A Branch and Cut Approach for Multiple Failure Diagnosis Problem with Imperfect State Information and Spreading Failures

by

Kaan Pekel

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

KOÇ ÜNİVERSİTESİ

January 5, 2020

A Branch and Cut Approach for Multiple Failure Diagnosis Problem with Imperfect State Information and Spreading Failures

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Kaan Pekel

and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the final examining committee have been made.

Committee Members:

Asst. Prof. Barış Yıldız (Advisor)

Prof. Metin Türkay

Asst. Prof. Hande Küçükaydın

Date:

to my family

ABSTRACT

A Branch and Cut Approach for Multiple Failure Diagnosis Problem with Imperfect State Information and Spreading Failures
Kaan Pekel
Master of Science in Industrial Engineering
January 5, 2020

Failure detection in complex systems is a crucial task that attracts significant attention from both industry and academia. Accurate detection of the failed component(s) and equally importantly the failure spread path(s) are critical to take corrective actions (in time) to restore a malfunctioning system and improve its design. In this thesis, we focus on multiple failure detection that relaxes the simplifying assumption of a single component failure, at the time of inspection, which is difficult to justify for many real world problems that involve fault-tolerant systems with little opportunity of maintenance during their operation. We also aim to relax the commonly used perfect information assumption (accurately detecting all the symptoms) and consider the cases where only a (random) subset of possible symptoms can be successfully detected, due to possible failures in the sensors as well. To address this urgent yet challenging problem we introduce a novel approach that uses graph theory concepts to model the diagnosis problem with an Integer programming formulation and devise a branch-and-cut algorithm to solve it efficiently. Extensive numerical experiments on realistic problem instances attests 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.

ÖZETÇE

Yayılan Hataların Bulunduğu ve Sistem Bilgisinin Eksik Olduğu Durumlarda Çoklu Arıza Tespiti Problemi için bir Dal Kesi Yöntemi
Kaan Pekel
Endüstri Mühendisliği, Yüksek Lisans
5 Ocak 2020

Karmaşık sistemlerde arıza tespiti, hem endüstride hem de akademide büyük ilgi gören önemli bir problemdir. Arızalı bileşenlerin doğru bir şekilde tespiti ve aynı derecede önemlisi, arızanın yayılma yollarının ortaya çıkarılması, zamanında doğru önlemler alarak sistemi tamir etmek ve sistemin tasarımını iyileştirmek için kritik öneme sahiptir. Bu çalışmada, operasyon sırasında çok az bakım fırsatına sahip, arızaya dayanıklı sistemleri içeren birçok gerçek dünya problemini göz önüne alarak, tek bir parça arızasının basitleştirici varsayımını gevşeterek çoklu arıza tespitine odaklanıyoruz. Ayrıca, yaygın olarak kullanılmakta olan sistem durumu hakkında mükemmel bilgiye sahip olma varsayımını gevşetmeyi ve sensörlerde olası arızalar nedeniyle, olası semptomların sadece (rastgele) bir alt kümesinin ortaya çıkabileceği durumları dikkate almayı amaçlıyoruz. Bu önemli ancak zorlu problemi grafik teorisi konseptini kullanarak geliştirdiğimiz bir tamsayılı programlama formülasyonu ile modelliyor ve bu modeli verimli bir şekilde çözmebilmek için bir dal kesi algoritması öneriyoruz. Gerçekçi problem örnekleri üzerinde yaptığımız kapsamlı sayısal deneyler, hem hesaplama verimliliği hem de tahmin doğruluğu açısından, literatürdeki en son teknolojiye kıyasla yaklaşımımızın üstün performansını ortaya koymaktadır.

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my advisor Dr. Barış Yıldız for his invaluable guidance and expertise. I appreciate all of his lasting patience and never-ending positivity throughout my study. Without his persistent encouragement and endless support, it would have been really impossible to complete this research.

I also would like to thank to Prof. Metin Türkay and Asst. Prof. Hande Küçükaydın for taking part in my thesis committee and sharing their precious time and expertise for my dissertation defense.

I wish to acknowledge the support and love of my family, my mother Gönül, my father Fahri, and my sister Fulya. They were with me from beginning to end and they always encouraged me to do what I love.

I would like to express special thanks to Müge for being present all the time and making me smile even in my most stressful moments. Her support was invaluable. I feel really lucky to have her in my life.

I also wish to thank my friends, Çağkan and Tezcan for the great working atmosphere we set at countless nights. Working beside you pushed my limits even further. I would also like to thank my friend Talha for the early times we worked together and decided to take a master's degree. I started to learn Java along with him without realizing how important it would be in this study.

I am also grateful to my graduate colleagues, Begüm and Hakan for co-working with me in course projects, sharing their experiences and making my graduate life enjoyable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

MFD	Multiple Fault Diagnosis
SC	Set Covering
wSC	Weighted Set Cover
BC	Branch-and-cut
IP	Integer Programming
IPT	Integer Programming Tree
BN	Bayesian Network
MWPT	Minimum Weight Plausible-Tree
MCT	Maximum Coverage Tree
TP	True Positive
FP	False Positive
FN	False Negative

Chapter 1

INTRODUCTION

Most of the technologies that make up our modern lives are large and complex systems in which several components work interdependently. As a matter of course, these components can fail for any reason at any time, whether the system is mechanic or biologic. 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 considerably increasing attention both in the application and research domains due to the obvious practical motivation and interesting theoretical properties of the problem [Ding et al., 2011]. When considering fault-tolerant systems (equipped with several back-up mechanisms or redundant components) with little or no opportunity of maintenance during their operations (e.g., aircraft, reactors, power grids), the simplifying assumption, at most, a single fault in the system between consecutive maintenance episodes is not realistic [Shakeri et al., 2000a]. As such, *multiple fault diagnosis* (MFD) problem emerges as a practically important and theoretically challenging extension of the single fault diagnosis problem, which is the main focus of this study.

In many real-world systems, there is not a unique mapping between the possible faults and the observed symptoms. Usually, several symptoms are common among various faults. As a result, it is possible to provide many alternative explanations (faults) for a given symptom set, which makes MFD a quite challenging problem because the number of combinations to consider grows exponentially with the number of faults [Vedam and Venkatasubramanian, 1997]. Moreover, the system information may not be perfect as only a subset of the (known) symptoms may surface (or get successfully detected), which considerably complicates the diagnosis problem. Although, there are various studies in the literature that address the combinatorial nature of the diagnosis problem in MFD, to the best of knowledge our study is the first one to consider the case of imperfect state information.

Besides the relation between fault-symptom pairs, the inter-dependent relation among the system components is also important for the MFD problem. 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]. However, not all failures (in all systems) spread so fast, i.e. the rate of propagation may be several orders of magnitude slower than the response time of the observer [Tu et al., 2003]. The spread of chronic illnesses in biological systems (e.g., diabetes causing kidney failure, vessel damages, heart problems or cataracts) and malfunctions in the mechanical systems that 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 of such "lazy spread" of failures. Even though our suggested methodology can also work in the case of cascading failures, in this study our focus is on the MFD problems with the lazily spreading failures.

Despite the importance of spreading failure diagnosis, there are not enough studies in the literature that consider the inter-dependencies between the system components and suggest efficient methods to handle realistic size problem instances. Moreover, the solutions are generally unable to explain how a particular set of faults occurred or what was the spread path that resulted in the given set of failed components, which can obviously provide very useful information to improve system design to eliminate such failures in the future. Focusing on this significant 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) given a symptom set, even when the state information of the system is not perfect (the symptom set is not complete). For that purpose, we introduce a novel approach that uses graph theory concepts to model the diagnosis problem with an Integer programming (IP) formulation and suggest an efficient branch-and-cut algorithm to solve it. In particular, the contributions of this study can be summarized as follows.

- We suggest a novel approach to solve MFD, which can effectively consider 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). Detecting this failure path for each failed component is not only helpful to increase diagnostic accuracy but also critical to improve system design to eliminate future failures (or restrict the spread).

- We conduct extensive numerical experiments that are generated to represent a wide range of real-world systems. Our experiments show that for the considered instances:

  - Our methodology achieves a superior performance against the state of the art in the literature, both in computational efficiency and prediction accuracy, especially when there are non-negligible interactions between the system components, as is the case with many real-world settings.
  - Our approach can provide high accuracy explanations with missing system information. For the problem instances we consider in our numerical experiments, even only a small fraction (i.e., less than 25%) of the symptoms show up (successfully detected), the accuracy of our branch-and-cut algorithm stays almost the same with the perfect information case, where all the respective symptoms for the failed components are successfully detected.

The remainder of the thesis is organized as follows. In Chapter 2, we review the related literature by focusing on multiple fault diagnosis methodologies with applications in various fields such as chemical plants, electronic circuits, telecommunication networks and biologic systems. In Chapter 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 Chapter 4, we present our performance measure metrics and interpret the results of the computational study. In Chapter 5, we extend the research and propose an extension for MFD in the case of wrong system information about the component-symptom associations. Finally, in Chapter 6, we conclude with some final remarks and a discussion of future work.

Chapter 2

LITERATURE REVIEW

In this chapter, we will review the studies that suggest efficient ways to solve the diagnosis problem dealing with multiple failures.

MFD falls in the broad class of fault diagnosis (detection) problems. For the vast literature on the topic we refer the reader to comprehensive reviews by [Venkatasubramanian et al., 2003], [Hwang et al., 2009] and more recently by [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 diagnosis problem.

A typical MFD problem consists of multiple failures that emerge in a system and several symptoms that arise as a result of these faults. The most commonly used approaches to address MFD include statistical methods (Bayes classifier, decision trees), approximation methods(polynomial classifiers), density-based methods (geometrical classifiers) and artificial intelligence such as artificial neural networks and fuzzy classifiers ([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 their work provides a theoretical foundation for failure diagnosis from first principles. Later on, [de Kleer et al., 1992] analyze the concept of diagnosis in detail and explores the notions of implicate/implicant and prime implicate/implicant in the context of MFD.

Failure spread is a very common phenomenon in complex electronic systems such as networks or circuits and MFD is a widely researched subject in electrical engineering field, since in this type of systems, spread of a failure is inevitable. MFD has been an active research area especially for analog and digital circuits. There are several studies in the literature that use effect-cause analysis ([Abramovici and

Breuer, 1980]), multilayer perceptrons ([Maidon et al., 1997], [Ogg et al., 1998]) and many more ([Boppana et al., 1999], [Veneris et al., 2002], [Wang et al., 2003], [Tadeusiewicz and Halgas, 2006], [Lin et al., 2007]).

Failure spreads are also very common in chemical plants. A problem in a unit (pipes, valves, pumps etc.) can cause another unit (reactors, heat exchangers etc.) to fail, which can be highly costly. Because of that, MFD is also a widely researched subject in chemical engineering fields. 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 to provide a solution for continuing the chemical process as quickly as possible. There have been a lot of work that uses methodologies such as signed digraph models ([Vedam and Venkatasubramanian, 1997], [Watanabe et al., 1994]), principal component analysis ([Raich and Çinar, 1995]), a combination of signed digraph and dynamic partial least squares ([Lee et al., 2004]) and artificial neural networks ([Venkatasubramanian and Chan, 1989]). Most of these works test their methodologies on Tennessee Eastman Process. Most of these methods benefit from data-driven approach and causal connectivity of fault-symptom pairs, and the failure interactions are not considered in these studies. With an aim to address this gap, [Chiang et al., 2015] propose a modified distance/causal dependency algorithm to solve MFD with spreading failures. They incorporate the propagation paths of the failures to present a more realistic schema for the failure spread mechanism. The authors consider four types of multiple faults: induced fault, independent multiple faults, masked multiple faults, and dependent faults.

Multiple fault diagnosis takes the form of multiple disease (disorder) diagnosis in the medical field. There have been a lot of studies and applications that suggest efficient ways to diagnose the patients correctly. [Szolovits and Pauker, 1978] investigate the benefits and drawbacks of categorical decision making and probabilistic reasoning algorithms in multiple diseases by comparing four expert systems. As a result, they propose to use both of these to optimize their benefits. [Heckerman, 1990] use causal probabilistic models while making an assumption that the diseases are marginally independent from each other. They present an algorithm which is called Quickscore to compute the posterior probabilities of each disease given a set of observed findings. Case-based reasoning (CBR) algorithms are used widely to mimic the diagnosis process of a physician. CBRs can be described as systems that diagnose the patients based on the information gathered from the previous patients that have similar symptoms and diagnosed correctly. We refer reader to [Macura and Macura, 1997] for the applications and opportunities of CBR in healthcare systems. [Wu, 1989], [Wu, 1990] and [Wu, 1991] use a procedure of symptom decomposition and clustering for diagnosing multiple disorders. Instead of considering each symptom at their own and find their related diseases, they group the observable symptoms in possible clusters (as an intermediate step), where each cluster of symptoms indicate only one disease. This procedure increases the performance of the method compared to the candidate generation type of procedures. [Suojanen et al., 2001] proposes a new method for diagnosing multiple diseases using causal probabilistic models. They apply their method for the diagnosis of neuromuscular disorders. In their approach, they consider to examine the cases using multiple phases, first by assuming there is one disease at a time and then increase the assumed disease count one by one. They consider diagnosis as a classification procedure whose objective is to assign the cases to predefined diseases based on the acquired symptoms. They also compare the disease diagnosis with fault diagnosis in the mechanical systems and share the common attributes of them. The first attribute they have in common is that the prior probabilities of each disease (or a failed mechanic component) are low, since at any time most of the people are healthy (most of the systems are undamaged). Secondly, the organs inside a human body can also cause other organs to fail as in the mechanic systems and therefore the presence of multiple failures are quite frequent. The third attribute is that both of the diseases and faults have common symptoms that are shared with so many other diseases and faults as well. Fourth, the failure process goes generally in one direction by unification, i.e. a disease or fault do not counteract a prior symptom that is already present in the system. Each disease brings its symptoms to the system. Finally, in both cases the diagnosis is made under uncertainty, i.e. not all the symptoms necessarily show up in the system in case of failure. In our approach, we mention all of these five attributes and show

how our algorithm can perform in such situations.

[Yu et al., 2003] suggest three algorithms for diagnosis in the Quick Medical Reference-Decision Theoretic Network (QMR-DT), namely a lagrangian relaxation algorithm, a primal heuristic algorithm and an approximate belief revision algorithm that can handle over 1000 symptoms at a time. [Hsu and Ho, 2004] proposes a framework that is a combination of case-based reasoning, neural networks, fuzzy theory and decision theory. The specification made by the patients are retrieved as input and these inputs are used to select the most useful candidate cases from a historical data. Using the features of these previous cases, a decision tree is constructed to reveal the most possible diseases. [Bayati et al., 2015] consider the multiple disease prediction from a different angle and they try to diagnose a patient correctly while reducing the number of biocheckers taken from the patient (test results, blood samples, etc.), which can be highly costly. They propose a statistical data-driven method to maximize the prediction accuracy while minimizing the number of biocheckers. Their solution use a combination of group dimensionality reduction and multi-task learning.

Dynamic MFD in large systems, when the test outcomes are unreliable and imperfect, is studied in various studies. [Shakeri et al., 1998] and [Shakeri et al., 2000b] proposes several test sequencing algorithms to find a solution for obtaining the posterior probabilities due to computational complexity in the case of multiple failures. They assume independent failure states, since dependent failures increases the complexity even further. Instead of using a diagnostic tree, they suggest to use a diagnostic directed graph (digraph) to overcome the computational burden. [Tu et al., 2003] proposes two efficient algorithms for MFD in large graph-based systems to obtain the most plausible fault set. To reduce the computational complexity of multiple failures, they present a heuristic algorithm based on Lagrangian relaxation and sub-gradient optimization. [Ligeza and Kościelny, 2008] investigates the drawbacks of classical binary diagnostic matrix of failures and their associated symptoms. They use algebraic and rule-based models and they propose a new formulation for the diagnostic matrix that takes into account the co-occurring symptoms and complementary ones. [Singh et al., 2009] and [Ruan et al., 2009] and [Kodali et al., 2013] suggest a dynamic programming formulation when it is highly possible to get false or missed alarms due to unreliable sensors and the test outcomes are delayed. They use the Viterbi algorithm and the Max-Sum algorithm to find the most possible hidden failure states.

Bayesian Networks (BN) are successfully used over the decades for all type of diagnosis methodologies. We refer the reader to [Cai et al., 2017] for a broad review on how they are utilized as a data-driven approach using historical data by backward analysis for MFD. In their work, they present a bibliographical review on the use of BNs in many areas (with a focus on engineering systems) in the last decades. Due to their success, and natural fit to the problem context, BN methodologies are considered as a benchmark in many studies to evaluate the performances of the suggested approaches. One such example is [Kandula et al., 2005], one of the closest studies to our work, where authors investigate the MFD in internet protocol (IP) networks and present a tool for root cause analysis of faults. To establish the efficiency of their methods, authors test their results by comparing them with Bayesian classification methods and minimum set cover algorithm, which we discuss next.

From the modeling perspective, our study is closely related with the classical minimum set covering (SC) problem. SC is one of the oldest and most studied optimization problems in the literature and it is used in many applications such as minimal speech corpus design ([Chevelu et al., 2008]), anesthesia performance ([Sawa and Ohno-Machado, 2001]), IP networks ([Bejerano and Rastogi, 2006]), vehicle routing ([Beasley, 1987]), airline scheduling ([Beasley and Jørnsten, 1992]) and many more. For a more broad review, we refer interested reader to [Christofides and Korman, 1975] and [Caprara et al., 2000], which put comprehensive surveys of the set covering algorithms.

For diagnostic expert systems, using the general model of SC is first proposed by [Reggia et al., 1983] and [Reggia et al., 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 emerged in the system. They propose a general model that consists of two conflicting goals. Firstly, the subset of disorders should be able to 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, but they assume that these disorders are independent of each other. Following these studies, [Peng and Reggia, 1987a] and [Peng and Reggia, 1987b] propose a model that combines the Bayesian classification to set covering theory. They suggest a formulation that uses both the structural knowledge (faults, symptoms and associations between them) and the probabilistic knowledge (prior probabilities of faults and strength of the fault-symptom associations). To rank the plausible explanations and find the most likely one, they use the probabilities of causal links between the fault-symptom pairs. Later on, [Peng and Reggia, 1989] proposes a competition-based connectionist model for the same problem where multiple simultaneous disorders increases the computational complexity. However, their formulation does not guarantee to find the global optimum, i.e. the most plausible explanation among the others.

Most of the aforementioned studies disregard the failure interactions (spread) between the system components. Our study differs from the general set covering models studied in combinatorial optimization literature as well as the ones 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 component failures are independent from each other. To take such interdependencies into account, we consider the probabilistic relations to find the most likely subset of faults that can cover the symptom set. 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 novel formulation that includes additional constraints (exponentially many) for the classical set covering formulation. To solve this challenging extension of the already difficult (NP-Hard) SC problem [Garey and Johnson, 2002], we propose an efficient branch-and-cut algorithm that can solve realistic problem instances. In that perspective, our work is also related with 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].

Graph based models are also used to model influence spread in the social networks. In such studies, the objective is to maximize the spread by minimizing the subset of seed nodes. [Gaye et al., 2015] and [Gaye et al., 2017] suggest an independent cascade model, which extract the information of acyclic spanning graphs of a network. The method uses centrality measures to discover the best plausible nodes in the system. [Wilder et al., 2018] suggest an exploratory influence maximization algorithm which queries the individual network nodes and the node-links with an objective of locating a subset of seeds that is similarly influential as the global optimum. For an extensive review about the topic, we refer reader to [Chen et al., 2009], [Nguyen and Zheng, 2012] and [Kempe et al., 2015].

Chapter 3

MATHEMATICAL MODEL

In this chapter, we provide the mathematical model of our formulation. We will first define the MFD problem and present the notation used in the paper. We will finish the chapter by discussing the details of our solution approach.

3.1 Problem Definition and Notation

In this section, we provide definitions and notation pertinent throughout the thesis. Additional definitions and notation will be listed on a need basis.

We consider a complex system with a set of components denoted by C. At any time, a component $i \in C$ may either fail by its own (spontaneously), or it may fail due to 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 abnormal behaviors (symptoms) among a known set of symptoms M_i. The component-symptom associations are represented by the collection $\mathcal{M} = \{M_i : i \in C\}$. For the notational convenience, for a set $S \subseteq C$ we define $M(S) = \cup_{i \in S} M_i$. 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 $\langle C, P, \mathcal{M} \rangle$. When a set of components $S \subseteq C$ fail, due to imperfect state information, a random (possibly a proper) subset M^+ of the plausible symptoms $M(S)$ is observed.

A rooted tree formed by a subset of components C is called a *failure-chain*.

Table 3.1: Outline of Notation

Notation		Description
C	:	Set of components
$P(i)$	:	Spontaneous failure probability of a component $i \in C$
$P(i,j)$	:	Failure spread probability from component $i \in C$ to $j \in C \setminus \{i\}$
M_i	:	Set of symptoms associated with the failure of a component $i \in C$
$\mathcal{M}$	:	Collection of component-symptom associations; $\mathcal{M} = \{M_i : i \in C\}$
M	:	Set of all symptoms; $M = \cup_{i \in C} M_i$
$M(S)$	:	Set of symptoms associated with a set $S \subseteq C$; $M(S) = \cup_{i \in S} M_i$
$C(m)$	:	Set of components that can generate symptom $m \in M$
M^+	:	Set of observed symptoms
$P(\phi)$	:	Probability of a failure-chain, $P(\phi) = P(c_1)\prod_{\ell=2}^{n} P(c_\ell, c_{\bar{\ell}})$
$C(\epsilon)$	:	Set of components of an explanation; $C(\epsilon) = \cup_{k=1}^{n} \phi_k$
$M(C(\epsilon))$	:	Set of symptoms that can cover the explanation ϵ
$P(\epsilon)$	:	Likelihood of an explanation; $P(\epsilon) = \prod_{\phi \in \epsilon} P(\phi)$
G	:	Spread-graph representing the network
N	:	Set of nodes in G
s	:	Special node (root of the spontaneous failures of components)
A	:	Set of arcs in G
w_{ij}	:	Weight of the arc $(i,j) \in A$
$C(T)$	:	Set of components in a rooted tree T

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

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

$C(\epsilon) = \{c : c \in \phi_k,$ for some $k = 1, \ldots, n\}$. A symptom m is called to be covered by an explanation ϵ, if $m \in M(C(\epsilon))$. For a given set of observed symptoms $M^+ \subset M$, an explanation ϵ is called as a *plausible-explanation*, if all the symptoms in M^+ are covered by ϵ, 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 example, 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 3.1, i.e., $M_1 = \{a, b, c\}, M_2 = \{b, c, e, g\}, M_3 = \{c, d, e, g\}, M_4 = \{f, g, h\}, M_5 = \{g, h, i\}$.
- Observed symptoms: $M^+ = \{c, g, h\}$ (marked green in Figure 3.1).

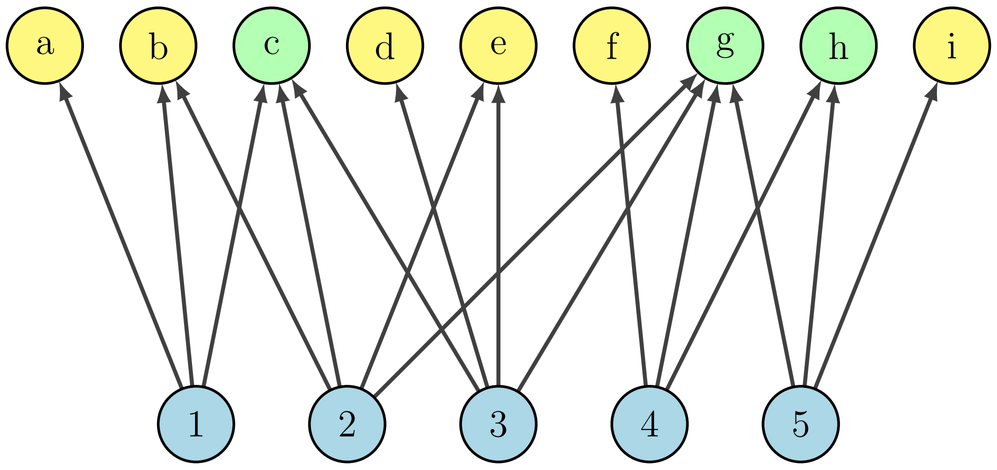
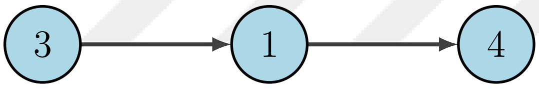
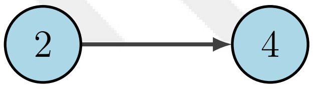


Figure 3.1: Component-symptom associations.

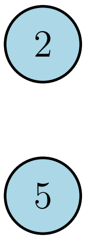
For this small example, one can build several plausible-explanations for the observed symptoms $M^+ = \{c, g, h\}$. In Figure 3.2, we present four such plausible-explanations ($\epsilon_1, \epsilon_2, \epsilon_3$ and ϵ_4) with different number of failure chains and number of nodes in each chain. The 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. As we see in this small 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 more accurate estimates for the failed components. As we can observe in this example, among other explanations ϵ_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 MFD as follows.



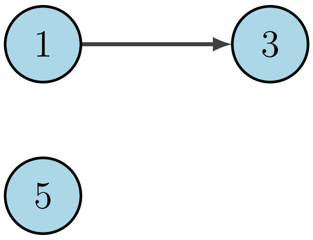
(a) $P(\epsilon_1) = P(3)P(3,1)P(1,4) = 2.5E^{-5}$



(b) $P(\epsilon_2) = P(2)P(2,4) = 2E^{-3}$



(c) $P(\epsilon_3) = P(2)P(5) = 1E^{-4}$



(d) $P(\epsilon_4) = P(1)P(1,3)P(5) = 1E^{-5}$

Figure 3.2: Some plausible-explanations for $M^+ = \{c, g, h\}$.

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

In the next section, we will present our solution approach to MFD problem.

3.2 Solution Approach

In this section, we present the integer programming (IP) formulation we develop to model MFD and the branch-and-cut algorithm we devise to solve it.

3.2.1 IP Formulation

We model MFD 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, one can construct the spread-graph as shown in Figure 3.3.

Note that each explanation presented in Figure 3.2 can be represented by a tree (rooted at s) in the spread graph as shown in Figure 3.4, 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.*

Definition 2. *For a given MFD instance $\langle C, P, \mathcal{M}, M^+ \rangle$, a tree T in the respective spread graph G, rooted in s, is called a plausible-tree if the components included in it $C(T)$ 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 derived for a MFD instance $\langle C, P, \mathcal{M}, M^+ \rangle$. Then the set of failure-chains (sub*

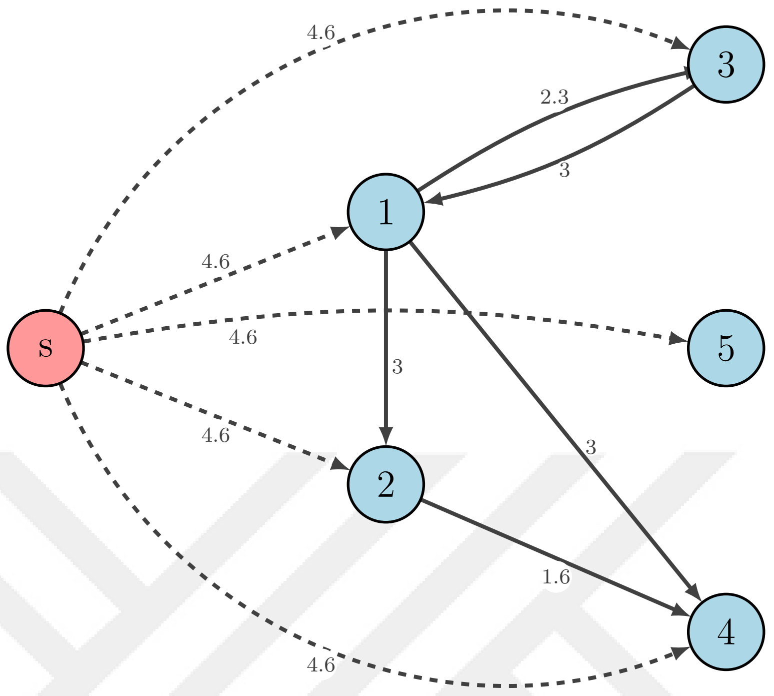
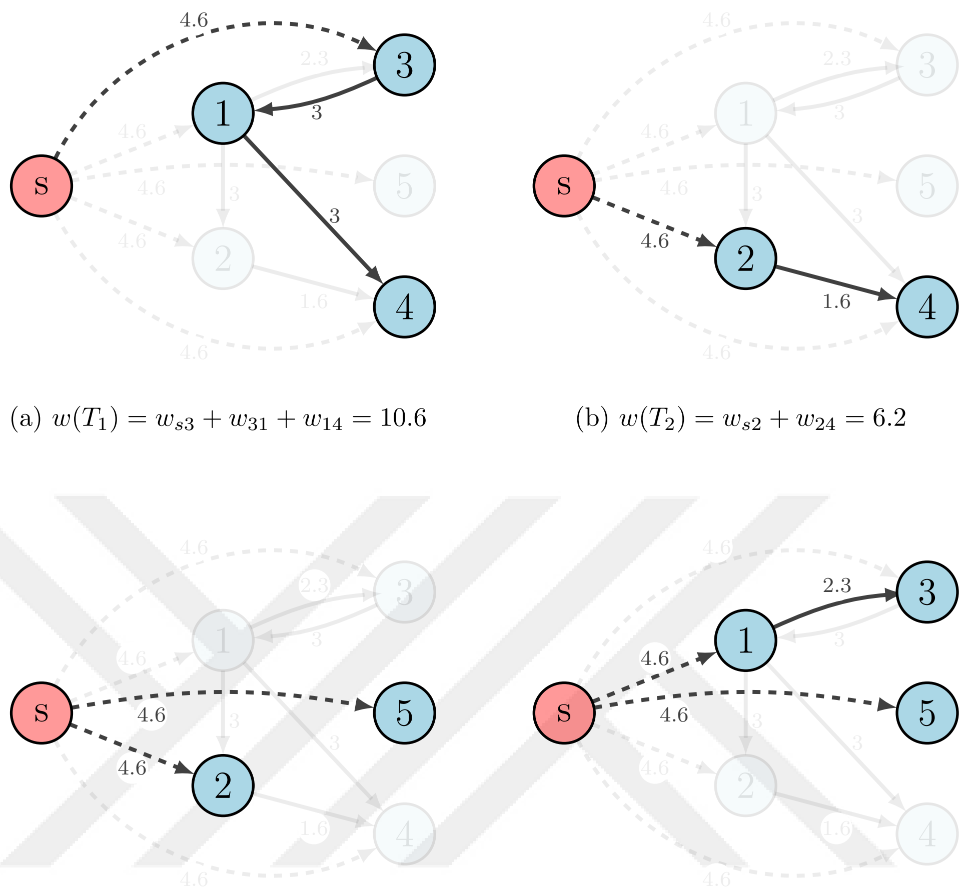


Figure 3.3: Spread-graph for the small problem instance.

trees) obtained from T^* by removing the root node s, forms a plausible-explanation ϵ^*, which is an optimal solution for the given MFD problem.

Proof. Clearly, ϵ^* is a plausible-explanation simply due to T^* being a plausible-tree by definition. So to complete the proof, we just need to show that ϵ^* 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 MFD instance by finding the



(a) $w(T_1) = w_{s3} + w_{31} + w_{14} = 10.6$ (b) $w(T_2) = w_{s2} + w_{24} = 6.2$

(c) $w(T_3) = w_{s2} + w_{s5} = 9.2$ (d) $w(T_4) = w_{s1} + w_{13} + w_{s5} = 11.5$

Figure 3.4: Some plausible-trees for $M^+ = \{c, g, h\}$.

smallest weight plausible-tree in the respective spread-graph. Equipped with this result, we now present our IP formulation that aims to find the minimum weight plausible-tree (MWPT) in a given spread-graph, hence, it can be used to solve a given MFD 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, all of the problem parameters are listed in Table 3.1. The formal definition of the IP formulation IPT is as the following.

IPT:

$$\min \sum_{(i,j)\in A} w_{ij}x_{ij} \tag{3.1}$$

$$\text{s.t.} \sum_{i\in C(m)} y_i \geq 1 \qquad \forall m \in M^+, \tag{3.2}$$

$$\sum_{(i,j)\in A} x_{ij} = y_j \qquad \forall j \in C, \tag{3.3}$$

$$\sum_{(i,j)\in A(S)} x_{ij} \leq \sum_{i\in S\setminus\{k\}} y_i \qquad \forall S \subseteq N, \forall k \in S, \tag{3.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A, \tag{3.5}$$

$$y_i \in \{0,1\} \qquad \forall i \in N \tag{3.6}$$

The objective is to minimize the total weight of the solution (a plausible-tree rooted tree in s). Constraints (3.2) 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 (3.3) indicate a necessary condition that if a component is included in the solution than 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 (does not include cycles), for that purpose we add the cycle elimination constraints (3.4) which are proposed by [Lee et al., 1996] to formulate Steiner-Tree problems and can completely characterize the respective spanning-tree poly-tope 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 in (3.5) and (3.6).

Note that in this formulation, the set of constraints (3.4) may get very large in number as the number of components in the system $|C|$ grows. So, it is not practical to solve the problem directly. To overcome this difficulty, we suggest a branch-and-cut approach and add additional constraints iteratively. In the following section, we define the details of our branch-and-cut algorithm.

3.2.2 Branch and Cut Algorithm

In this section we discuss the details of our branch-and-cut algorithm (BC) to solve IPT iteratively.

Branch-and-cut can be considered as an approach of combinatorial optimization for solving integer programming problems. The method uses two techniques namely branch-and-bound (an enumeration tree of candidate solutions) and cutting planes. To solve the IP problem, the relaxations of the original problem are considered and to obtain an integer solution new cutting planes are added repeatedly. The cutting planes are found by solving the separation problem, which finds the feasible and infeasible solutions.

For our formulation, at each iteration, we solve IPT with a subset of the cycle elimination constraints (3.4) and solve a separation problem (which separates feasible and infeasible solutions) 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 $IPTr$ which does not include any of the cycle-cancellation Constraints (3.4). Through out the branch-and-bound algorithm, we consider the following procedure to detect violated inequalities for a given solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$.

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 $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ 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 $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. We call $\bar{G} = (\bar{N}, \bar{A})$ as the induced-sub-graph of G for the given solution and run a connectivity check on $\bar{G}$ to detect the connected components $\mathcal{K} = \{K_1, \ldots, K_\ell\}$ 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 (3.7) are violated to add into the model. The pseudo code for the CC algorithm is provided in Algorithm 1.

$$\sum_{(i,j)\in A(K)} x_{ij} \leq \sum_{i\in K\setminus\{k\}} y_i \qquad \forall K \in \mathcal{K}, \forall k \in K. \tag{3.7}$$

```
Algorithm 1: Connectivity Check
    input  : $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$
    output: $\langle \mathcal{V} \rangle$
1   Initialize the set of violated inequalities $\mathcal{V} = \emptyset$;
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       for $k \in K$ do
7           if $\sum_{(i,j)\in A(K)} x_{ij} > \sum_{i \in K\setminus\{k\}} y_i$ then
8               Add the inequality $\sum_{(i,j)\in A(K)} x_{ij} \leq \sum_{i \in K\setminus\{k\}} y_i$ to $\mathcal{V}$;
9   return $\mathcal{V}$
```

Note that when the solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ 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 basic search algorithm the run time complexity of the CC algorithm is linear in the number of arcs in the induced graph, which is usually much smaller than that of original separation-graph. As we discuss in the next section in more detail, having an efficient algorithm to solve the separation problem contributes greatly to superior computational efficiency of the BC algorithm, compared to the state of the art methodologies in the literature.

As an alternative to CC, separation problem can be also solved by solving a maximum flow problem on a bipartite whose node set is composed of the binary decision variables of IPT. For the details of such an approach we refer reader to [Lee et al., 1996]. However, our preliminary studies have indicated that, for the problem instances we considered in our computational studies, the best computational performance is achieved when the separation problem is solved only for the integer solutions by using the CC algorithm.

Chapter 4

COMPUTATIONAL STUDIES

In this chapter, we present the details of our extensive numerical experiments we conducted to test the computational efficiency and the prediction accuracy (detecting failed components correctly) of the BC algorithm against the state of art in the literature. In particular, we consider the Shrink algorithm [Kandula et al., 2005], Bayes Classifier, implemented using the Bayes Net Toolbox [Murphy, 2001], and the classical minimum cardinality set cover approach [Reggia et al., 1983], which we generalize by our BC algorithm. We also tested a weighted Set Covering (wSC) extension for 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 minimizes the cardinality of the failed component set, whereas wSC minimizes the total weight of the components predicted to be failed to cover the given set of symptoms, where components 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 Appendix A.

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 to represent various real-world settings with differing fault interactions and fault-symptom associations. In our experiments, we 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, which makes 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 μ number of symptoms from M, where μ is random variable that is uniformly distributed in $[\underline{\mu}, \overline{\mu}]$. In our computational experiments, we set $\underline{\mu}$ and $\overline{\mu}$ to 10 and 20, respectively. The pseudocode of how we generate the collection of component - symptom associations are presented in Algorithm 2.

Algorithm 2: Component-Symptom Association Generation

```
input  : $\langle C, M, \underline{\mu}, \overline{\mu} \rangle$
output: $\langle \mathcal{M} \rangle$
1 foreach $i \in C$ do
2     Draw a random integer $\mu$ between $\underline{\mu}$ and $\overline{\mu}$;
3     Set $j = 0$;
4     while $j < \mu$ do
5         Draw a random symptom $m \in M$;
6         if $m \notin M_i$ then
7             $m \in M_i$;
8             $j = j + 1$;
9 return $\mathcal{M}$
```

We assume that all the faults can occur spontaneously. We assign a random probability $P(i), \forall i \in C$, where $P(i)$ is a random variable that is drawn from a Pareto Distribution ([Arnold, 2015]) with a support $(0, \overline{P}]$ such that 80% of malfunctions in the system are expected to be initiated by 20% of the faults. In our instances we consider the case with $\overline{P} = 5E^{-4}$. As it will be more clear when we describe how do we simulate the generation of faults, we consider such a small values for $\overline{P}$ to obtain problem instances where a relatively small fraction of the components are to be faulty at the time of inspection.

In our instances, 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 possible interactions, which we choose from $\{0.01, 0.025, 0.05, 0.075\}$. 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 $\overline{\Omega} \in \{1.25E^{-2}, 2.5E^{-2}, 6.25E^{-2}, 12.5E^{-2}, 18.75E^{-2}\}$ 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, it is important to note that as the 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 the average) one needs to consider $\overline{\Omega}$ values that are much higher than $\overline{P}$, to obtain problem instances where the component failures happen due to a small number of failure-chains (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 $[0.1, \overline{\lambda}]$, where we consider the problem instances with $\overline{\lambda} \in \{0.4, 0.55, 0.7, 0.85, 1\}$ in our experiments.

After the system parameters fixed, we randomly generate the faults with a simple simulation 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 performed to detect the components that have failed during this time interval. The details of our fault generation method are presented in Algorithm 3. As expected, some simulations return no faults at the end and we simply ignore them. The number of faults that emerged in the system when the simulation ends is denoted by p in our analyses.

Once we determine the failed components $\bar{C} = F \setminus \{s\}$ we generate the set M^+

Algorithm 3: Fault Generation

```
   input  : $\langle G, t \rangle$
   output: $\langle F \rangle$
 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                 $\bar{F} = \bar{F} \cup \{j\}$;
 9     $F = F \cup \bar{F}$;
10 return $F$
```

by considering the symptom expression probabilities by following the steps indicated in Algorithm 4.

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 running Windows 7. As mentioned before we considered (tested) minimum cardinality set cover (SC), minimum weight set cover (wSC), BayesNet and Shrink algorithms as benchmarks to assess the performance of BC algorithm we develop in this study. We implemented the BC, SC and wSC algorithms in Java using CPLEX 12.9. 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, 2013]), 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].

Algorithm 4: Symptom Generation

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

4.3 Experimental design and analysis of the results

In this section, we will first provide the parameters for each set of experiments we conducted. Then, we will analyze the results of these experiments one at a time.

4.3.1 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.
- Considering multiple explanations to increase diagnosis accuracy

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 following parameter values $|M| = 150$, $d = 0.05$, $\overline{\Omega} = 0.125$ and $\overline{\lambda} = 0.7$.

The first set of experiments E1 aims to investigate the impact of the number of plausible-explanations on the computational complexity and diagnosis accuracy for 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 that are related with a given symptom decreases in $|M|$.

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 four different levels for the density parameter as $d \in \{0.01, 0.025, 0.05, 0.075\}$.

The third set of experiments E3 aims to investigate the impact of spread probabilities on the performances of the considered algorithms. E3 contains five different maximum spread probability values as $\overline{\Omega} \in \{0.0125, 0.025, 0.0625, 0.125, 0.1875\}$.

Finally, 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. For that matter, E4 contains problem instances with five different levels for the symptom expression probabilities with $\overline{\lambda} \in \{0.4, 0.55, 0.7, 0.85, 1\}$.

For each configuration specified in E1, E2, E3 and E4, we generate $40,000$ instances to obtain enough observations (i.e., more than 20) for various number of faults ($p \in \{1, 2, \ldots, 6\}$) in the system at the time of inspection. Here it worths mentioning that a significant portion of the instances (around 30,000 among the attempted 40,000) generated by our failure spread simulation (Algorithm 3) contains no faults, which we simply disregard in our experiments.

4.3.2 Analysis of Results

In this subsection, we present the results of our numerical experiments and discuss their practical implications. Before proceeding with the results, we first want to explain how we 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 components that are mistakenly labeled as non-faulty.

Using TP, FN and FP, we calculate recall ($\frac{TP}{TP+FN}$) and precision ($\frac{TP}{TP+FP}$) ([Powers and Ailab, 2011]). Recall of the solution shows the fraction of correctly predicted faults to real faults, whereas precision shows the fraction of correctly predicted faults to total predicted faults, respectively. By using recall and precision as a measure, we can consider two important dimensions of the accuracy. Both of them are important in our setting, since we focus on the cases where most of the components actually are non-faulty and simply identifying all the components as faulty would give a very high accuracy score without much of a 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 widely used in the literature for evaluating the performance of classification algorithms with a single metric ([Powers and Ailab, 2011]). To get a feeling about the relevance of using F1 score to evaluate diagnosis performance in our setting we present some simple examples in Table 4.1, which indicates the F1 scores for different diagnoses, considering a problem instance with three failed components: 1, 2 and 3 among the five other.

Table 4.1: An example of how we measure each algorithm when $F = \{1, 2, 3\}$

Guess	**TP**	**FN**	**FP**	**Recall**	**Precision**	**F1 Score**
1, 2	2	1	0	0.67	1	0.8
1, 2, 3	3	0	0	1	1	1
1, 2, 3, 4	3	0	1	1	0.75	0.86
1, 2, 5	2	1	1	0.67	0.67	0.67
2, 4, 5	1	2	2	0.33	0.33	0.33

In the sequence, we will discuss how the F1 scores vary for 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.

Figure 4.1 shows the run-time of the considered algorithms for E1 instances. We report the run times of the algorithms for various p (actual number of failures in the system) values. In each graph, the run time of considered algorithms is illustrated 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, (generally) more faults should be considered to explain them all. In both cases, the number of alternative explanations grows high and diagnosis problem gets harder, as we discuss next in more detail. The y-axis is on the logarithmic scale to be able to display the differences between the algorithms in a larger range. 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.

Figure 4.1 presents interesting results about the computational efficiencies of the studied algorithms, which present important insights about their applicability in different settings. As expected, for the Shrink algorithm, we observe that the cardi-

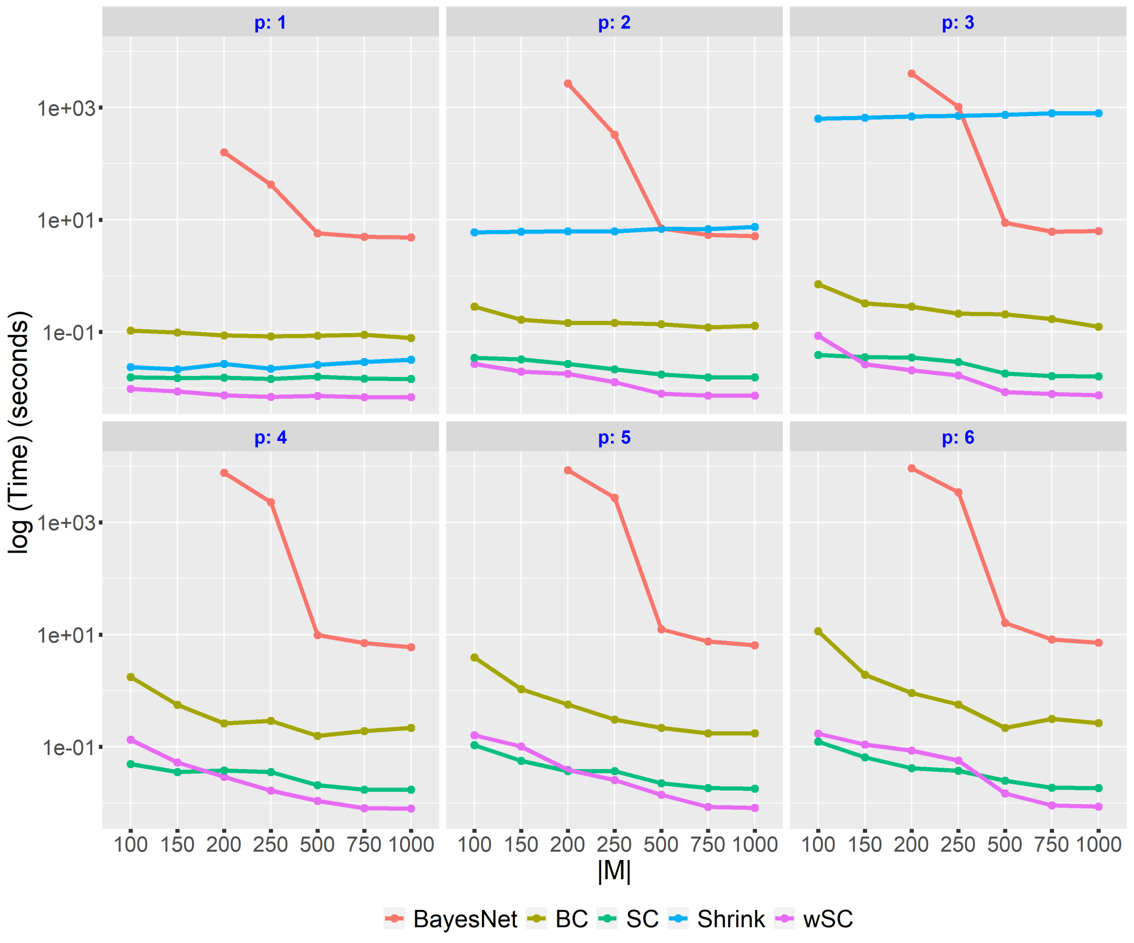


Figure 4.1: Run time results for the E1 instances.

nality of the symptom set has a much smaller impact on the run time compared to the number of actual faults in the system. Being basically an enumeration algorithm with a run time complexity of $O(|M|^p)$, Shrink cannot provide solutions (within a time limit of one hour) for instances with more than 3 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) to complicate the diagnosis problem. As mentioned above, for $|M|$ values that are less than 200 the BayesNet cannot provide a solution within one hour. On the other hand, in almost all the cases the set covering family (BC, SC, and wSC) has 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 has similar runtimes. However, it is quite 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 exponential number of constraints). We attribute this result mostly to the high efficiency of the separation procedure CC (Algorithm 1), which is a polynomial-time ($O(|A|)$) algorithm.

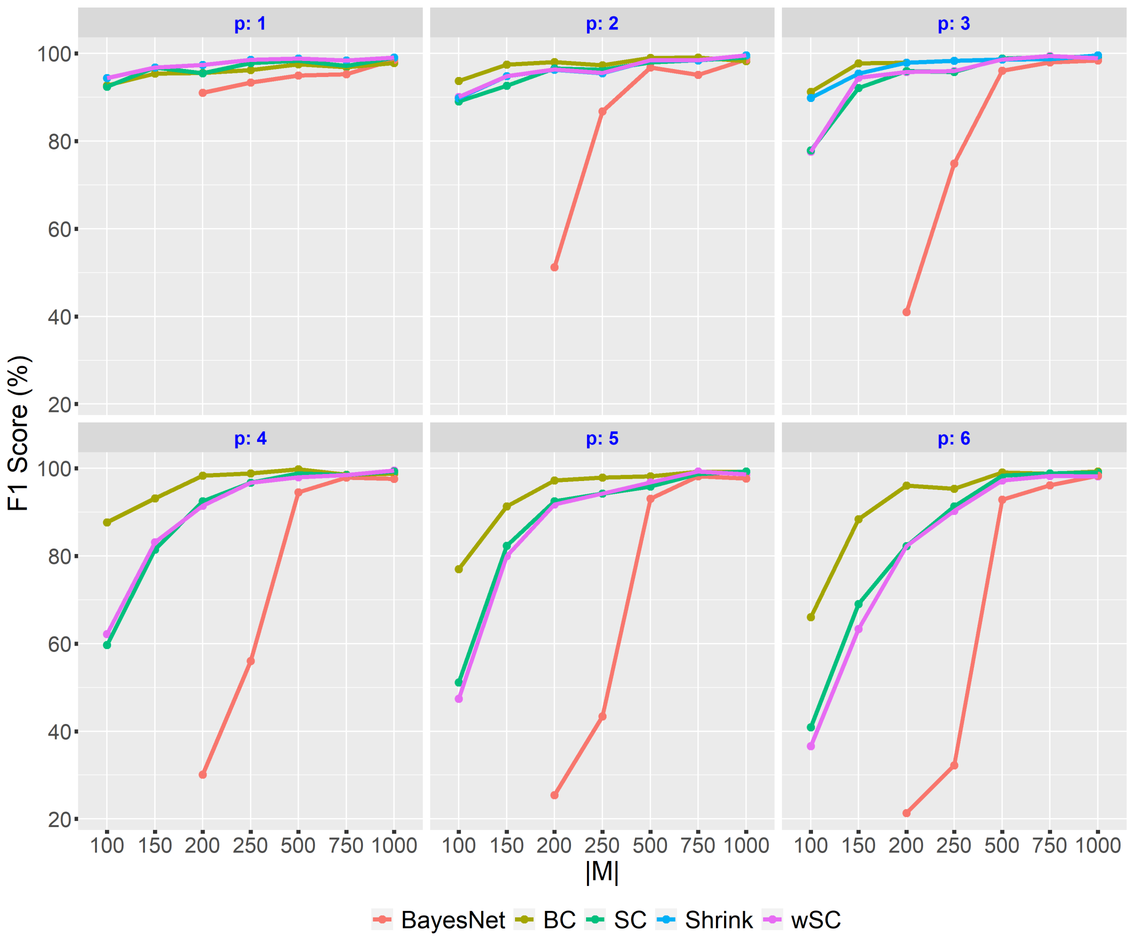


Figure 4.2: Diagnostic performance results (F1 scores) for the E1 instances.

The F1 Scores for the E1 experiments are given in Figure 4.2 (recall and precision results for E1 experiments are provided in Appendix B). As expected, E1 results show that due to the increasing number of alternative explanations, the diagnostic performances decrease as p increases or $|M|$ decreases. Considering the run time and diagnostic performances together (Figures 4.1 and 4.2), we can clearly 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 F measures are already over 90% for all considered algorithms. Although the BayesNet can provide solutions for instances with more than 3 failed components, its diagnostic performance is quite poor for $|M| < 250$. For example, we can see that the average F score of the BayesNet for $p = 4$ and $|M| = 200$ is around 30 % while 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 clearly outperforms SC and wSC, especially when $|M|$ is less than 250, indicating that it is worthwhile to incur the relatively small extra computational burden to solve BC.

Another important 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 in our E2 experiments. Figure 4.3 illustrates the F1 scores of considered algorithms for the problem instances we study in E2 (recall and precision results for E2 are presented in Appendix C). As expected, these results show that for small p the impact of interconnectivity is not very pronounced. In particular, when $p = 1$, the impact of d is negligible since there is no spread in the system. However, as p gets larger (i.e., $p \geq 3$), BC clearly outperforms the other alternatives as the interactions between the components become to play an important role to initiate chains of failures which BC is tailored to capture. Interestingly, we see that BC scores are much better even for the smallest d value of =0.01, which indicates that BC can be a better choice whenever there is a non-negligible number of failure interactions between the system components. We also observe a decline in the F1 score of BC as d increases, especially for higher p values. This is mainly due to the fact that with the higher d value the number of failure chains that BC needs to consider increases, which impacts the BC's performance adversely.

Note that while the parameter d controls the number of failure interactions between the system components. The intensity of those relations is controlled by $\bar{\Omega}$,

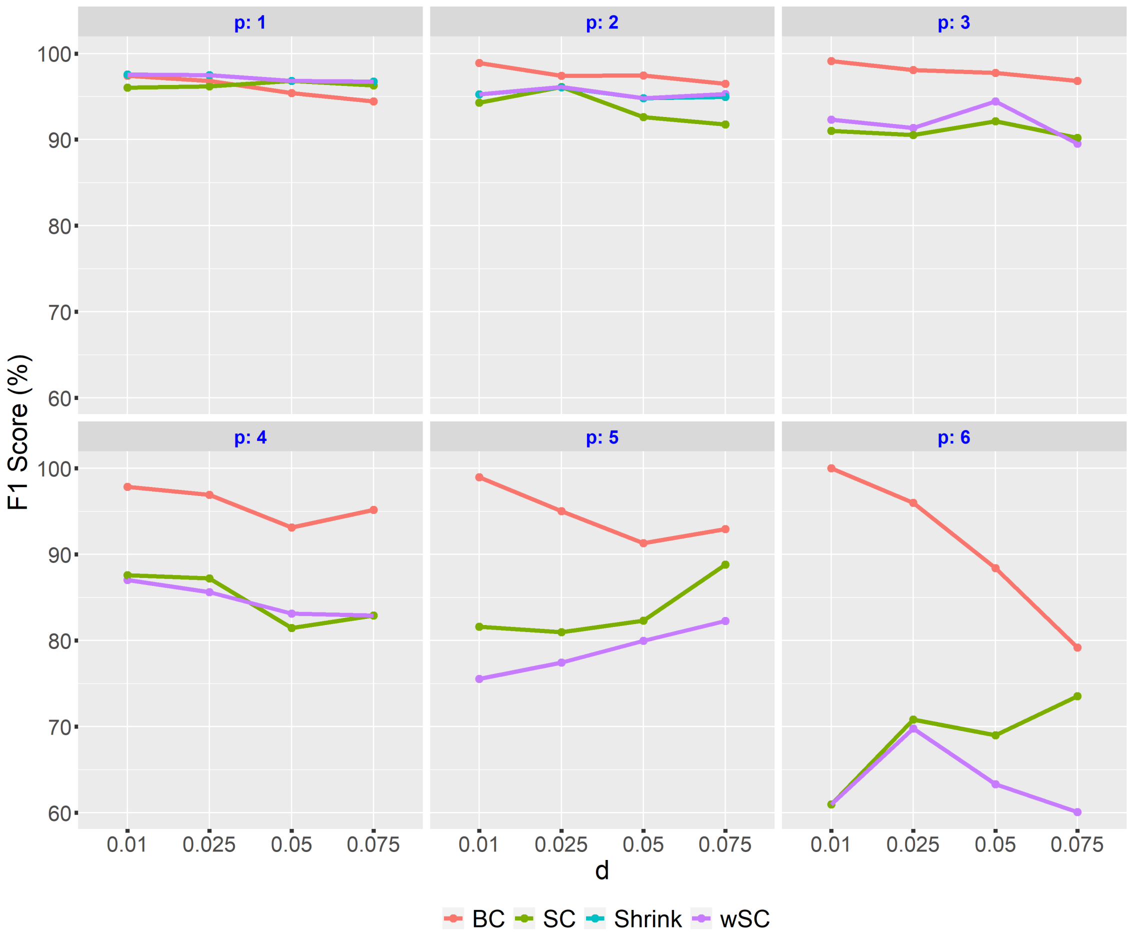


Figure 4.3: Diagnostic performance results (F1 scores) for the E2 instances.

which denotes the upper limit for the probability of a failure spread between two components and controls the proportion of spontaneous failures versus failures due to spread. The results for E3 experiments are given in Figure 4.4 for various levels of $\overline{\Omega}$ (recall and precision results for E3 are presented in Appendix D). As expected, for $p = 1$, i.e. only one component that fails spontaneously, the F1 scores of the algorithms are almost the same. However, for larger values of Ω, we see a slight decrease in BC. 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. As an interesting result, we also observe that changing $\bar{\Omega}$ does not have a significant impact on the BC's performance. Providing an interesting insight, these results indicate if a failure in one component has some non-negligible potential to initiate failure in some other component, taking into account this relation highly improves the diagnosis accuracy.

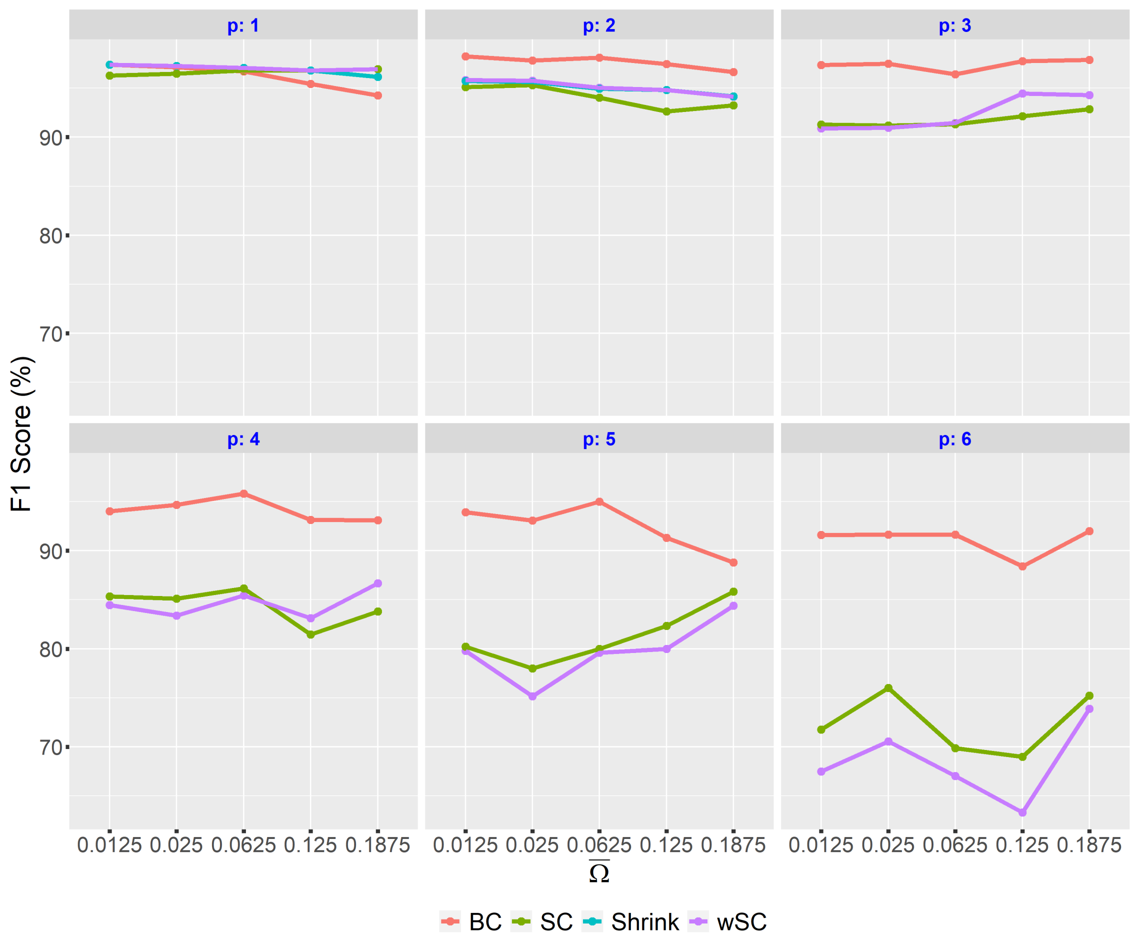


Figure 4.4: Diagnostic performance results (F1 scores) for the E3 instances.

Fourth, we analyze the F1 scores of the problem instances in E4, presented in Figure 4.5 (recall and precision results are presented in Appendix E), to understand the impact of imperfect state information on the diagnosis performance. Recall that in E4 we study different levels for the problem parameter $\overline{\lambda}$, which controls the symptom expression (successful detection) probabilities. As we see in Figure 4.5, missing information about the system state (symptoms) impacts the diagnostic performance very significantly. In general, all of the algorithms perform better as $\overline{\lambda}$ increases. This is expectable since the algorithms use more information to predict the most likely explanation as more symptoms show up (detected) in the system. In addition, as p increases and the problem becomes more challenging, the impact of missing information on the F1 scores becomes more pronounced. We also observe that BC outperforms the other algorithms with a large margin for lower values of $\overline{\lambda}$ and higher values of p, where the diagnostic problem gets more challenging. It is interesting to see that when $\lambda = 0.4$, 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 informations.

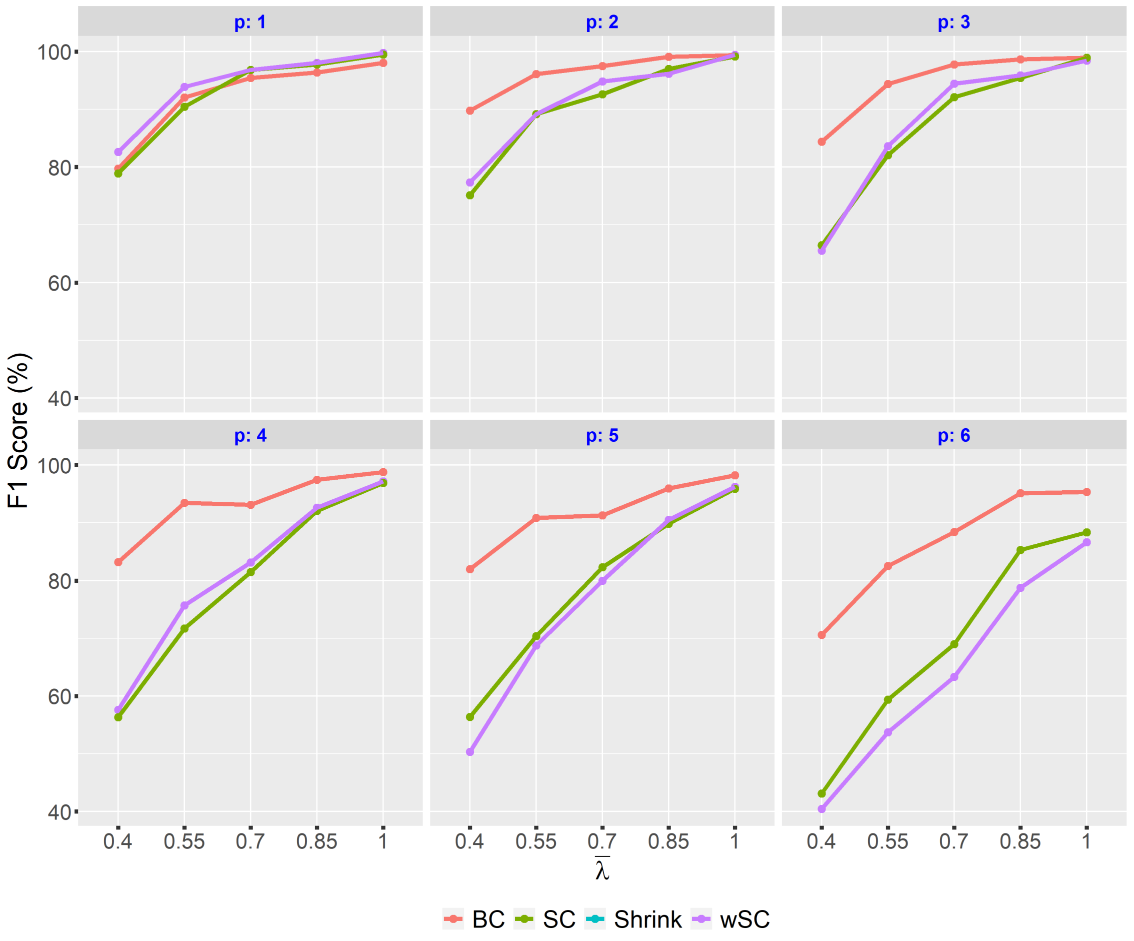


Figure 4.5: Diagnostic performance results (F1 scores) for the E4 instances.

Lastly, we want to analyze how much the predictions accuracy can be improved by considering alternative explanations one can easily obtain with the BC algorithm that builds on a mathematical formulation. In particular, we are also interested to see how the diagnostic performances would increase by considering second and third best plausible explanations, instead of the optimal solution. For that purpose, we solved the base case instances and considered the top three explanations (respective to their likelihood score). To obtain these solutions, we solve BC with an additional constraint that forces to find an alternative explanation that is different than the previous solutions.

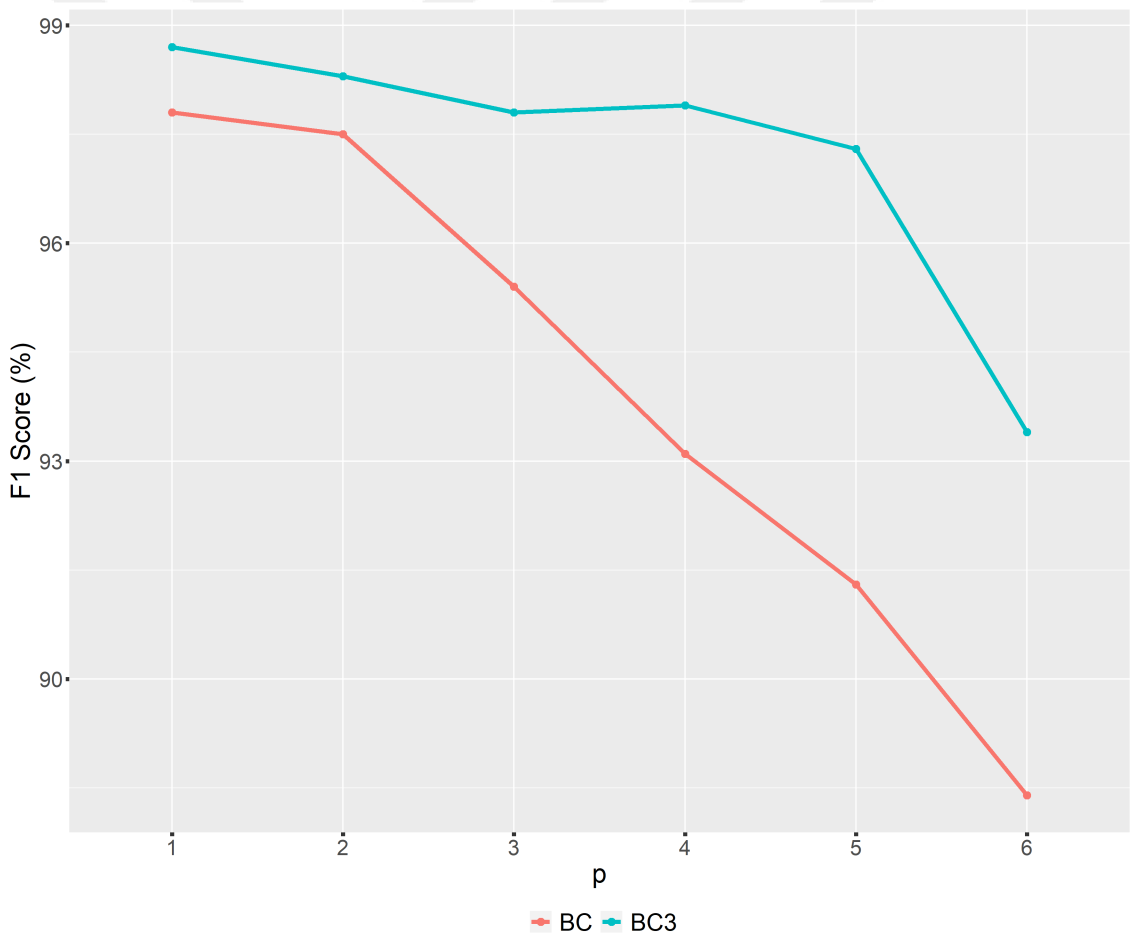


Figure 4.6: F1 scores for the E5 instances.

We show the results of these experiments in Figure 4.6, where BC3 indicates the highest F1 score for the top three explanations we obtain with BC (recall and precision results are presented in Appendix F).

As expected, we see that BC3 has a higher performance than BC for each p value, especially when $p \geq 3$. Since the computational effort which is needed to obtain such additional explanations (less than a second in these set of experiments) is minimal, increasing the diagnostic accuracy by considering multiple explanations emerges as a viable option to achieve higher diagnostic accuracy.

Chapter 5

SOLVING MFD WITH INACCURATE FAILURE-SYMPTOM ASSOCIATIONS

In this chapter, we extend our solution approach to solve the MFD instances with the inaccurate failure-symptom associations, which requires to modify "set covering" concept to find correct explanations as we discuss in detail next.

5.1 Finding explanations with the maximum symptom coverage

In the MWPT problem, one of the main assumptions is that we know each symptom for all of the components in the collection $\mathcal{M}$ for certain. Thus, when a symptom m shows up in the system, our formulation puts at least one fault that is associated with m in the plausible-explanation. However, in real-world systems, the information about these associations may not be perfect. There may be cases that an expert thinks there is an association between a component and a symptom, but in reality, there is not (wrong system information). The exact opposite might be true as well. The expert may think that there is no association between a component and a symptom, but in reality, there is a relation (missing system information). In such cases, the plausible-explanation does not necessarily cover each symptom, since these symptoms may be unreliable. We relax this coverage constraint and suggest a modified formulation that aims to find the tree (in the spread graph) that can cover the highest number of symptoms without exceeding a given weight limit.

To account for the missing/wrong system information, we introduce a new problem parameter $\theta \in [0, 1]$, which indicates the level of accuracy (i.e., expert knowledge) to detect component-symptom associations. When $\theta = 1$, the expert knowledge is perfect and all the information between component-symptom pairs correctly identified. Note that this is the case BC finds the most plausible-explanation by covering all the symptom set M^+. For $\theta < 1$, our new formulation aims to cover the highest number of symptoms in M^+. In addition we would like to introduce a budget parameter B which specifies the maximum weight that a plausible tree can get.

We determine the weight limit B using the formula $B = b \cdot \overline{w}_{ij} + \overline{w}_{0j}$, where $\overline{w}_{ij} = -log(\frac{\overline{\Omega}}{2})$, $\overline{w}_{0j} = -log(\frac{\overline{P}}{2})$, b is an integer between $[0, n]$, and $\overline{w}_{ij}$ and $\overline{w}_{0j}$ are the average weights of spread failures and spontaneous failures, respectively. We initiate b to 0 and increase it one by one until θ percent of the symptoms in M^+ are covered. The budget parameter basically is to limit the number of faults in the plausible explanation. When budget is low, only a few symptoms are covered by the set of faults. By increasing it iteratively, the algorithm can cover more of the symptoms by introducing new faults. A low budget favors precision whereas a high budget favors recall. By covering θ percent of the symptoms in M^+, we find a balance between these two to obtain a better F1 Score.

Lastly, we define the symptom inclusion variable z_m, which takes the value of one if a symptom $m \in M^+$ is covered by the plausible-tree and zero otherwise, and build the maximum coverage formulation MCT as follows.

MCT:

$$
\begin{aligned}
\max \quad & \sum_{m \in M^+} z_m & & \quad (5.1)\\
\text{s.t.} \quad & \sum_{i \in C(m)} y_i \geq z_m & \forall m \in M^+, & \quad (5.2)\\
& \sum_{(i,j) \in A} w_{ij} x_{ij} \leq B & \forall (i,j) \in A, & \quad (5.3)\\
& \sum_{(i,j) \in A} x_{ij} = y_j & \forall j \in C, & \quad (5.4)\\
& \sum_{(i,j) \in A(S)} x_{ij} \leq \sum_{i \in S \setminus \{k\}} y_i & \forall S \subseteq N, \forall k \in S, & \quad (5.5)\\
& x_{ij} \in \{0,1\} & \forall (i,j) \in A, & \quad (5.6)\\
& y_i \in \{0,1\} & \forall i \in N & \quad (5.7)\\
& z_m \in \{0,1\} & \forall m \in M^+ & \quad (5.8)
\end{aligned}
$$

The objective (5.1) is to maximize the number of symptoms to be covered. Constraints (5.2) make sure that there is at least one fault included in the tree for each covered symptom. Constraints (5.3) ensure that the total weight of the graph is less than or equal to the budget we specified earlier. The constraints (5.4), (5.5), (5.6) and (5.7) are the same as in IPT formulation. Lastly, Constraints (5.6)- (5.8) describe variable domains.

The right hand side of the constraints (5.3) are updated at each iteration until θM of the observable symptoms are covered.

As in the MWPT formulation, the constraints (5.5) may get very large in number as the number of components in the system increases. So, suggest to use a branch-and-cut algorithm $\overline{BC}$, which basically follows the same steps discussed in the Subsection 3.2.2, to solve MCT formulation.

Algorithm 5: Missing System Information

```
input : $\theta$, $\mathcal{M}$
output: $\mathcal{M}^\theta$
1 Initialize $\mathcal{M}^\theta = \mathcal{M}$;
2 foreach $i \in C$ do
3     foreach $m \in M$ do
4         if $m \in M_i^\theta$ then
5             Draw a uniform random variable $r$ between 0 and 1;
6             if $r > \theta$ then
7                 $m \notin M_i^\theta$
8 return $\mathcal{M}_\theta$
```

5.2 Instance Generation and Experimental Design

In this section, we present how we generate the instances for the cases containing θ portion wrong or missing system information. We conducted two additional groups of experiments, E6 and E7, that are built on the base case setting we defined in Subsection 4.3.1.

In the set of experiments E6, we assume that θ portion of our information about the component symptom associations is missing, i.e. we do not know some of the relations, but associations that we know are hundred percent true. In particular, we remove θ portion of the known associations and generate the collection $\mathcal{M}^\theta$; $\mathcal{M}^\theta = \{M_i^\theta : i \in C\}$, which is our guess about the component-symptom associations. The pseudo-code of the algorithm is given in Algorithm 5.

In the set of experiments E7, we assume that θ portion of our information about the component symptom associations is wrong, i.e we do not know some of the existing relations and some of the relations that we aware of are wrong. We generate the collection $\mathcal{M}^\theta$; $\mathcal{M}^\theta = \{M_i^\theta : i \in C\}$, that is our guess about the component-symptom associations. The pseudo-code of the algorithm is given in Algorithm 6.

Algorithm 6: Wrong System Information

```
    input  : $\theta$, $\mathcal{M}$
    output: $\mathcal{M}^\theta$
 1  Initialize $\mathcal{M}^\theta = \mathcal{M}$;
 2  foreach $i \in C$ do
 3      foreach $m \in M$ do
 4          if $m \in M_i^\theta$ then
 5              Draw a uniform random variable $r$ between 0 and 1;
 6              if $r > \theta$ then
 7                  $m \notin M_i^\theta$
 8          if $m \notin M_i^\theta$ then
 9              Draw a uniform random variable $r$ between 0 and 1;
10              if $r > \theta$ then
11                  $m \in M_i^\theta$
12  return $\mathcal{M}_\theta$
```

For both of the experiments, we tested our formulation for six levels of θ; $\theta \in \{0.75, 0.8, 0.85, 0.9, 0.95, 1\}$. We include $\theta = 1$ for comparison purposes and it shows the original BC results.

5.3 Analysis of the results

In this section, we will give the results for the E6 and E7 instances. We use the F1 Score for the evaluation of the results as explained in Subsection 4.3.2.

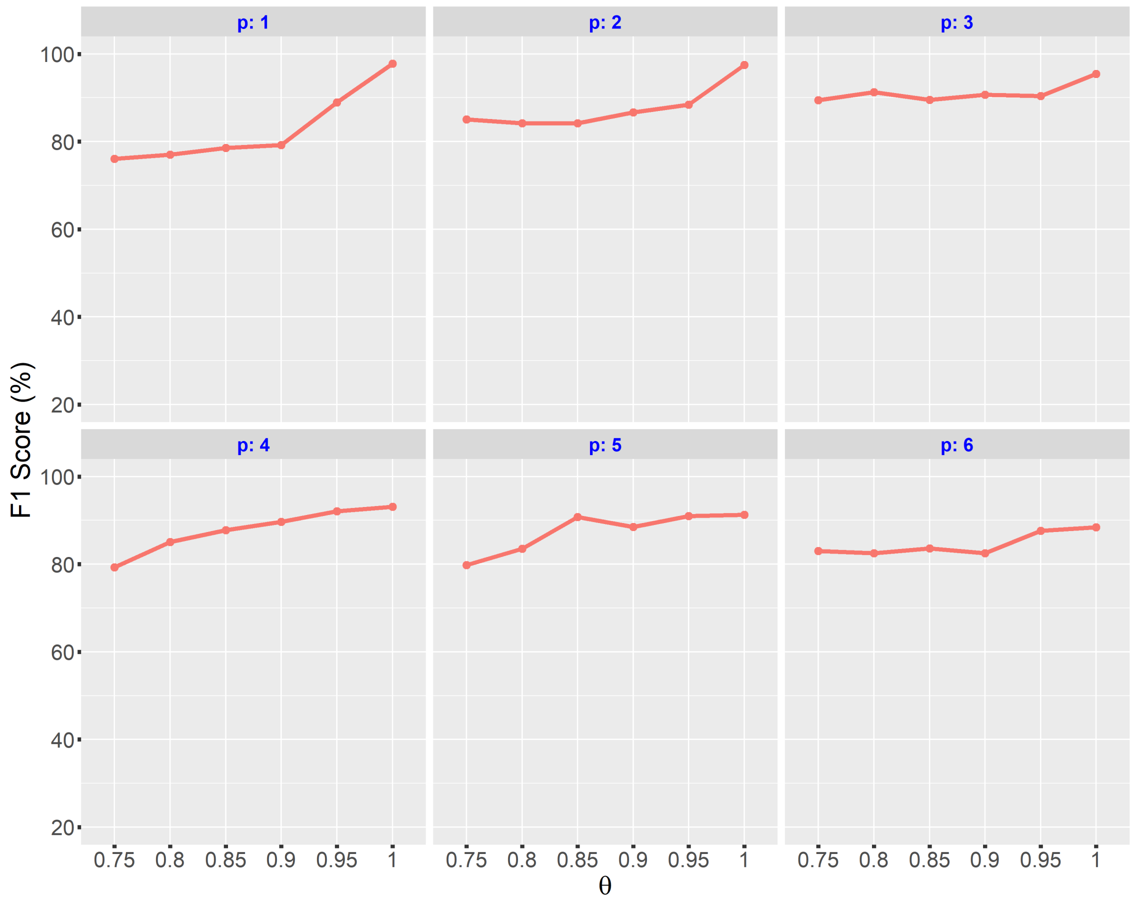


Figure 5.1: F1 scores for the E6 instances.

The F1 scores for E6 instances are given in Figure 5.1 (recall and precision results are provided in Appendix G). As expected, there is a drop in accuracy with the decrease of θ. However, we do not see a significant drop which indicates the benefit of the suggested approach. We see that even 25% of the component-symptom associations are missing, the F measures are above 80% for $p \geq 2$. In real systems where missing information is highly expected, $\overline{BC}$ is a viable alternative to solve the diagnosis problem.

Figure 5.1 shows the F1 scores for the E7 instances (recall and precision results are provided in Appendix G). We can see that the performance is drastically decreased compared to the E6 instances. When more than 10% of the component - symptom associations are wrong, the performance of the algorithm drops significantly. Considering the E6 and E7 together, we can conclude that $\overline{BC}$ is a good candidate only if the experts knowledge about the existing associations are mostly correct but the expert does not necessarily know all of the existing associations in the system.

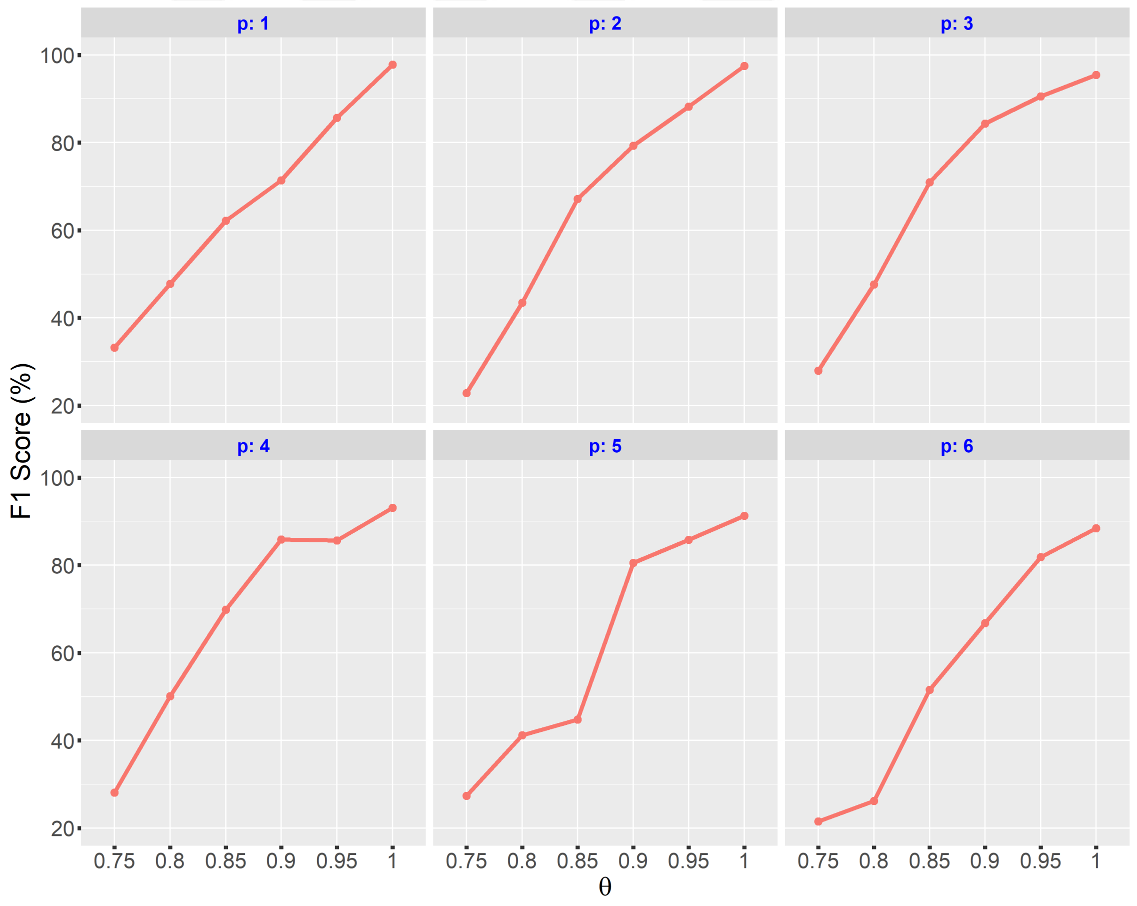


Figure 5.2: F1 scores for the E7 instances.

Chapter 6

CONCLUSION

In this thesis, we study the multiple fault diagnosis problem considering spreading failures and imperfect system state information. We propose a novel approach to address this urgent yet challenging diagnosis problem that extends the literature in several directions. Representing the failure interactions between the components through a directed weighted graph, we propose a novel integer programming formulation to model diagnosis problem for the spreading failures and use graph theoretical results to device 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. As indicated by the result of these experiments, the suggested methodology can provide accurate diagnoses (much better than the state-of-the-art algorithms in the literature) in a computationally efficient way, for all the set of experiments we consider in this study. 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 detection of associated symptoms for the faulty components.

Providing a significant example for the huge 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 not only for the researchers and practitioners who work on diagnosis problems but also for the operations research community in general. Our work essentially introduces a new extension for the classical set covering problem with interesting theoretical properties and practical applications.

As future research directions, building on the mathematical formulations we suggest for the diagnosis problem, several extensions can be studied to further improve diagnostic accuracy and applicability of the suggested approach. One such direction is to develop new mechanisms to consider the cases where some of the symptom fault associations are not correctly defined, i.e., some symptoms may be wrongly associated with some faults which require to modify the "set covering" concept, as the diagnostic accuracy in those cases can be improved by disregarding some symptoms. Devising fast heuristic approaches to detect plausible trees with high likelihood scores in the spread graph, instead solving the formulations IPT and MCT exactly, would also be of interest to be able to extend the applicability of the proposed solution approach for much larger problem instances.

BIBLIOGRAPHY

[Abramovici and Breuer, 1980] Abramovici and Breuer (1980). Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis. *IEEE Transactions on Computers*, C-29(6):451–460.

[Arnold, 2015] Arnold, B. C. (2015). *Pareto Distributions*. Chapman and Hall/CRC, 2nd edition.

[Bayati et al., 2015] Bayati, M., Bhaskar, S., and Montanari, A. (2015). A Low-Cost Method for Multiple Disease Prediction. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2015:329–338.

[Beasley, 1987] Beasley, J. (1987). An algorithm for set covering problem. *European Journal of Operational Research*, 31(1):85 – 93.

[Beasley and Jørnsten, 1992] Beasley, J. and Jørnsten, K. (1992). Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58(2):293 – 300. Practical Combinatorial Optimization.

[Bejerano and Rastogi, 2006] Bejerano, Y. and Rastogi, R. (2006). Robust monitoring of link delays and faults in ip networks. *Networking, IEEE/ACM Transactions on*, 14:1092 – 1103.

[Boppana et al., 1999] Boppana, V., Mukherjee, R., Jain, J., Fujita, M., and Bollineni, P. (1999). Multiple Error Diagnosis based on Xlists. *Proceedings - Design Automation Conference*, pages 660–665.

[Cai et al., 2017] Cai, B., Huang, L., and Xie, M. (2017). Bayesian Networks in Fault Diagnosis. *IEEE Transactions on Industrial Informatics*, 13(5):2227–2240.

[Caprara et al., 2000] Caprara, A., Toth, P., and Fischetti, M. (2000). Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–.

[Chen et al., 2010] Chen, S., Ljubić, I., and Raghavan, S. (2010). The regenerator location problem. *Networks: An International Journal*, 55(3):205–220.

[Chen et al., 2015] Chen, S., Ljubić, I., and Raghavan, S. (2015). The generalized regenerator location problem. *INFORMS Journal on Computing*, 27(2):204–220.

[Chen et al., 2009] Chen, W., Wang, Y., and Yang, S. (2009). Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 199–208, New York, NY, USA. Association for Computing Machinery.

[Chevelu et al., 2008] Chevelu, J., Barbot, N., Boëffard, O., and Delhay, A. (2008). Comparing set-covering strategies for optimal corpus design.

[Chiang et al., 2015] Chiang, L. H., Jiang, B., Zhu, X., Huang, D., and Braatz, R. D. (2015). Diagnosis of multiple and unknown faults using the causal map and multivariate statistics. *Journal of Process Control*, 28:27–39.

[Christofides and Korman, 1975] Christofides, N. and Korman, S. (1975). A computational survey of methods for the set covering problem. *Management Science*, 21(5):591–599.

[Crucitti et al., 2004] Crucitti, P., Latora, V., and Marchiori, M. (2004). Model for cascading failures in complex networks. *Phys. Rev. E*, 69:045104.

[de Kleer et al., 1992] de Kleer, J., Mackworth, A. K., and Reiter, R. (1992). Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222.

[de Kleer and Williams, 1987] de Kleer, J. and Williams, B. C. (1987). Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130.

[Ding et al., 2011] Ding, S. X., Zhang, P., Jeinsch, T., Ding, E., Engel, P., and Gui, W. (2011). A survey of the application of basic data-driven and model-based methods in process monitoring and fault diagnosis. *IFAC Proceedings Volumes*, 44(1):12380–12388.

[Dueñas-Osorio and Vemuru, 2009] Dueñas-Osorio, L. and Vemuru, S. M. (2009). Cascading failures in complex infrastructure systems. *Structural Safety*, 31(2):157 – 167. Risk Acceptance and Risk Communication.

[Edmonds, 2003] Edmonds, J. (2003). Submodular functions, matroids, and certain polyhedra. In *Combinatorial Optimization—Eureka, You Shrink!*, pages 11–26. Springer.

[Farahani et al., 2012] Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., and Goh, M. (2012). Covering problems in facility location: A review. *Computers & Industrial Engineering*, 62(1):368–407.

[Gao et al., 2015] Gao, Z., Cecati, C., and Ding, S. X. (2015). A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3757–3767.

[Garey and Johnson, 2002] Garey, M. R. and Johnson, D. S. (2002). *Computers and intractability*, volume 29. wh freeman New York.

[Gaye et al., 2015] Gaye, I., Mendy, G., Ouya, S., and Seck, D. (2015). Spanning graph for maximizing the influence spread in social networks. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1389–1394.

[Gaye et al., 2017] Gaye, I., Mendy, G., Ouya, S., and Seck, D. (2017). *An Approach to Maximize the Influence Spread in the Social Networks*, pages 207–228.

[Heckerman, 1990] Heckerman, D. (1990). A Tractable Inference Algorithm for Diagnosing Multiple Diseases. *Machine Intelligence and Pattern Recognition*, 10(C):163–171.

[Hsu and Ho, 2004] Hsu, C. C. and Ho, C. S. (2004). A new hybrid case-based architecture for medical diagnosis. *Information Sciences*, 166(1-4):231–247.

[Hwang et al., 2009] Hwang, I., Kim, S., Kim, Y., and Seah, C. E. (2009). A survey of fault detection, isolation, and reconfiguration methods. *IEEE transactions on control systems technology*, 18(3):636–653.

[Isermann, 2011] Isermann, R. (2011). *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer.

[Kandula et al., 2005] Kandula, S., Katabi, D., and Vasseur, J. P. (2005). Shrink: A Tool for Failure Diagnosis in IP networks. *Proceedings of ACM SIGCOMM 2005 Workshops: Conference on Computer Communications*, pages 173–178.

[Kempe et al., 2015] Kempe, D., Kleinberg, J., and Tardos, E. (2015). Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147.

[Kodali et al., 2013] Kodali, A., Singh, S., and Pattipati, K. (2013). Dynamic Set-Covering for Real-Time Multiple Fault Diagnosis with Delayed Test Outcomes. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 43(3):547–562.

[Lee et al., 2004] Lee, G., Han, C., and Yoon, E. S. (2004). Multiple-Fault Diagnosis of the Tennessee Eastman Process Based on System Decomposition and Dynamic PLS. *Industrial and Engineering Chemistry Research*, 43(25):8037–8048.

[Lee et al., 1996] Lee, Y., Chiu, S. Y., and Ryan, J. (1996). A branch and cut algorithm for a steiner tree-star problem. *INFORMS Journal on Computing*, 8(3):194–201.

[Ligeza and Kościelny, 2008] Ligeza, A. and Kościelny, J. M. I. (2008). A New Approach to Multiple Fault Diagnosis: A Combination of Diagnostic Matrices, Graphs, Algebraic and Rule-Based Models. The Case of Two-Layer Models. *International Journal of Applied Mathematics and Computer Science*, 18(4):465–476.

[Lin et al., 2007] Lin, Y. C., Lu, F., and Cheng, K. T. (2007). Multiple-Fault Diagnosis Based On Adaptive Diagnostic Test Pattern Generation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(5):932–942.

[Ljubić and Gollowitzer, 2013] Ljubić, I. and Gollowitzer, S. (2013). Layered graph approaches to the hop constrained connected facility location problem. *INFORMS Journal on Computing*, 25(2):256–270.

[Macura and Macura, 1997] Macura, R. T. and Macura, K. (1997). Case-based reasoning: opportunities and applications in health care. *Artificial Intelligence in Medicine*, 9(1):1–4.

[Maidon et al., 1997] Maidon, Y., Jervis, B. W., Dutton, N., and Lesage, S. (1997). Diagnosis of multifaults in analogue circuits using multilayer perceptrons. *IEE Proceedings: Circuits, Devices and Systems*, 144(3):149–154.

[Murphy, 2001] Murphy, K. (2001). The bayes net toolbox for matlab. *Computing science and statistics*, 33.

[Nguyen and Zheng, 2012] Nguyen, H. and Zheng, R. (2012). Influence spread in large-scale social networks – a belief propagation approach. In Flach, P. A., De Bie, T., and Cristianini, N., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 515–530, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Ogg et al., 1998] Ogg, S., Lesage, S., Jervis, B., Maidon, Y., and Zimmer, T. (1998). Multiple fault diagnosis in analogue circuits using time domain response features and multilayer perceptrons. *IEE Proceedings - Circuits, Devices and Systems*, 145(4):213.

[Peng and Reggia, 1986] Peng, Y. and Reggia, J. (1986). Plausibility of diagnostic hypotheses: The nature of simplicity. volume 1, pages 140–147.

[Peng and Reggia, 1987a] Peng, Y. and Reggia, J. A. (1987a). A Probabilistic Causal Model for Diagnostic Problem Solving-Part I: Integrating Symbolic Causal Inference with Numeric Probabilistic Inference. *IEEE Transactions on Systems, Man and Cybernetics*, 17(2):146–162.

[Peng and Reggia, 1987b] Peng, Y. and Reggia, J. A. (1987b). A Probabilistic Causal Model for Diagnostic Problem Solving Part II: Diagnostic Strategy. *IEEE Transactions on Systems, Man and Cybernetics*, 17(3):395–406.

[Peng and Reggia, 1989] Peng, Y. and Reggia, J. A. (1989). A Connectionist Model For Diagnostic Problem Solving. *IEEE Transactions on Systems, Man and Cybernetics*, 19(2):285–298.

[Powers and Ailab, 2011] Powers, D. and Ailab (2011). Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *J. Mach. Learn. Technol*, 2:2229–3981.

[R Core Team, 2013] R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Raich and Çinar, 1995] Raich, A. C. and Çinar, A. (1995). Multivariate statistical methods for monitoring continuous processes: assessment of discrimination power of disturbance models and diagnosis of multiple disturbances. *Chemometrics and Intelligent Laboratory Systems*, 30(1):37–48.

[Reggia et al., 1983] Reggia, J. A., Nau, D. S., and Wang, P. Y. (1983). Diagnostic expert systems based on a set covering model. *International Journal of Man-Machine Studies*, 19(5):437–460.

[Reggia et al., 1985] Reggia, J. A., Nau, D. S., and Wang, P. Y. (1985). A Formal Model of Diagnostic Inference. I. Problem Formulation and Decomposition. *Information Sciences*, 37(1-3):227–256.

[Reiter, 1987] Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95.

[Ruan et al., 2009] Ruan, S., Zhou, Y., Yu, F., Pattipati, K. R., Willett, P., and Patterson-Hine, A. (2009). Dynamic Multiple-Fault Diagnosis With Imperfect Tests. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 39(6):1224–1236.

[Sawa and Ohno-Machado, 2001] Sawa, T. and Ohno-Machado, L. (2001). Generation of dynamically configured check lists for intra-operative problems: Using a set covering algorithm. *Proceedings / AMIA ... Annual Symposium. AMIA Symposium*, 9:593–7.

[Shakeri et al., 1998] Shakeri, M., Pattipati, K. R., Raghavan, V., and Patterson-Hine, A. (1998). Optimal and Near-Optimal Algorithms for Multiple Fault Diagnosis with Unreliable Tests. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 28(3):431–440.

[Shakeri et al., 2000a] Shakeri, M., Raghavan, V., Pattipati, K. R., and Patterson-Hine, A. (2000a). Sequential testing algorithms for multiple fault diagnosis. *IEEE transactions on systems, man, and cybernetics-part a: systems and humans*, 30(1):1–14.

[Shakeri et al., 2000b] Shakeri, M., Raghavan, V., Pattipati, K. R., and Patterson-Hine, A. (2000b). Sequential Testing Algorithms for Multiple Fault Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 30(1):1–14.

[Singh et al., 2009] Singh, S., Kodali, A., Choi, K., Pattipati, K. R., Namburu, S. M., Sean, S. C., Prokhorov, D. V., and Qiao, L. (2009). Dynamic Multiple

Fault Diagnosis: Mathematical Formulations and Solution Techniques. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, 39(1):160–176.

[Suojanen et al., 2001] Suojanen, M., Andreassen, S., and Olesen, K. G. (2001). A Method for Diagnosing Multiple Diseases in MUNIN. *IEEE Transactions on Biomedical Engineering*, 48(5):522–532.

[Swamy and Kumar, 2004] Swamy, C. and Kumar, A. (2004). Primal–dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269.

[Szolovits and Pauker, 1978] Szolovits, P. and Pauker, S. G. (1978). Categorical and Probabilistic Reasoning in Medical Diagnosis. *Artificial Intelligence*, 11(1-2):115–144.

[Tadeusiewicz and Halgas, 2006] Tadeusiewicz, M. and Halgas, S. (2006). An algorithm for multiple fault diagnosis in analogue circuits. *International Journal of Circuit Theory and Applications*, 34(September 2006):607–615.

[Tu et al., 2003] Tu, F., Pattipati, K. R., Deb, S., and Malepati, V. N. (2003). Computationally Efficient Algorithms for Multiple Fault Diagnosis in Large Graph-based Systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 33(1):73–85.

[Vedam and Venkatasubramanian, 1997] Vedam, H. and Venkatasubramanian, V. (1997). Signed Digraph Based Multiple Fault Diagnosis. *Computers and Chemical Engineering*, 21(SUPPL.1):655–660.

[Veneris et al., 2002] Veneris, A., Liu, J. B., Amiri, M., and Abadir, M. S. (2002). Incremental Diagnosis and Correction of Multiple Faults and Errors. *Proceedings -Design, Automation and Test in Europe, DATE*, pages 716–721.

[Venkatasubramanian and Chan, 1989] Venkatasubramanian, V. and Chan, K.

(1989). A Neural Network Methodology for Process Fault Diagnosis. *AIChE Journal*, 35(12):1993–2002.

[Venkatasubramanian et al., 2003] Venkatasubramanian, V., Rengaswamy, R., Yin, K., and Kavuri, S. N. (2003). A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & chemical engineering*, 27(3):293–311.

[Wang et al., 2003] Wang, Z., Tsai, K. H., Marek-Sadowska, M., and Rajski, J. (2003). An Efficient and Effective Methodology on the Multiple Fault Diagnosis. *IEEE International Test Conference (TC)*, pages 329–338.

[Watanabe et al., 1994] Watanabe, K., Hirota, S., Hou, L., and Himmelblau, D. M. (1994). Diagnosis of Multiple Simultaneous Fault via Hierarchical Artificial Neural Networks. *AIChE Journal*, 40(5):839–848.

[Wilder et al., 2018] Wilder, B., Immorlica, N., Rice, E., and Tambe, M. (2018). Maximizing influence in an unknown social network.

[Wu, 1989] Wu, T. D. (1989). Symptom Clustering and Syndromic Knowledge in Diagnostic Problem Solving. *Proceedings - Annual Symposium on Computer Applications in Medical Care*, pages 45–49.

[Wu, 1990] Wu, T. D. (1990). Efficient Diagnosis of Multiple Symptom Disorders Based on a Symptom Clustering Approach. *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 357–364.

[Wu, 1991] Wu, T. D. (1991). A problem decomposition method for efficient diagnosis and interpretation of multiple disorders. *Computer Methods and Programs in Biomedicine*, 35(4):239–250.

[Yıldız and Karaşan, 2015] Yıldız, B. and Karaşan, O. E. (2015). Regenerator location problem and survivable extensions: A hub covering location perspective. *Transportation Research Part B: Methodological*, 71:32–55.

[Yıldız and Karaşan, 2017] Yıldız, B. and Karaşan, O. E. (2017). Regenerator location problem in flexible optical networks. *Operations Research*, 65(3):595–620.

[Yu et al., 2003] Yu, F., Tu, F., Tu, H., and Pattipati, K. (2003). Multiple Disease (fault) Diagnosis with Applications to the QMR-DT Problem. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2:1187–1192.

Appendix A

MATHEMATICAL FORMULATIONS OF SC AND WSC

The mathematical formulation of the SC algorithm is given below:

$$\min \sum_{i \in C} y_i \tag{A.1}$$

$$\text{s.t.} \sum_{i \in C(m)} y_i \geq 1 \qquad \forall m \in M^+, \tag{A.2}$$

$$y_i \in \{0, 1\} \qquad \forall i \in C \tag{A.3}$$

The mathematical formulation of the wSC algorithm is given below:

$$\min \sum_{i \in C} w_i y_i \tag{A.4}$$

$$\text{s.t.} \sum_{i \in C(m)} y_i \geq 1 \qquad \forall m \in M^+, \tag{A.5}$$

$$y_i \in \{0, 1\} \qquad \forall i \in C \tag{A.6}$$

Appendix B

RECALL AND PRECISION GRAPHS FOR E1

In Figure B.1 we show the recall and in Figure B.2 we show the precision of the considered algorithms for E1.

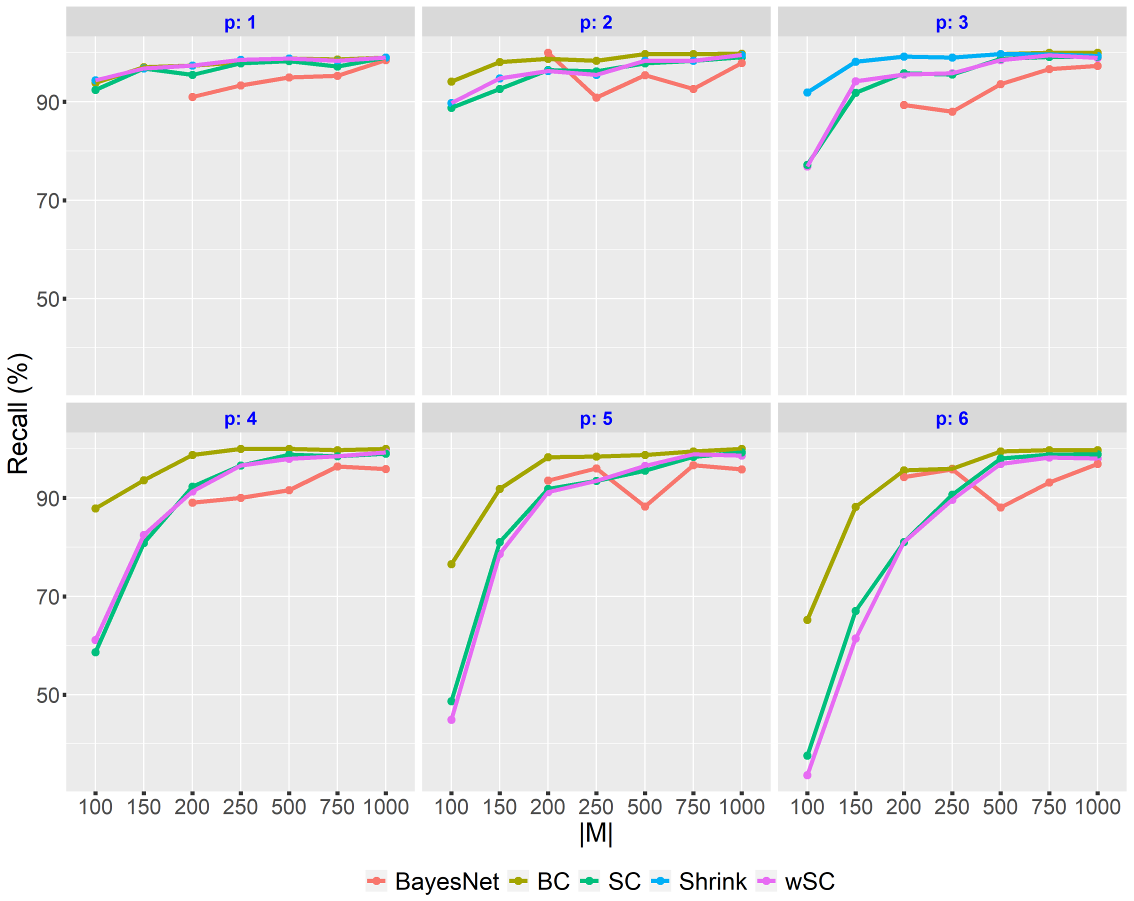


Figure B.1: Recall metrics for E1 instances.

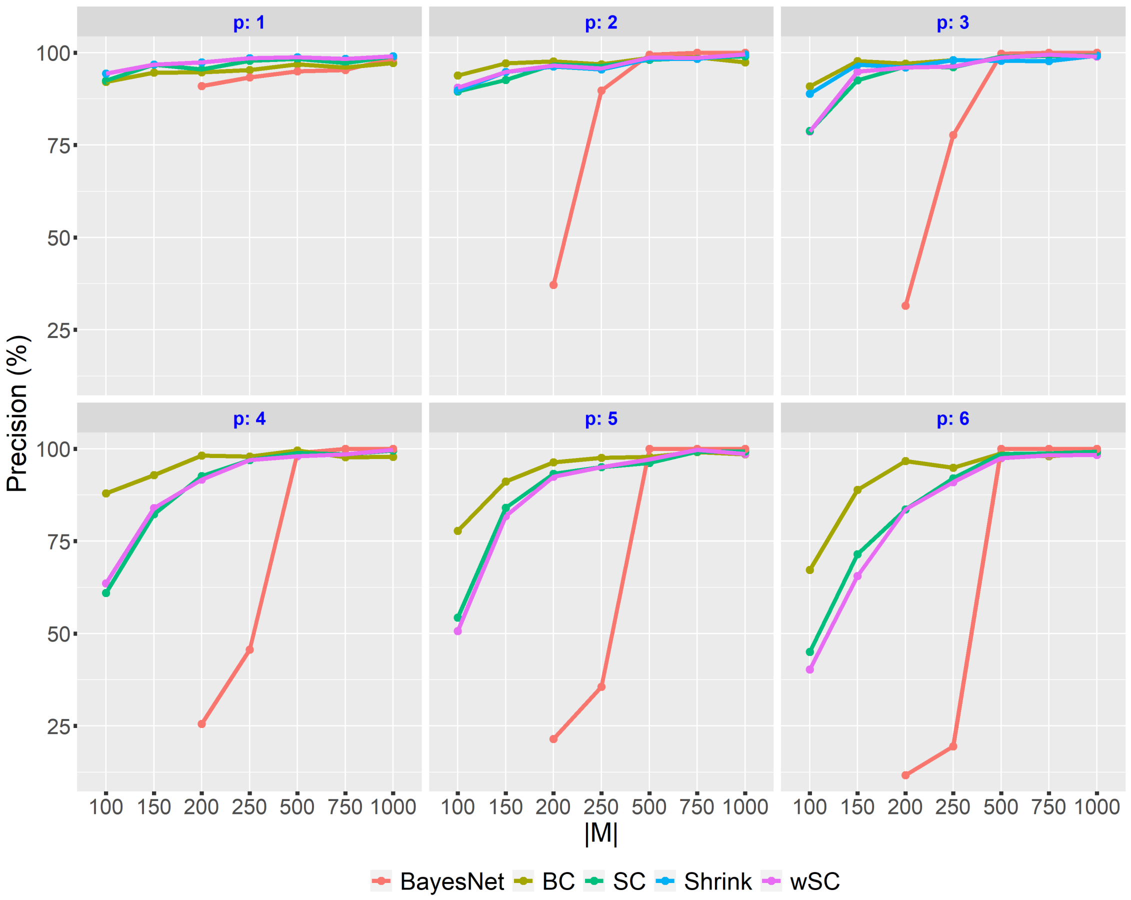


Figure B.2: Precision metrics for E1 instances.

Appendix C

RECALL AND PRECISION GRAPHS FOR E2

In Figure C.1 we show the recall and in Figure C.2 we show the precision of the considered algorithms for E2.

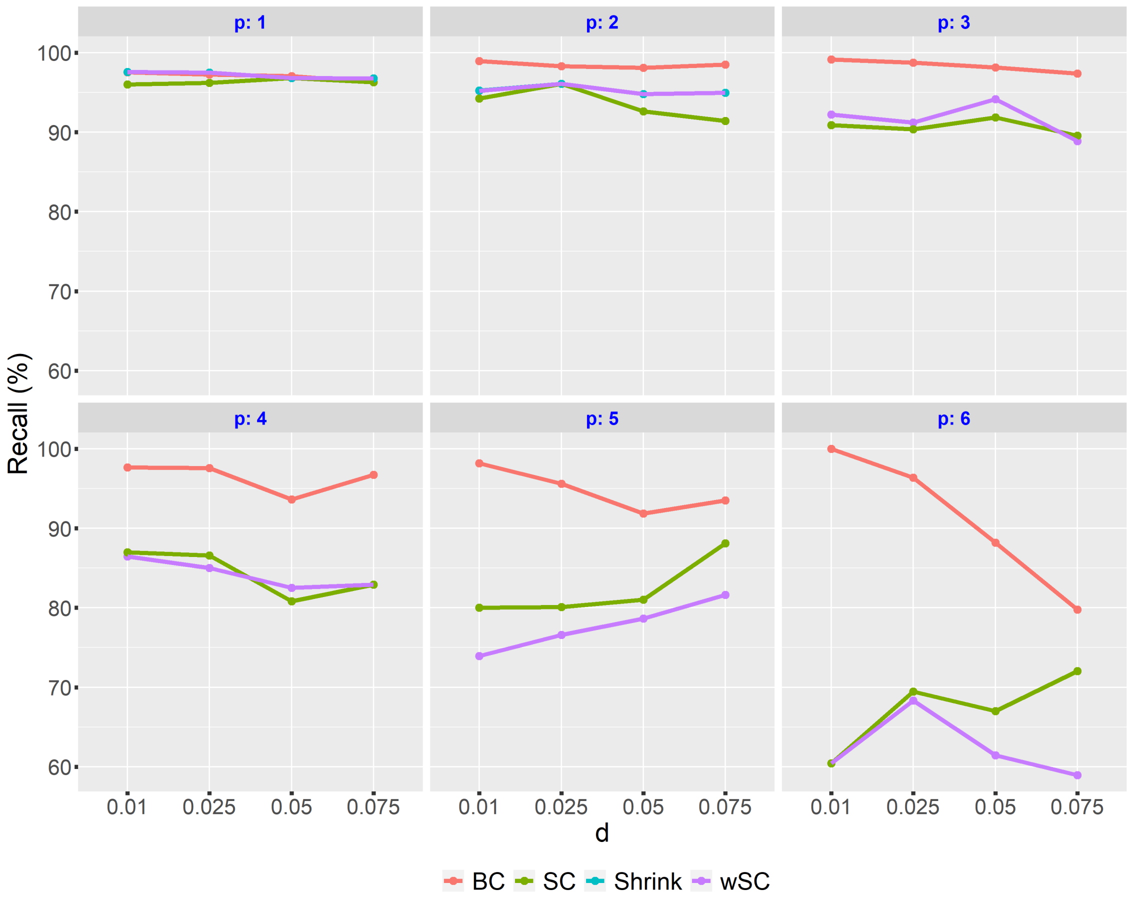


Figure C.1: Recall metrics for E2 instances.

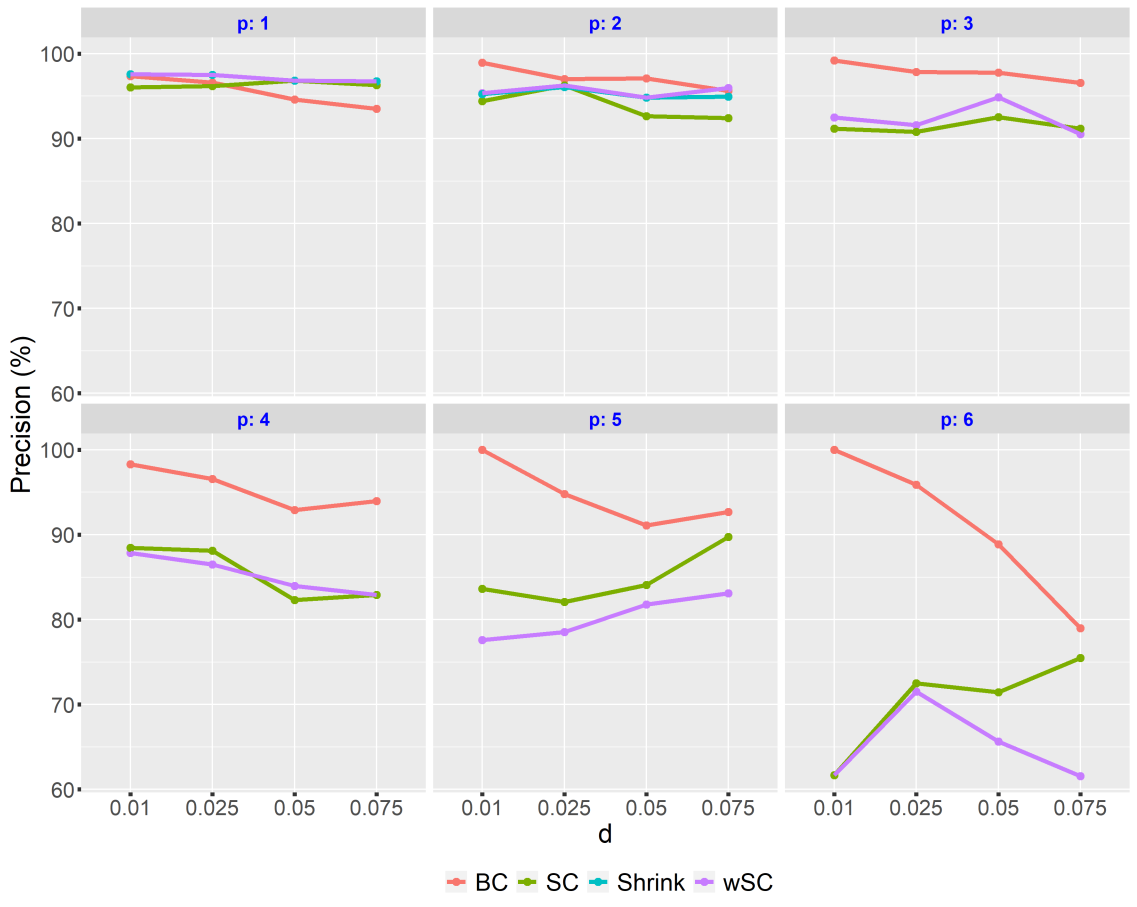


Figure C.2: Precision metrics for E2 instances.

Appendix D

RECALL AND PRECISION GRAPHS FOR E3

In Figure D.1 we show the recall and in Figure D.2 we show the precision of the considered algorithms for E3.

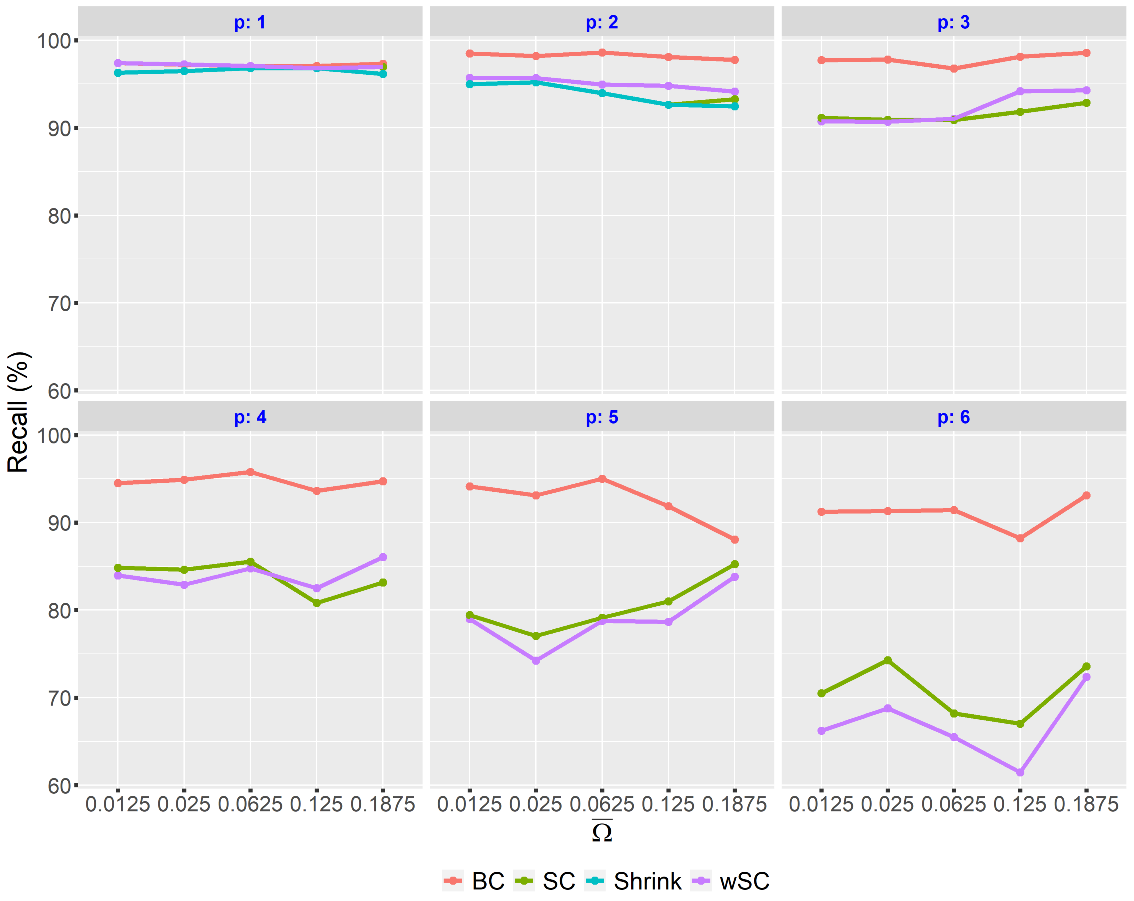


Figure D.1: Recall metrics for E3 instances.

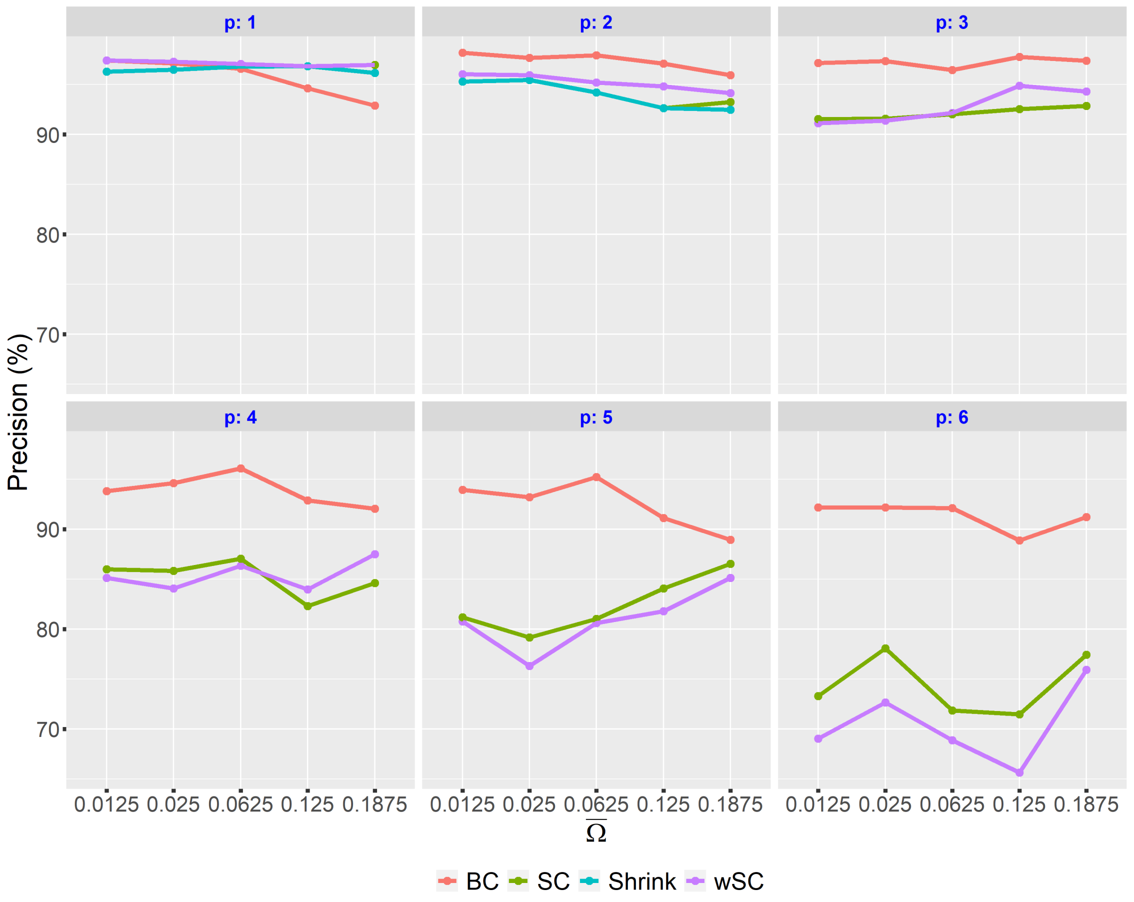


Figure D.2: Precision metrics for E3 instances.

Appendix E

RECALL AND PRECISION GRAPHS FOR E4

In Figure E.1 we show the recall and in Figure E.2 we show the precision of the considered algorithms for E4.

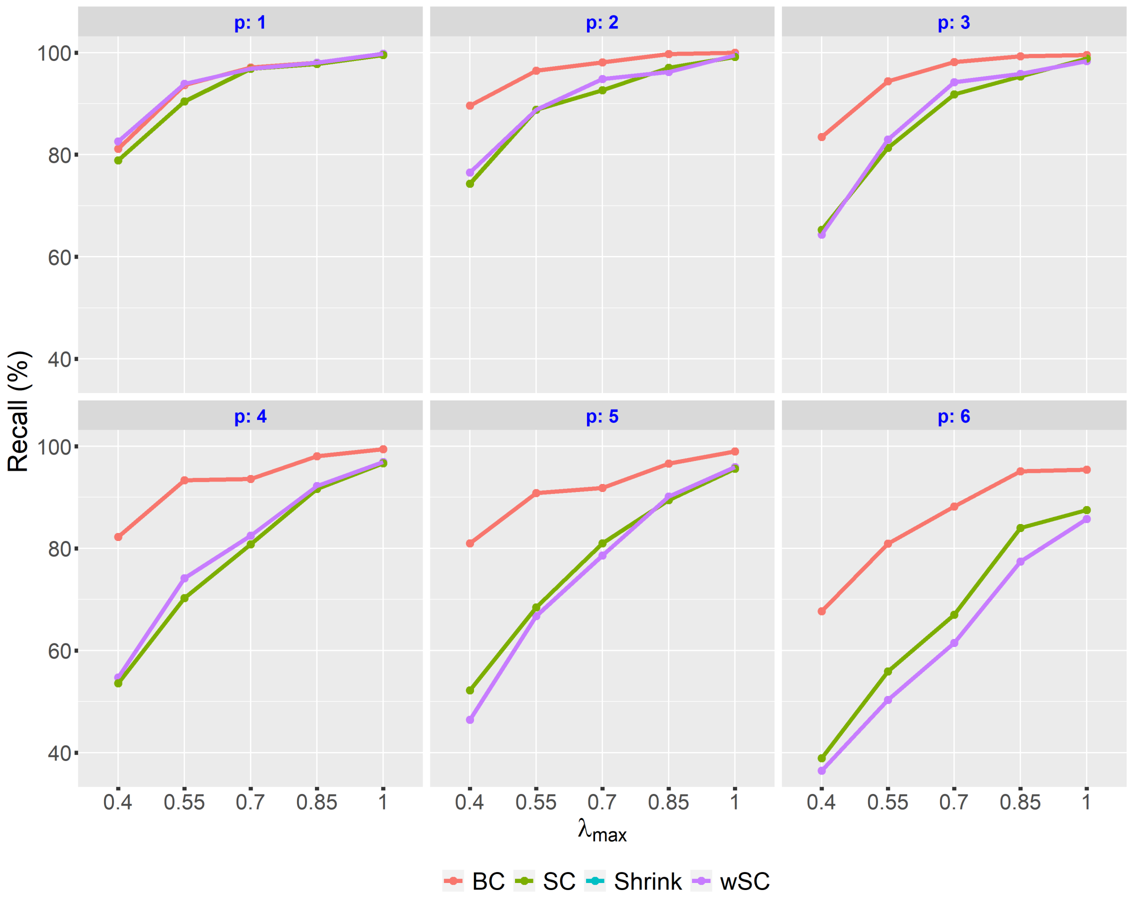


Figure E.1: Recall metrics for E4 instances.

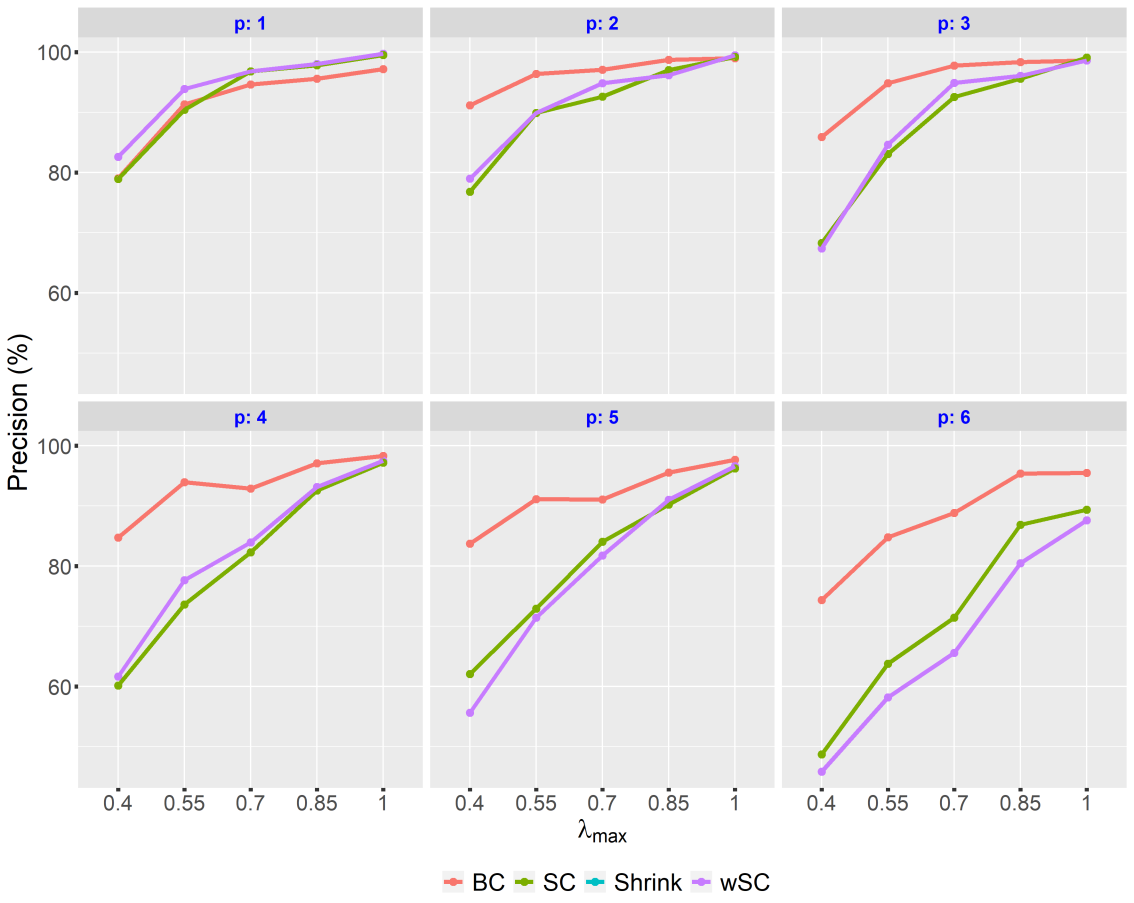


Figure E.2: Precision metrics for E4 instances.

Appendix F

RECALL AND PRECISION GRAPHS FOR E5

In Figure F.1 we show the recall and in Figure F.2 we show the precision of the considered algorithms for E5, i.e. one guess and three best guesses of BC.

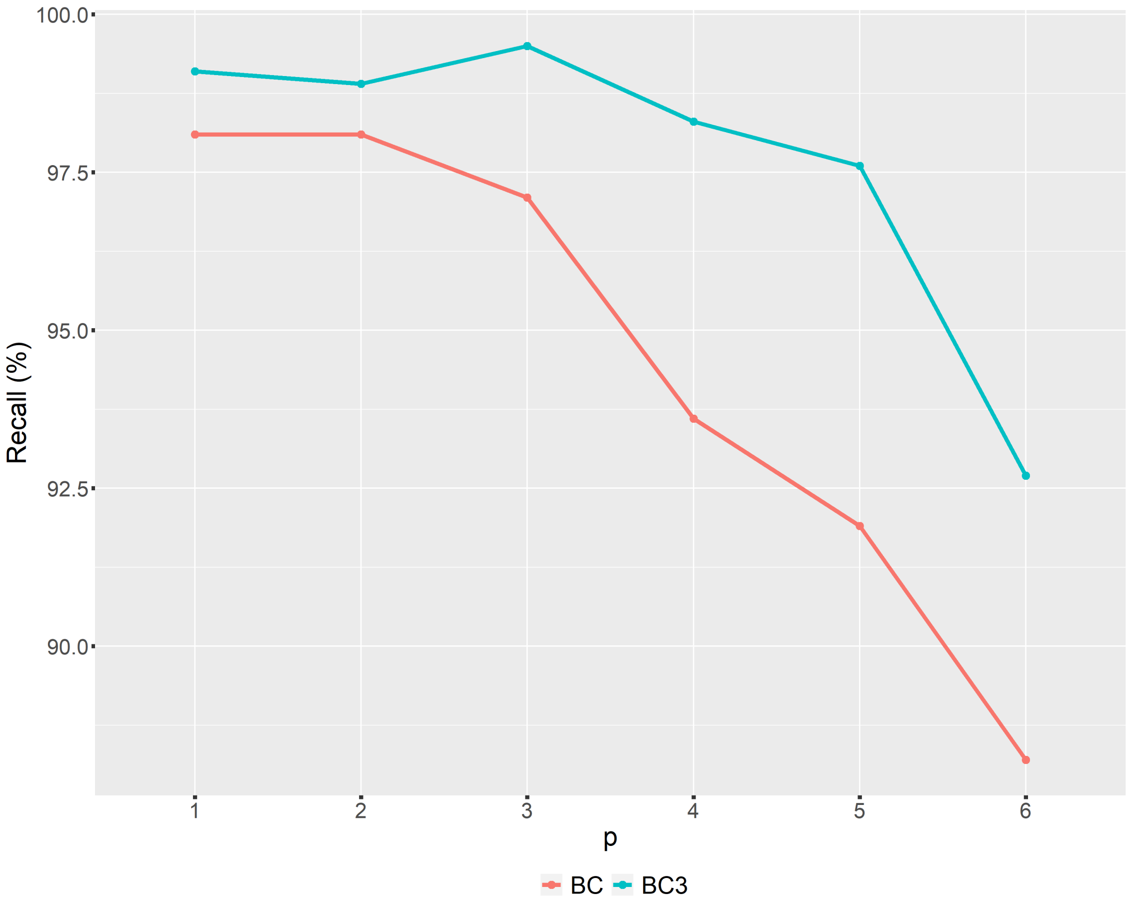


Figure F.1: Recall metrics for E5 instances.

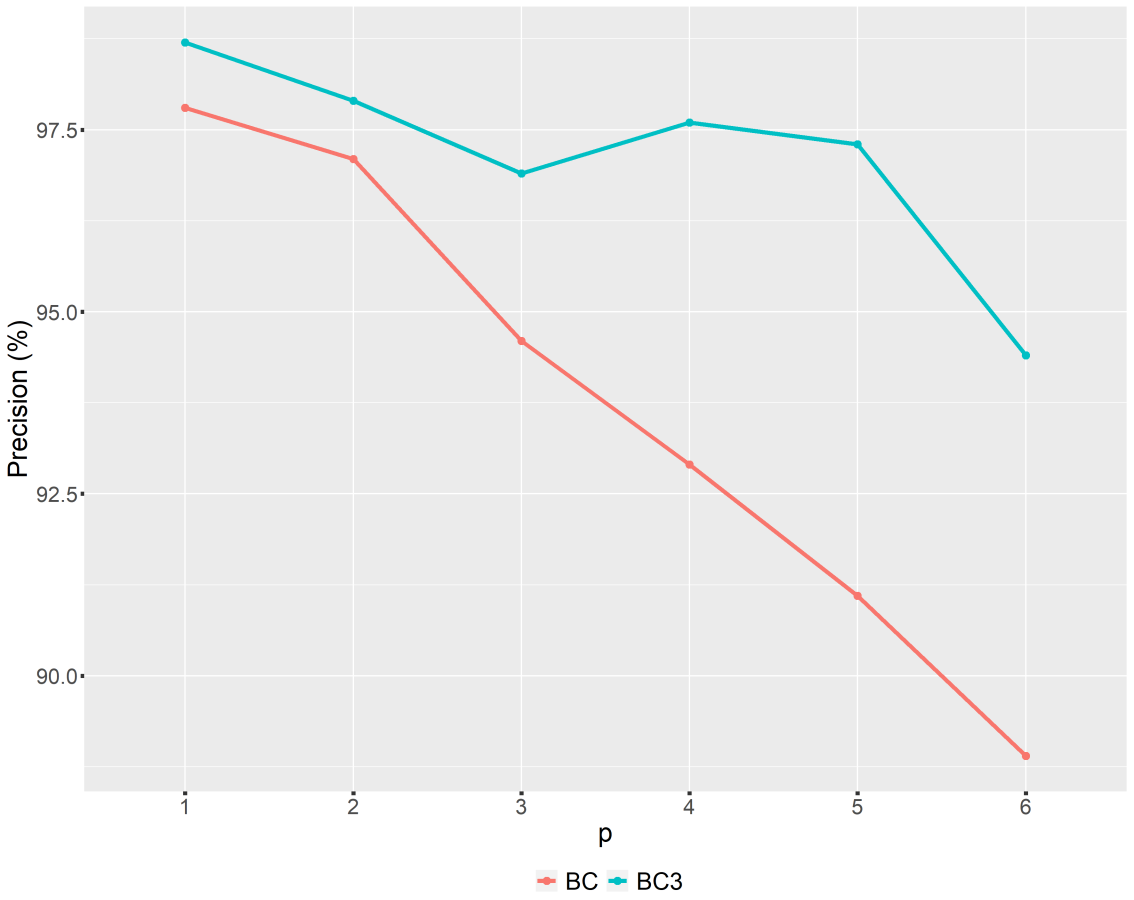


Figure F.2: Precision metrics for E5 instances.

Appendix G

RECALL AND PRECISION GRAPHS FOR E6

In Figure G.1 we show the recall and in Figure G.2 we show the precision of our MCT algorithm for various θ and p.

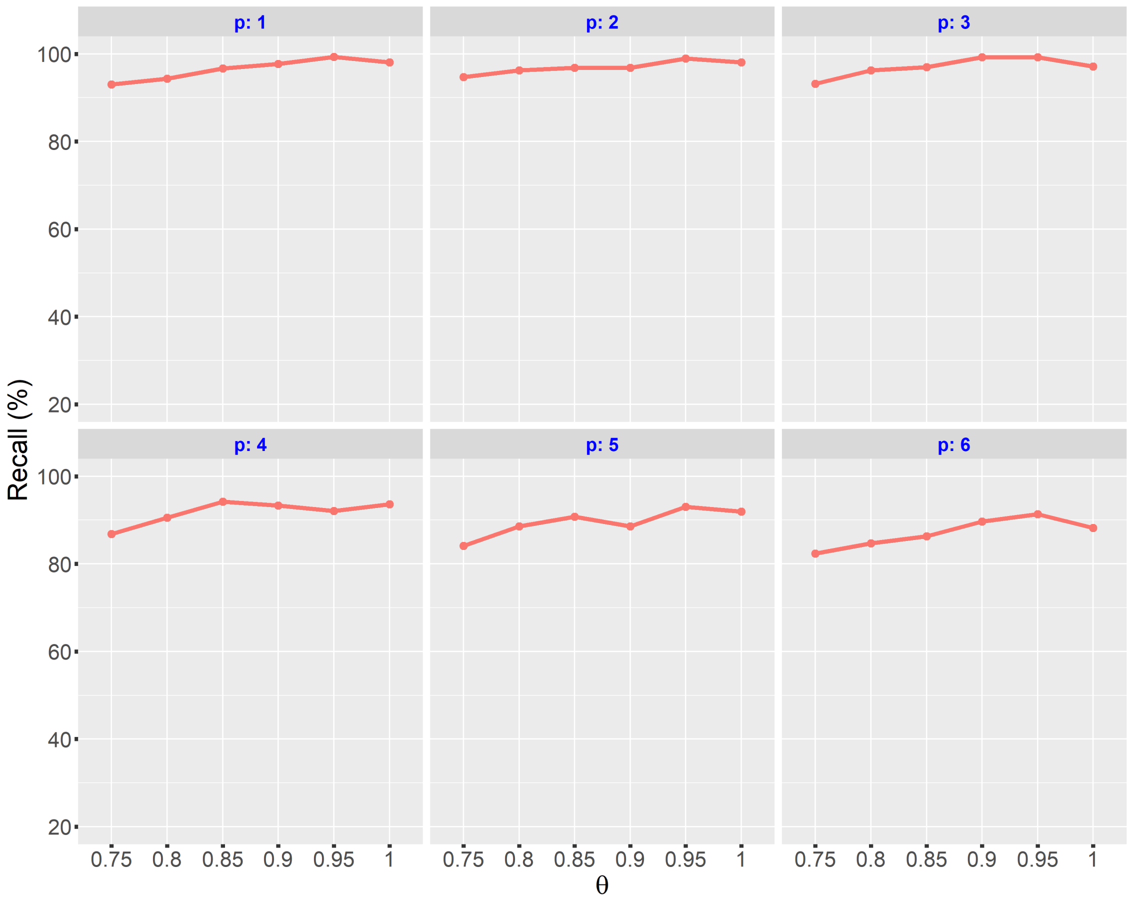


Figure G.1: Recall metrics for E6 instances.

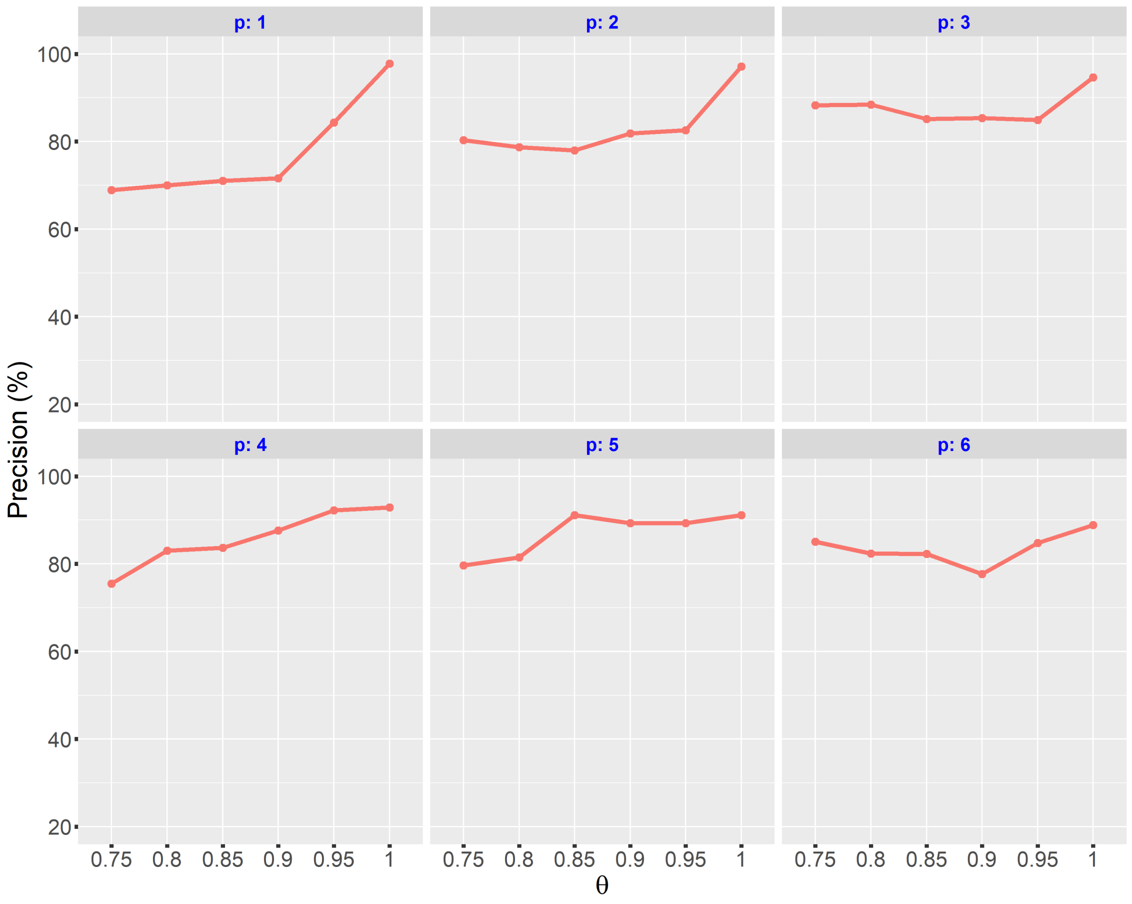


Figure G.2: Precision metrics for E6 instances.

Appendix H

RECALL AND PRECISION GRAPHS FOR E7

In Figure H.1 we show the recall and in Figure H.2 we show the precision of our MCT algorithm for various θ and p.

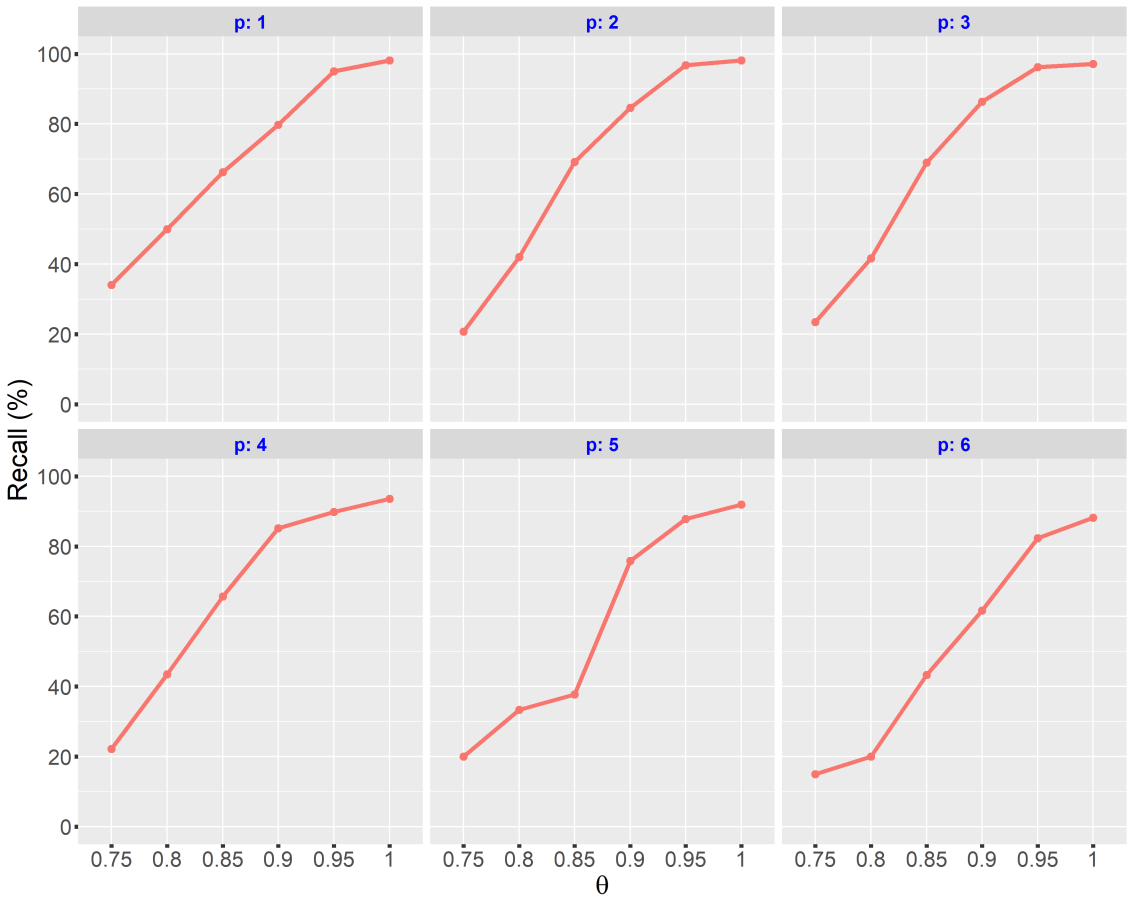


Figure H.1: Recall metrics for E7 instances.

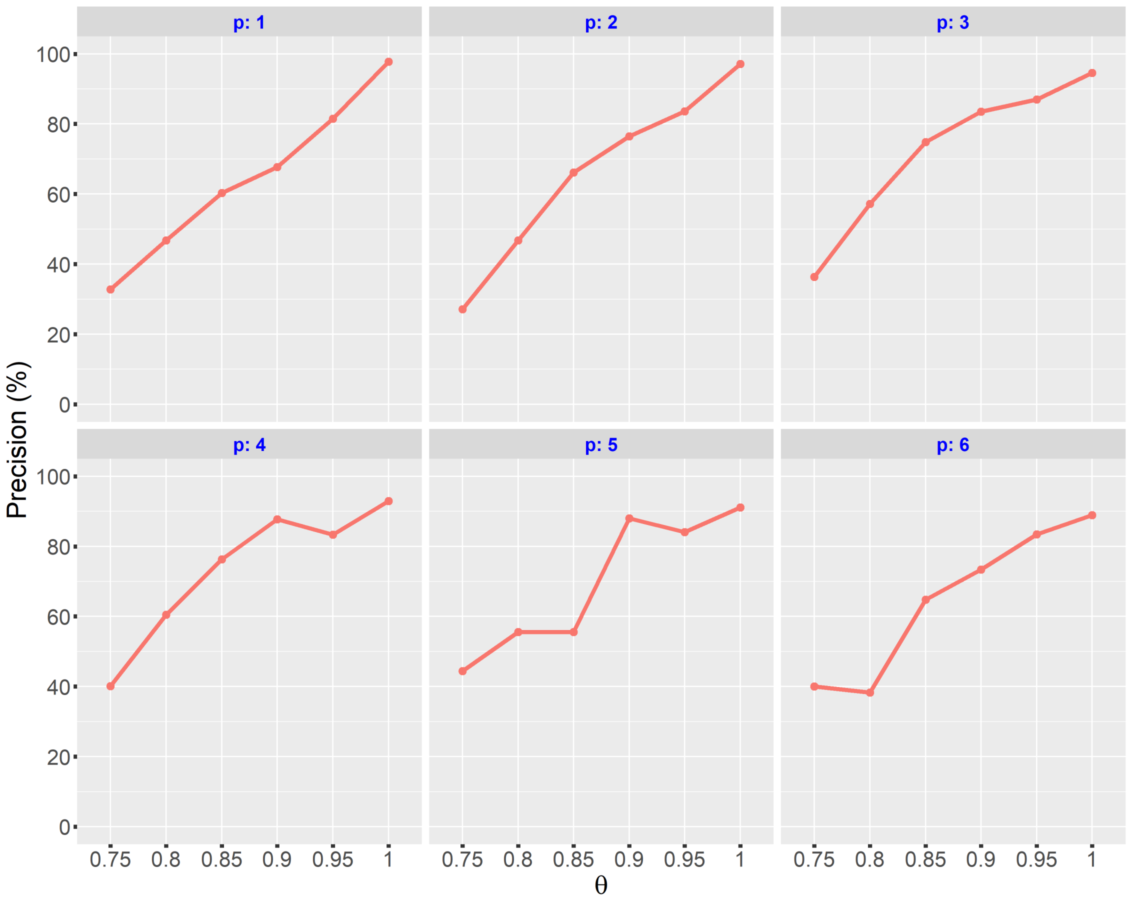


Figure H.2: Precision metrics for E7 instances.