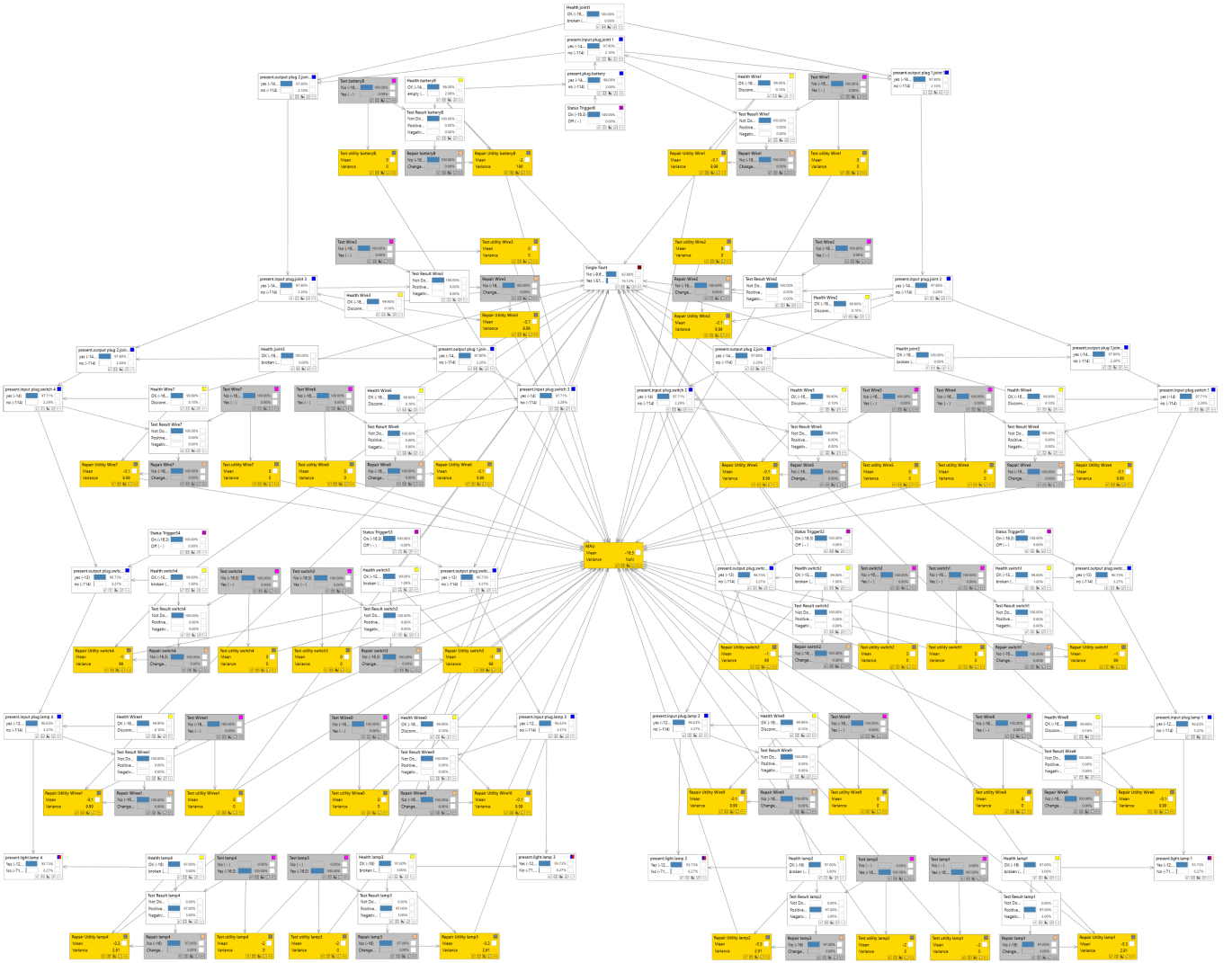


Figure 6.2: Influence diagram of the quadruple light circuit depicted in the software

## 6.2 Results

For each example, different values of the components' prior probability  $P(H)$  and cost  $C$  are considered (Table 6.3). The cost of each test ( $c_1$ ) is fixed at 2. As pointed out in Section 5.1.4, when solving the evaluated IDs with the discrete ID, one has to consider all the different sequences of tests since a prespecified order is required. Therefore, the maximum expected utility over the set of possible sequences is kept, where the set of possible sequences is the number of test decision nodes ( $n$ ) factorial,  $n!$ . The average time it takes to find the optimal sequence is also monitored.

Solving the evaluated IDs with the LIMID + Myopic approach, each possible failure scenario (e.g., battery is broken, rest is OK) needs to be resolved (perform until *End* is reached, in Figure 5.1) as Section 5.1.3 suggests. The results show a weighted expected utility because the



solution to each possible scenario is weighed against the likelihood of that scenario occurring. The weighted average utility is the sum of all possible failure scenarios, where the total number of failure scenarios equals 2 to the power of the number of components minus one,  $2^{|components|} - 1$ . Given that each component has two states and the scenario in which all components are OK is excluded because it is not relevant given a system failure. In this approach, the amount of time it on average takes to suggest the next best test is kept. The results obtained for the different evaluated IDs using the two methods are shown in Table 6.4 and 6.5.

Table 6.3: The parameters for the numerical examples

(a) Model 1: Single Light					(b) Model 2: Quadruple Light				
Components	$c_2$	$c_3$	$c_4$	$P(H_n = 0)$	Components	$c_2$	$c_3$	$c_4$	$P(H_n = 0)$
Battery	10	6	50	0.980	Battery	10	6	100	0.980
Wire	10	6	50	0.999	Wire	10	6	100	0.999
Switch	10	6	50	0.990	Switch	10	6	100	0.990
Lamp	10	6	50	0.970	Lamp	10	6	100	0.970

Table 6.4: Comparison of total expected utility and run time for the solution in the discrete ID and LIMID + Myopic methods for the single light model

Expected utility [€]					Run time [sec]		
<i>Single light model (Figure 6.1a, isolation)</i>							
Discrete ID					Discrete ID		
SFA					SFA		
Possible Sequences	=	120	;	120	<i>time</i>	=	29.46 ; 31.27
Maximum Expected Utility	=	-14.45	;	-13.47			
LIMID + Myopic					LIMID + Myopic		
SFA					SFA		
Number Scenarios	=	31	;	5	<i>time</i>	=	0.23 ; 0.34
Weighted Average Utility	=	-14.68	;	-13.47			
<i>Single light model (Figure 6.1b, additional)</i>							
Discrete ID					Discrete ID		
SFA					SFA		
Possible Sequences	=	720	;	720	<i>time</i>	=	1,008.10 ; 1,065.61
Maximum Expected Utility	=	-14.73	;	-13.76			
LIMID + Myopic					LIMID + Myopic		
SFA					SFA		
Number Scenarios	=	31	;	5	<i>time</i>	=	0.21 ; 0.31
Weighted Average Utility	=	-14.98	;	-13.76			

Table 6.5: Comparison of total expected utility and run time for the solution in the discrete ID and LIMID + Myopic methods for the quadruple light model

		Expected utility [€]		Run time [sec]	
<i>Quadruple light model (Figure 6.2)</i>					
		Discrete ID		Discrete ID	
		SFA		SFA	
Possible Sequences	=	$2.43 * 10^{18}$	;	$2.43 * 10^{18}$	<b>Memory Error</b>
Maximum Expected Utility	=				
		LIMID + Myopic		LIMID + Myopic	
		SFA		SFA	
Number Scenarios	=	1,048,575	;	20	
Weighted Average Utility	=	-16.34	;	-15.24	<i>time</i> = 4.51 ; 36.92

In each table, a distinction is made between the results obtained with the Single Fault Assumption (SFA) and the multiple-faults one. The SFA is introduced because the algorithm keeps performing tests before corrective maintenance actions if both are present in the set of potentially observable variables. By introducing the SFA, the algorithm will stop suggesting tests when a single fault is found. It is noted that the single fault scenarios are also considered in the multiple fault ones. The number of scenarios in Table 6.4, having 5 components, is therefore  $2^{|5|} - 1 = 31$  relaxing the SFA and  $|5| = 5$  assuming only a single fault. The number of possible sequences remains untouched.

Even though there are advanced inference algorithms, when the number of decision (test) nodes becomes larger, the exact (discrete ID) method will have memory issues. In the LIMID + Myopic, only those nodes with links to the decision nodes are assumed to be known at the time of making the decision. This assumption can strongly reduce the computational effort when optimizing the decisions. With increasing memory, i.e., with an increasing number of links to the decision nodes, the policy domains of the decision nodes increase, making the solution of the optimization problem intractable. On the other hand, reducing the number of information links toward the decision node leads to suboptimal solutions.

The discrete ID model outperforms the LIMID + Myopic model with a higher maximum expected utility in all scenarios in which the SFA is removed, as indicated by the values in Table 6.4, except for the values where the discrete ID cannot compute the expected utility for the network, as seen in Table 6.5. From this table, it is observed that the discrete is not scalable for networks where the total number of test nodes is large, whereas the Myopic can solve this problem within an acceptable amount of time. The average percentage of change between the max values of the discrete ID models ( $-14.45$ ;  $-14.73$ ) in Table 6.4 and the weighted average utility of the LIMID + Myopic models ( $-14.68$ ;  $-14.98$ ) is 1.59% and 1.70%, respectively. Assuming a SFA, the performance of the discrete ID and LIMID + Myopic models is equal. More details about the differences in their scalability and decision-making performance can be found in Section 6.3 and 6.4. The inequality between the models in Table 6.4 is that an additional observation to determine the health of the trigger is needed

for the *additional* model. Therefore, the expected utility is slightly lower for the latter.

### 6.3 Scalability (Computational Time)

In this section, the scalability of the two methods is tested to verify the applicability of the model construction in a scalable way. A problem with a range of components and functions in a series is tested in isolation. See Figure 6.3 for an abstract representation of the network. This network relates to the IDs of Model 1, since these networks also consist of a series of components and functions. However, the network of Model 1 consists of only 5 or 6 test decision nodes, and the example of Figure 6.3 continues until there are 20 test decision nodes. Model 2 also consists of 20 test decision nodes. However, in contrast to the configurations of Model 2, the network does not have parallel structures and consists of a series where all the probabilities and costs are equal. The number of components  $N$  and tests  $M$  is also equal. Hence, the scalability of both methods is compared based on inference times.

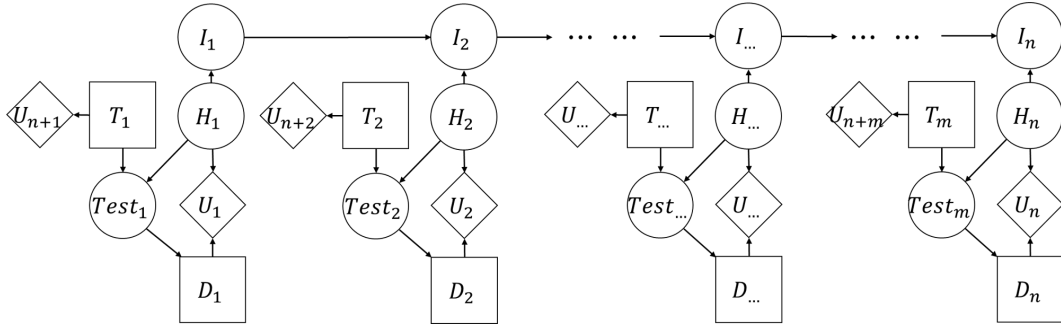


Figure 6.3: Abstracted influence diagram for scalability comparison

In describing an approach to find a solution to a problem with a final decision table (full decision sequence specified as in the discrete ID) requires a lot of computing time and is intractable in common software applications. Extending the problem to 10 available tests would involve a final decision table of over 61 billion entries and the number of possible sequences of over 3.6 million. Considering the Java virtual machine with a heap space size of 12 GB, the experiment returns an out of memory error when nine decision variables are employed. Thus, Figure 6.4 considers only the discrete ID model up to 8 decision nodes and is not further considered because of the memory limitation.

The average time to come up with the next best test is, in all cases, an advantage with the LIMID + Myopic. The LIMID + Myopic average time is considered acceptable when the SFA is removed. The average inference time when under the SFA drastically increases for the number of test nodes  $\geq 17$ . This phenomenon has already been noticed in Table 6.4 and 6.5, where the run time is longer when the SFA is used. Modeling the SFA explicitly in the network in the current way is by linking all health nodes to the **Single Fault** node, which fosters an increase in density in the network, since finding a failed component will set the other components to OK, causing nodes to no longer be regarded as independent of each other. The increase in density causes longer inference times because dense networks are computationally more expensive (Kjærulff and Madsen, 2008).

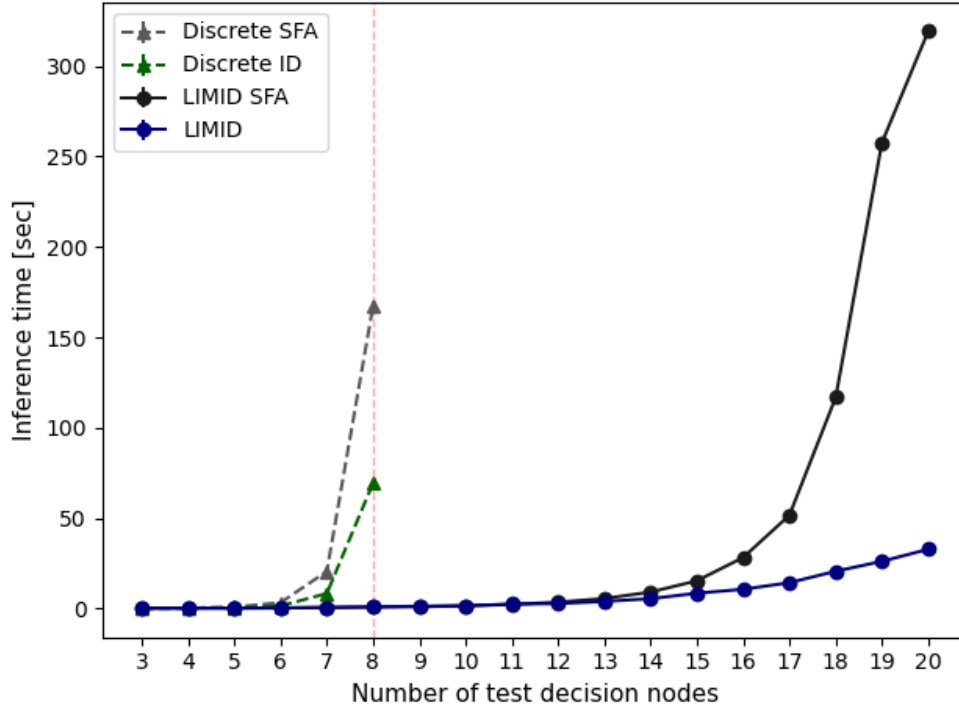


Figure 6.4: Inference time comparison of single sequence of the discrete ID and average time to come up with the next best test in the LIMID + Myopic

When diving deeper in the individual values, not observable from the figures, the main increase in inference time is found when initializing the first evidence (e.g. **Single Fault**= *yes* and error states). Where for the SFA considering 20 decision nodes (Model 2: Quadruple Light), it is on average 2500 seconds and for the relaxation of the assumption, it is 15 seconds. When the evidence is set and the next best test algorithm can be executed, the average inference time is slightly higher than when the single fault is removed.

It has also been discovered that every test adds value ( $T_n = \text{yes}$ ) in this case. This assumes that, for every test, the expected value is checked. Therefore, the average inference time is also increasing for larger values since it needs to check every test, excluding the ones already performed. Considering different network structures (as in Table 6.5), the set of suggested tests is smaller and, therefore, the average time to suggest the best next test is less.

## 6.4 Policy Evaluation

A sensitivity and performance analysis of both approaches is described in this section to demonstrate the degree of effectiveness (accuracy) of the advised diagnostic action. The examples in this analysis consist of a network in a series where the probabilities are equal, ascending, descending, and random, accompanied by different cost parameters. The example used for both analyses is the *Single light model* (Figure 5.4, *isolation*).

### 6.4.1 Sensitivity analysis

Different prior probabilities ( $P(H)$ ) are chosen for the components' failure probability. And various cost parameters ( $C$ ) are considered for each example. These are specified as follows:

Table 6.6: Sensitivity parameter for the numerical examples

(a) Probabilities $P(H_n = 0)$					
Type	$P(H_{battery})$	$P(H_{wire1})$	$P(H_{trigger})$	$P(H_{wire2})$	$P(H_{lamp})$
Equal	0.920	0.920	0.920	0.920	0.920
Ascending	0.850	0.880	0.920	0.950	0.990
Descending	0.990	0.960	0.920	0.890	0.850
Randomized	Random uniform between 0.850 - 0.990				

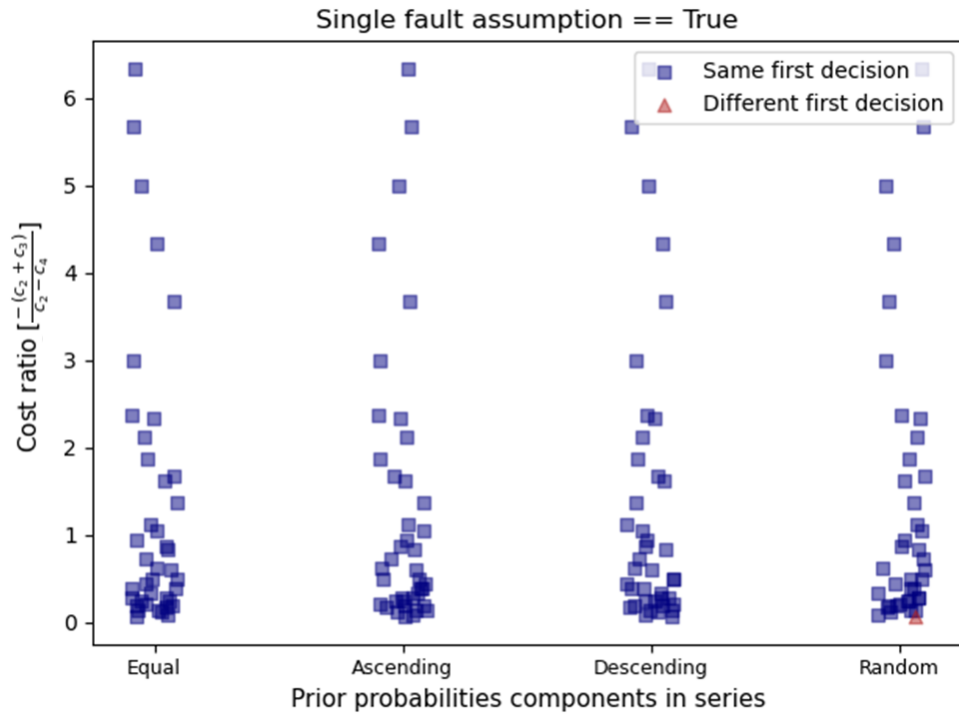
  

(b) Costs				
Name	Definition	Range	Step size	
Cost of carrying out a test	$c_1$	0	-	
Replacement cost	$c_2$	10	-	
Incorrect replacement penalty	$c_3$	5, 15, ..., 75	10	
Failure-to-repair cost	$c_4$	25, 50, ..., 400	*2	

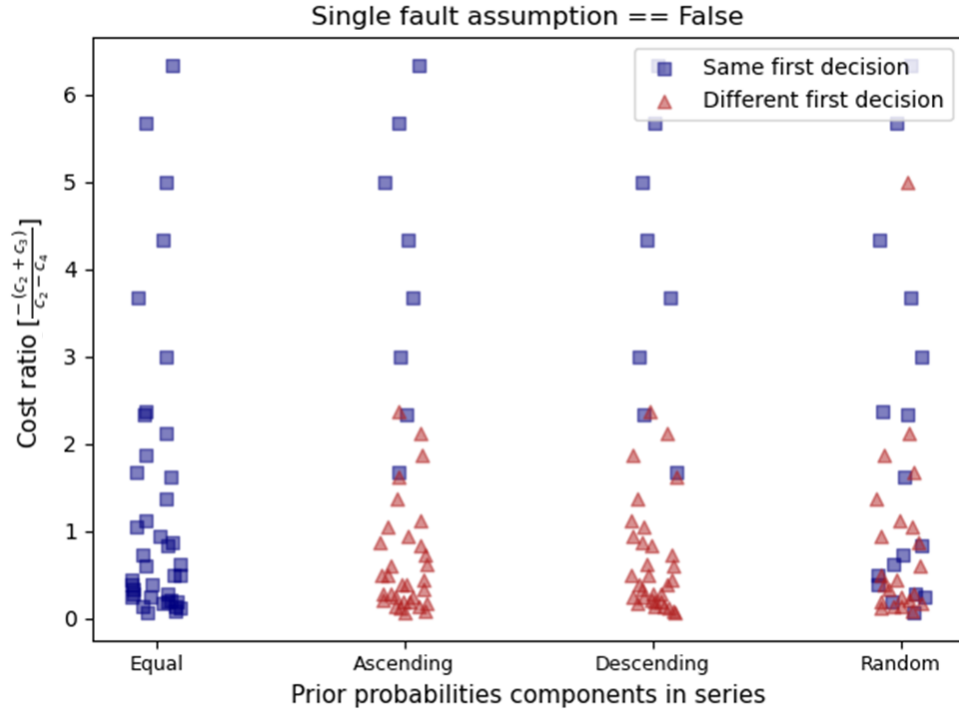
A total of 40 unique cost configurations appear and for each cost configuration, four different probability types can be applied. This increases the total of unique cost-probability configurations to 160. When the probability type is *Randomized*, different probability draws are made based on samples from a uniform distribution between 0.850 and 0.990 for each component's health. A sensitivity analysis is considering the policy for both the discrete ID and LIMID + Myopic. The graphs in Figure 6.5 display the first decision policy, examining whether the exact method represents the same first decision as the approximate one, bearing in mind the difference in ratio and prior probabilities for both with and without the SFA.

Figure 6.5a is the model that assumes a single fault. All different costs and prior configurations revolve around the same decision policy. This indicates that the LIMID + Myopic generates an optimal solution for the decision problem considering the SFA. Where the SFA is removed (Figure 6.5b) and the replacement cost and incorrect replacement penalties are larger than the failure-to-repair cost (cost ratio  $\geq 2.5$ ), the models choose an optimal first decision. The same holds when the prior probabilities are equal. However, cost ratios of  $> 1$  and equal failure probabilities of components will probably not emerge in real-life scenarios. Cost ratios of  $> 1$  suggest that the inability to fix a component means that the entire system will have to be replaced. This will not be possible for systems considered in this report.

For the models where there is a greater difference in prior probabilities, the LIMID + Myopic model (removing the SFA) tends to be less accurate. These models more often choose a suboptimal path. This behavior comes from the fact that sometimes a single test does not yield anything by itself (Myopic), whereas its outcome may be crucial for selecting a second, very informative test. This behavior is captured when the full sequence is specified since it includes these informational links.



(a) Single fault assumption



(b) Single fault assumption removed

Figure 6.5: Policy change for different costs and prior probabilities

### 6.4.2 Comparison

The final analysis is done using each possible configuration of the models in Subsection 6.4.1. The previous subsection compares the suggested first test of the LIMID + Myopic to any other suggested test in the network. Here, proper analysis is done for the models where the single fault assumption is removed (Figure 6.5b). In these models, many of the first decisions deviate from the optimal decision, especially in the interested area (cost ratio  $\leq 1$  and different probabilities). It is thus determined how far they deviate from the ideal when considering a full solution (not only first decision) because this method would choose a different order but can still come up with an efficient solution. Figure 6.6 and 6.7 depict the normalized difference between the expected utility of the best sequence of the discrete ID ( $DID_{EU}$ ) and the expected utility of the LIMID + Myopic ( $LIMID_{EU}$ ) on the y-axis. The range of the distance is determined by the expected utility when all components are tested ( $ALL_{EU}$ ). The values that are plotted on the y-axis are then specified as follows:

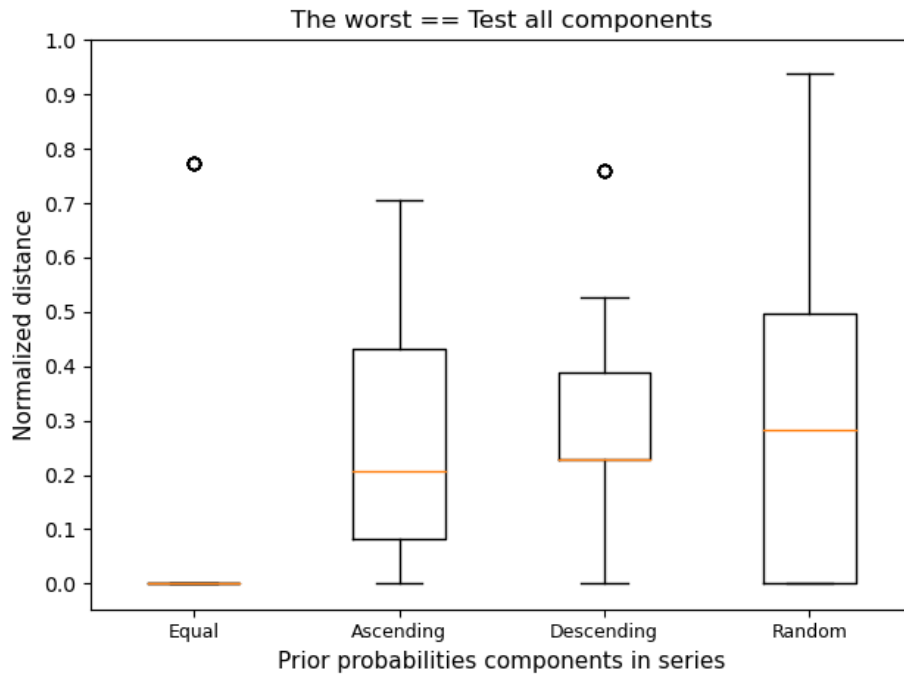
$$\text{Normalized distance} = \left| \frac{LIMID_{EU} - ALL_{EU}}{DID_{EU} - ALL_{EU}} - 1 \right|$$

where 0 indicates the same expected utility (discrete ID and LIMID + Myopic). Each box in Figure 6.6 and 6.7 extends from the data's lower to upper quartiles, with an orange line at the median. The whiskers extend from the box to show the range of the data. Flier points (circles outside each box) are those past the end of the whiskers. Based on the 160 unique cost-probability configurations found in Section 6.4.1, the number of data points in each of the four columns of Figure 6.6 is as follows: 40; 40; 40; and 40, where each column has equal priors but different cost ratios. And the number of data points in each of Figure 6.7's five columns is: 28; 36; 36; 36; and 24, where each column has similar cost ratios but many priors can be present.

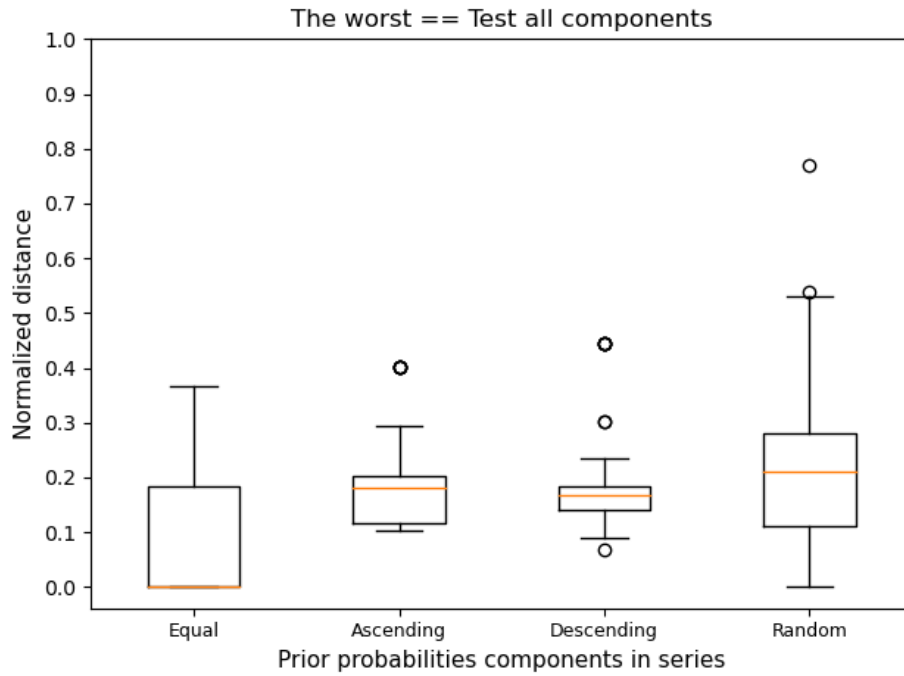
It was found that in discrete ID, the number of tests suggested is more than one. The expected utility is therefore reduced and dependent on the outcomes of the initial set of first tests. Hence, it, for example, suggests testing *wire*<sub>1</sub>, *wire*<sub>2</sub>, and *trigger*, whereas the LIMID + Myopic is forced to only perform one test. Even if the probability of a defect in a component has the greatest likelihood. The difference is that, since we assume to relax the single fault assumption, if one component is likely to be faulty, the other components are likely to be faulty with its original prior probability (as seen in Appendix A). It proposes to still reduce the uncertainty in these nodes due to the higher cost, in contrast to the low cost for testing. The discrete ID suggests, therefore, that you simply test multiple components, whereas the LIMID + Myopic only accepts one test at a time. The ratio between replacement cost, incorrect replacement penalty and the failure to fix cost is of fewer importance considering these assumptions.

The difference between the test cost and failure to fix cost is of much greater importance. Hence, this is when the difficulty of the corrective maintenance action comes to mind. When the cost of failure to fix becomes larger, accompanied by the assumption to always perform a test before a corrective maintenance action if both are suggested, the method will keep testing until no more tests are suggested or left before performing such an action. This behaviour can be seen when comparing the boxplots of Figure 6.7a with 6.7b. Higher test cost ( $c_1$ ) leads on average to an decrease in distance between the discrete ID and LIMID + Myopic since it

will decrease the number of tests it performs.



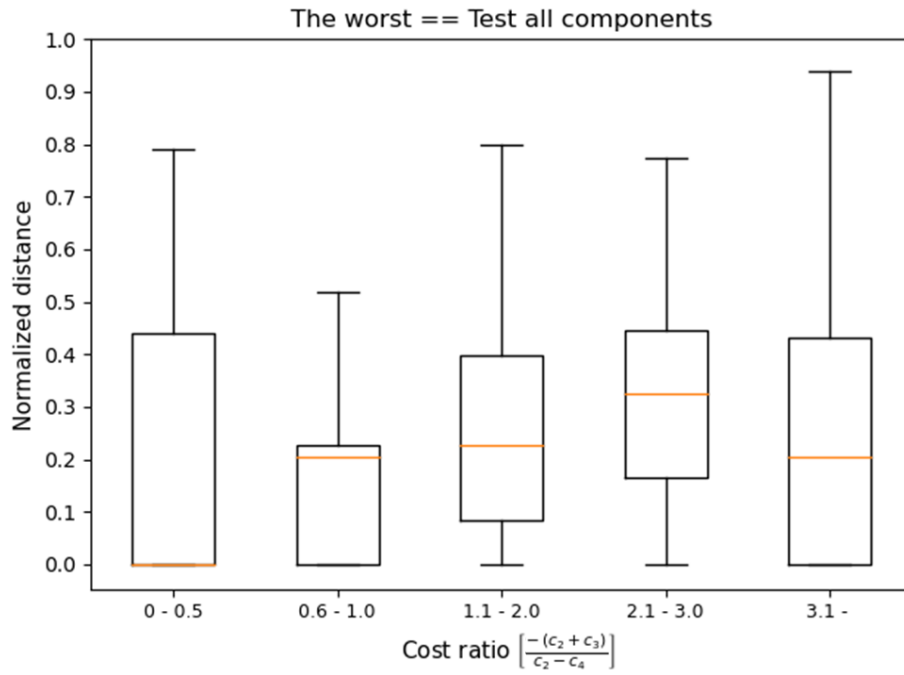
(a)  $c_1 = 2$



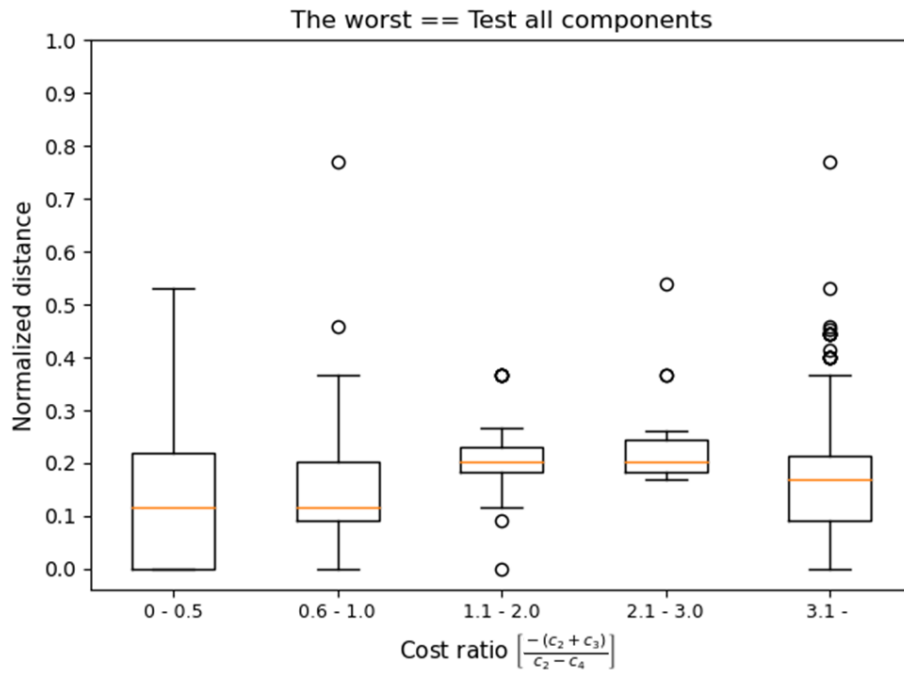
(b)  $c_1 = 5$

Figure 6.6: Normalized distance between the discrete ID and LIMID + Myopic in the range of the discrete ID and testing all components per probability specification





(a)  $c_1 = 2$



(b)  $c_1 = 5$

Figure 6.7: Normalized distance between the discrete ID and LIMID + Myopic in the range of the discrete ID and testing all components per ratio binned

## 6.5 Chapter Conclusion

The proposed LIMID + Myopic solution approach and its potential were evaluated on different criteria. The evaluation procedure covered both performance and scalability. The performance under the Single Fault Assumption (SFA) is similar to the exact solution as found using the discrete ID. However, the discrete ID is not scalable both with and without the SFA, whereas the LIMID + Myopic is only scalable by removing the SFA. When in the LIMID + Myopic, the SFA is considered, their inference times increase drastically since the domain that is evaluated for the inference is larger and requires computationally more storage. The decision policy without the SFA more often chooses a suboptimal path. However, their normalized distance to the exact solution is relatively close and particularly in the region we are interested where the cost ratio is between 0 – 0.5. In conclusion, the solution approach provided by the LIMID + Myopic method appeared to be a good approximation and provides sufficient insight into the scalability and performance of a decision support method based on an influence diagram for assisting service engineers in their diagnostics tasks. Nevertheless, to emphasize that in the end, corrective maintenance actions are not included in the set of next best tests when both tests and corrective actions are suggested. Therefore, the proposed method's applicability for real-life applications will require some more adjustments to the network before actual usage.

## Chapter 7

# Conclusion and Future Work

The goal of this thesis was to develop a decision support method for assisting service engineers in their diagnostics tasks by advising an improved diagnostic action. This concluding chapter starts off with a summary of the main contributions of this thesis. This section is followed by several limitations of the existing contribution. The final section introduces some points for future work within this context.

### 7.1 Conclusion

In this work, an introduction to the extension of Bayesian networks into influence diagrams to model decision processes is given. Influence diagrams are a compact yet powerful tool that allows both for the elicitation of complex processes by non-computer experts and their sound analysis using the known paradigm of Bayesian inference.

The first research sub-question; *How can an influence diagram be applied in this domain?*, has received little attention by literature. Hence, this report has proposed a method for solving large and complex decision-making problems to come up with a decision sequence. The method uses a novel implementation of a Limited Memory Influence diagram accompanied by an approximate Myopic approach to suggest the next best test. Such implementation has not been seen before in the literature. The solution framework is constructed to be fully generic and therefore applicable to any problem in this domain. This framework is also compatible with the software used by ESI (TNO) and can be modified and operated through Python. The representation of a decision process in a Limited Memory Influence diagram explicitly distinguishes the information known to a decision-maker at the time of the decision from that which is not. This allows us to memory efficiently model diagnostic information in these influence diagrams which yields a computationally tractable solution.

The second sub-question; *How to build such a model in a scalable way?*, is also investigated. The building blocks for modeling influence diagrams are derived from medical applications. Because the current approach uses a Bayesian network for diagnostics, a transformation from the existing Bayesian network to an influence diagram is therefore proposed. This transformation only depends on a mapping between nodes in the network and the set of possible tests. The framework is therefore independent of the Bayesian network structure and is also applicable to other sequencing problems. The question on the use of the structural or functional

model is not applicable since the modeling is independent of the network, as long as a good test-node relationship is described. In the ideal case, these influence diagrams are directly generated from existing technical information without the need to first have a Bayesian network. Although the process begins with identifying hard-system down scenarios, all design decisions are made with the goal of extending performance and predictability. Because influence diagrams can also be used in a dynamic environment for state-transition models.

The main research question; *How to develop a decision support method based on an influence diagram for assisting service engineers in their diagnostics task by advising an efficient diagnostic action?*, is partly answered. The diagnostic efficiency is the final criteria. The solution of the proposed method is equal to the exact solution of the decision problem under the single fault assumption. Relaxing this assumption does lead to a performance decrease but is considered comparable since its normalized distance to the exact method is, in all cases, relatively close. The suggested method outperforms the exact solution for larger networks because of the computational restrictions arising in the exact method. For larger networks, the computation time to generate the next best test is very appealing when the single fault assumption is relaxed since it suggests the next test in less than 40 seconds. The spike in computation time considering the single fault assumption is mainly dependent on the time it takes to initialize and execute the model with the initial evidence. After the initialization, the computation times are very similar. Testing different topologies reinforce our belief that the method is suitable for different network configurations.

## 7.2 Limitation

As seen in the results, the Myopic approach does not guarantee an optimal sequence. Sometimes a single test does not yield anything by itself, whereas its outcome may be crucial for selecting a second, very informative test. The expected value of a pair of tests may be greater than that of a single test. In that case, the Myopic approach is misleading. This phenomenon is mentioned in Subsection 6.4.1. Where the discrete influence diagram solution approach produces different first decisions as seen in Figure 6.5b.

The model's size is not shrunk to the minimum. As mentioned in Section 5.1.2, the size can be further reduced by limiting the number of possible states of a node. For the test result nodes, the number of states can be reduced from 3 to 2, excluding the *not\_done* state, but this was not done. Different values in Section 6.3 could therefore be obtained since the number of entries into a (final) decision node is reduced.

In the current approach, the assumption holds that one should always perform a test before a corrective maintenance action if both are suggested in the case where the single fault assumption is relaxed. What if a corrective maintenance action can also be considered a test? Since the replacement of a part can be seen as an intervening action, this is currently modelled as the final node (action) if there are no more tests to perform. It may be the case that just replacing the component is more cost-efficient than testing another component to further increase the reliability of your diagnosis. This is currently not handled.

Finally, there is a lack of concise evaluation criteria that use the expected utility of the

worst possible sequence as the lower benchmark rather than the expected utility when all components are tested. This is because it is self-evident that when the test costs increase, the Myopic moves further and further away from simply performing all the tests. An evaluation that takes the worst possible sequence was not feasible within the time frame of this thesis. It requires some immense simulation time since a proper analysis of the choices should consist of an analysis of all possible sequences of tests (including the empty sequence). Therefore, the choice of taking the expected utility when all components are tested is a proper initial benchmark.

### 7.3 Further Work

Despite the fact that influence diagrams (IDs) have been around a long time, there are still directions for future work. In IDs where the full decision sequence is specified, the outcomes of all nodes must be conditioned on all the outcomes of their parents. This implies that when generating the equivalent tree, all decision scenarios must be explored, including redundant paths. Such a tree can be beneficial to help with the explainability of the diagnostic actions. However, as the problem is dealt with an increasing scale, the usefulness of a discrete influence diagram diminishes. A point is reached where the complexity of the interactions creates data overload (memory error). Frequently, the constraint is merely the size of the diagram and the inability to deal with the entire domain of influences in one model.

A line for further research could be to explore approximate or faster and more memory-efficient inference algorithms when the full decision sequence is specified. When such an algorithm can be employed, one can explore the possibility of not having to go through all the possible test sequences. A smarter algorithm that reduces the number of possible sequences can be introduced. Because the time required to find the correct sequence is  $\mathcal{O}(m!)$ , where  $m$  denotes the number of tests in the network.

Another line could be to explore what happens to the decision policy if we remove the test cost independence assumption and use another alternative of Model 1: Single Light that does not test components in isolation but also performs tests on functions. The discrete ID also takes future decisions into account in their decision-making and the LIMID + Myopic only considers a one-time step ahead. Therefore, by considering test cost dependence, the discrete ID should therefore have more advantages in order to perhaps perform a more expensive test first, which then entails a set of cheaper but very efficient tests.

Some important modeling issues can be tackled in further research. Firstly, an important distinction should be made between probing or intervening actions. Probing is a passive observation of the state of a variable, and intervention is an active action that forces a variable to be in a certain state (Kjærulff and Madsen, 2008). Intervening actions change the system. Intervening actions are not discussed in this report but can be modeled as we merely do with isolated tests. The replacement of a part can also be seen as an intervening action. However, because of the assumption that one should always perform a test before a corrective maintenance action if both are suggested, one does not consider corrective maintenance actions in the set of possible next actions. A research opportunity is, therefore, to include corrective maintenance actions in the set of possible next actions.

And the final point to improve the model is that the current representation does not include the cost of testing the entire system again after some maintenance. It immediately checks if the system is operational after some corrective maintenance action is performed. Since testing/checking the entire system could take a substantial amount of time for typical high-tech machinery, and if you only replace a small, simple component, is it then necessary to check the entire system, or is it more beneficial to check other components first.

# Bibliography

- Alzghoul, A., Backe, B., Löfstrand, M., Byström, A., and Liljedahl, B. (2014). Comparing a knowledge-based and a data-driven method in querying data streams for system fault detection: A hydraulic drive system application. *Computers in Industry*, 65.
- Atoui, M. A. and Cohen, A. (2021). Coupling data-driven and model-based methods to improve fault diagnosis. *Computers in Industry*, 128:103401.
- Bayes Server Limited (2022). Computer software. <https://www.bayesserver.com/>. [Online; accessed 2022-01-03].
- Bielza, C., Fernández del Pozo, J., and Lucas, P. J. (2008). Explaining clinical decisions by extracting regularity patterns. *Decision Support Systems*, 44:397–408.
- Bielza, C., Gomez-Olmedo, M., and Shenoy, P. (2011). A review of representation issues and modeling challenges with influence diagrams. *Omega*, 39:227–241.
- Cai, B., Huang, L., and Xie, M. (2017). Bayesian networks in fault diagnosis. *IEEE Transactions on Industrial Informatics*, PP:1–1.
- Clemen, R. (1996). *Making Hard Decisions: An Introduction to Decision Analysis*.
- Cobb, B. R. (2021). Statistical process control for the number of defectives with limited memory. *Decision Analysis*, 18(3):203–217.
- Cooper, H., Iyengar, G., and Lin, C.-Y. (2019). *Deep Influence Diagrams: An Interpretable and Robust Decision Support System*, pages 450–462.
- Diez, F., Yebra, M., Bermejo, I., Palacios-Alonso, M., Arias, M., Luque, M., and Pérez Martín, J. (2017). Markov influence diagrams: A graphical tool for cost-effectiveness analysis. *Medical Decision Making*, 37:183–195.
- ESI (TNO) (2022). Technical memo rd. Unpublished confidential document. [Accessed from; Van Gerwen, Emile. Cited with permission].
- Fernández del Pozo, J., Bielza, C., and Gómez-Olmedo, M. (2005). A list-based compact representation for large decision tables management. *European Journal of Operational Research*, 160:638–662.
- Gómez, M. (2004). Real-world applications of influence diagrams. In *Advances in Bayesian Networks*, pages 161–180. Springer Berlin Heidelberg.

- Hovgaard, M. and Brincker, R. (2016). Limited memory influence diagrams for structural damage detection decision-making. *Journal of Civil Structural Health Monitoring*, 6:205–215.
- Isermann, R. (2006). Fault diagnosis systems an introduction from fault detection to fault tolerance. *SERBIULA (sistema Librum 2.0)*.
- Jensen, F. (2001). *Bayesian Network and Decision Graphs*.
- Jensen, F., Lauritzen, S., and Olesen, K. (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 5.
- Jensen, F., Nielsen, T., and Shenoy, P. (2006). Sequential influence diagrams: A unified asymmetry framework. *International Journal of Approximate Reasoning*, 42:101–118.
- Jolandie Konig (2021). Knowing the difference. <https://www.eaglecmms.com/knowing-the-difference-between-predictive-preventive-and-corrective-maintenance/>. [Online; accessed 2022-04-08].
- Khaled, A., Hansen, E., and Yuan, C. (2013). Solving limited-memory influence diagrams using branch-and-bound search. *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2012*.
- Kjærulff, U. and Madsen, A. (2008). *Bayesian Networks and Influence Diagrams*.
- Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*.
- Lacave, C., Luque, M., and Diez, F. (2007). Explanation of bayesian networks and influence diagrams in elvira. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 37:952–65.
- Lauritzen, S. and Nilsson, D. (2001). Representing and solving decision problems with limited information. *Management Science*, 47.
- Luque, M., Diez, F., and Carlos, D. (2016). Optimal sequence of tests for the mediastinal staging of non-small cell lung cancer. *BMC Medical Informatics and Decision Making*, 16.
- Mauá, D., Campos, C., and Zaffalon, M. (2011). Solving limited memory influence diagrams. *Journal of Artificial Intelligence Research*, 44.
- Meltzer, D. (2001). Addressing uncertainty in medical cost-effectiveness analysis: Implications of expected utility maximization for methods to perform sensitivity analysis and the use of cost-effectiveness analysis to set priorities for medical research. *Journal of Health Economics*, 20(1):109–129.
- Mirnaghi, M. S. and Haghighat, F. (2020). Fault detection and diagnosis of large-scale hvac systems in buildings using data-driven methods: A comprehensive review. *Energy and Buildings*, 229:110492.
- Mohd Amiruddin, A., Zabiri, H., Taqvi, S. A. A., and Tufa, L. D. (2020). Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems. *Neural Computing and Applications*, 32.



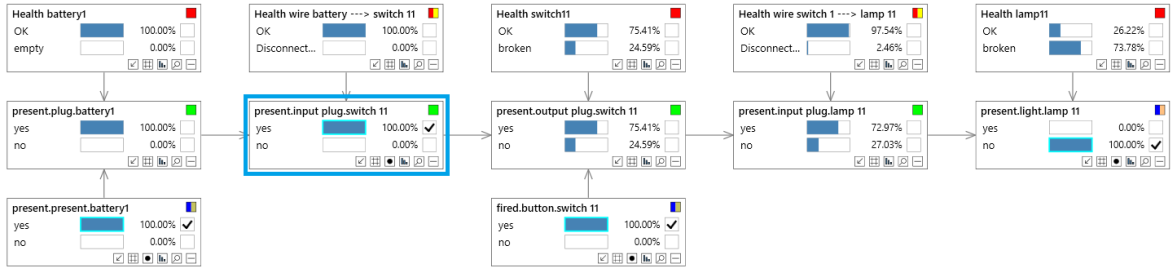
- Namburu, S., Wilcutts, M., Chigusa, S., Qiao, L., Choi, K., and Pattipati, K. (2006). Systematic data-driven approach to real-time fault detection and diagnosis in automotive engines. pages 59 – 65.
- Nielsen, J. and Sørensen, J. (2010). Planning of o&m for offshore wind turbines using bayesian graphical models. *Reliability, Risk and Safety: Back to the Future*, pages 1081–1088.
- Pearl, J. (2000). Causality: Models, reasoning, and inference, second edition. *Causality*, 29.
- Pearl, J. (2005). Influence diagrams-historical and personal perspectives. *Decision Analysis*, 2.
- Shachter, R. D. (1986). Evaluating influence diagrams. *Oper. Res.*, 34:871–882.
- Thierno M. L., D., Henry, S., Ouzrout, Y., and Bouras, A. (2018). Data-based fault diagnosis model using a bayesian causal analysis framework. *International Journal of Information Technology & Decision Making*, 17.
- Tidriri, K., Chatti, N., Verron, S., and Tiplica, T. (2016). Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges. *Annual Reviews in Control*, 42.
- Venkatasubramanian, V., Rengaswamy, R., Kavuri, S., and Yin, K. (2003). A review of process fault detection and diagnosis part i: Quantitative model-based methods. *Computers & Chemical Engineering*, 27:293–346.
- Zhang, N. and Poole, D. (1994). A simple approach to bayesian network computations. *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pages 171–178.
- Zhao, Y., Li, T., Zhang, X., and Zhang, C. (2019). Artificial intelligence-based fault detection and diagnosis methods for building energy systems: Advantages, challenges and the future. *Renewable and Sustainable Energy Reviews*, 109:85–101.

# Appendices

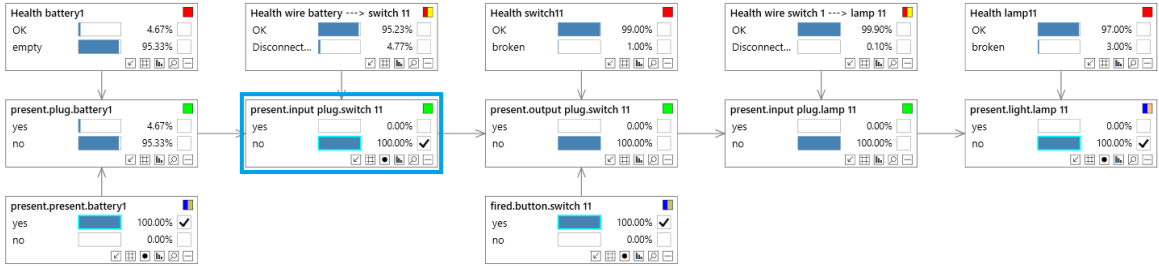
## A Model 1: Single Light visualization in software (BayesServer)

### Function test

The model utilized in this example is the Bayesian network represented in Figure 4.2a. The initial evidence as mentioned in Section 4.1, that the trigger is on and there is a battery present, is already modeled in the Figures A.1 and A.2. After deciding to execute a test to gather further information about one of the network's states, the decision-maker decides to measure the voltage at the output of  $I_{wire1}$  or  $I_{wire2}$  (Figures A.1 and A.2).



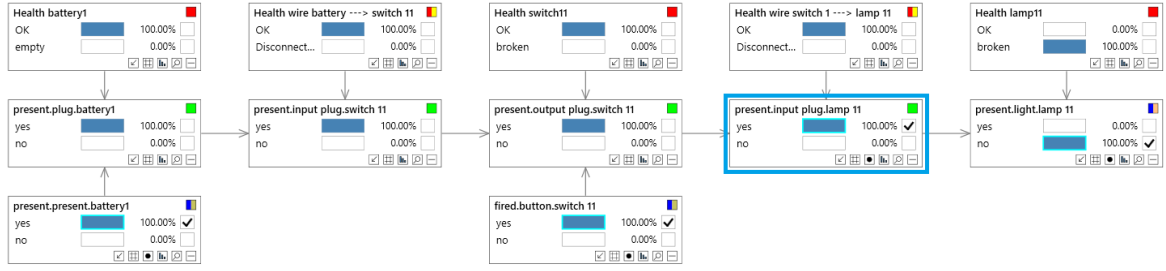
(a) Measuring voltage at bold circled node



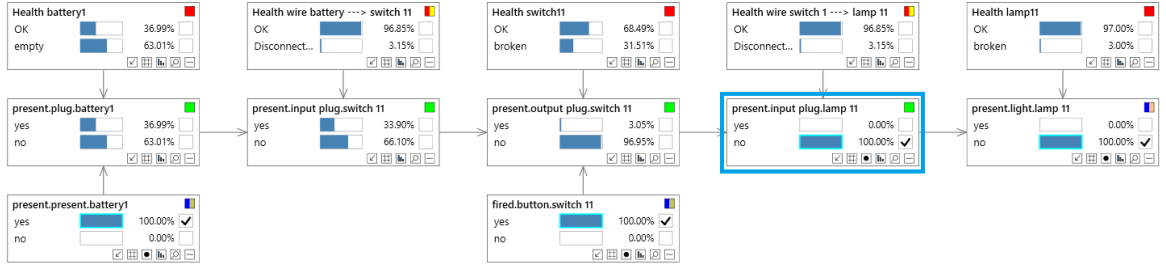
(b) Not measuring voltage at bold circled node

Figure A.1: Posterior network after observation on  $I_{wire1}$

The presence of electricity at a wire end indicates that the system is operational at all preceding nodes of that wire end. As can be seen in Figures A.1a and A.2a, all of the previous nodes' posterior health probabilities (state = *OK*) are equal to 100 percent. The latter observation,  $I_{wire2} = \text{yes}$ , identifies the source of the problem quickly, specifically that the light is broken ( $P(H_{light} = \text{broken}) = 100\%$ ).



(a) Measuring voltage at bold circled node



(b) Not measuring voltage at bold circled node

Figure A.2: Posterior network after observation on  $I_{wire2}$ 

While measuring no voltage at  $I_{wire1}$  (Figure A.1b), it implies a defect in  $H_{bat}$  and/or  $H_{wire1}$ , as well as a possibility (prior probability) that  $H_{trig}$ ,  $H_{wire2}$ , and  $H_{lamp}$  are also broken. This is due to the fact that there is a chance that several defective components will arise. The change may be subtle at first, but it is present. The measurement of indicating no voltage at  $I_{wire1}$  reduces the change of  $P(H_{light} = broken)$  and increases the likelihood of the others. However, the prior probability of being broken persists.

## B Link specification algorithms

Algorithm to generate the given sequence in a network

---

**Algorithm 1:** Generate the specified sequence in the network

---

**Function** GenerateSequenceNetwork( $B, N, S, T, R$ ):

**Data:**  $B$ : BayesServer represents the Java modules,  $N$ : the network representing the influence diagram,  $S$ : one possible sequence of decision nodes accompanied by corresponding test result nodes,  $T$ : set of test decision nodes, and  $R$ : set of diagnostic decision nodes

**Result:** A fully specified network structure with correct distributions

```
for  $s \in S$  do
  for  $node\_from \in s$  do
     $add\_link\_to\_test = \text{list}(\text{of all subsequent decision nodes})$  ;
    if  $add\_link\_to\_test$  is not empty then
      for  $node\_to \in add\_link\_to\_test$  do
        | Add link  $node\_from$  to  $node\_to$  ;
      end
      for  $r \in R$  do
        | if link between  $r$  and  $node\_from$  not exists then
        | | Add link  $node\_from$  to  $r$  ;
      end
    end
  end
end

for  $node \in T + R$  do
  | normalize distribution for  $node$  ;
end

Return  $N$ 

End Function
```

---

## C Next best test algorithm

Additional algorithm (Myopic) on top of the LIMID solution that finds the next best test

---

**Algorithm 2:** Myopic test selection given a network configuration

---

**Function** NextBestTest( $B, N, TR, EU$ ):

**Data:**  $B$ : BayesServer represents the Java modules,  $N$ : the network representing the influence diagram,  $TR$ : set of test decision nodes at 100% accompanied by their test result node, and  $EU$ : current expected utility of the network

**Result:** Returns the node of the next best test to perform

$expected\_values \leftarrow$  (empty list) ;

**for**  $test\_result\_node \in TR$  **do**

$ev\_value \leftarrow$  (empty list) ;

$prob\_result =$  probability table of current test result ;

**if** 1.0 **not in**  $prob\_result$  **then**

**for**  $state \in test\_result\_node$  **do**

            set evidence of state to 100% ;

            retrieve the expected utility of the network given that evidence ;

            multiply the found expected utility with probability of the test result ;

            append the value to the  $ev\_values$  list ;

            remove evidence of state ;

**end**

**end**

    append sum of  $ev\_value$  to  $expected\_values$  ;

**end**

$best\_index = \arg \max(expected\_values)$  ;

**if**  $EU > \max expected\_values$  **then**

**Return** (empty list)

**else**

**Return**  $TR[best\_index]$

**end**

**End Function**

---