A Branch-and-Cut Approach to Solve the Fault Diagnosis Problem with Lazy Spread and Imperfect System Information

Kaan Pekel\textsuperscript{a}, Yılmazcan Özyurt\textsuperscript{b}, Barış Yıldız\textsuperscript{a,*}

\textsuperscript{a}Department of Industrial Engineering, Koç University, Istanbul, Turkey
\textsuperscript{b}Department of Computer Science, ETH Zürich, Switzerland

Abstract

This paper presents a new approach to solving the Fault Diagnosis Problem with Lazy Spread (FDPL) that arises in many fault-tolerant real-world systems with few opportunities for maintenance during their operations and significant failure interactions between the subsystems/components. As opposed to cascading faults that spread to most of the system almost instantaneously, FDPL considers fault resistant systems where the spread of failures is relatively slow (lazy), i.e., only a small fraction of the components are faulty at the time of inspection, and accurate diagnosis of the faulty components is of critical importance to restore system performance and stop further damage. Introducing an extra level of difficulty, FDPL needs to be solved under imperfect (missing and wrong) system information in most real-world settings. To address this challenging prediction problem, we use graph theory concepts to develop an integer programming formulation and devise an efficient branch-and-cut algorithm for its solution. Extensive numerical experiments on realistic problem instances attest to the superior performance of our approach, in terms of both computational efficiency and prediction accuracy, compared to the state-of-the-art in the literature.

Keywords: Multiple failure diagnoses, spreading failures, combinatorial optimization, set covering, integer programming, branch-and-cut

1. Introduction

Most of the technologies that make up our modern lives are large and complex systems made of several components that work interdependently. As a matter of course, these components can fail for any reason at any time. When failures occur, some indications (or symptoms) are observed as a result. A timely analysis of these symptoms to correctly detect failed component(s) is of critical importance to be able to restore the system performance to normal operational conditions or isolate the failures to limit the negative impacts. This gives rise to the diagnosis problem, which is receiving increasing attention both in the application and research domains due to the apparent practical motivation and interesting theoretical properties of the problem (Ding et al., 2011). When considering fault-tolerant systems (equipped with several backup mechanisms or redundant components) with little or no opportunity for maintenance during their operations (e.g., aircraft, spacecraft, chemical reactors, power grids), the simplifying assumption: at most a single fault to occur in the system between consecutive maintenance episodes is not realistic (Shakeri

\*Corresponding author

\textit{Email address: byildiz@ku.edu.tr} (Barış Yıldız)

et al., 2000). Moreover, in complex real-world systems, a failure of a component rarely stays isolated and is very likely to induce failures in other (related) components as well, giving rise to the failure-spread phenomenon. On the one extreme are cascading failures that spread rapidly and damage most of the components in a system or cause severe capacity loss before any corrective action can be taken, e.g., blackouts in large power grids caused by cascading failures in transformers (Crucitti et al., 2004; Dueñas-Osorio and Vemuru, 2009). In most of such cases, the failed components are already known and the main goal (for system management) is to determine the most efficient way to prevent cascading failures before they occur or develop an efficient recovery plan to restore system performance as quickly as possible when they occur despite all the preventive measures (Nedic et al., 2006; Ash and Newth, 2007; Wang and Chen, 2008).

Not all failures, in all systems, spread so quickly. In many fault-resistant systems, the rate of propagation is much slower and only a small fraction of the system components are faulty at the time of inspection (Tu et al., 2003). Clearly, in such cases, the accurate diagnosis of the failed components (among many non-faulty ones) is a critical and challenging task for the system management to restore system performance and stop further damage. The spread of chronic illnesses in biological systems (e.g., diabetes causing kidney failure, vessel damage, heart problems, or cataracts) and malfunctions in the mechanical systems speed up the wear and tear in the interacting systems (e.g., a clogged radiator fin causing damage in plastic components in an engine) can be considered as examples for such “lazy” spread of failures. These types of failures give rise to Fault Diagnosis Problem with Lazy Spread (FDPL), which is the main focus of this study. Indicating the critical importance of the spread of failures in complex systems with high reliability and redundancy measures, Rutledge and Mosley (Rutledge and Mosleh, 1995) show that spatial or functional interactions account for more than 15% of the multiple sub-system failures in Space Shuttle Orbiter, according to the flight anomaly data they study. The crucial impact of the correlations between the failure modes of different components on the system reliability has been the focus of many studies, considering various real-world mechanical and electronic systems (Park et al., 2015; Cheng and Du, 2016; Gu et al., 2019).

There are three primary sources of difficulties in developing efficient algorithms to solve FDPL. First, there is not a unique mapping between possible faults and observed symptoms. It is common for different faults to share several symptoms in many real-world systems. As a result, it is possible to provide many alternative explanations (combinations of component failures) to account for a given symptom set, which significantly increases the computational complexity because the number of combinations to consider grows exponentially with the number of faults (Vedam and Venkatasubramanian, 1997). Second, accounting for the failure interactions requires considering several spread paths to provide an explanation for observed symptoms. As the number of possible spread paths can grow exponentially with the number of system components and failure interactions between them, FDPL instances with a high number of components and failure interactions emerge as a challenging combinatorial problem to solve. Furthermore, the information about the true system state is not always available. For various reasons, e.g., due to possible failures in the sensors, only a subset of the (known) failure symptoms may surface (or get successfully detected). On top of that, false alarms in the sensors may introduce unrelated symptoms,
further complicating the diagnosis problem.

Despite the practical urgency, there are not enough studies in the literature that focus on FDPL, primarily due to the computational challenges mentioned above. With an aim to address this gap in the literature, the main goal of this paper is to develop an efficient methodology that can provide the correct explanations (accurately detect the set of failed components) for a given set of symptoms, even when the state information of the system is not perfect (missing and wrong). For that purpose, we introduce a novel approach that uses graph theory concepts to model FDPL with an Integer programming (IP) formulation and suggest an efficient branch-and-cut algorithm (BC) to solve it. In particular, the contributions of this study can be summarized as follows.

- We suggest a new approach to model and solve FDPL with imperfect system information, which considers the inter-dependency relations between the system components and can accurately detect where the failure chain starts (the root cause) and how it spreads (spread path).

- We conduct extensive numerical experiments that are generated to represent a wide range of real-world systems. Our results show that:
  - In terms of computational efficiency and prediction accuracy, BC achieves a superior performance against the state-of-the-art in the literature, especially when there are significant failure interactions between the system components, as is the case with many real-world applications.
  - BC can provide high accuracy explanations in the case of incorrect system information. For the problem instances, we consider in our numerical experiments: i) Even when only a small proportion (i.e., less than 35%) of the symptoms show up (successfully detected), BC can still provide accurate predictions with more than 90% recall and precision scores, for most of the cases. ii) When a significant portion (i.e., more than 10 percent) of the known symptoms are false positives, BC can still identify the relevant symptoms and provide accurate predictions with more than 85% recall and precision scores.

The remainder of the paper is organized as follows. In Section 2, we review the related literature by focusing on multiple fault diagnosis methodologies with applications in various fields such as chemical plants, electric circuits, telecommunication networks and biologic systems. In Section 3, we provide the notation we use to describe our methodology, present a formal problem definition, introduce the mathematical model and the solution methodology we develop to solve it. In Section 4, we present and interpret the results of our computational studies. Finally, in Section 5, we conclude with some final remarks and a discussion of future work.

2. Related Work

FDPL falls in the broad class of fault diagnosis, particularly multiple failure diagnosis (MFD) problems. For the vast literature on the topic, we refer the reader to comprehensive reviews presented in
(Venkatasubramanian et al., 2003; Hwang et al., 2009; Gao et al., 2015). Here, we focus on studies involving settings similar to the ones we consider (spreading failures) or involving methodologies similar to the ones we propose to address the diagnosis problem.

2.1. Applications

Failure spread is a common phenomenon in complex electronic systems (networks, circuits) and chemical plants (reactors), and therefore, MFD is a widely researched subject in electrical and chemical engineering fields. It has been an active research area, especially for analog and digital circuits (Abramovici and Breuer, 1980; Maidon et al., 1997; Tadeusiewicz and Halgas, 2006; Lin et al., 2007). In large chemical plants, where an enormous number of units such as reactors, pipes, and valves operate simultaneously, solving the MFD problem is critical to find malfunctioned parts and provide a solution for continuing the chemical process as quickly as possible. MFD takes the form of comorbidity diagnosis in the medical field. Various studies in this literature focus on different aspects of the problem such as symptom decomposition and clustering (Wu, 1990, 1991); causal probabilistic modeling (Heckerman, 1990; Suojanen et al., 2001); case-based reasoning (Macura and Macura, 1997; Hsu and Ho, 2004).

2.2. Solution methods

The most commonly used approaches to address MFD include statistical methods, approximation methods, density-based methods and artificial neural networks (Isermann, 2011). de Kleer and Williams (1987) develop consistency-based reasoning algorithms for multiple fault diagnosis that were mostly applied to static systems with multiple failed components. Later, Reiter (1987) generalizes the research and provides a theoretical foundation for diagnosis from first principles. de Kleer et al. (1992) analyze the concept of diagnosis in detail and explores the notions of implicate/implicant and prime implicate/implicant. Alternative methodologies such as signed digraph models (Vedam and Venkatasubramanian, 1997; Watanabe et al., 1994), principal component analysis (Raich and Çinar, 1995); dynamic partial least squares (Lee et al., 2004); and artificial neural networks (Venkatasubramanian and Chan, 1989) have been used to solve MFD instances in different applications. Most of these methods benefit from a data-driven approach and causal connectivity of fault-symptom pairs, and the failure interactions are not considered. Lin and Goebel (1990) is the first study to consider fault interactions and represent them on a directed graph to model MDF as a Steiner Tree problem. A specialized algorithm is proposed, which runs in polynomial time for a fixed number of observations. However, no computational experiments have been reported to assess the efficiency of the suggested methodology to solve realistic size problem instances. Tu et al. (2003) also consider graph-based modeling for MDF and develop a sophisticated sub-gradient algorithm to solve it. Extensive numerical experiments have been conducted to investigate the efficacy of the suggested methodology to solve large-size problem instances. Focusing on zero-time systems, Le and Hadjicostis (2007) propose max-product and sequential max-product algorithms to solve the MDF in the case of unreliable sensor information and provide both analytical and numeric analysis to assess the efficiency of the suggested methodologies. In a more recent study, Chiang et al. (2015) propose a modified distance/causal dependency algorithm to solve MFD with spreading failures. The authors
consider four types of multiple faults: induced faults, multiple independent faults, multiple masked faults, and dependent faults. In recent studies, deep learning approaches are widely used to detect failures (Li et al., 2019b; He and He, 2017; Liu et al., 2018; Wen et al., 2018; Li et al., 2019a; Wu and Zhao, 2018; Jiang et al., 2019; Lei et al., 2019; Lei et al., 2016). Unlike ours, none of these studies consider false positives and negatives in the symptom set and provide the root cause along with the spreading path, which is critical to know for improving the system’s resilience.

Bayesian Networks (BN) have been successfully used over the decades for diagnosis methodologies. We refer the reader to Cai et al. (2017) and Cai et al. (2017) for a broad review of how BN models are utilized as a data-driven approach to address various diagnosis problems. Due to their success and natural fit to the problem context, BN methodologies are considered a benchmark in many studies to evaluate the performance of alternative fault detection methods. In one of the closest studies to our work, Kandula et al. (2005) investigate MFD in communication networks and present a tool for root cause analysis of faults. The authors compare their algorithm’s performance against the Bayesian classification methods and a general minimum set cover algorithm, one of the most used methodologies to solve MFD, as we discuss next.

2.3. Set covering models

From the modeling perspective, our study is closely related to the classical minimum set covering (SC) problem (Wolsey, 1998). SC is one of the oldest and most studied optimization problems in the literature. The interested reader is referred to Caprara et al. (2000) for a comprehensive survey on alternative approaches to solving SC. For diagnostic expert systems, using the general model of SC was first proposed by Reggia et al. (1983, 1985). In these seminal studies, authors use the causal relationship between disorders and their symptoms, and they define the term explanation as finding a subset of disorders that can explain the symptoms that emerged in the system. They propose a general model that consists of two conflicting goals. Firstly, the subset of disorders should cover all of the manifestations. Secondly, this explanation should be the smallest set that can explain it since the simplest explanation (involving the fewest entities) is the most acceptable one according to the Principle of Parsimony or as known as the Ockham’s Razor (Peng and Reggia, 1986). In their formulation, they assume that it is possible to have multiple disorders at a time. However, none of the aforementioned studies consider the fault interactions between the system components (spreading failures).

Our study differs from the general set covering models studied in the combinatorial optimization literature and from those used for diagnosis. As one of the main contributions of our study, we investigate multiple failures in a system, but we do not assume that the components’ failures occur independently. Taking fault interactions into account by considering failure spread probabilities, we suggest a new approach to solve FDPL. As we discuss in detail, when we introduce our mathematical model in the next section, such an approach requires to impose a specific structure for the failure set to choose (to cover a given set of symptoms), which motivates our formulation that includes additional constraints (exponentially many) on top of the classical set covering formulation. To solve this challenging extension of an already difficult (NP-Hard) problem (Garey and Johnson, 2002), we propose an efficient branch-and-cut
algorithm that can solve realistic problem instances. In this perspective, our work is also related to the studies that investigate connected facility location problems that arise in various applications (Swamy and Kumar, 2004; Chen et al., 2010; Farahani et al., 2012; Ljubić and Gollowitzer, 2013; Yıldız and Karaşan, 2015; Chen et al., 2015; Yıldız and Karaşan, 2017; Kızıl and Yıldız, 2023).

3. Mathematical Model

3.1. Problem Definition and Notation

In this section, we provide definitions and notation pertinent throughout the paper. Additional definitions and notation will be listed on a need basis. To simplify the exposition, in the sequence, we will first introduce our model for the basic case where we assume that all detected symptoms indeed exists in the system (no false positives), although some of the failure symptoms may not have been detected (i.e., there may be false-negatives). We then relax this assumption and show how to extend our basic approach to solve the problem instances with both false-positive and false-negative observations.

We consider a complex system with a set of components denoted by $C$. At any time, a component $i \in C$ may either fail on its own (spontaneously), or it may fail due to the failure of another related component in the system. We use the term spread for the latter. The spontaneous failure probability of a component $i$ during some given time interval is denoted by $P(i)$ and the probability of spread of failure from component $i$ to any other component $j \in C \setminus \{i\}$ is denoted by $P(i, j)$. We call two nodes $i, j \in C$ related, if $P(i, j) > 0$. When a component $i$ fails, the system may show a set of symptoms among a known set $M_i$. The component-symptom associations are represented by the collection $M = \{M_i : i \in C\}$. For the notational convenience, for a set $S \subseteq C$ we define $M(S) = \bigcup_{i \in S} M_i$, as the plausible set of symptoms for the failures of the components included in $S$. The set of all symptoms is denoted by $M$, i.e., we define $M = M(C)$. A symptom $m$ can be associated with more than one component, and we denote the set of components whose failures can result in the observation of $m$ with $C(m)$. A given system is characterized by a three-tuple $(C, P, M)$. When a set of components $S \subseteq C$ fail, due to imperfect state information, a random subset $M^+ \subseteq M(S)$ of the plausible symptoms emerges (is detected).

A rooted tree formed by a subset of components $C$ is called a failure-chain. More formally, we define a failure-chain $\phi = \{c_1, \ldots, c_n\}$ as an ordered subset of $C$, where $\ell$ denotes order in which the component $c_\ell$ fails in the chain and for all $c_\ell \in \phi, \ell > 1$, there exists a parent $c_{\bar{\ell}}$ such that $\bar{\ell} < \ell$ and $P(c_{\bar{\ell}}, c_\ell) > 0$. The component $c_1$, which does not have a parent, is called the root-cause of the chain. The likelihood of a failure-chain $\phi = \{c_1, \ldots, c_n\}$ is defined as $P(\phi) = P(c_1) \prod_{\ell=2}^{n} P(c_\ell, c_{\bar{\ell}})$.

A collection of failure-chains with distinct sets of components is called an explanation. For an explanation $\epsilon = \{\phi_1, \ldots, \phi_n\}$ we define its component set as $C(\epsilon) = \bigcup_{k=1}^{n} \phi_k$. A symptom $m$ is called to be covered by an explanation $\epsilon$, if at least one of the failures in $\epsilon$ can account for emergence of $m$, i.e., $m \in M(C(\epsilon))$. For a given set of observed symptoms $M^+ \subseteq M$, an explanation $\epsilon$ is called as a plausible-explanation, if all the symptoms in $M^+$ are covered by $\epsilon$, i.e., $M^+ \subseteq M(C(\epsilon))$. The likelihood of an explanation $\epsilon = \{\phi_1, \ldots, \phi_n\}$ is denoted by $P(\epsilon) = \prod_{\phi \in \epsilon} P(\phi)$.

Consider the following small problem instance Example-1, which we will also refer to in the rest of the section to explain our solution approach.
• Component set: $C = \{1, 2, 3, 4, 5\}$,
• Spontaneous failure probabilities: $P(i) = 0.01$ for all $i \in C$.
• Spread probabilities: $P(1, 3) = 0.1$, $P(3, 1) = 0.05$, $P(1, 2) = 0.2$, $P(1, 4) = 0.05$, $P(2, 4) = 0.2$ and zero for the rest of the component pairs.
• Component-symptom associations are as indicated in Figure 1, i.e., $M_1 = \{a, b, c\}$, $M_2 = \{b, c, e, g\}$, ..., $M_5 = \{g, h, i\}$.
• Observed symptoms: $M^+ = \{c, g, h\}$ (marked green in Figure 1).

For this small example, one can build several plausible explanations for the observed symptoms $M^+ = \{c, g, h\}$. In Figure 2 we present four such plausible explanations ($\epsilon_1$, $\epsilon_2$, $\epsilon_3$ and $\epsilon_4$) with a different number of failure chains and a different number of nodes in each one of them. The figure also shows the likelihood calculations for the respective explanations, revealing the basic intuition to look for an explanation with the highest likelihood to find the failed components. Note that, in this example, one needs to consider at least two failed components to account for the observed symptoms. When the probability of a failure due to spread is much higher than a spontaneous failure, explanations that consider failure chains with high likelihood scores are more likely to provide the correct explanation for the observed symptoms. For instance, in Example-1, among other explanations, $\epsilon_2$ has a higher likelihood score as it considers the strong probability of spread between components 2 and 4 (from 2 to 4), which can jointly account for the observed symptoms. Building on this intuition, we formally define the FDPL as follows.

Definition 1. For a given system $\langle C, P, M \rangle$ and the set of observed symptoms $M^+$, FDPL is to find the plausible-explanation for $M^+$ with the highest likelihood score.

We want to conclude this part by noting that the likelihood scores we consider in our method do not consider false-negative probabilities. One heuristic approach to achieve this would be updating the arc weights by considering the unobserved symptoms of the target node. However, in that case, the likelihood scores would then suffer from “double counting” if the missing symptom (e.g., due to a failed sensor) is shared by multiple component failures. Indeed in our preliminary studies where we implemented the mentioned update, we observed that the prediction performance of the algorithm is decreased because
of this double counting issue. Therefore, a more sophisticated approach, which would possibly require some significant changes in the methodology we develop in this study, is required to update the likelihood scores to consider false negatives properly. We leave this to future work.

Next, we present the integer programming (IP) formulation we develop to model FDPL and the branch-and-cut algorithm we devise to solve it.

3.2. Solution approach

3.2.1. IP formulation

We model FDPL over a spread-graph $G = (N, A)$, which is a weighted directed graph with a node set $N$, arc set $A$, and weights $w_{ij}$ for each arc $(i, j) \in A$. The node set contains the set of components $C$ as well as a special node $s$, which we use to represent spontaneous failures of the components, i.e., $N = C \cup \{s\}$. The arc set $A$ is composed of two groups of arcs $A_1$ and $A_2$, where $A_1 = \{(s, i) : i \in C, P(i) > 0\}$ represents the spontaneous failures of the components and $A_2 = \{(i, j) : i, j \in C, i \neq j, P(i, j) > 0\}$ represent the initiation of failures by spread. For the reasons which will be more clear when we explain the details of our solution approach, we define $w_{si} = -\log(P(i))$ for an arc $(s, i) \in A_1$ and $w_{ij} = -\log(P(i, j))$ for an arc $(i, j) \in A_2$. For the small problem instance presented in the previous subsection (Example-1), one can construct the spread-graph as shown in Figure 3.
Note that each explanation presented in Figure 2 can be represented by a tree (rooted at $s$) in the spread graph as shown in Figure 4, where the tree weights are equal to the negative natural logarithm of the likelihood scores for the respective explanations. We next formalize this observation with the following proposition which lays the foundation for our IP formulation. Before proceeding with the proposition we first define the notion of a plausible-tree.

![Diagram of plausible-trees](image)

**Figure 4:** Some plausible-trees for $M^+ = \{c, g, h\}$ (Example 1)

**Definition 2.** For a given FDPL instance $(C, P, M, M^+)$, a tree $T$ in the respective spread graph $G$, rooted in $s$, is called a plausible-tree if the components included in $T$ can cover the symptom set $M^+$, i.e., $M^+ \subseteq M(C(T))$.

**Proposition 1.** Let $T^*$ be a minimum weight plausible-tree in the spread-graph $G$ of a FDPL instance $(C, P, M, M^+)$. Then the plausible-explanation $\epsilon^*$ that is derived from $T^*$, by removing the root node $s$, is an optimal solution for the given FDPL instance.

**Proof.** Clearly, $\epsilon^*$ is a plausible-explanation simply due to $T^*$ being a plausible-tree by definition. So we just need to show that $\epsilon^*$ indeed has the highest likelihood score among all the plausible-explanations.
We establish this by showing that one would reach a contradiction otherwise. Let $\bar{\epsilon}$ be a plausible-explanation such that $P(\bar{\epsilon}) < P(\epsilon^*)$. Then one can build a plausible-tree $\bar{T}$ in $G$, by joining the node $s$ with the failure-chains in $\bar{\epsilon}$. But then we would then have $\sum_{(i,j)\in \bar{T}} w_{ij} < \sum_{(i,j)\in T^*} w_{ij}$, simply due to our assumption $P(\bar{\epsilon}) < P(\epsilon^*)$, which contradicts $T^*$ being the smallest weight plausible-tree in $G$. Hence, the result follows.

Proposition 1 establishes that one can solve a given FDPL instance by finding the smallest weight plausible-tree in the respective spread-graph. Equipped with this result, we now present our integer programming formulation that aims to find the minimum weight plausible-tree (MWPT) in a given spread-graph, and thereby solve a given FDPL instance.

We define the following binary decision variables to formulate the MWPT problem.

- Component inclusion variables $y_i$ takes the value of one, if a component $i \in C$ is included in the tree and zero otherwise.
- Spread variables $x_{ij}$ takes the value of one, if the arc $(i,j) \in A$ is included in the tree and zero otherwise.

For a quick reference, the problem parameters are listed in Table 1, followed by the formal definition of the integer programming formulation $IPT$.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>Set of components</td>
</tr>
<tr>
<td>$P(i)$</td>
<td>Spontaneous failure probability of a component $i \in C$</td>
</tr>
<tr>
<td>$P(i,j)$</td>
<td>Failure spread probability from component $i \in C$ to component $j \in C \setminus {i}$</td>
</tr>
<tr>
<td>$M_i$</td>
<td>Set of known symptoms associated with the failure of a component $i \in C$</td>
</tr>
<tr>
<td>$M$</td>
<td>Collection of component-symptom associations; $M = {M_i : i \in C}$</td>
</tr>
<tr>
<td>$M(S)$</td>
<td>Set of symptoms associated with a set $S \subseteq C$; $M(S) = \cup_{i \in S} M_i$</td>
</tr>
<tr>
<td>$C(m)$</td>
<td>Set of components whose failure can generate a symptom $m \in M$</td>
</tr>
<tr>
<td>$M^+$</td>
<td>Set of observed symptoms</td>
</tr>
<tr>
<td>$P(\phi)$</td>
<td>Probability of a failure-chain, $P(\phi) = P(c_1) \prod_{k=2}^n P(c_k, c_{k-1})$</td>
</tr>
<tr>
<td>$C(\epsilon)$</td>
<td>Collection of components of an explanation; $C(\epsilon) = \cup_{k=1}^n \phi_k$</td>
</tr>
<tr>
<td>$M(C(\epsilon))$</td>
<td>Set of symptoms that can cover the explanation $\epsilon$</td>
</tr>
<tr>
<td>$P(\epsilon)$</td>
<td>Likelihood of an explanation; $P(\epsilon) = \prod_{\phi \in \epsilon} P(\phi)$</td>
</tr>
<tr>
<td>$G$</td>
<td>Spread-graph representing the network</td>
</tr>
<tr>
<td>$N$</td>
<td>Set of nodes in the spread graph</td>
</tr>
<tr>
<td>$A$</td>
<td>Set of arcs in in the spread graph</td>
</tr>
<tr>
<td>$w_{ij}$</td>
<td>Weight of an arc $(i,j) \in A$</td>
</tr>
<tr>
<td>$C(T)$</td>
<td>Set of components in a rooted tree $T$</td>
</tr>
</tbody>
</table>

\[
\min \sum_{(i,j)\in A} w_{ij} x_{ij} \tag{1}
\]
The objective is to minimize the total weight of the solution (a plausible tree rooted in $s$). Constraints (2) ensure that for each observed symptom, there is at least one associated fault included in the tree; hence, the solution is a plausible tree for the given problem instance. Constraints (3) indicate a necessary condition that if a component is included in the solution, then exactly one of its incoming arcs should be active (included in the solution) so that the result is a tree in $G$. Although necessary, these constraints are not sufficient to ensure the solution is a tree in $G$, for that purpose, we add the cycle elimination Constraints (4) which are proposed by (Lee et al., 1996) to formulate Steiner-Tree problems and can completely characterize the respective spanning-tree polytope when the given values of the component inclusion variables $y_i, \in N$ (Edmonds, 2003). Lastly, the decision variables are binary and their domains are defined by (5) and (6).

Note that the suggested modeling approach does not consider the symptoms’ detection times to decide whether a given explanation is plausible. One straightforward way to update the proposed formulation to achieve this is to replace the set $C(m)$ in the coverage Constraint (3) with $C(m, M_+) = \{c \in C(m) : c \notin C(\bar{m}) \text{ for all } \bar{m} \in M, t_m \neq t_{\bar{m}}\}$, where $t_m$ indicates the time that a symptom $m \in M_+$ is recorded. Clearly, if such information is available, the suggested update would significantly reduce the number of plausible alternative explanations for the given symptom set and the number of the non-zero entries in the mathematical program. The resulting model would provide more accurate predictions in the shorter run times. However, in many cases, for example, with scheduled or after mission maintenance checks, the symptoms are detected simultaneously without any recorded information about the time they occurred in the system. To make our approach general enough to be applicable in such cases, we did not consider such an update in this initial study that introduces the FDLP in its most general form.

3.2.2. Branch-and-cut algorithm

In IPT, the set of constraints (4) may get very large in number as the number of components in the system ($|C|$) grows. So, it is not practical to solve it directly and to overcome this difficulty, we suggest a branch-and-cut approach to include the cycle cancelation constraints iteratively, as they are needed. In this section, we discuss the details of our branch-and-cut algorithm (BC) to solve IPT iteratively. At each iteration, we solve IPT with a subset of the cycle cancellation constraints (4) and solve a separation problem to detect violated inequalities to include in the model for the next iteration. We solve IPT with a branch and bound approach by starting the algorithm with the relaxed formulation IPT$_r$, which does...
Connectivity-check algorithm (CC). The main idea behind the CC algorithm is to consider a sub-graph \( \bar{G} \) of \( G \) induced by the solution \((\bar{x}, \bar{y})\) and conduct a connectivity check to detect violated inequalities in an efficient way. Let \( \bar{A} = \{(i, j) \in A : \bar{x}_{ij} > 0\} \) and \( \bar{N} = \{i \in N : \bar{y}_i > 0\} \) be the set of arcs and nodes of \( G \) included in the solution \((\bar{x}, \bar{y})\). We define \( \bar{G} = (\bar{N}, \bar{A}) \) as the induced-sub-graph of \( G \) for the given solution \((\bar{x}, \bar{y})\) and run a connectivity check on \( \bar{G} \) to detect the connected components \( K = \{K_1, \ldots, K_\ell\} \) in it. If there is more than one connected component in \( \bar{G} \), i.e., \( \ell > 1 \), we check if any of the following constraints are violated to add into the model.

\[
\sum_{(i,j) \in A(K)} x_{ij} \leq \sum_{i \in K \setminus \{k\}} y_i \quad \forall K \in \mathcal{K}.
\]  

The pseudo code for the CC algorithm is provided in Algorithm 1.

**Algorithm 1: Connectivity Check**

```plaintext
input : (\bar{x}, \bar{y})
output: \(V\)
1. Initialize the set of violated inequalities \( V = \emptyset \);
2. Set \( \bar{A} = \{(i, j) \in A : \bar{x}_{ij} > 0\} \) and \( \bar{N} = \{i \in N : \bar{y}_i > 0\} \);
3. Build the induced graph \( \bar{G} = (\bar{N}, \bar{A}) \);
4. Find the set of connected components \( \mathcal{K} \) in \( \bar{G} \);
5. for \( K \in \mathcal{K} \) do
6. \hspace{1em} for \( k \in K \) do
7. \hspace{2em} if \( \sum_{(i,j) \in A(K)} x_{ij} > \sum_{i \in K \setminus \{k\}} y_i \) then
8. \hspace{3em} Add the inequality \( \sum_{(i,j) \in A(K)} x_{ij} \leq \sum_{i \in K \setminus \{k\}} y_i \) to \( V \);
9. return \( V \)
```

Note that when the solution \((\bar{x}, \bar{y})\) is binary, i.e., no variable assume fractional values, CC can solve the separation problem exactly. Clearly, for the fractional solutions, if CC fails to detect a violation, one can continue the branch and bound algorithm by performing regular branching cuts. It is also worth mentioning that as a graph search algorithm, the run time complexity of the CC is \( O(|\bar{A}|) \), i.e., grows linearly in the number of arcs in the induced-graph \( \bar{G} \), which is typically much smaller than that of original separation-graph \( G \). As we discuss in the next section in more detail, having such an efficient procedure to solve the separation problem contributes greatly to the overall computational efficiency of the BC algorithm. As an alternative, the separation problem can also be solved as a maximum flow problem on a bipartite graph whose node-set is composed of the binary decision variables of \( IPT \). For details of this approach, we refer the reader to Lee et al. (1996). However, our preliminary studies indicated that, for the problem instances we considered in our computational experiments, the best computational performance is achieved when the separation problem is solved only for the integer solutions using the CC algorithm.
We want to conclude by noting that, as implied by the NP-Hardness results for the general Steiner
tree problems (Garey and Johnson, 2002), there is no algorithm known to solve the separation problem
exactly for the fractional solutions in polynomial time (Wolsey, 1998), and thus the BC algorithm has an
exponential run time in general. However, despite this unfavorable worst-case (theoretical) performance
bound, as we show in the following section, our algorithm can actually solve realistic problem instances
in reasonable run times, much better than what is achieved by the state-of-the-art, showing its practical
value.

3.2.3. Solving the problem instances with false-positive observations

In order to account for the wrong observations about the true system state, we modify the IPT
formulation by introducing a new problem parameter $\beta$, which indicates the expected percentage of
correctly detected symptoms as a result of the noisy or defective sensors. Clearly, the system management
would not know the exact value of this parameter, but as we show in this section, a reasonable “estimate”
of this critical system parameter can be used to make accurate predictions.

The formulation $IPT - F$ that considers the false-positive observations to solve FDPL is defined as
follows:

\[
\begin{align*}
\min & \sum_{(i,j)\in A} w_{ij}x_{ij} \\
\text{s.t.} & \sum_{m\in M^+} z_m \geq |M^+| - (1 - \beta)|M| \\
& \sum_{i\in C(m)} y_i \geq z_m \quad \forall m \in M^+, \\
& \sum_{(i,j)\in A} x_{ij} = y_j \quad \forall j \in C, \\
& \sum_{(i,j)\in A(S)} x_{ij} \leq \sum_{i\in S \setminus \{k\}} y_i \quad \forall S \subseteq N, \forall k \in S, \\
& x_{ij} \in \{0, 1\} \quad \forall (i,j) \in N, \\
& y_i \in \{0, 1\} \quad \forall i \in N, \\
& z_m \in \{0, 1\} \quad \forall m \in M
\end{align*}
\]

Unlike $IPT$, formulation $IPT - F$ does not force itself to cover all the symptoms observed from the
system but instead accounts for the fact that $(1 - \beta)$ portion of false symptoms can be recorded as a
result of the noisy or defective sensors. Two constraints regulate this extension: Constraint (9) determines
which symptoms should be considered as accurate, therefore to be covered. Then Constraints (10) ensure
that for each considered symptom, there is at least one associated fault included in the tree; hence the
solution is a plausible tree for the given problem instance.

As we mentioned before, not knowing the exact value of the $\beta$, an estimate $\hat{\beta}$ can be used instead of
\( \beta \) in \( IPT - F \) to predict faulty components. As such, \( \hat{\beta} \) can be considered a user-defined parameter for our prediction model, which can be determined by a k-fold validation approach as we show to work well in our numerical experiments. Clearly, the same branch-and-cut approach we defined to solve \( IPT \) also works for solving the \( IPT - F \), with obvious technical modifications which we skip here for the sake of brevity.

Finally, we would like to underline that the solution approach we develop assumes that the failure probability of components is not changing over time, which is the case for exponential distribution between the failures. Even if the failure times do not have the memoryless property, in practice, the suggested approach can still work well if the expected lifetimes of the components are much longer than the periods of maintenance inspections, which is the case we consider in this study. To show this, in our numerical experiments, we study problems instances with different failure arrival distributions and show that the suggested approach can provide accurate predictions also in such cases.

4. Computational Studies

In this section, we present the details of the extensive numerical experiments conducted to test the computational efficiency and the prediction accuracy of the BC algorithm against the state-of-the-art in the literature. In particular, as the benchmarks from the literature, we consider the Shrink algorithm by Kandula et al. (Kandula et al., 2005), Bayes Classifier (Murphy, 2001), and the classical minimum cardinality set cover approach first suggested by Reggia et al. (Reggia et al., 1983), which is essentially generalized by the BC. We also tested a weighted Set Covering (wSC) extension of the SC, which considers the spontaneous failure probabilities of the components to determine component weights and then solves a weighted set cover problem to determine failed components. To be more precise, SC aims to cover the observed symptoms with a minimum-cardinality component set, while wSC aims to cover those symptoms with a minimum-weight component set, where the component weights are defined as \( w_i = -\log(P(i)) \), \( \forall i \in C \). Mathematical formulations we use to solve SC and wSC are presented in the online supplementary file.

Before proceeding with the analysis of the results of our numerical experiments, we first present the details about the instance generation and the implementations of the considered algorithms.

4.1. Instance Generation

We generate our instances considering various potential real-world applications with different numbers of fault interactions and fault-symptom associations such as nuclear power plants (Kvam and Miller, 2002), spacecraft (Rutledge and Mosleh, 1995), air-handling systems (Yan et al., 2017), vehicles (Zhang and Horigome, 2001), virtual private networks (Bennacer et al., 2014), and process monitoring (Liu et al., 2013). Instead of basing our computational study parameters on a single application area, we conduct our experiments for a large range of critical parameter values to better assess the suitability of the proposed approach as a general methodology to solve FDLP.

Our experiments consider a system with 150 components (i.e., \( |C| = 150 \)). For the size of the symptom set \( M \), we consider seven levels where \( |M| \in \{100, 150, 200, 250, 500, 750, 1000\} \). Here we want to note
that for a fixed number of components, a smaller symptom set implies more symptoms to be shared between various components, making the diagnosis problem harder, as the number of alternative plausible explanations increases when the same symptom is associated with a larger number of faults. For each component, we randomly choose $\mu$ number of symptoms from the symptom set $M$, where $\mu$ is a uniformly distributed random variable in $[\underline{\mu}, \overline{\mu}]$. In our computational experiments, we set $\mu$ and $\overline{\mu}$ to 10 and 20, respectively.

We draw spontaneous failure probabilities, $P(i), i \in C$, from a Pareto Distribution (Arnold, 2015) with support $(0, \overline{P})$, such that 80% of the failure chains in the system are expected to be initiated from the 20% of the components. In our experiments, we consider the case with $\overline{P} = 0.0005$. As it will be more evident when we describe how we simulate the generation of faults, we consider such a small value for $\overline{P}$ to obtain problem instances where a relatively small fraction of the components are to be faulty at the time of inspection.

We control the number of fault interactions between the system components with a density parameter $d$, which indicates the density of the resulting spread-graph for a given problem instance. More specifically, $d$, controls the total number of interactions in terms of the percentage of maximum number of failure interactions, which we choose from $\{0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06\}$. For a given $d$ value, we randomly choose $d|N|(|N| - 1) - |C|$ number of component pairs for which we consider a positive spread probability. For such component pairs $(i, j)$, we draw a random number from the interval $(0, \overline{\Omega})$, where we consider the cases with $\overline{\Omega} \in \{0.00125, 0.0025, 0.125, 0.1875\}$ to control the relative likelihoods of spontaneous failures versus the failures due to spread. Clearly, for the higher values of $\overline{\Omega}$, a higher proportion of the failed components fail due to the spread. However, we note that the number of components in the system (150) is much higher than the out-degree of a component node in the spread-graph (between 1.5 and 7.5, on average). Therefore, one needs to consider $\overline{\Omega}$ values that are much higher than $\overline{P}$, to obtain problem instances where the component failures happen mostly due to spread (i.e., less than 20 percent of the failed components brake down due to spontaneous failures.)

To account for the imperfect state information we assign different expression probabilities $P(i, m)$, for each $i \in C$ and $m \in M_i$, which denotes the probability that the symptom $m$ will be observable if component $i$ fails. We randomly choose expression probabilities from the interval $[0, 1]$, where we consider the problem instances with $\overline{\lambda} \in \{0.4, 0.55, 0.7, 0.85, 1\}$ in our experiments.

As explained before, to account for the false symptoms, we introduce $\beta \in [0, 1]$ as the probability of correct measurement for each symptom. Clearly, when $\beta = 1$, all the symptoms are correctly measured. Using various $\beta$ values, we generate a set of component-symptom pairs and for each $\beta$, we test different $\hat{\beta}$ values, where $\hat{\beta}$ is our estimation for $\beta$.

After the system parameters are fixed, we randomly generate the faults with a simple simulation (presented in Algorithm 2) where the faults occur spontaneously or by spread considering the respective probabilities. We consider a case where a fault-free system is run for $t = 30$ time units until the maintenance check is performed to detect the components that have broken down during this time interval. If a simulation return no faults at the end we ignore the resulting problem instance in our experiments. The number of faults that emerged in the system when the simulation ends is denoted by $p$ in our anal-
yses. Once we determine the failed components $\bar{C} = F \setminus \{s\}$ we generate the set $M^+$ by considering the symptom expression probabilities following the steps indicated in Algorithm 3.

**Algorithm 2: Fault Generation**

input : $(G, t)$
output: $(F)$

1. Initialize $F = \{s\}$;
2. for $t \in [1, 30]$ do
   3. Set $\bar{F} = \emptyset$;
   4. for $i \in F$ do
      5. for $(i, j) \in A$ do
         6. Draw a uniform random variable $r$ between 0 and 1;
         7. if $r \leq P(i, j)$ then
            8. $\bar{F} = F \cup \{j\}$;
      9. $F = F \cup \bar{F}$;
10. return $F$

**Algorithm 3: Symptom Generation**

input : $(M(C), \bar{\lambda})$
output: $(M^+)$

1. Initialize $M^+ = \emptyset$;
2. foreach $m \in M(C)$ do
   3. Draw a uniform random variable $r$ between 0 and 1;
   4. Draw a uniform random variable $\lambda$ between 0.1 and $\bar{\lambda}$;
   5. if $r \leq \lambda$ then
      6. $M^+ = M^+ \cup \{m\}$;
7. return $M^+$

### 4.2. Implementation Details

All computational experiments are performed on a computer with 16 GB of RAM and 3.6 GHz Intel Core i7-4790 processor. As mentioned before, we tested minimum cardinality set cover (SC), minimum weight set cover (wSC), BayesNet, and Shrink algorithms as benchmarks to assess the performance of the BC algorithm we developed in this study. We implemented the BC, SC, and wSC in Java using CPLEX 12.9 to solve the respective integer programs. For BC, we used the LazyCutCallBack feature of CPLEX to detect and add the violated inequalities for the integral solutions found during the branch and bound search. We implemented the Shrink algorithm, using R (R Core Team, 2018), as described by (Kandula et al., 2005). For BayesNet, we implemented the naive Bayes classifier algorithm by using the Bayes Net Toolbox for Matlab as suggested in (Murphy, 2001).
4.3. Experimental design

Our main goal with the numerical experiments is to understand how the diagnosis accuracy and the computational performance of the studied approaches are impacted by the following problem properties.

- Number of alternative plausible-explanations,
- Number of fault interactions between the system components (number of possible failure chains in the system),
- Intensity of the fault interactions between the system components (proportion of simultaneous failures versus failure due to spread),
- Level of information about the true system state.
- Effect of false symptoms introduced to the system.

To try to find the answers to these questions, we have conducted four groups of experiments, each builds on the base case setting (BS) which we define by the parameter values: \( |M| = 150, d = 0.05, \overline{\Omega} = 0.125, \bar{\lambda} = 0.7 \) and \( \beta = 1 \).

E1: Computational performance analysis. The first set of experiments, E1, aims to investigate the impact of the number of plausible explanations on the computational complexity and diagnostic accuracy of the BC algorithm and the benchmark algorithms from the literature. For that purpose, we consider seven different levels for the total number of symptoms \( |M| \in \{100, 150, 200, 250, 500, 750, 1000\} \).
Note that fixing \( |C| = 150 \), the higher number of symptoms decreases the number of plausible explanations as the number of component failures related to a given symptom decreases in \( |M| \).

E2: Impact of the number of fault interactions on the diagnosis performance. In the second set of experiments, E2, we aim to see how the number of fault interactions between the components impacts the performances of the diagnosis algorithms we consider in this study. For that purpose, E2 contains instances with seven different levels for the density parameter as \( d \in \{0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06\} \).

E3: Impact of the strength of fault interactions on the diagnosis performance. The third set of experiments, E3, aims to investigate the impact of spread probabilities on the performances of the considered algorithms. For that purpose, E3 contains five different maximum spread probability values as \( \overline{\Omega} \in \{0.0125, 0.025, 0.0625, 0.125, 0.1875\} \).

E4: Impact of imperfect system information on the diagnosis performance. In the fourth set of experiments, E4, we aim to observe the impact of imperfect information about the true system state, which is controlled by the maximum expression probabilities of the symptoms. E4 contains problem instances with five different levels for the symptom expression probabilities with \( \bar{\lambda} \in \{0.4, 0.55, 0.7, 0.85, 1\} \).

E5: Impact of falsely detected symptoms on the diagnosis performance. Finally, in the final set of experiments E5, we aim to observe the impact of the false symptoms that can be observed in the system. For that matter, E5 contains six different \( \beta \) values, where \( \beta \in \{0.95, 0.96, 0.97, 0.98, 0.99, 1.0\} \).
In all set of experiments, for each specific level of the varying problem parameter we run our failure spread simulation (Algorithm 3) as many times as needed to generate at least 150 instances for each \( p = 1, \ldots, 6 \).

4.4. Analysis of Results

In this part, we present the results of our numerical experiments and discuss their practical implications. Before proceeding with the results, we first want to explain the measures we consider to evaluate the diagnosis performance.

Each algorithm predicts a set of faulty components and provides a diagnosis. To measure the diagnosis accuracy of the algorithms, we compare the predicted fault set with the real fault set considering the following metrics:

- Number of true positives (TP): Number of failed components that are correctly identified.
- Number of false positives (FP): Number of non-faulty components that are mistakenly labeled as failed.
- Number of false negatives (FN): Number of failed component that are mistakenly labeled as non-faulty.

Using TP, FN, and FP, we calculate recall \( \frac{TP}{TP+FN} \) and precision \( \frac{TP}{TP+FP} \) scores of a given explanation, where recall indicates the fraction of correctly predicted faults to real faults and precision shows the fraction of correctly predicted faults to total predicted faults, respectively. Evaluating recall and precision scores together, one can consider two respective dimensions of the prediction accuracy, which are both critical in our setting with mostly non-faulty components at the time of inspection. Clearly, in such a setting, simply identifying all the components as non-faulty would give a very high accuracy score without much practical use. So we use the F1 Score (or F Measure), which is defined as the harmonic mean of the recall and precision measures and is widely used for evaluating the performance of classification algorithms with a single metric (Powers and Ailab, 2011).

In the sequence, we will discuss how the F1 scores vary, for the different algorithms, in the four experimental settings we specified earlier. But before starting to focus on the diagnosis accuracy we first want to investigate the computational performances of the algorithms we study.

4.5. Experiments

4.5.1. Computational performance analysis

Figure 5 shows the run-time of the considered algorithms for E1 instances, reporting the run times of the algorithms for various \( p \) values. Each graph shows the run time of considered algorithms (in logarithmic scale) for different cardinalities of the symptom set \(|M|\). Note that a smaller value of \(|M|\) indicates that the number of alternative plausible explanations in the system is high, and a larger \( p \) indicates that more symptoms are observed in the system, and thus, more faults should be considered for explaining them all. In both cases, the number of alternative explanations grows high, and diagnosis
problem gets harder to solve. The results for BayesNet are given only for $|M| = \{200, 250, 500, 750, 1000\}$ and Shrink results are provided only for $p \leq 3$, since BayesNet and Shrink algorithms were not able to provide a solution, within one hour, for the problem instances with other parameter values. Since the run times of the considered algorithms do not change significantly by the problem parameters other than the size of the symptom set ($|M|$) and the number of the failures to look for in the system ($p$), for the sake of brevity we do not present the run times of the experiments E2-E4, which present the same insights we discuss in the sequence.

Figure 5 presents critical insights into the applicability of considered algorithms in different problem settings. As expected, for the Shrink algorithm, we observe that the cardinality of the symptom set has a much smaller impact on the run time than the number of actual faults in the system. Being basically an enumeration algorithm with run time complexity of $O(|M|^p)$, Shrink cannot provide solutions (within a time limit of one hour) for instances with more than three failed components, where the diagnosis problem essentially gets harder. On the contrary, we see that the run times for BayesNet are not worsened much by the increase in $p$, but they get much larger as the number of alternative explanations grows ($|M|$ decrease). As mentioned above, for $|M|$ values less than 200, the BayesNet cannot provide a solution within one hour. In almost all the cases, the set covering family (BC, SC, and wSC) have the smallest run times (orders of magnitude better than Shrink and BayesNet), which can scale up well with the increasing problem size and complexity. As expected, we see that SC and wSC have similar runtimes. However, it is interesting to see that the difference between BC and the other two set covering algorithms is not as much, considering the fact that BC solves a much larger IP formulation (with an exponential number of constraints). We attribute this result mainly to the high efficiency of the separation procedure CC (Algorithm 1).

The F1 Scores for the E1 experiments are given in Figure 6 (recall and precision results for E1 experiments are provided in the online supplementary file). As expected, E1 results show that the diagnostic performances decrease as $p$ increases or $|M|$ decreases due to the increasing number of alternative expla-
nations. Considering the run time and diagnostic performances together (Figures 5 and 6), we can see that set covering family (SC, wSC, and BC) emerge as a better fit for the considered set of problems. Both the Shrink and BayesNet algorithms suffer from computational efficiency limitations and can only provide solutions for relatively easy problem instances (i.e., \( p \leq 3 \) or \( M \geq 500 \)), where F1 measures are already over 90% for all considered algorithms. Although the BayesNet can provide solutions for instances with more than three failed components, its diagnostic performance is relatively poor for \( |M| < 250 \). For example, we can see that the average F1 score of the BayesNet for \( p = 4 \) and \( |M| = 200 \) is around 30% while the F1 score of BC for the same instances is almost 100%. Comparing the performances of the set covering algorithms between each other, we see that BC outperforms SC and wSC, especially when \( |M| \) is less than 250, indicating that it is worthwhile to incur the relatively small extra computational cost to use BC instead of SC or wSC to solve FDPL.

4.5.2. Impact of the number of fault interactions on the diagnosis performance

Another critical problem parameter we aim to investigate in our experiments is the number of fault interactions between the system components, which is controlled by the parameter \( d \). Figure 7 illustrates the F1 scores of considered algorithms for the problem instances we study in E2 (recall and precision results for E2 are presented in the online supplementary file). As expected, these results show that for small \( p \), the impact of interconnectivity is not very pronounced. In particular, when \( p = 1 \), the density parameter has no impact since there is no spread in the system. However, as \( p \) gets larger (i.e., \( p \geq 3 \)), BC starts outperforming other alternatives as the interactions between the components play an important role in initiating chains of failures that BC is tailored to capture.

Interestingly, we see that F1 scores for BC are much better than the rest of the algorithms, even for small density values (i.e., \( d \) value of \( = 0.01 \)). We also observe a decline in the F1 scores for all the algorithms. As expected, for SC and wSC, ignoring failure interactions results in less accurate diagnosis as more of the failures happen due to spread (as \( d \) increases). However, it is interesting to see that after
a threshold, higher values of $d$ have a negative impact on the accuracy of the BC algorithm due to the increasing number of failure chains that the algorithm needs to consider.

4.5.3. Impact of the strength of fault interactions on the diagnosis performance

Note that, while the parameter $d$ controls the number of failure interactions between the system components, the intensity of those relations is controlled by the parameter $\Omega$, which denotes the upper limit for the probability of a failure spread between two components. The results for E3 experiments are given in Figure 8 for various levels of $\Omega$ (recall and precision results for E3 are presented in the online supplementary file). Note that by changing the maximum spread probability $\Omega$, we basically control the proportion of the failed components due to spread. A larger $\Omega$ indicates that more of the components result in failure due to spread (a higher level of failure interactions). As expected, for $p = 1$, i.e., only one component that fails spontaneously, the F1 scores of the algorithms are almost the same. However, we see a slight decrease in BC for larger values of $\Omega$. Since the spread probabilities are much higher at these values, BC occasionally finds an explanation containing more than one fault, whose likelihood score is higher than the correct explanation with one fault. However, as $p$ increases, the F1 score difference between BC and the basic set covering algorithms tends to go up, as a higher portion of failures happen due to spread, which BC is tailored to capture. Interestingly, we also observe that BC has a significant advantage over simple set covering approaches even for the systems with relatively low-intensity failure interactions.

4.5.4. Impact of imperfect system information on the diagnosis performance

The F1 scores of alternative algorithms for the problem instances in E4 are presented in Figure 9 (recall and precision results are presented in the online supplementary file), showing the impact of imperfect system information on the diagnosis performance. Recall that in E4, we study different levels for the problem parameter $\lambda$, which controls the symptom expression (successful diagnosis) probabilities.
As we see in Figure 9, missing information about the system state (symptoms) impacts the diagnostic performance significantly. As expected, all algorithms perform better as $\lambda$ increases. We also see that, as $p$ increases and the prediction problem becomes more complex, the impact of missing information on the F1 scores becomes more pronounced. We observe BC outperforms the other algorithms with a large margin for lower values of $\lambda$ and higher values of $p$, where the diagnostic problem becomes more challenging to solve. It is interesting to see that when $\lambda = 0.4$, i.e., only 25% of the failure symptoms are detected on the average, the F1 scores of the BC algorithm are still over 80% for all the $p$ values except for $p = 6$, which indicates the robustness of BC algorithm against the missing symptom information.
4.5.5. Impact of falsely detected symptoms on the diagnosis performance

In this part, we present the prediction accuracy for the E5 instances in Figure 10. Recall that we have the base case setting when $\beta = 1$. As false symptoms are introduced to the system (i.e., $\beta$ is reduced), the F1 scores decrease for all $p$, as expected. However, the decrease appears to be more critical when $p$ is small (i.e., $p \leq 2$) since adding false symptoms to a relatively small set of accurate observations has a more substantial impact. As such, for $p \geq 2$, we see that F1 scores are above 75 in almost all the cases. As an example to show the high prediction power of the BC, we see that for the instances with $p = 3$ and $\beta = 0.98$, the F1 score is close to 90, even though in the average, there are three false positives besides the 18 correctly detected symptoms (i.e., more than 14% of the detected symptoms are false positive).

As discussed at the end of the previous section, theoretically, our model assumes exponential inter-arrival times for the failures, and the problem instances we analyzed so far are generated accordingly. However, our algorithm can still make accurate predictions in practice even if this is not the case. To support this argument, we conduct additional experiments using the Weibull distribution to model failure inter-arrival times, as considered by many studies in reliability literature (McCool, 2012). As the results of our experiments in Appendix B show, the prediction accuracy of the BC algorithm dropped only slightly in these experiments, where the interarrival times for the failures are relatively shorter compared to the inspection periods, as we assume to be the case for the systems we focus in this study.

![Figure 10: F1 Scores vs. $\beta$ for $p = \{0.95, 0.96, 0.97, 0.98, 0.99, 1.0\}$](image)

5. Conclusion

In this paper, we study the fault diagnosis problem considering spreading failures and imperfect system state information. We propose a novel approach to address this challenging diagnosis problem and extend the literature in several directions. Representing the failure interactions between the components through a directed weighted graph, we propose a new integer programming formulation to model diagnosis
problems for the spreading failures and use graph theoretical results to devise an efficient branch-and-cut algorithm to solve it.

We conduct extensive numerical experiments to assess the potential of this new methodology both from the computational efficiency and diagnostic accuracy perspectives. To show the suitability of the proposed approach as a general methodology to solve FDLP instances that arise in different applications ranging from nuclear powerplants to spacecraft to virtual private networks, with different problem parameters, we tested a wide range of problem parameter values in our experiments, instead of focusing on specific values one would have in a particular application. As indicated by the result of these experiments, our proposed method can provide more accurate diagnoses in shorter run times compared to the state-of-the-art, for almost all instances we study in this paper with a wide range of problem parameter values. It is particularly interesting to observe that the superior performance of our method becomes more pronounced as the diagnosis problem gets more challenging (i.e., more symptoms are shared among different faults, more components are faulty at the time of inspection, or less accurate diagnosis of associated symptoms for the faulty components). Providing an example of the significant potential of applying advanced optimization techniques to model and solve complex classification problems that arise in many applications, we believe that the modeling approach we study in this paper will be of interest to researchers and practitioners working on similar diagnosis problems.

One limitation of our approach is disregarding the increasing fail probabilities of components by time, i.e., assuming a memoryless property for the interarrival times for failures. Although, this may not be unrealistic when the failure interarrival times are much longer compared to the inspection frequencies, justifying a constant failure rate, for those systems with relatively longer inspection times relaxing this assumption would be necessary for a high diagnosis performance. Extending the proposed algorithmic approach to achieve this provides an interesting research direction we plan to follow in our future studies. As another significant limitation of the proposed approach, the likelihood scores we consider in our method do not consider false-negative probabilities, i.e.; we do not take advantage of the information regarding the symptoms that have not been observed to update the probability of a component being in the failure mode. As we discussed when we introduced our algorithmic approach, due to the possible double-counting issues, it is not possible to address this problem with a simple graph transformation, requiring to develop new sophisticated approaches to properly include such information in the likelihood calculations, which presents an interesting new research direction to consider for future studies.

In this initial study that introduces a new graph-theoretical approach to model FDLP, we developed a branch-and-cut algorithm for the exact solution of the resulting integer program. Although we show that such an approach can solve practically relevant problem instances, with the underlying combinatorial complexity of the problem, the branch-and-cut algorithm may become impractical for larger problem instances. In this regard, devising fast heuristic approaches to detect plausible trees with high likelihood scores in the spread graph, instead of solving the integer programming formulations exactly, would be of interest to be able to extend the applicability of the proposed solution approach.
References

1. Miron Abramovici and Melvin A. Breuer. Multiple fault diagnosis in combinational circuits based on an


4. Leila Bennacer, Yacine Amirat, Abdelghani Chibani, Abdelhamid Mellouk, and Laurent Ciavaglia. Self-
diagnosis technique for virtual private networks combining bayesian networks and case-based reasoning.


6. Baoping Cai, Lei Huang, and Min Xie. Bayesian Networks in Fault Diagnosis. *IEEE Transactions on


10. Yao Cheng and Xiaoping Du. System reliability analysis with dependent component failures during early

11. Leo H. Chiang, Benben Jiang, Xiaoxiang Zhu, Dexian Huang, and Richard D. Braatz. Diagnosis of
multiple and unknown faults using the causal map and multivariate statistics. *Journal of Process


    apr 1987.


Kaibo Liu, Xi Zhang, and Jianjun Shi. Adaptive sensor allocation strategy for process monitoring and

Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. Artificial intelligence for fault di-
agnosis of rotating machinery: A review. Mechanical Systems and Signal Processing, 108:

Ivana Ljubić and Stefan Gollowitzer. Layered graph approaches to the hop constrained connected facility

Robert T Macura and Katarzyna Macura. Case-based reasoning: opportunities and applications in health

Y. Maidon, B. W. Jervis, N. Dutton, and S. Lesage. Diagnosis of multifaults in analogue circuits using


Dusko P Nedic, Ian Dobson, Daniel S Kirschen, Benjamin A Carreras, and Vickie E Lynch. Criticality

Chanyoung Park, Nam H Kim, and Raphael T Haftka. The effect of ignoring dependence between failure
modes on evaluating system reliability. Structural and multidisciplinary optimization, 52(2):251–268,
2015.

Yun Peng and James A Reggia. Plausibility of diagnostic hypotheses: The nature of simplicity. In AAAI,

David Powers and Ailab. Evaluation: From precision, recall and f-measure to roc, informedness, marked-

R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical

A. C. Raich and A. Çinar. Multivariate statistical methods for monitoring continuous processes: assess-
ment of discrimination power of disturbance models and diagnosis of multiple disturbances. Chem-

James A. Reggia, Dana S. Nau, and Pearl Y. Wang. Diagnostic expert systems based on a set covering


Appendix A. Mathematical Formulations of SC and wSC

We solve the following integer program to find SC diagnoses.

\[
\begin{aligned}
\min & \quad \sum_{i \in C} y_i \\
\text{s.t.} & \quad \sum_{i \in C(m)} y_i \geq 1 \quad & \forall m \in M^+, \\
& \quad y_i \in \{0, 1\} \quad & \forall i \in C
\end{aligned}
\]  

(A.1) (A.2) (A.3)

We solve the following integer program to find wSC diagnoses.

\[
\begin{aligned}
\min & \quad \sum_{i \in C} w_i y_i \\
\text{s.t.} & \quad \sum_{i \in C(m)} y_i \geq 1 \quad & \forall m \in M^+, \\
& \quad y_i \in \{0, 1\} \quad & \forall i \in C
\end{aligned}
\]  

(A.4) (A.5) (A.6)
Appendix B. F1 Scores for Weibull inter-arrival times

Figure B.11 shows the F1 scores of the BC algorithm in the cases of time-dependent (BC-Weibull) and time-independent (BC) failure generation, for the E1 problem setting. For the time dependent failure generation we use the Weibull distribution with parameters $1 \leq \eta \leq 2$ and $1 \leq \beta \leq 2$. The pseudocode for time-dependent fault generation simulation is given in Algorithm 4.

![Figure B.11: F1 Score Comparison for $p = \{1, 2, 3, 4, 5, 6\}$](image)

---

**Algorithm 4:** Time-Dependent Fault Generation

input : $(G_t, t)$

output: $(F)$

1. Initialize $F = \{s\}$;
2. for $i \in F$ do
   3. for $(i, j) \in A$ do
      4. Draw uniform random variables $\eta$ and $\beta$ between 1 and 2. Set them as parameters of Weibull Distribution. Use $P(i, j, t, \eta, \beta)$ for the spread probabilities;
   5. for $t \in [1, 30]$ do
      6. Set $\bar{F} = \emptyset$;
      7. for $i \in F$ do
         8. for $(i, j) \in A$ do
            9. Draw a uniform random variable $r$ between 0 and 1;
            10. if $r \leq P(i, j, t, \eta, \beta)$ then
                11. $\bar{F} = F \cup \{j\}$;
                $F = F \cup \bar{F}$;
         12. return $F$;
Appendix C. Recall and Precision Graphs for E1

In Figure C.12 we show the recall and in Figure C.13 we show the precision of the considered algorithms for E1.

Figure C.12: Recall vs. $|M|$ for $p = \{1, 2, 3, 4, 5, 6\}$

Figure C.13: Precision vs. $|M|$ for $p = \{1, 2, 3, 4, 5, 6\}$
In Figure D.14 we show the recall and in Figure D.15 we show the precision of the considered algorithms for E2.

Figure D.14: Recall vs. $d$ for $p = \{1, 2, 3, 4, 5, 6\}$

Figure D.15: Precision vs. $d$ for $p = \{1, 2, 3, 4, 5, 6\}$
Appendix E. Recall and Precision Graphs for E3

In Figure E.16 we show the recall and in Figure E.17 we show the precision of the considered algorithms for E3.

Figure E.16: Recall vs. $\Omega$ for $p = \{1, 2, 3, 4, 5, 6\}$

Figure E.17: Precision vs. $\Omega$ for $p = \{1, 2, 3, 4, 5, 6\}$
Appendix F. Recall and Precision Graphs for E4

In Figure F.18 we show the recall and in Figure F.19 we show the precision of the considered algorithms for E4.

Figure F.18: Recall vs. $\lambda$ for $p = \{1, 2, 3, 4, 5, 6\}$

Figure F.19: Precision vs. $\lambda$ for $p = \{1, 2, 3, 4, 5, 6\}$