Università degli Studi di Torino
Scuola di Dottorato in Scienza e Alta Tecnologia

Tesi di Dottorato di Ricerca in Scienza e Alta Tecnologia
Indirizzo: Informatica

RELIABILITY ANALYSIS OF PROBABILISTIC NETWORKS

Roberta Terruggia

Advisor: Prof. Andrea Bobbio
Tutor: Prof. Giuliana Franceschinis

XXII Ciclo, Gennaio 2010

Contents

Chapter 1

Introduction

In the last decades, due to the progress in computer power and the growing consciousness that both man-made structures and human interactions can be viewed as networks, the interest in this field has been enhanced to a wider scientific audience [15]. The increased complexity of real networked systems requires new analytic approaches to afford their new scale and predict their behavior.

A peculiar feature of the modern organization of the human life is the increasing level of the interconnectivity. Social, economical and technological structures tend to increase in dimension and to aggregate in complex interconnected systems. Connected systems can be abstracted in the form of network where the vertices are the entities of the system and the edges the physical or relational links among them. One relevant property of networks, that make them a preferential structure both in natural and technological systems, is that the connection between any two nodes of the networks can be usually achieved through a number of redundant paths, thus making the connection intrinsically reliable. For this reason, the reliability of networks

has always been a major concern since the earliest times of reliability engineering [72, 5].
Since the early 60s the exponential growth of the size and number of networks of all types has led many researchers to study the various characteristics, the topology, and the reliability of the networks.
Today the analysis of the network reliability is a major concern for the creation, validation and maintenance of many real systems such as communication, computer or electrical networks. The reliability of these complex systems is of increasing importance since the failure of some components may lead to disastrous results.
The IEEE 90 standard [3] defines the reliability as "the ability of a system or component to perform its required functions under stated conditions for a specified period of time".
The reliability of complex systems is an important element because a simple accident can create more catastrophic consequences. System accidents may have serious financial, environmental or human effects.
For this reason it is important to provide methods to help people to determine the probability of occurrence of such accidents. For example, the failure of the power network in North American August 14, 2003 plunged into darkness more than 50 million people in the United States and Canada. The failure of certain components of the power transmission in the state of Ohio create a cascading collapse of 22 nuclear and 80 thermal plants in less than 10 seconds. In telecommunication networks the maximum acceptable of unavailability is approximately five minutes of out of service in a year.

The reliability of the systems that we analyse depends on the reliability of the components and on the system topology. We take into account the possible failure of components (links and nodes) knowing that system components are subject to failure. In the worst case the failure of certain

sets of components may cause the failure of the network itself.

A network can be represented by means of a graph whose elements (both vertices and edges) are considered as binary objects characterized by a working (up) or a failed (down) condition. If a probability measure is assigned to the up or down state of each element, a probability measure associated to the connectedness of the whole graph can be computed. The network reliability is defined as the probability that all nodes, or a subset of the nodes, of the graph communicate through at least one path of working edges. The computational techniques proposed in the literature to solve the network reliability problem always assume that the network elements are statistically independent and can be classified in two main categories; *i)* - approaches in which the desired network reliability measure is directly calculated (series-parallel reduction [20], pivotal decomposition using keystone components [66, 46]) and *ii)* - approaches in which all possibilities through which two specified nodes can communicate (or not communicate) with each other are first enumerated (path/cut set search [12, 58]) and then the reliability expression is evaluated, resorting to different techniques [5], like inclusion-exclusion method or sum of disjoint products [4, 80, 69, 49]. An algorithm based on the construction and saving of the pathsets in disjoint binary form is presented in [43].
In many cases the analysis of the performance of the system requires a richer representation by associating to each arc a weight representing a specific attribute of the arc (e.g.capacity, resistance, cost, length). For example, the amount of traffic characterizing the connections in communication or transport systems or the distance between nodes in a highway network are fundamental for a full description of these networks. The thesis explores the problem of the quantitative evaluation of reward functions in stochastic weighted networks, where the weights assigned to the arcs may have

different physical interpretations. In this thesis we discuss two types of interpretation of weights: weights as distances and weights as capacities. Correspondingly, two different algorithms based on a data structure called Algebraic Decision Diagram (ADD), are discussed and presented. The first evaluates the probability that the terminal node can be reached from the source within a determinate distance or cost. The second computes the probability that a flow greater than a threshold can be transmitted between the source and the sink. The algorithms have been tested with several examples and with some benchmark networks taken from the literature.

In many real-life situations components and/or systems can be in one of more than two states (up and down). Systems that are characterized by different levels of performance are known as multi-state systems (MSS).

Examples of MSS are power systems and computer systems, where the performance of the elements is characterized by generating capacity and data processing speed respectively. Most of reliability analysis and optimization models assume that system consists of binary-state components, where the states are functioning and failure. Binary-state reliability models don't allow to adequately describe some aspects of the MSS as MSS operation processes, remaining resources, system state evolution, reasons of system failures and mechanisms of their prevention.

The computational techniques to compute the MSS reliability assessment, which are presented in literature, are based on four different approaches: the extension of the Boolean models to the multi-valued case, the stochastic processes (mainly Markov and semi-Markov) approach, the universal generating function approach, and the Monte-Carlo simulation technique.

The main difficulty in the MSS reliability analysis is the "dimension damnation" since each system element can have many different states (not only two states as the binary-state system).

This makes the known approaches overworked and time consuming, because the number of system states increases dramatically with the increase in the number of system elements.
In this thesis we illustrate an analysis technique based on Multi-valued Decision Diagrams [53].
In the last decades, Decision Diagrams (BDD, ADD, MDD) [30] [11] [53] have provided an extraordinarily efficient method to represent and manipulate Boolean functions [31], and have been also exploited to model the connectivity of a Boolean networks. The present thesis discusses various approaches that can be followed to compute the reliability of a network starting from a Decision Diagrams representation and presents a preliminary tool for the network analysis.

The thesis is organized as follows: Chapter 2 describes the state of the art of some existing methods of network reliability present in literature.
In Chapter 3 we introduce the data structure of Binary Decision Diagrams.
Chapter 4 presents some methods of analysis of Probabilistic Unweighted Networks using the Binary Decision Diagrams. Two real world studies are analysed and results are reported.
In Chapter 5 the Probabilistic Weighted Network analysis is proposed. By means of new operations defined over the Algebraic Decision Diagram it is possible to study the network reliability by considering the weight assigned to the edges.
In Chapter 6 complex networks are described and studied. The peculiar properties of Regular Networks, Random Networks and Scale Free Networks are illustrated. An exact and a simulative algorithm is shown in order to compute the network reliabilty of these networks.
The results of the analysis of a case study are presented in Chapter 7. In Chapter 8 a new algorithm is designed in order to analyse the Probabilis-

tic Weighted Multi-State Network by means of the Multi-valued Decision Diagrams.
The two tools implemented to validate and experiment the theoretical results developed in this thesis, are described in Chapter 9.
Finally in Chapter 10 the conclusions are outlined.

Chapter 2

State of the Art

2.1 Introduction

The study of reliability of complex systems, that means systems containing a rather large number of interacting elements, is interesting for people of different disciplines: from the technological to economical and biological areas.
Network reliability encompasses a range of issues related to the design and analysis of networks which are subject to the random failure of their components.
Relatively simple, and yet quite general, network models can represent a variety of applied problem environments. Network classes include data communications networks, voice communications networks, transportation networks, computer architectures, electrical power networks command and control systems and new emerging ad hoc wireless networks .
The advent of the digital computer led to significant reliability modeling efforts. Early computer memories were made up of large numbers of in-

dividual components such as relays or vacuum tubes. Computer systems which failed whenever a single component failed were extremely unreliable, since the probability of at least one component out of thousands failing is quite high, even if the component failure probability is low. Much initial work in highly reliable systems concentrated on systems whose failure could cause massive damage or loss of human life. Examples include aircraft and spacecraft systems, nuclear reactor control systems and defense command and control systems. More recently, it has been recognized that very high reliability systems make economic sense in a wide range of industries. Examples include telecommunications networks, banking systems, credit verification systems and order entry systems. The ultimate objective of research in the area of network reliability is to give design engineers procedures to enhance their ability to design networks for which reliability is an important consideration.

2.2 Systems

We consider systems of elements that satisfy the following hypotheses:

- The system may be decomposed into n elements $1, \ldots, n$
- In a given instant the state of each element may be only one of the following: up (working) or down (faulty)
- In a given instant the state of the system may be only functioning or faulty and depends only on the state of its components.

A "structure function" allows to describe behavior of system reliability depending on the efficiency of its components.

2.3 Structure Function

We can consider a system S composed of n components e_i, $i = 1, 2, \ldots, n$. We associate with each element e_i a state variable x_i such that:

$$x_i = \begin{cases} 1 & \text{if the component } e_i \text{ is in a good state} \\ 0 & \text{if the component } e_i \text{ is faulty} \end{cases}$$

The set of component can be identified as $\mathbf{e} = \{e_1, \ldots, e_n\}$ and the state of set of components, denoted by (x), as the n-tuple $(x_1, x_2 \ldots . x_n)$.

$$(x) = (x_1, x_2, \ldots, x_n) \tag{2.1}$$

There are 2^n possible n-tuple of (x) due to the 2^n different states of the set of elements.
Let y be the state variables of the system such that:

$$y = \begin{cases} 1 & \text{if the system is up} \\ 0 & \text{if the system is down} \end{cases}$$

The variable y depends on (x) and exists a function $(x) \rightsquigarrow y$ depending on n variable called "structure function".
We denote the structure function by

$$y = \varphi(x) \text{ or } y = \varphi(x_1, x_2, \ldots x_n) \tag{2.2}$$

2.3.1 Graphical Representation of Networks

In very general terms a network is any system that admits an abstract mathematical representation as a graph whose nodes (vertices) identify the

elements of the system and in which the set of connecting links (edges) represent the presence of a relation or interaction among those elements. Clearly such a high level of abstraction generally applies to a wide array of systems. In this sense, networks provide a theoretical framework that allows a convenient conceptual representation of relations in the systems where the system level characterization implies the mapping of interactions among components.

2.3.2 Graph

A network can be represented in the form of a graph $G = (V, E)$, where V is the set of vertices (or nodes) and E the set of edges (or arcs). We denote by N_V the number of vertices and by N_E the number of edges. A network is undirected when the edges can be traversed in both directions, while the network is directed if the edge can be traversed only in one direction indicated by an arrow. For undirected networks without self loop the presence of at least one edge per node guarantees that all the nodes are connected. Usually real networks are much more connected than this minimal threshold thus allowing multiple paths among any pair of vertices. For directed graphs the connectivity property is more cumbersome since vertices can pertain to three categories: strongly connected vertices (subset of vertices that can be reached from any other vertex in the subset following the direction of the edges); transient nodes that have only outgoing arcs and cannot be reached from any other vertex; absorbing vertices that have only ingoing arcs and once reached cannot be left.

	1	2	3	4
1	0	1	1	0
2	0	0	1	1
3	0	0	0	1
4	0	0	0	0

Table 2.1: Matrix representation of graph in Figure 2.1

2.3.3 Graph Representation

A graph can be graphicaly represented as a set of circle (the vertices) connected by a set of lines (the edges) as depicted in Figure 2.1.

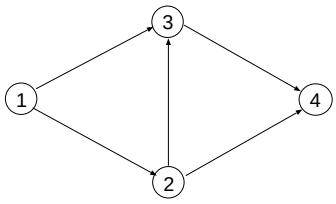

Figure 2.1: Graphical representation

Adjacency Matrix

In order to describe the graph a boolean matrix can be used, a square matrix of order $N_V \times N_V$, where N_V is the number of vertices, whose elements $a_{i,j}$ are 0 or 1 according as if exist an edge between the pair of nodes i and j. For directed networks the element $a_{i,j}$ is equal to 1 if there is an edge coming from the vertex i to the vertex j. The adjacency matrix

Node	List
1	2 3
2	3 4
3	4
4	

Table 2.2: Adjacent List representation of graph in Figure 2.1

of an undirected network is symmetrical: $a_{i,j} = a_{j,i}$. The diagonal elements are equal to zero ($a_{i,i} = 0$)
Table 2.1 represents the adjacency matrix of graph in Figure 2.1.

Adjacency List

A graph may also be described by an adjacent list where for each node v_i of the graph the list of its neighbours (nodes connected by an edge with v_i) is given.
Table 2.2 represents the adjacency list of graph in Figure 2.1.

An absolute best way in order to represent a graph does not exists : if graph has a large number of edges, then the adjacency matrix doesn't waste much space, it represents the best solution also because it indicates whether an edge (i, j) exists with one access (rather than following a list). However, if graph is sparse (not many of its vertex pairs have edges between them), then an adjacency list becomes preferable.

2.3.4 Cuts and Paths

Consider a systems with components:

$$e = \{e_1, e_2, \ldots, e_n\} \tag{2.3}$$

and defined by the following structure function:

$$y = \varphi(x_1, x_2, \dots x_n) \tag{2.4}$$

Let

$$p = \{e_i, i \in I\} \subset e \tag{2.5}$$

be a subset of components defined by the subset of indices $I \subset \{1, 2, \dots, n\}$ One say that p is a ***path*** if

$$\left.\begin{array}{ll} x_i = 1, & i \in I \\ x_i = 0, & i \notin I \end{array}\right\} \Rightarrow \; y \; = \; 1 \tag{2.6}$$

A **path** is thus a subset of components such that the system is working if all the elements of the subset are up and the other component are not working.

Considering the same system, let

$$c = \{e_j, j \in J\} \subset e \tag{2.7}$$

be a subset of components defined by the subset of indices $J \subset \{1, 2, \dots, n\}$ One say that c is a ***cut*** if

$$\left.\begin{array}{ll} x_j = 0, & j \in J \\ x_j = 1, & j \notin J \end{array}\right\} \Rightarrow \; y \; = \; 0 \tag{2.8}$$

A **cut** is thus a subset of components such that the system is down if all the elements of the subset are down and the other component are working.

Minimal path A path p

$$p = \{e_{p1}, e_{p2}, \ldots, e_{pI}\} \tag{2.9}$$

is a minimal path (or minpath) if does not exist a subset of p that is also a path.

Minimal cut A cut c

$$c = \{e_{c1}, e_{c2}, \ldots, e_{cJ}\} \tag{2.10}$$

is a minimal cut or mincut if does not exist a subset of c that is also a cut.

2.4 Network Reliability

Traditional reliability analysis techniques in the area of networked systems used to look at networks of tens or hundreds of vertices [54, 4, 66, 12]. These techniques were based on exhaustive search algorithms intended to provide qualitative and quantitative information on the network connectivity, dependability, and vulnerability. A literature survey indicates that the approaches, which have been used to compute two-terminal reliability could broadly be classified into two paradigms: i) - the paradigm in which desired network reliability is directly calculated (series-parallel reduction [20], pivotal decomposition using keystone components [66, 46]) and ii) - the paradigm in which all possibilities through which the two specified nodes can communicate (or not communicate) with each other are first

enumerated (path/cut set search [12, 58]) and then reliability (unreliability) expression is evaluated.

2.4.1 Analysis of Network Reliability

In topological network design, a main concern is to design networks that operate effectively in the presence of component failures. The first ingredient of effective operation is connectivity. In real environments, problems with performance are more prevalent and hence more widely addressed. However, while problems with connectivity are less frequent, they are more catastrophic. Network reliability is primarily concerned at the present time with issues of connectivity [6]. In addressing network reliability, three main questions must be answered:

1. What is the structure of the network?

2. What causes components to fail?

3. When is the network considered operational?

Any measure of "reliability" should be a (useful) quantitative indicator of the ability of the network to remain operational in its environment, despite the component failures. Naturally, any more precise definition requires precise answers to the three questions above. It comes as no surprise that hundreds of seemingly natural definitions arise by examining the plethora of different types of networks, causes and types of failures, and levels and types of operation.
One should not expect to find a single definition for reliability that accommodates the many real situations of importance. Here, we elect to focus on one of the most basic and widely studied classes of reliability measures.

For our purposes we adopt a simple network model known as the probabilistic graph: a network is a a graph G with vertex set V and edge set E as described in [35]; we let $N_V = |V|$ and $N_E = |E|$. Vertices of G represent network nodes, and edges represent bidirectional or direct connections. In our model, edges are subject to random failure, while vertices never fail. Each edge $e \in E$ has a known probability p_e of being operational; otherwise it is failed. Operations of different edges are statistically independent.
We address one basic network operation. Let $K \in V$ be a set of target vertices; we imagine that these vertices are to be able to communicate with each other. The network G is k-terminal operational for K when the subgraph of operating edges of G has all vertices of K in the same connected component. When $K = V$, this is an all-terminal operation; when $|K| = 2$, it is a two-terminal operation. At any instant of time, only some edges of G are operational. A state of G is a subgraph (V, E') with$E' \subseteq E$, where E' is the set of operational edges. For any fixed set K of targets, each state is classified as operational or failed, according to whether it is k-terminal operational for K.

2.4.2 Computational Complexity

An algorithm is a polynomial time algorithm if for a problem of size n, its running time is bounded by a polynomial in n. Any algorithm that is not a polynomial time algorithm is generally referred to as an exponential time algorithm. In combinatorics, the so-called "satisfiability" problem is in the class NP-complete, i.e., given an arbitrary Boolean expression in product of sums form, determine whether or not there exists an assignment of values TRUE or FALSE to the variables that makes the entire expression TRUE (a Boolean expression can be obtained for every network in terms of Boolean

indicators for the edges and nodes).
A problem P is said to belong to the class NP-complete if :

1. given a proposed solution its validity can be checked in polynomial time,

2. the existence of an algorithm to solve P in polynomial time implies the existence of algorithms to solve the satisfiability problem in polynomial time.

It is generally believed that no polynomial time algorithm exists for any of the NP-complete problems. Any problem that is not NP-complete, but that can be proved to be at least as hard as NP- complete problem is known as an NP-hard problem.
Most network reliability problems are, in the worst case, NP-hard ([13], [14]). Network reliability problems are, in a sense, more difficult than many standard combinatorial optimization problems. That is, given a tentative solution to a combinatorial problem, often its correctness can be determined in polynomial time. However, given a purported solution to a reliability problem, it cannot even be checked without computing the reliability of the network from the beginning.
Although the above remarks sound very discouraging, there are in fact linear and polynomial time algorithms for network reliability problems of special structure.

2.5 Survey of Existing Methods to Compute Network Reliability

In this section we present a survey of existing methods used in order to compute the reliability of networks. More details of the major methods are

described in next sections.
We can classify methods for network reliability computation in the following categories:

- enumeration methods,
- transformation methods (reduction and decomposition),
- direct methods,
- approximation methods.

It is important to note that these categories are not mutually exclusive and are often combined with each other.
Firstly, we discuss methods for the representation of the structure function. Complete state enumeration is the simplest, but also the most inefficient method (the number of possible states usually grows exponentially with the network size). More sophisticated is a minpath or mincut enumeration, which results in a corresponding DNF (Disjunctive Normal Form) representation (see Section 3.2) of the structure function. This is more efficient than the enumeration of all states, but still may be infeasible in practice. Whether a minpath- or mincut-based representation of the structure function is more suitable depends on the actual network structure. More recently different approaches based on Binary Decision Diagrams (BDDs) are introduced, where a structure function representation is obtained by means of Shannon's decomposition principle, a common factoring method [30]. Closely related are the approaches based on Edge Expansion Diagrams (EEDs) (see [56]).
Most enumeration techniques require in a second step some disjoint form of the resulting terms in order to calculate the exact reliability. The classical

way to accomplish this is by the application of some sum of disjoint products (SOP) method, see [4], [49], [48], and [10]. Another classical method is the standard inclusion-exclusion expansion, which is very inefficient on its own right, but some sophisticated variations exist in the context of the domination graph theory [74]. More recently we observe the construction of a particular type of BDDs, so-called Ordered BDDs (OBDDs), from min-path, mincut, or EED representations, see [56] for the latter. Of course, it is also possible to generate an OBDD directly from the graph representation of the network, see [23], [21], [19], [46] and [83]. An interesting special case among network relilability problems is obtained when equal edge reliabilities are assumed. This case is treated sperarately with the so-called theory of reliability polynomials. So when all probabilities are equal to p, a pathset containing i edges occurs with probability $p^i \cdot (1-p)^{m-i}$, where $m = N_e$. The reliability can be now written as $R(p) = \sum_{i=0}^{m} N_i p^i (1-p)^{m-i}$, where N_i is the number of i-edge pathsets. This is a polynomial in p of degree m. The computation of network reliability is thus reduced to the problem of computing the coefficients N_i. For more details on reliability polynomials see [35], [34], [32] and [44]. In addition to all of the above mentioned evaluation methods, various decomposition and reduction methods exist. These methods can be used independently of the above described methods. The general idea behind decomposition methods is to divide the graph into subgraphs, and then to evaluate the subgraphs independently of each other. The overall reliability is then calculated based on the reliabilities of the corresponding subgraphs. Reductions are special transformation techniques which allow to simplify the topological structure of a given network. The goal of such techniques is to reduce the network size in order to improve the performance of further reliability calculations. Reductions are reliability preserving and thus particularly qualified for combined use

with exact methods. For certain classes of graphs, the reliability can be computed very efficiently by the sole application of reductions. The most important types of reductions are series-parallel and polygon-to-chain, see [57]. Beside the exact reliability methods, corresponding approximation techniques exist. This is due to the fact that essentially all reliability problems of interest are NP-hard [79]. Many sophisticated methods have been developed, both for the estimation of network reliability by Monte Carlo sampling techniques [38] as well as for the computation of reliability bounds.

2.6 Series Systems

A series system is a configuration such that, if any one of the system components fails, the entire system fails.
A series system is weaker than its weakest link. Figure 2.2 shows a graphical description of a series system.

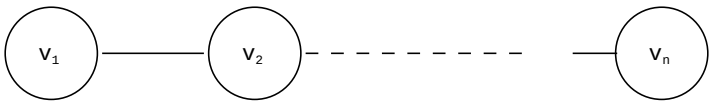

Figure 2.2: Graphical representation

The structure function of a series system is:

$$y = \varphi(x_1, x_2, \ldots x_n) = x_1 \cdot x_2 \cdot \cdots \cdot x_n = \prod_{i=1}^{n} x_i \tag{2.11}$$

This system functions only under the condition that all components of the system are working:

$$(\forall i, x_i = 1) \Longrightarrow (y = 1) \tag{2.12}$$

$$(\exists i, x_i = 0) \Longrightarrow (y = 0) \tag{2.13}$$

In series networks each component is a cut. The only path is composed of all the components.
Assuming for each component i a different probability p_i to be working, the reliability of the entire system can be computed as:

$$R_s = p_1 \times p_2 \times \ldots \times p_n = \prod_{i=1}^{n} p_i \tag{2.14}$$

If all components have the same probability p_i we have:

$$R_s = (p_i)^n \tag{2.15}$$

We have system success only is we have success of every individual component

2.7 Parallel Systems

A parallel systems is a configuration such that, as long as not all of the system components fail, the entire system works.
The total system reliability is higher then the reliability of any single system component. A graphical description of a parallel system is shown in Figure 2.3.

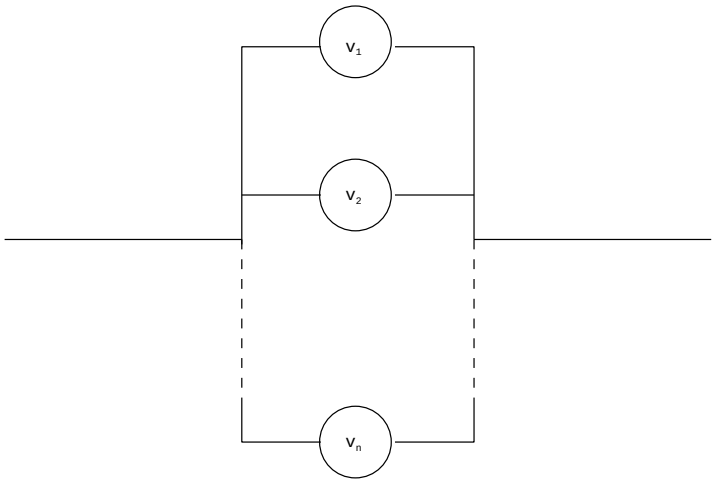

Figure 2.3: Graphical representation

The structure function of a parallel system is:

$$y = \varphi(x_1, x_2, \dots x_n) = 1 - (1 - x_1) \cdot (1 - x_2) \cdot \dots \cdot (1 - x_n) = 1 - \prod_{i=1}^{n} (1 - x_i) \tag{2.16}$$

This system functions only under the condition that at least one of the components of the system is working:

$$(\exists i, x_i = 1) \Longrightarrow (y = 1) \tag{2.17}$$

$$(\forall i, x_i = 0) \Longrightarrow (y = 0) \tag{2.18}$$

In parallel networks each component is a path. The only cut is composed of all the components.
Assuming for each component i a different probability p_i to be working,

the reliability of the entire system can be computed as:

$$R_s = 1 - (1 - p_1) \times (1 - p_2) \times \ldots (1 - p_n) = 1 - \prod_{i=1}^{n} (1 - p_i) \qquad (2.19)$$

If all components have the same probability p_i we have:

$$R_s = 1 - (1 - p_i)^n \qquad (2.20)$$

The classes for which efficient exact algorithms are known are quite sparse, and do not appear to be those in which we expect to find most practical problems. Nevertheless, the presence of such exact algorithms even for sparse classes can be used to accelerate exact algorithms for larger classes that simplify the network in some way.

2.8 Series-Parallel and Parallel-Series Systems

By a *series-parallel system* we mean a system composed of n subsystems in series, where subsystem i consists of m_i elements in parallel, $i = 1, 2, \ldots, n$. The system functions successfully if and only if all subsystems function successfully. Further, each subsystem functions successfully if at least one of its elements functions successfully.
Assuming for each componet j of the subsystem i a different probability p_{ij} the reliability of the series-parallel system can be computed as:

$$R_s = \prod_{i=1}^{n} (1 - (\prod_{j=1}^{m_i} (1 - p_{ij}))). \qquad (2.21)$$

If in the same subsystem all components have the same probability p_i we have:

$$R_s = \prod_{i=1}^{n}(1 - (1 - p_i)^{m_i}). \tag{2.22}$$

If, in addition, all components of the system have the same probability p and all n subsystems have the same number of components (m) then we have:

$$R_s = (1 - (1 - p)^m)^n \tag{2.23}$$

By a *parallel-series system* we mean a system composed of m subsystems in parallel, where each subsystem consists of n_j elements in series, $j = 1, 2, \ldots, m$.
The system functions successfully if at least one of the subsystems function successfully. Further, each subsystem functions successfully if and only if all its elements functions successfully.
Assuming for each component a different probability p_{ij} the reliability of the parallel-series system can be computed as:

$$R_s = (1 - \prod_{i=1}^{m}(1 - \prod_{j=1}^{n_i} p_{ij})). \tag{2.24}$$

If in the same subsystem all components have the same probability p_i we have:

$$R_s = 1 - \prod_{i=1}^{m}(1 - (p_i)^{n_i}). \tag{2.25}$$

If, in addition, all components of the system have the same probability p and all m subsystems have the same number of components (n) then we have:

$$R_s = 1 - (1 - p^n)^m \tag{2.26}$$

2.9 Non Series-Parallel Systems

When reliability-preserving transformations fail to reduce the network into a restricted class for which an efficient exact method is known, we are forced to resort to potentially exponential time methods. The first main class of these exact methods examines the possible states of the network.

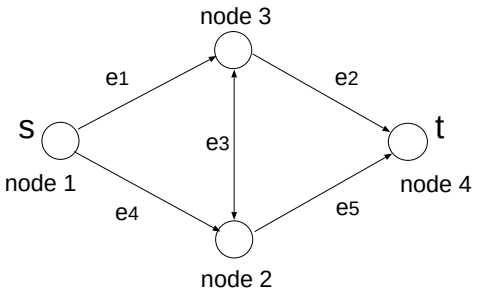

Figure 2.4: Bridge graph

2.9.1 Complete State Enumeration

Complete state enumeration (or Boolean truth table method) is a solution method based on the idea to determine all operational states or all non-operational states of the system, and to sum the probability of each state. If we consider the operational states we obtain the reliabilty of the system, otherwise if non-operational states are used, then the result is the unreliability of the system. A state of a network $G = (V, E)$ is a subset $S \subseteq E$ of operational edges. The conceptually simplest exact algorithm is complete state enumeration. Let be PS the set of all operational states (pathsets).

Then

$$R(G) = \sum_{S \in PS} \prod_{e \in S} p_e \prod_{e \notin S} (1 - p_e) \tag{2.27}$$

By generating all states, and determining which is operational, we can easily (but not efficiently) compute the reliability.

Table 2.3: Operational states of graph

$< e_1, e_2, e_3, e_4, e_5 >$	**Probability**
$< 1, 1, 0, 0, 0 >$	$p_1 p_2 q_3 q_4 q_5$
$< 0, 0, 0, 1, 1 >$	$q_1 q_2 q_3 p_4 p_5$
$< 1, 1, 1, 0, 0 >$	$p_1 p_2 p_3 q_4 q_5$
$< 1, 1, 0, 1, 0 >$	$p_1 p_2 q_3 p_4 q_5$
$< 1, 1, 0, 0, 1 >$	$p_1 p_2 q_3 q_4 p_5$
$< 1, 0, 1, 0, 1 >$	$p_1 q_2 p_3 q_4 p_5$
$< 1, 0, 0, 1, 1 >$	$p_1 q_2 q_3 p_4 p_5$
$< 0, 1, 1, 1, 0 >$	$q_1 p_2 p_3 p_4 q_5$
$< 0, 1, 0, 1, 1 >$	$q_1 p_2 q_3 p_4 p_5$
$< 0, 0, 1, 1, 1 >$	$q_1 q_2 p_3 p_4 p_5$
$< 1, 1, 1, 1, 0 >$	$p_1 p_2 p_3 p_4 q_5$
$< 1, 1, 1, 0, 1 >$	$p_1 p_2 p_3 q_4 p_5$
$< 1, 1, 0, 1, 1 >$	$p_1 p_2 q_3 p_4 p_5$
$< 1, 0, 1, 1, 1 >$	$p_1 q_2 p_3 p_4 p_5$
$< 0, 1, 1, 1, 1 >$	$q_1 p_2 p_3 p_4 p_5$
$< 1, 1, 1, 1, 1 >$	$p_1 p_2 p_3 p_4 p_5$

Let boolean variable $x_i = 1$ denotes the event that component i is operational and $x_i = 0$ the event that component i has failed, than the state of system can be representd by $< x_1, x_2, \ldots, x_n >$.

The probability of each state is $\prod_{i=1}^{n} k_i$, where k_i is the probability p_i that the component i is working if $x_i = 1$ else k_i is the probability $q_i = 1 - p_i$ that the component i has failed if $x_i = 0$.
Let $p_i = Pr\{x_i = 1\}$ and $q_i = Pr\{x_i = 0\}$; the complete state enumeration of operational states of network of Figure 2.4 is shown in Table 2.3. The system reliability is computed as the sum of the entries in the right hand column of the table.
The drawnback of this method is that its computation complexity is exponential in the number of components: complete state enumeration requires the generation of all 2^n states of system, where n is the number of components.

2.9.2 Factoring Method

The factoring method generates the states in a more clever way: factoring method concentrates on the state of an individual link . A basic ingredient in this is the Factoring Theorem:

$$R(G) = p_e R(G \cdot e) + (1 - p_e) R(G \setminus e) \tag{2.28}$$

or

$$R(G) = p_e R(G|e \textit{ is working}) + (1 - p_e) R(G|e \textit{ is not operational}) \tag{2.29}$$

where p_e is the probability that edge e in graph G is working. This is applied for any edge e of G. Factoring, also called pivotal decomposition, was explicitly introduced in [59].
The idea is to choose an arc and break the graph down into two cases:

- assume the edge is working

- assume the edge has failed

For each case a new graph is built by taking into account the behavior of the chosen arc.
Factoring is carried out until all arcs of the network are analysed. However, some simple observations result in improvements. When $G \setminus e$ is failed, any sequence of contractions and deletions results in a failed network, and hence there is no need to factor $G \setminus e$. Moreover, although we may be unable to simplify G with a reliability-preserving transformation, we may well be able to simplify $G \cdot e$ or $G \setminus e$.
Different factoring algorithms have been developed.
This method can be employed using directly graph G without knowing the minimal path set or the minimal cutset.

For example, for the network in Figure 2.4 if we choose the edge e_3 to

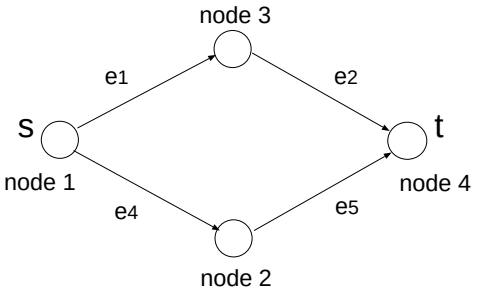

Figure 2.5: e_3 down

factor on we obtain the two graphs shown on Figure 2.5 and in Figure 2.6. Both the resulting graphs are series-parallel, so the efficient formulae for series-parallel reliability can be applied for their analysis.
Let $p_i = Pr\{e_i = 1\}$ and $q_i = Pr\{e_i = 0\}$,

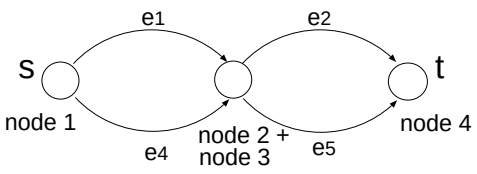

Figure 2.6: e_3 up

- e_3 up: $R_{e_3up} = (1 - q_1q_4)(1 - q_2q_5)$
- e_3 down:$R_{e_3down} = 1 - (1 - p_1p_2)(1 - p_4p_5)$

then the reliability of the system,R, is:

$$R = p_3R_{e_3up} + q_3R_{e_3down} \tag{2.30}$$

2.9.3 Minpaths and Mincuts

Different approaches require that all possible path/ cut sets are first enumerated and then the reliability expression is evaluated [16].
A path is defined as a set of elements such that if these elements are all working, the systems is working. A path is minimal (minpath) if it has not proper subpaths. Let T_i, $1 \leq i \leq n_p$, be a minpath of a system with n_p minpaths, then

$$Reliability\ of\ System\ = Pr(\bigcup_{i=1}^{n_p} T_i = 1) \tag{2.31}$$

The minpaths of the graph of Figure 2.4 are $\{e_1, e_2\}, \{e_4, e_5\}, \{e_1, e_3, e_5\}, \{e_4, e_3, e_2\}$.
A cut is defined as a set of elements such that if these elements are all down, the system is down. A cut is minimal (mincut) if it has not proper subcuts.

Let C_i, $1 \leq i \leq n_c$, be a mincut of a system with n_c mincuts , then

$$Unreliability\ of\ System\ = Pr(\bigcup_{i=1}^{r} C_i = 1) \tag{2.32}$$

The mincuts of the graph of Figure 2.4 are $\{e_1, e_4\}, \{e_2, e_5\}, \{e_1, e_3, e_5\}, \{e_4, e_3, e_2\}$.

2.9.4 The Inclusion-Exclusion Method

Let A_i denote the event that all elements in the i^{th} minimal path are functional, so we can say that A_i represents the event that minimal path i works. $\bar{A}_i$ denote the complement of this event. The probability that the minimal path i works can be expressed as

$$P(A_i) = \prod_{k \in\ element\ of\ k} p_i \tag{2.33}$$

Let n_p be the number of minimal path sets. A system with n_p minimal paths works if and only if at least one of the minimal paths works. System success corresponds to the event $\bigcup_{i=1}^{n_p} A_i$. The reliability of the system is equal to the probability of the union of the n_p minpaths, namely

$$R_s(G) = P(\bigcup_{i=1}^{n_p} A_i) \tag{2.34}$$

Let

$$S_k = \sum_{1 \leq i_1 < \ldots < i_k \leq n_p} P(A_{i_1} \bigcap A_{i_2} \bigcap \ldots \bigcap A_{i_k}) \tag{2.35}$$

Then, S_k represents the sum of the probablilities that any k minimal paths are simultaneously working. By the Inclusion-Exclusion principle (see [40]),

the reliability of the system, which is equal to the probability of the union of the n_p minpaths, can be expressed as:

$$R_s = \sum_{k=1}^{n_p} (-1)^{(k-1)} S_k \tag{2.36}$$

If there are n_p pathsets, then this calculation involves 2^{n_p-1} terms. In some cases, two different intersections of A_i's will correspond to the same event so that these different intersections of A_i's will have the same probability. If one intersection consists of an odd number of A_i's and another intersection consists of an even number of A_i's, they will cancel. Satyanarayana [73] and Satyanarayana and Prabhaka in [74] give algorithms that generate only the non cancelling terms.

For the graph of Figure 2.4, let $p_i = Pr\{e_i = 1\}$ and $q_i = Pr\{e_i = 0\}$, using minpaths $MP_1 = \{e_1, e_2\}, MP_2 = \{e_4, e_5\}, MP_3 = \{e_1, e_3, e_5\}, MP_4 =$

$\{e_2, e_3, e_4\}$, then the system reliability R is:

$$\begin{aligned}
S_1 &= P(MP_1) + P(MP_2) + P(MP_3) + P(MP_4) \\
S_2 &= P(MP_1MP_2) + P(MP_1MP_3) + P(MP_1MP_4) \\
&\quad + P(MP_2MP_3) + P(MP_2MP_4) + P(MP_3MP_4) \\
S_3 &= P(MP_1\ MP_2\ MP_3) + P(MP_1\ MP_2\ MP_4) \\
&\quad + P(MP_1\ MP_3\ MP_4) + P(MP_2\ MP_3\ MP_4) \\
S_4 &= P(MP_1\ MP_2\ MP_3\ MP_4) \\
R_s &= S_1 - S_2 + S_3 - S_4 \\
R_s &= P(e_1e_2) + P(e_4e_5) + P(e_1e_3e_5) + P(e_2e_3e4) - P(e_1e_2e_4e_5) \\
&\quad - P(e_1e_2e_3e_4) - P(e_1e_2e_3e_4) - P(e_1e_3e_4e_5) - P(e_2e_3e_4e_5) \\
&\quad - P(e_1e_2e_3e_4e_5) + P(e_1e_2e_3e_4e_5) + P(e_1e_2e_3e_4e_5) + P(e_1e_2e_3e_4e_5) \\
&\quad + P(e_1e_2e_3e_4e_5) - P(e_1e_2e_3e_4e_5) \\
&= p_1p_2 + p_4p_5 + p_1p_3p_5 + p_2p_3p_4 - p_1p_2p_4p_5 - p_1p_2p_3p_5 \\
&\quad - p_1p_2p_3p_4 - p_1p_3p_4p_5 - p_2p_3p_4p_5 + 2p_1p_2p_3p_4p_5 \qquad (2.37)
\end{aligned}$$

2.9.5 Sum of Disjoint Products Method

Let E_i be the event that all the edges in minpath MP_i are working. We can compute the reliability expression $R(G)$ as:

$$R(G) = P(E_1) + P(\bar{E}_1E_2) + \ldots + P(\bar{E}_1\bar{E}_2 \ldots \bar{E}_{p-1}E_p) \qquad (2.38)$$

where p is the total number of minpaths, $\bar{E}_i$ denotes the complement of event E_i, and $P()$ is the probability function.
SDP methods involve adding probabilities; however, the calculation of each constituent probability is, in general, quite involved. It is also important to

emphasize that the effectiveness of these methods can be highly dependent on the specific ordering given to the events E_i.
In order to apply SDP technique the problem is to compute $P(\sigma)$ where $\sigma_i = \bar{E}_1\bar{E}_2 \ldots \bar{E}_{i-1}E_i$. If we define the events in terms of minpaths, σ_i represents the events that all components in minpath MP_i are working and each minpath $MP_1, MP_2, \ldots, MP_{i-1}$contains at least one failed component. If E_j and E_i are independent for $1 \leq j \leq i$, then

$$P(\sigma_i) = \prod_{k \in MP_i} p_k \prod_{j=1}^{i-1} [1 - \prod_{s \in MP_J} p_s] \tag{2.39}$$

where p_k is the reliability of edge k. This equation defines how each minpath MP contributes in the final reliability, considering a parallel structure or disjoint path (without common elements).
If the minpaths share some common edges the solution of $P(\sigma_i)$ appears quite involved.
In this case we compute $P(\sigma_i)$ as

$$P(\sigma_i) = P(E_i)P(\bar{E}_1|E_i)P(\bar{E}_2|E_iE_1)\ldots P(\bar{E}_{i-1}|E_iE_1E_2\ldots E_{i-2}) \tag{2.40}$$

Let D_j be a conditional event such that minpath MP_j is down given that minpaths $MP_i, MP_1, \ldots MP_{j-1}$ are working. We have

$$P(\sigma_i) = P(E_j) \prod_{j=1}^{i-1} P(D_j) \tag{2.41}$$

$P(E_i)$ represents the probability of event E_i and can be simply computed since failures are assumed to be statistically independent. Opposite, evaluating $P(D_j)$ is a complex problem in the system reliability field. For the

graph of Figure 2.4, let $p_i = Pr\{e_i = 1\}$ and $q_i = Pr\{e_i = 0\}$, using minpath, then the system reliability R is:

$$\begin{aligned} R &= P(e_1e_2) + P(\overline{e_1e_2}\ e_4e_5) + P(\overline{e_1e_2}\ \overline{e_4e_5}\ e_1e_3e_5) \\ &\quad + P(\overline{e_1e_2}\ \overline{e_4e_5}\ \overline{e_1e_3e_5}\ e_2e_3e_4) \\ &= p_1p_2 + q_1p_4p_5 + q_2p_4p_5 + q_2q_4p_1p_3p_5 + q_1q_5p_2p_3p_4 \end{aligned} \quad (2.42)$$

If we replace q_i with $1 - p_i$ we have the same result obtain with Inclusion-Exclusion method in equation 2.37.

Chapter 3

Binary Decision Diagrams

3.1 Boolean Expression

Boolean expressions consist of a set of *Boolean variables* $x, y, \ldots$, the *constant true* 1 and the *constant false* 0 and the *operator of conjunction* $\wedge$, *disjunction* $\vee$, *negation* $\neg$, *implication* $\Rightarrow$ and *bi-implication* $\Leftrightarrow$ [9].
Formally, the following grammar generates Boolean expression:

$$t ::= x \mid 0 \mid 1 \mid \neg t \mid t \vee t \mid t \wedge t \mid t \Rightarrow t$$

where x is a Boolean variable. This is the *abstract syntax* of Boolean expression, but usually in order to solve ambiguites a concrete syntax is used that includes parentheses. Moreover, it is assumed that operators follow the common convention priority. Starting from the highest the priorities are:

$$\neg,\ \wedge,\ \vee,\ \Leftrightarrow,\ \Rightarrow$$

x	$\neg x$
0	1
1	0

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	$x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	$x_1 \Rightarrow x_2$
0	0	1
0	1	1
1	0	0
1	1	1

Figure 3.1: Truth Tables

Hence, for example

$$x_1 \wedge \neg x_2 \vee x_3 \Leftrightarrow x_4 \ = ((x_1 \wedge (\neg x_2)) \vee x_3) \Leftrightarrow x_4$$

A Boolean expression with variables $x_1, \ldots, x_n$ denotes for each assignment of truth values to the variables itself a truth value according to the standard truth tables, see Figure 3.1. Truth assignments are written as sequences of assignments of values to variables, e.g., $[0/x_1; 1/x_2; 0/x_3; 1/x_4]$ which assigns 0 to x_1 and x_3, 1 to x_2 and x_4.

3.2 Normal Forms

3.2.1 Disjunctive Normal Form (DNF)

A statement is in *Disjunctive Normal Form (DNF)* if it is expressed as a series of disjunctions (sequence of ORs) when each disjunct is either a simple proposition (or negation of) or a conjunction (ANDs) of simple propositions

and negations of simple propositions.

$$(t_1^1 \wedge t_2^1 \wedge \ldots \wedge t_{n_1}^1) \vee \ldots \vee (t_1^i \wedge t_2^i \wedge \ldots \wedge t_{n_i})$$

where each t_j^k is either a variable x_j^k or a negation of a variable $\neg x_j^k$.
A more compact representation is the following:

$$\bigvee_{k=1}^{i} (\bigwedge_{j=1}^{n_k} t_j^k)$$

3.2.2 Conjunctive Normal Form (CNF)

A statement is in *Conjunctive Normal Form (CNF)* if it is is expressed as a series of conjunctions (sequence of ANDs) when each conjunct is either a simple proposition (or negation of) or a disjunction (ORs) of simple propositions and negations of simple propositions.

$$(t_1^1 \vee t_2^1 \vee \ldots \vee t_{n_1}^1) \wedge \ldots \wedge (t_1^i \vee t_2^i \vee \ldots \vee t_{n_i})$$

where each t_j^k is either a variable x_j^k or a negation of a variable $\neg x_j^k$.
A more compact representation is the following:

$$\bigwedge_{k=1}^{i} (\bigvee_{j=1}^{n_k} t_j^k)$$

3.3 Binary Decision Diagrams

The Binary Decision Diagram [71] representing a function F with a set of n boolean variables $x_1, x_2, x_3, ...x_n$, is an acyclic direct binary graph. We have two types of vertices:

- **terminal nodes**: node 0 and node 1 representing the *constant false* and the *constant true*.
- **non terminal nodes** : a set of variable nodes x_i of out-degree two. The two outgoing arcs are given by two functions $F(x_i = 0)$ (right edge or 0-edge) and $F(x_i = 1)$(left edge or 1-edge).

In picture the left edge is indicated by solid lines and right edges by dashed lines. The Shannon decomposition is defined in terms of the ternary If Then Else (ITE) connective

$$F = x_i \cdot F_{x_i=1} + \bar{x} \cdot F_{x_i=0} \tag{3.1}$$

where x_i is one of the decision variables. The functions $F_{x_i=1}$ and $F_{x_i=0}$ are Boolean functions evaluated at $x_i = 1$ and $x_i = 0$ respectively. $F_{x_i=1}$ and $F_{x_i=0}$ contain one variable less than F.
We apply recursively the decomposition on F until all the variables $x_1, x_2.....x_n$ have been considered.

3.3.1 BDD

The sequence of decompositions can be represented in graphical form using a binary tree. Each non-sink node is labeled with a Boolean variable v and has two out-edges labeled 1 (if then) and 0 (or else). Each non-sink node represents the Boolean function corresponding to its 1-edge if $v = 1$, or the Boolean function corresponding to its 0-edge if $v = 0$. An un-simplified BDD is basically a Binary Decision Tree which contains $2^n - 1$ non-terminal nodes. For example the function $f(x_1, x_2, x_3)$ depicted by the truth table 3.1 can be represented by the Binary Decision Tree in Figure 3.2.
Starting from root node and traversing the tree if we follow the branch

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Table 3.1: True table

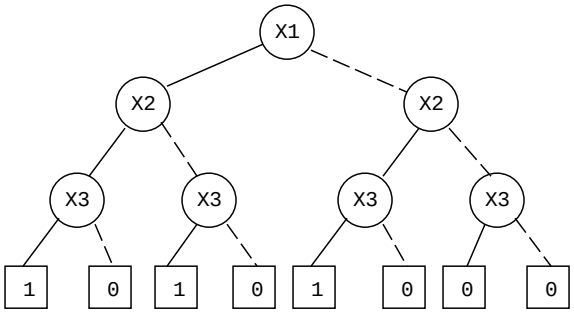

Figure 3.2: Binary Decision tree

representing variable value we reach the terminal node 0 or the terminal node 1 according with the function value for the assignment.

3.3.2 Ordered BDD (OBDD)

An Ordered BDD (OBDD) is a BDD in which each variable is encountered no more than once in any path. The order of variables is the same along each path. The tree in Figure 3.2 considers the variables in the order $x_1 \prec x_2 \prec x_3$ where $x_i \prec x_j$ means that inside the tree the variable x_i is closer to the root node than x_j

3.3.3 Reduced OBDD (ROBDD)

A Reduced OBDD (ROBDD) is an OBDD that is reduced by three reduction rules: deletion rule and terminal and non-terminal merging rules. These reduction rules remove redundancies from the OBDD

- **Merge equivalent leaves**. Remove duplicate terminals: eliminate all but one terminal vertex with a given label and redirect all arcs to the eliminated vertices to the remaining one. An example depicted in Figure 3.3 is obtained from the binary decision tree of Figure 3.2. In this case only one terminal node with label 1 and one with label 0 remain all the other are deleted.

- **Merge Rule**: If two or more non-terminal nodes of the same label have the same 0 and 1 successors i.e. their left and right sons are equivalent then they can be merged in a single node as shown in Figure 3.4. Remove duplicate nonterminals: if nonterminals u and v have $var(u) = var(v)$, $low(u) = low(v)$ and $high(u) = high(v)$,

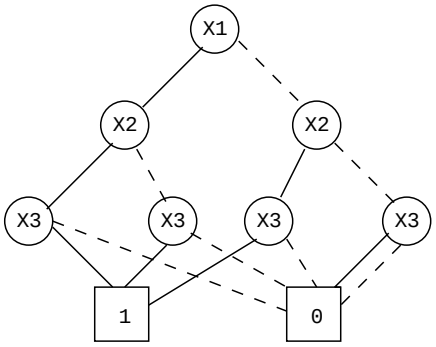

Figure 3.3: Merge equivalent leaves

eliminate one of the two vertices and redirect all incoming arcs to the other vertex.

Starting from the BDD in Figure 3.3, applying this reduction, we obtain the BDD in Figure 3.4. For example three of the four subtrees rooted in the node x_3 are equal so two of them can be deleted. The outgoing edges of x_2 are modified in order to point at the only remained subtree.

- **Deletion Rule**: If one or more non-terminal nodes are such that both their branches corresponding to 0 and 1 lead to a non-terminal successor node or to a terminal node then this node can be deleted.
  We remove redundant tests: if nonterminal vertex v has low(v) = high(v), eliminate v and redirect all incoming arcs to low(v)
  If we apply this rule to the BDD in Figure 3.4 we obtain the BDD in Figure 3.5. We remove the outgoing edges of the node x_2 if x_1 is true so that the node x_1 points to x_3.
  Besides if we consider x_1 and x_2 false the outgoing edges of node x_3

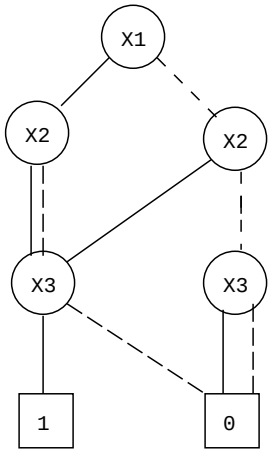

Figure 3.4: Merge Rules

can be deleted: now node x_2 is redirected to terminal node 0. We remove also non connected nodes x_2 and x_3 .

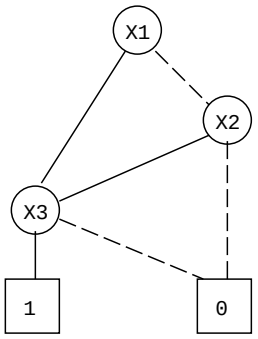

Figure 3.5: Merging Rules

Canonicity of ROBDDs

Two Boolean functions are equivalent iff their reduced OBDDs (ROBDD) are identical with respect to any variable ordering [30].

Theorem 1 *For any Boolean function f, there is a unique (up to isomorphism) ROBDD representing f and any other OBDD representing f contains more nodes.*

In other words, if G_0 and G_1 are two ROBDD representations of a Boolean function f, then G_0 and G_1 are isomorphic.
Two ROBDDs G_0 and G_1 are isomorphic if there is a one-to-one mapping σ from the vertexes of G_0 onto the vertexes of G_1.
Suppose that v_0 is a vertex of G_0 and $v_1 = \sigma(v_0)$ is the corresponding vertex in G_1. One of the following must hold:

- both v_0 and v_1 are terminal vertexes with

$$value(v_0) = value(v_1)$$

- both v_0 and v_1 are non-terminal vertexes with:
  - $index(v_0) = index(v_1)$,
  - $\sigma(low(v_0)) = low(v_1)$
  - $\sigma(high(v_0)) = high(v_1)$

A canonical form is obtained by applying the transformation rules until no further application is possible.
Due to the canonicity of ROBDD different boolean operations can be efficient implementated. Possible applications can be i.e.:

- checking equivalence: verify isomorphism between ROBDDs,
- non satisfiability: verify if ROBDD has only one terminal node, labeled by 0 and
- tautology: verify if ROBDD has only one terminal node, labeled by 1

3.3.4 Construction of Binary Decision Diagrams

BDDs are constructed by first creating individual BDDs for each variable of the function and then using the APPLY operation to build the BDD from these individual variable BDDs. The APPLY operation requires two BDDs and a Boolean operation to be applied to these BDDs as inputs and produces a resulting BDD as output. One efficient way to implement APPLY as described by K. S. Brace, R. L. Rudell, and R. E. Bryant in [29] is to use the If-Then-Else (ITE) operator. The ITE operator is a recursive form of the Shannon expansion theorem
In a BDD, a function is represented recursively with triples $(x_i; f|_{x_i=1}; f|_{x_i=0})$,

which are a consequence of Shannon's decomposition theorem :

$$f = x_i \cdot f|_{x_i=1} + \bar{x_i} \cdot f|_{x_i=0} \tag{3.2}$$

The If-Then-Else or ITE is a three-variable Boolean operator, defined as:

$$ITE(f; g; h) = f \cdot g + \bar{f} \cdot h \tag{3.3}$$

With ITE, all two-variable Boolean operations can be implemented (see Table 3.2). Shannon's decomposition theorem can be shortly written as $f = ITE(x_i; f|_{x_i=1}; f|_{x_i=0})$, and therefore it forms a basic operation in BDD.

$$F = ite(x_1, F_{x_1=1}, F_{x_1=0}) \tag{3.4}$$

$$ite(F, G, H) = (F \wedge G) \vee (\bar{F} \wedge H) \tag{3.5}$$

All of the possible binary Boolean operators can be implemented as ITE expressions as illustrated in Table 3.2.

Table	Name	Expression	ITE
0000	0	0	0
0001	AND(F,G)	$F \cdot G$	ite(F,G,0)
0010	$F > G$	$F \cdot \bar{G}$	ite(F,$\bar{G}$,0)
0011	F	F	F
0100	$F < G$	$\bar{F} \cdot G$	ite(F,0,G)
0101	G	G	G
0110	XOR(F,G)	$F \oplus G$	ite(F,$\bar{G}$,G)
0111	OR(F,G)	F + G	ite(F,1,G)
1000	NOR(F,G)	$\overline{F + G}$	ite(F,0,$\bar{G}$)
1001	XNOR(F,G)	$\overline{F \oplus G}$	ite(F,G,$\bar{G}$)
1010	NOT(G)	$\bar{G}$	ite(G,0,1)
1011	F $\geq G$	F+ $\bar{G}$	ite(F,1,$\bar{G}$)
1101	F$\leq G$	$\bar{F}$ +G	ite(F,G,1)
1110	NAND(F,G)	$\overline{F \cdot G}$	ite(F,$\bar{G}$,1)
1111	1	1	1

Table 3.2: Two variable functions realized with ITE

3.3.5 How to Represent a Graph by Means of BDD

The BDD is assembled using the logic operation ricursively in a bottom-up fashion. Each element in the network has associated single-node BDD (see Figure 3.6). For example, the BDD for element a and element b are shown in Figure 3.7. A BDD is constructed for each element, and then combined according to the structure of the network. The BDD for "a or b" is costructed by applying the OR Boolean function to the BDD for element a and the BDD for element b. Since a is first in the relation, it is considered as the "root" node. The union of b BDD is with each "child" node of a.
First consider the terminal node 0 of element a in Figure 3.7. Accordingly, since $0 + X = X$ and $1 + X = 1$, the left child reduces to b and the right child reduces to 1 as shown in Figure 3.8.
Now consider the intersection operation applied to elements a and b as shown in Figure 3.9. Note that $0 \cdot X = 0$ and that $1 \cdot X = X$. Thus, the reduced BDD for "a and b" is shown in Figure 3.9.

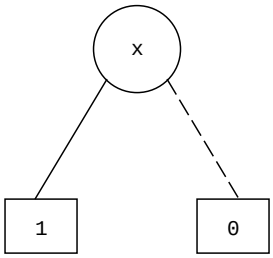

Figure 3.6: Single component BDD

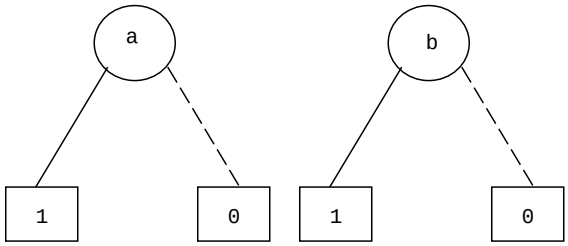

Figure 3.7: a and b node BDD

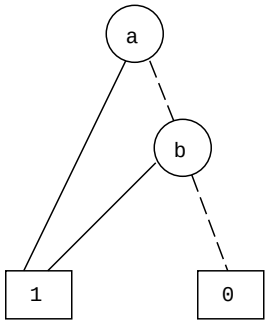

Figure 3.8: a OR b

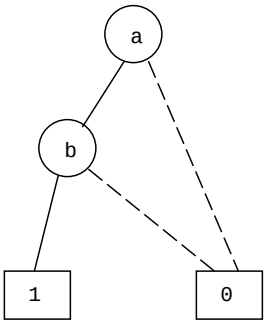

Figure 3.9: a AND b

3.4 Complexity

While BDDs are compact representations for many functions, they can reach exponential size with regard to the number of inputs for some functions. There are several reasons why this occurs. One of the reasons is that a bad ordering was chosen for the variables when the BDD was constructed. The size attained by a BDD is influenced greatly by the variable ordering chosen, however finding the best variable ordering during BDD construction is NP-complete.

Given a function with n inputs, one input ordering may require exponential number of vertices in ROBDD, while other may be linear in size.

It has also been shown that ROBDDs can be extremely sensitive to the variable ordering. Bryant [30] showed that the logic function of $2n$ variables

$$a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3 \tag{3.6}$$

with order

$$a_1 \prec b_1 \prec a_2 \prec b_2 \prec a_3 \prec b_3$$

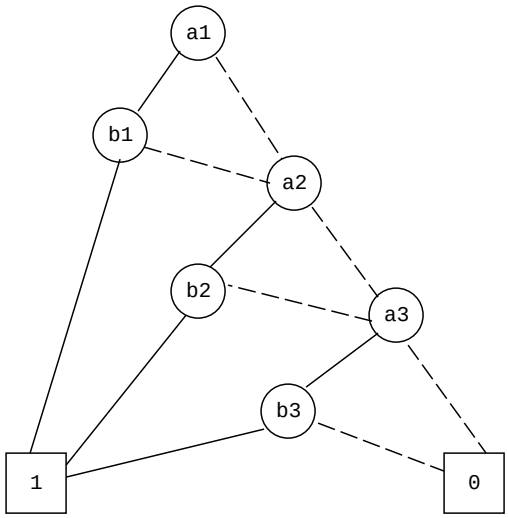

Figure 3.10: First ordering

produces a graph with $2n + 2$ nodes as showed in Figure 3.10, while with order

$$a_1 \prec a_2 \prec a_3 \prec b_1 \prec b_2 \prec b_3$$

produces a graph with 2^{n+1} nodes as showed in Figure 3.11.

The poor ordering can significantly affect the size of the BDD, thus the reliability analysis time for large systems. Currently there is no rule-based means of determining the best way of ordering basic events for a given structure, but fortunately heuristics can usually be used to find a reasonable variable ordering.

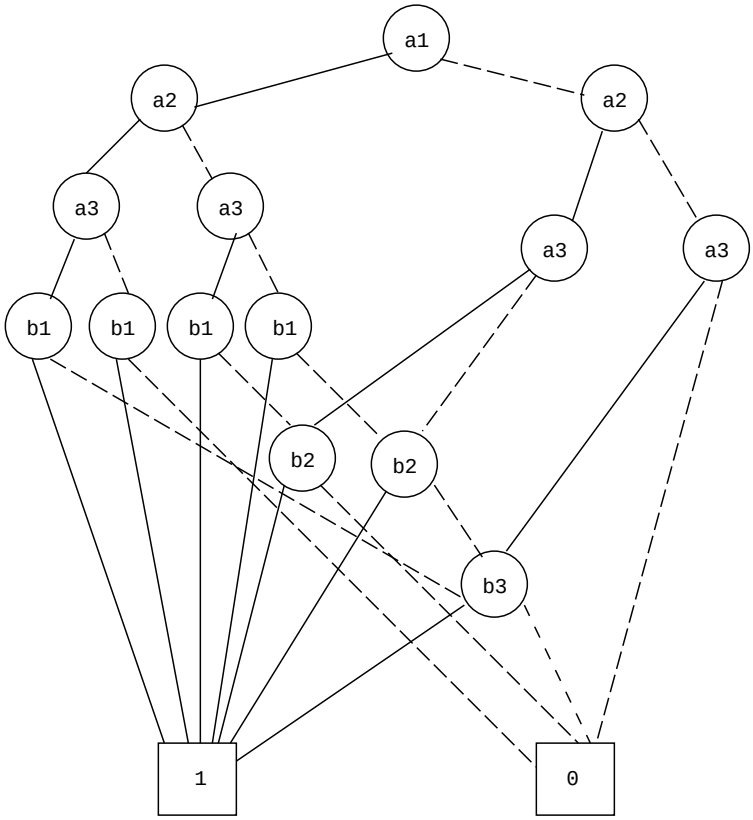

Figure 3.11: Second ordering

Chapter 4

Network Analysis

A peculiar feature of the modern organization of the human life is the increasing level of the interconnectivity. Social, economical and technological structures tend to increase in dimension and to aggregate in complex interconnected systems. Connected systems can be abstracted in the form of networks where the vertices are the entities of the system and the edges the physical or relational links among them. One relevant property of networks, that make them a preferential structure both in natural and technological systems, is that the connection between any two nodes of the networks can be usually achieved through a number of redundant paths, thus making the connection intrinsically reliable. For this reason, the reliability of networks has always been a major concern since the earliest times of reliability engineering [72, 5].

A network can be represented by means of a graph whose elements (both vertices and edges) are considered as binary objects characterized by a working (up) or a failed (down) condition. If a probability measure is assigned to the up or down state of each element, a probability measure

associated to the connectedness of the whole graph can be computed. The network reliability is defined as the probability that all nodes, or a subset of the nodes, of the graph communicate through at least one path of working edges [19]. The computational techniques proposed in the literature to solve the network reliability problem always assume that the network elements are statistically independent and can be classified in two main categories; *i)* - approaches in which the desired network reliability measure is directly calculated (series-parallel reduction [20], pivotal decomposition using keystone components [66, 46]) and *ii)* - approaches in which all possibilities through which two specified nodes can communicate (or not communicate) with each other are first enumerated (path/cut set search [12, 58]) and then the reliability expression is evaluated, resorting to different techniques [5], like inclusion-exclusion method or sum of disjoint products [4, 80, 69, 49].

In the last decades, Binary Decision Diagrams (BDD) [30] have provided an extraordinarily efficient method to represent and manipulate Boolean functions [31], and have been also exploited to model the connectivity of a Boolean networks. This chapter discusses various approaches that can be followed to compute the reliability of a network starting from a BDD representation and presents a preliminary tool for the analysis.

4.1 Network Analysis Algorithms

Various algorithms have been proposed in the literature to evaluate the network reliability. In this section, we first discuss the representation of Boolean functions based on BDD, and then we illustrate three different algorithms that exploit the BDD representation.

4.1.1 Calculating Reliability Using BDD

Each path through the BDD from the root to a leaf node represents a disjoint combination of component failures and non-failures. A path with a leaf node labeled with a 1 leads to system operation. Probabilities associated with arcs on each path are either $(1-p)$ (component failure probability) for the right branch or (p) for the left branch. The system unreliability is given by the sum of the probabilities for all paths from the root to a leaf node labeled 1.
Each path is disjoint by construction, so no subtraction is needed.

Consider a BDD branch

$$G = ite(x, G_1, G_2)$$

- If $G = 0$ then $P(G) = 0$ else
- If $G = 1$ then $P(G) = 1$
- Otherwise,

$$P(G) = p_x P(G_1) + (1 - p_x) P(G_2)$$

When x is the root node of the BDD, $P(G)$ gives the reliability of the system.

Furthermore, if we assign to every variable x_i a probability p_i of being true ($1 - p_i$ false), we can compute the probability $P\{F\}$ of the function F by applying recursively Equation 4.1.

$$\begin{aligned} P\{F\} &= p_1 P\{F_{x_1=1}\} + (1-p_1) P\{F_{x_1=0}\} \\ &= P\{F_{x_1=0}\} + p_1 (P\{F_{x_1=1}\} - P\{F_{x_1=0}\}) \end{aligned} \tag{4.1}$$

In network reliability analysis different approaches based on the use of BDD have already appeared in the literature [19] and can be categorized in three classes:

i) - algorithms based on the direct application of factoring algorithms [75, 46];

ii) - algorithms based on the search for minpaths (or mincuts);

iii) - algorithms based on the direct generation of the BDD through visiting algorithms on the graph [83].

4.1.2 K-terminal Reliability by Pivotal Decomposition

Factoring algorithms using a recursive pivotal decomposition have been exploited since a long time [5]. The direct derivation of the BDD structure from the network factorization has been presented in [75] and revisited in [46]. The term $f|_{x_i=1}$ in Equation (3.2) in Section 3.3.4 is the graph obtained by putting the edge x_1 to *true*, i.e. by considering that edge x_1 is perfectly reliable. This operation is usually called *contraction* [75] since it corresponds to remove the edge by joining the source and destination vertices. Similarly, the term $f|_{x_i=0}$ in Equation (3.2) is the graph obtained by putting the edge x_1 to *false*, i.e. by considering that edge x_1 as failed or open. This operation is usually called *deletion* [75] since it corresponds to removing the edge by keeping separated the source and destination vertices.

In term of graph factorization, the Shannon's decomposition (3.2) can be rewritten in the following way:

$$G = e_i\, T \vee \overline{e}_i\, F \tag{4.2}$$

where T is the graph obtained from G by contracting edge e_i and F is the graph obtained from G by deleting edge e_i. In the graphical representation of (4.2) as a binary tree we follow the usual convention of drawing the *contraction* as the left branch in solid line and the *deletion* as the right branch in dotted line.

Given a subset of $K\,(\leq\ N_V)$ vertices, the K-terminal connectedness (or K-connectedness) is a Boolean function that evaluates to true if the K vertices are connected (following the direction of the arcs in a directed network) and can be represented by a BDD resorting to a recursive application of the graph reduction step (Equation 4.2), following an (arbitrarily) ordered sequence of edges. At each step of the decomposition, we check the K-connectedness on the T and on the F reduced graphs obtained by applying one reduction step in Equation 4.2. If the K-connectedness on the T subgraph evaluates identically to true (for any combination of the remaining variables) the terminal 1 leaf of the BDD is reached, if the K-connectedness on the F subgraph evaluates identically to false the terminal 0 leaf of the BDD is reached, otherwise we proceed with a further decomposition step in Equation (4.2) by pivoting with respect to the subsequent edge in the ordered list. In directed networks the connectedness must be verified by following the direction of the arcs. We illustrate the construction of the BDD using the factorization algorithm on a simple example.

Example: Bridge Network 1 - A bridge network with directed arcs has the configuration presented in Figure 4.1 and we want to compute the `(s,t)` reliability. By assigning the ordering: $(e_2 \prec e_5 \prec e_3 \prec e_1 \prec e_4)$, the recursive application of the pivotal decomposition can be pictorially represented in Figure 4.2 (where contracted edges are drawn in bold line to keep track of the direction of the arcs).

The graph of Figure 4.2 can be represented as a binary tree in Figure 4.3

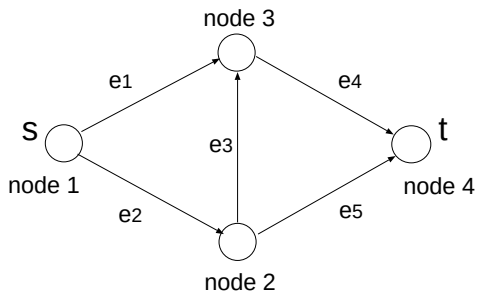

Figure 4.1: A directed bridge network

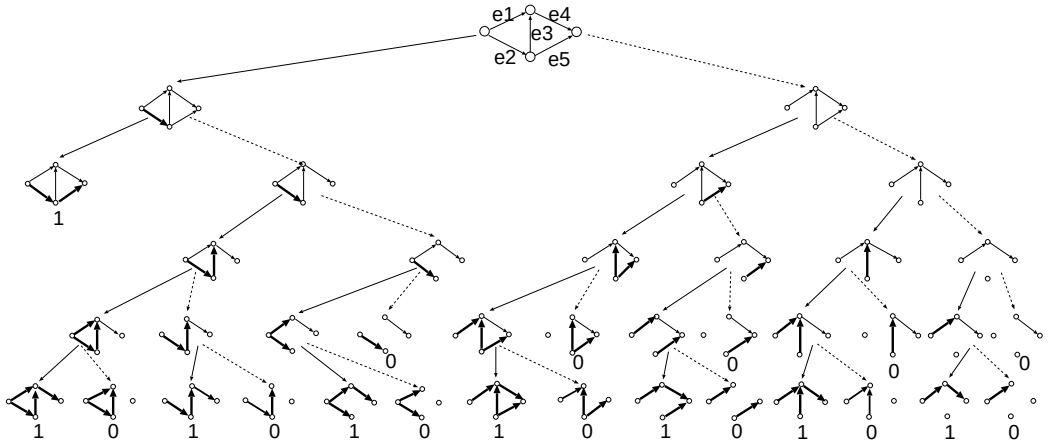

Figure 4.2: Factorization of the graph of Figure 4.1

where the labels into the circles evidence the pivoting logical variables at each step. By applying the reduction rules for BDD's and folding identical subtrees, the binary tree of Figure 4.3 reduces to the ROBDD of Figure 4.4.

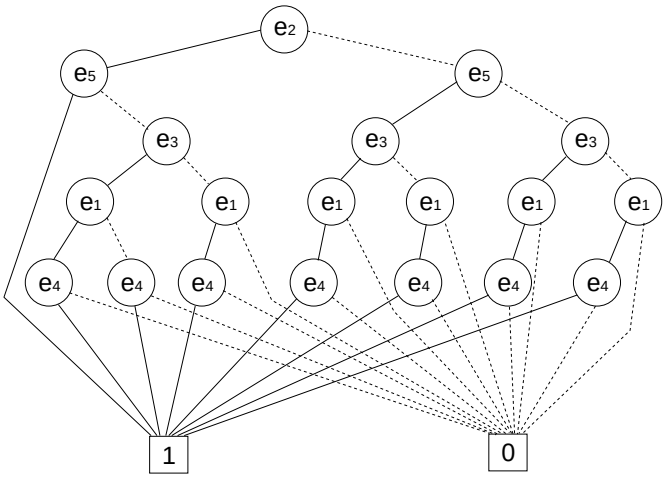

Figure 4.3: BDD-like representation of Figure 4.2

The computation of the probability of the BDD of Figure 4.4, can proceed recursively by resorting to Equation (4.1). Recursive application of Equation (4.1) is pictorially shown in Figure 4.5. Computation of the probability values at the intermediate nodes $(pr1, pr3, pr5)$ provides the value of the network reliability $R_{(\mathtt{s},\mathtt{t})} = pr2$ given in Formula (4.3).

$$\begin{aligned} R_{(\mathtt{s},\mathtt{t})} &= p_1p_4 + p_2p_3p_4 + p_2p_5 - p_1p_2p_3p_4 - p_2p_3p_4p_5 \\ &\quad -p_1p_2p_4p_5 + p_1p_2p_3p_4p_5 \end{aligned} \tag{4.3}$$

The K-terminal Reliability Algorithm

Given a graph G and the specification of the K-terminal reliability problem, we need to define a function `k_t_connectivity(G)`, that returns *true* or

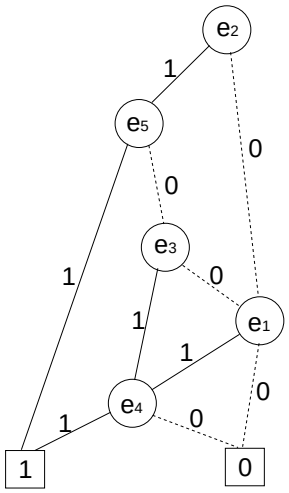

Figure 4.4: BDD representation of the bridge network of (4.1)

false whether G is K-terminal connected or not (taking also into account the possible direction of the arcs). Given the K-terminal connectivity function, the BDD construction algorithm proceeds according to the pseudocode of Algorithm 4.1. In the algorithm, G' is a graph initially empty, which is progressively filled with the contracted edges so that when the function `k_t_connectivity(G')` evaluates to true, the terminal 1 leaf is reached. The symbol `G/e` indicates the deletion of the arc e from G.

4.1.3 2-terminal Reliability via Minpath Analysis

Given a network $G = (V, E)$ and two nodes `(s,t)`, following the minpath definition [54] we can compute the network reliability by means of minpath list.

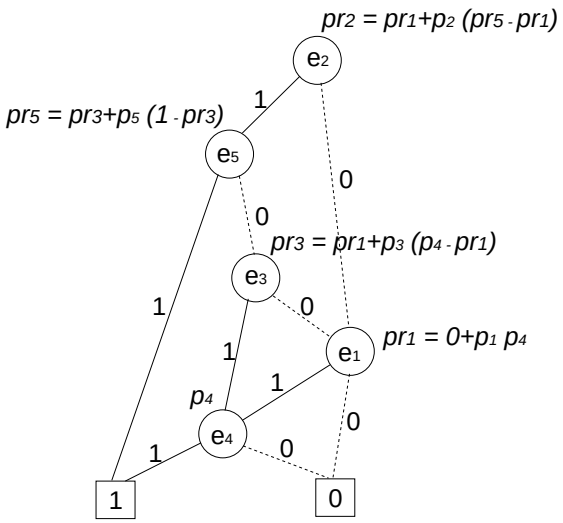

Figure 4.5: Probability computation from the BDD of Figure 4.4

Minpath Search Algorithm

Enumeration of all minimal paths between a terminal pair of a given graph is a fundamental step while computing terminal reliability. Algorithm 4.2 shows the algorithm used in order to search the minpaths.

Let $H_1, H_2, \ldots, H_h$ be the h minpaths between `s` and `t`. All the elements of a minpath must be working, therefore the elements of the minpath are logically connected in AND. However the network connectivity $S_{(\mathtt{s},\mathtt{t})}$ can be represented as the logical OR of its minpaths, since it is sufficient that any one of the minpaths is operational to make the network working.

$$S_{(\mathtt{s},\mathtt{t})} = H_1 \vee H_2 \vee \ldots H_h \tag{4.4}$$

Algorithm 4.1 Generation of the BDD via a recursive factoring algorithm

Require: $G = original\ graph, G' = empty$
Ensure: constr_bdd(G, G')

```
1: Let e be an edge in G and not in G'
2: if k_t_connectivity(G') then
3:    return 1
4: else
5:    if (! k_t_connectivity(G)) then
6:       return 0
7:    else
8:       T= constr_bdd(G,G'∨ e);
9:       E= constr_bdd(G/e,G');
10:      return BDD(e,T,E);
11:   end if
12: end if
```

The two terminal reliability can be calculated as:

$$R_{(\mathtt{s},\mathtt{t})} = P\{S_{(\mathtt{s},\mathtt{t})}\} = P\{H_1 \vee H_2 \vee \ldots H_h\} \tag{4.5}$$

Formula 4.5 shows that the network reliability can be evaluated as the probability of the union of non-disjoint events. It is known [5, 77] that Expression (4.5) can computed by different techniques:

- The inclusion-exclusion expansion algorithm [77];
- Sum of disjoint products [4, 58, 49];
- Binary Decision Diagram (BDD) [30, 19].

We only discuss the last alternative on the network of Figure 4.1.

Example: Bridge Network 2 - The minpath search Algorithm 4.2 applied to the network of Figure 4.1, produces the following list of minpaths

Algorithm 4.2 Minpath search algorithm

```
1: search_path (from, to)
2: this_path.insert(from);
3: if from==to then
4:    set_path.push_back(this_path);
5:    el_path.push_back(path);
6:    num_path++;
7:    this_path.erase(from);
8:    path.pop_back();
9: end if
10: node = first element of adjacency list of node from
11: while node != NULL do
12:    if node not in this_path then
13:       path.push_back(edge connecting node and node_to );
14:       search_path(node_to,to)
15:    end if
16:    node=next element of the list
17: end while
18: if !path.empty() then
19:    path.pop_back();
20:    this_path.erase(from);
21: end if
```

(that can be also verified by visual inspection):

$$H_1 = \{e_1e_4\} \;\; H_2 = \{e_2e_3e_4\} \;\; H_3 = \{e_2e_5\} \tag{4.6}$$

Replacing Eq. (4.6) into Eq. (4.4), the network connectivity can be expressed as:

$$S_{(\mathtt{s},\mathtt{t})} = \{e_1e_4\} \vee \{e_2e_3e_4\} \vee \{e_2e_5\} \tag{4.7}$$

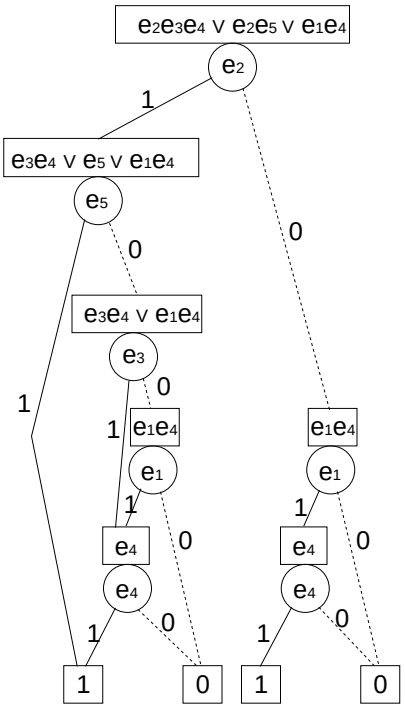

Figure 4.6: BDD Representation of Equation (4.9)

By using the same ordering as the preceding section, the Shannon's

decomposition of the Boolean connectivity expression (4.7) is built step by step in Figure 4.6. If the leaves 1 and 0 are merged and the identical subtrees (like those generated at node e_1e_4) are folded the same ROBDD of Figure 4.4 (and hence the same reliability expression) is obtained.

Example: Symmetric Network 1 - The minpath search algorithm applied to the network of Figure 4.7, assuming `s`=v_1 and `t`=v_5, shows that the graph possesses 5 minpaths (listed in order of their rank):

$$H_1 = \{6, 8\};\ H_2 = \{6, 3, 4\};\ H_3 = \{1, 7, 4\};\ H_4 = \{1, 2, 8\};\ H_5 = \{1, 2, 3, 4\} \tag{4.8}$$

Replacing (4.8) into (4.4), the network `(s,t)`-connectivity can be expressed

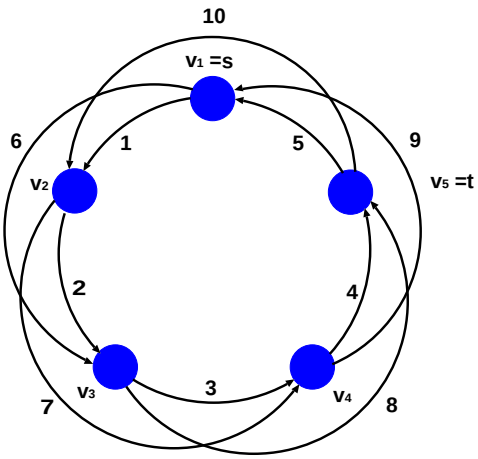

Figure 4.7: A symmetric directed network

as:

$$S_{(\mathtt{s},\mathtt{t})} = (6 \wedge 8) \vee (6 \wedge 3 \wedge 4) \vee (1 \wedge 7 \wedge 4) \vee (1 \wedge 2 \wedge 8) \vee (1 \wedge 2 \wedge 3 \wedge 4) \quad (4.9)$$

The connectivity expression (4.9) is a Boolean function for which the Shannon's decomposition can be applied and the related BDD constructed. The default construction takes the variable sequencing in the same order in which the minpaths are generated. From the BDD, the `(s,t)`-reliability can be finally computed from Equation 4.1.

A very similar approach to the one described in this section, could be followed by generating the mincuts instead of the minpaths, where a mincut is a subset of edges whose failure disconnects the `s` and `t` nodes.

4.1.4 2-terminal Reliability by Graph Visiting Algorithms

The BDD representation of the 2-terminal connectivity of a graph, can be directly derived without passing from a preliminary search for the minpaths or mincuts. In [83] an algorithm is proposed that generates the BDD directly, via a recursive visit on the graph, without explicitly deriving the Boolean expression. The algorithm is sketched in Algorithm 4.3, in pseudocode form.

The Implemented Algorithm

Given a graph $G = (V, E)$ and two nodes `(s,t)`, the algorithm starts from `s` and visits the graph (according to a given but arbitrary visiting strategy) until `t` is reached. The BDD construction starts recursively once the sink node `t` is reached. The BDD's of the nodes along a path from `s` to `t` are combined in AND, while if a node possesses more than one outgoing

Algorithm 4.3 Generation of the BDD via a recursive visit of the graph.

Require: start_node
Ensure: bdd_gen(start_node)

```
1: T_bdd = 0
2: set start_node in this_path
3: for edge_i in the set of edges starting from start_node do
4:    next_node = the other end of edge_i
5:    if next_node == sink_node then
6:       subpath_bdd = edge_i_bdd
7:    else if next_node is already in this_path then
8:       continue;
9:    else
10:      subpath_bdd = bdd_gen(next_node) and edge_i_bdd
11:      T_bdd = T_bdd or subpath_bdd
12:   end if
13: end for
14: clear start_node in this_path
15: return T_bdd
```

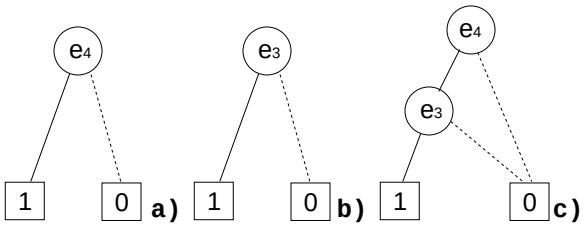

Figure 4.8: a) BDD arc e_4; b) BDD arc e_3; c) BDD $e_4 \wedge e_3$

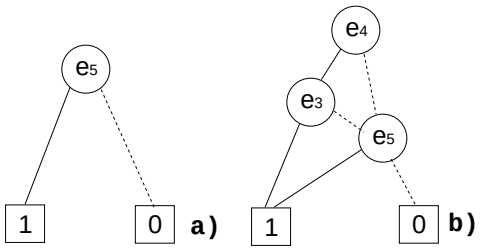

Figure 4.9: a) BDD arc e_5; b) BDD $((e_4 \wedge e_3) \vee e_5)$

edge the BDDs of the paths starting from each edge are combined in OR.

Example: Bridge Network 3 - The BDD construction algorithm of Algorithm 4.3 is illustrated step by step on the same directed bridge network of Figure 4.1. The graph is visited according to the progressive number assigned to the nodes. Starting from the source node `s` (node 1) the visit proceeds toward node 2 along edge e_2, toward node 3 along edge e_3 and toward the sink node `t` (node 4) along edge e_4. Once the sink is reached, the BDD construction starts with the BDD representing arc e_4 (Figure 4.8a).

Since node 3 has only e_4 as outgoing edge the visit returns to node 2. A new BDD for arc e_3 is created (Figure 4.8b) and the AND operator between the two created BDDs is applied $(e_4 \wedge e_3)$ (Figure 4.8c) .

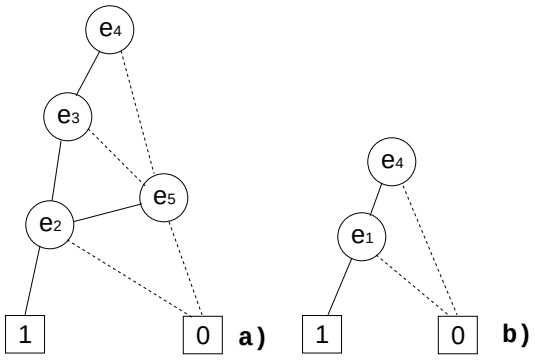

Figure 4.10: a) BDD $(((e_4 \wedge e_3) \vee e_5)) \wedge e_2)$; b) BDD $e_4 \wedge e_1$

Coming back to node 2 the algorithm visits arc e_5 reaching the sink and building the corresponding BDD (Figure 4.9a) that is combined in OR with the BDD created until now $((e_4 \wedge e_3) \vee e_5)$ (Figure 4.9b).

Since there are no more arcs from node 2, the visit returns to node 1 and the new BDD representing arc e_2 is combined in AND with the already constructed BDD $(((e_4 \wedge e_3) \vee e_5)) \wedge e_2)$ (Figure 4.10a).

The same procedure is used to create the BDD representing the path (e_1, e_4) between node 1 and node 4 (Figure 4.10b). The BDD for arc e_4 already exists, and it is not created again. Finally, the BDD's of the two paths outgoing from node 1 (Figure 4.10b and Figure 4.10a) are combined in OR to obtain the final BDD reported in Figure 4.11 that represents the `(s,t)` connectivity function

$$(e_1 \wedge e_4) \vee (((e_4 \wedge e_3) \vee e_5) \wedge e_2)$$

Even if any variable order could be preassigned in the BDD construction, the order in which the variables appear in the BDD of Figure 4.11 is

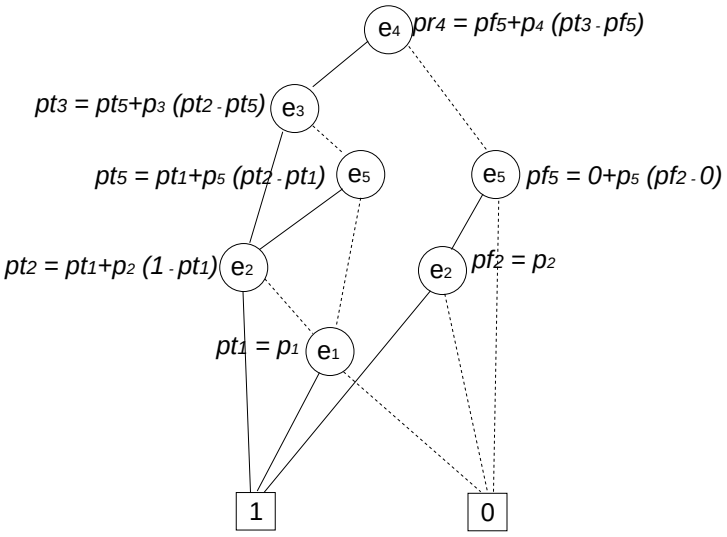

Figure 4.11: The final BDD

different from the one in Figure 4.4; hence, the two BDD's have a different aspect even if they represents the same Boolean function. The network reliability can be calculated by recursively applying equation (4.1), as illustrated in Figure 4.11.

Example: Symmetric Network 2 - The construction of the BDD by means of the algorithm sketched in Algorithm 4.3 is illustrated step by step on the same symmetric directed network of Figure 4.7. The graph is visited according to the progressive (but arbitrary) number assigned to the nodes. Starting from the source node $\mathtt{s} = v_1$ the visit proceeds along nodes v_2, v_3, v_4, until the sink node $\mathtt{t} = v_5$ is reached. Once the sink is reached, the construction starts with the BDD representing the last visited arc 4. Then, the algorithm makes one step back to node v_3 where it finds

a bifurcation and builds the BDD $((4 \wedge 3) \vee 8)$. Going one step back with the recursion, the algorithm revisits node v_2 and builds the BDD $(((4 \wedge 3) \vee 8) \wedge 2) \vee (4 \wedge 7)$. Finally, in the last step, the algorithm visits the source node v_1 and builds the BDD for the complete `(s,t)`-connectivity function.

$$S_{(\mathtt{s},\mathtt{t})} = (((((4 \wedge 3) \vee 8) \wedge 2) \vee (4 \wedge 7)) \wedge 1) \vee (((4 \wedge 3) \vee 8) \wedge 6) \quad (4.10)$$

It is easy to see that formula (4.10) contains replicated terms that are simplified and folded automatically during the construction of the ROBDD, that is reported in Figure 4.12. In the example the sequence in which the variables are ordered in the ROBDD is the same in which the arcs are encountered during the graph visit. However, any other ordered sequence could be followed.

Even if the algorithms depicted in Algorithm 4.1 and 4.3 derive directly the BDD's of the Boolean connectivity function, nevertheless it is sometime useful, or even required, to determine the list of the minpaths and/or of the mincuts of the network. The minpaths provide a qualitative information about the most favorable connections (in term of number of hops) between source and sink, whereas the mincuts give a useful information about the most critical catastrophic events that may isolate two nodes and disrupt the network into two non-communicating parts.

Any path in a BDD linking the root with the terminal leaf 1, where only the list of events corresponding to the *left* (or 1) traversed branches are included, is a path of the network (but not necessarily a minpath). Similarly, any path in a BDD linking the root with the terminal leaf 0, where only the list of events corresponding to the *right* (or 0) traversed branches are

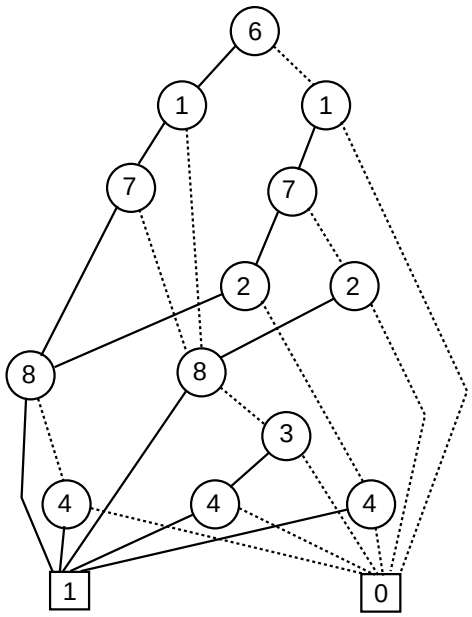

Figure 4.12: The final ROBDD of the network of Figure 6.2

included, is a cut of the network (but not necessarily a mincut). To derive the minpaths (mincuts) the list of paths (cuts) obtained from the BDD must be minimized. An alternative approach, proposed in [70], consists in transforming the original BDD into a new graph embedding all and only the minpaths (mincuts). Detail of the transformation algorithm are in [70]. In Figure 4.13 is shown the minimized BDD obtained from BDD of Figure 4.11. It is used in order compute the minpaths of bridge network of Figure 4.1.

In Figure 4.14 it is shown the minimized BDD obtained from the same BDD of the minpath case (Figure4.11). It is used in order to compute the

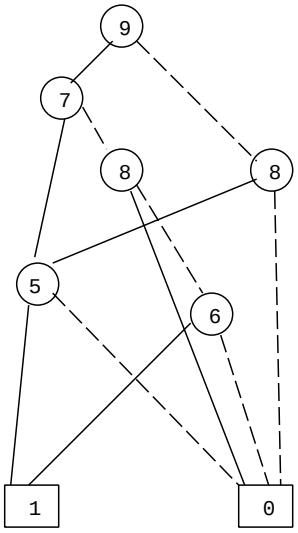

Figure 4.13: Minimized BDD for minpath computation

mincuts of bridge network of Figure4.1.

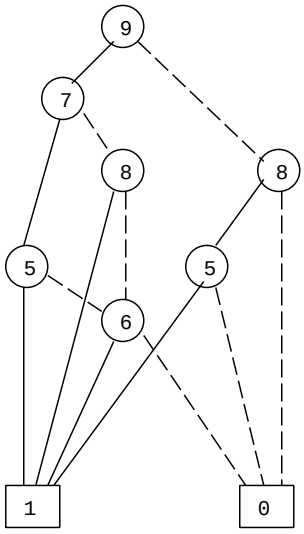

Figure 4.14: Minimized BDD for mincut computation

4.2 The Italian GARR Network

The GARR Network (whose acronym means "Gestione Ampliamento Rete Ricerca" - "Research Network Widening Management") is composed by all subjects representing the Italian Academic and Scientific Research Community, and manages their internet services. The layout of the GARR network and its main bandwidth characteristics are available from: http://www.garr.it/. The Italian Academic and Research Network GARR (http://www.garr.it/index.php) is based on scientific and academic collaboration projects between Italian Universities and public Research Organisations. The GARR network service is mainly provided for the GARR community.

Figure 4.15 shows the layout of the major connections of the GARR network. The network in the picture is composed of 42 nodes and 52 links.

The following measures can be computed on the network using the tool NRA (see Chapter 9):

1. *(s-t) Network reliability* - Given a source node `s` and a sink node `t` the (s-t) Network reliability is the probability that the source and the sink are connected by at least one path of working edges.

2. *List of minpaths* - Given a network and a source node `s` and a sink node `t`, a path H between `s` and `t` is a subset of edges that guarantees the source `s` and sink `t` to be connected if all the edges of the subset are functioning. A path H is a minpath if a subset of elements in H does not exist that is also a path. The NRA tool provides the list of all the minpaths ordered according to their size (number of edges forming the path).

3. *List of mincuts* - Given a network and a source node `s` and a sink node `t`, a cut K between `s` and `t` is a subset of edges whose failure

disconnects the source `s` and sink `t`. A cut K is a mincut if a subset of elements in K does not exist that is also a cut. The NRA tool provides the list of all the mincuts ordered according to their size (number of edges forming the cut).

The analysis technique used by the tool NRA (see chapter 9) to obtain the above measures is based on the representation of the connectivity function of the network by means of Binary Decision Diagram (BDD) as illustrated in paragraph 4.1.4 and in [21, 23].

4.2.1 Reliability Analysis of the GARR Network

We study the properties of the GARR network in two cases assuming two different couples of source and sink nodes:

- in the first case we assume the node TO as a source and the node CT as sink;
- in the second case we assume the node $TS1$ as a source and the node NA as sink (see Figure 4.15).

In the lack of available data about the reliability of the single elements of the GARR network, and with the goal of performing quantitative computations, we have assumed, arbitrarily, a uniform value p for the arcs of being up ($1 - p$ down). Usually we take $p = 0.9,\ 0.95,\ 0.99$. A summary of the results is presented in Table 4.1.

Figure 4.15: GARR network

Table 4.1: Reliability computations on the graph of Figure 4.15

Source Node	Terminal Node	Minpath	Mincut	No. BDD Nodes	Reliability Single Edge		
					0.9	0.95	0.99
TO	CT	196	481	1003	0.977428	0.994713	0.999797
TS1	NA	168	385	604	0.975771	0.994486	0.999795

The first two columns of Table 4.1 indicate the chosen source and sink nodes: the first row reports the first examined case, the second row the second case. The subsequent columns report the results obtained by running the NRA tool. In detail: the number of *minpaths*, the number of *mincuts* and the number of BDD nodes generated by using the generation algorithm based on a depth-first search on the graph [21, 23]. The final three columns report the (s-t) reliability value assuming that only the arcs can fail, all with identical probability p ($p = 0.9$, 0.95, 0.99, respectively).

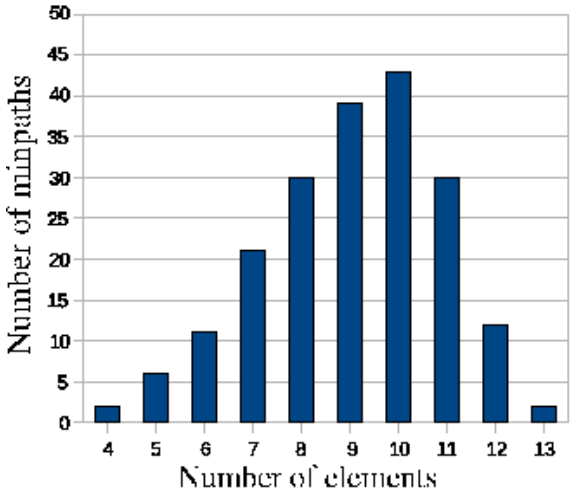

Figure 4.16: Histogram of the minpath length for the case s=TO t=CT

The connectivity properties depend not only on the number of *minpaths (mincuts)* but also on their order, that is the number of edges that form the *minpath (mincut)*. To show the order, we have derived the histograms of the distribution of the length of the *minpaths* and *mincuts* for the different

s-t cases reported in Table 4.1. For the case s=TO t=CT, Figure 4.16 reports the histogram of the length of the *minpaths* and Figure 4.17 the histogram of the order of the mincuts. It is evident that the minpaths of lower order are the most favorable to connect source and sink, while the mincuts of lower order are the most critical for interrupting the connection between source and sink. In any case, the tool NRA provides the complete list of edges that form each minpath and its probability of being up, and the complete list of edges that form each mincut and its probability of being down. Similar results are reported in the next two figures for the

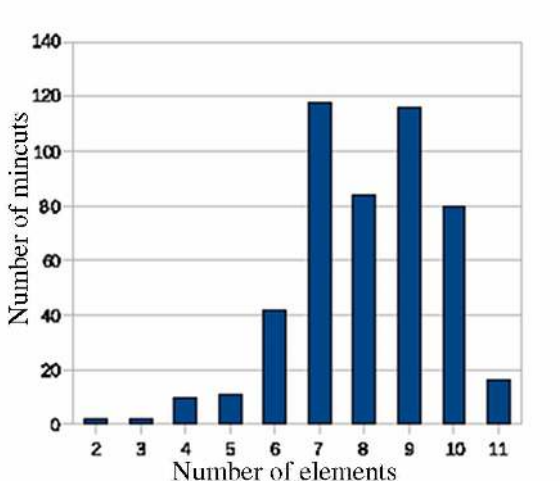

Figure 4.17: Histogram of the mincut order for the case s=TO t=CT

case s=TS1 t=NA. Figure 4.18 reports the histogram of the length of the minpaths , while Figure 4.19 reports the histogram of the order of the mincuts.

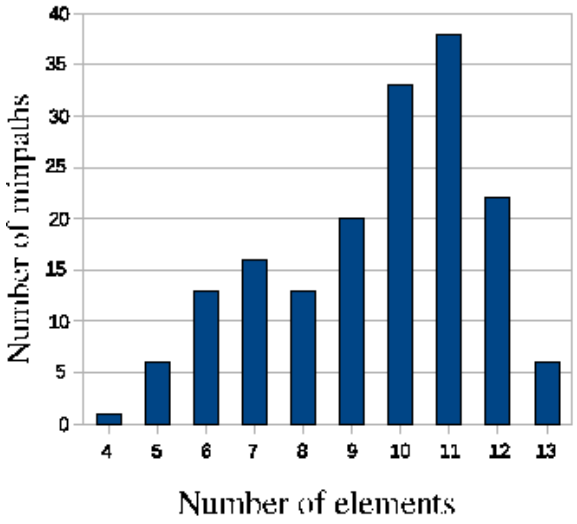

Figure 4.18: Histogram of the minpath length for the case s=TS1 t=NA

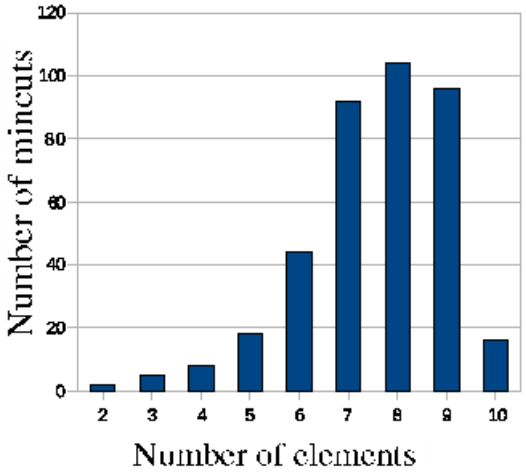

Figure 4.19: Histogram of the mincut order for the case s=TS1 t=NA

4.3 IEEE 118 Bus Test Case

The IEEE 118 Bus Test Case represents a portion of the American Electric Power System and was made available to the electric utility industry as a standard test case. Reference to the IEEE 118 Bus Test Case can be found in: *http://www.ee.washington.edu/research/pstca/*. We consider a portion of the IEEE 118 Bus Test Case represented in Figure 4.20.

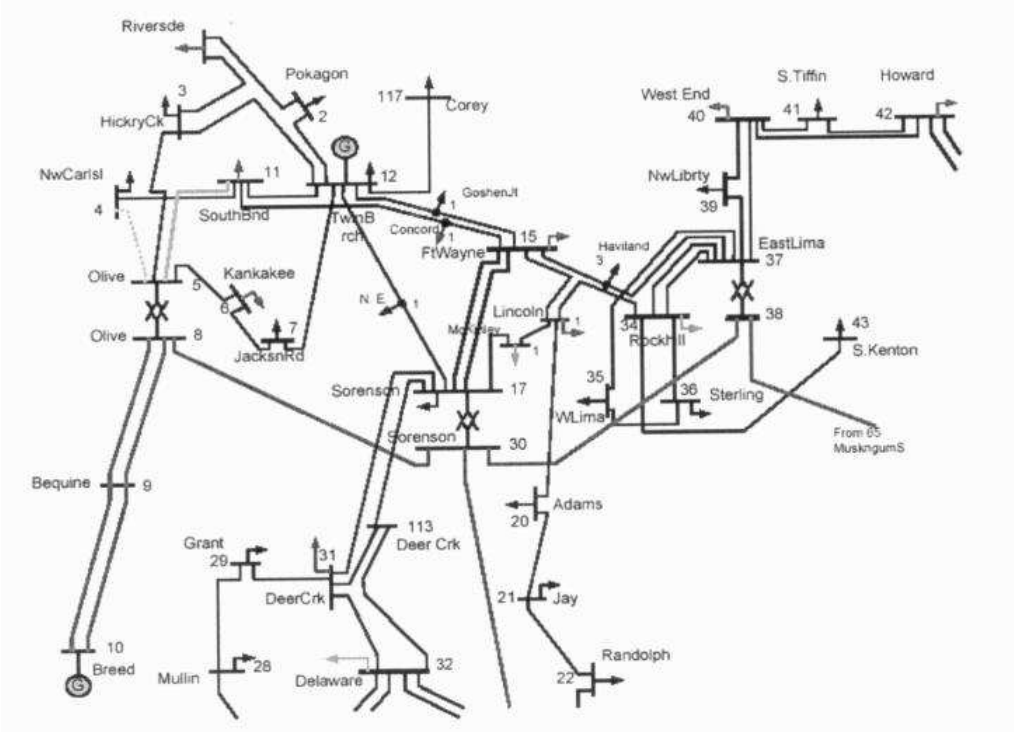

Figure 4.20: Portion of the Electrical Grid of the IEEE 118 Bus Test Case

The layout of the grid network of Figure 4.20 has been derived by su-

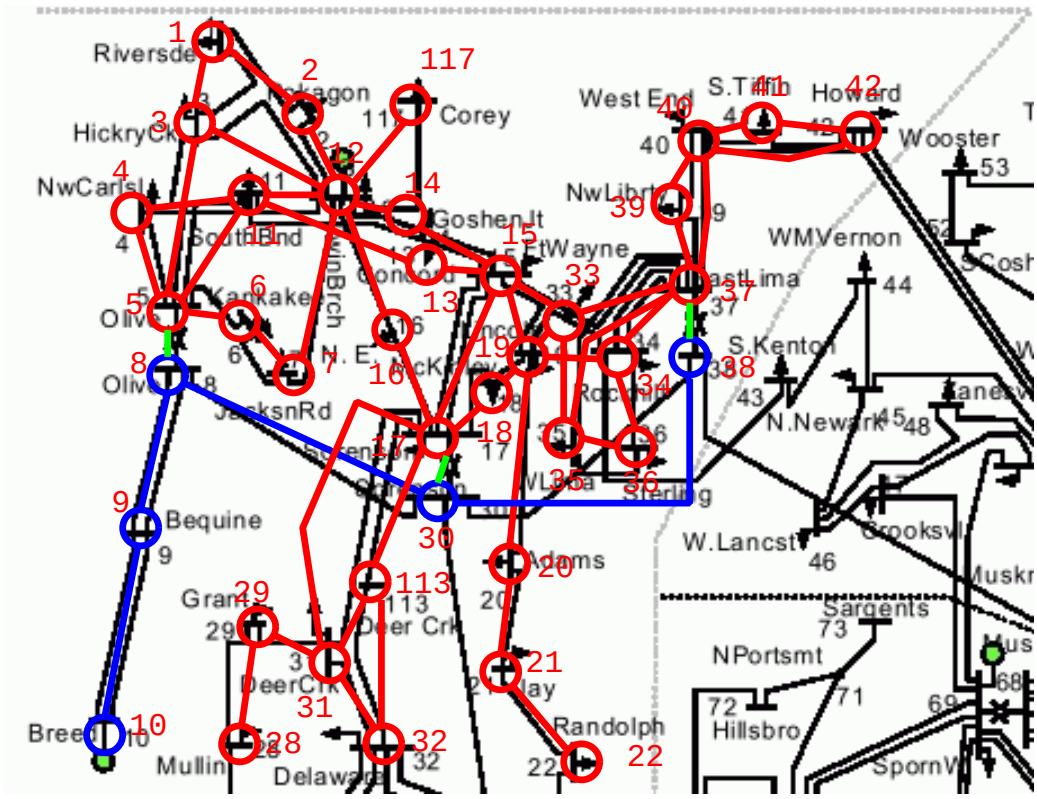

Figure 4.21: Graph layout of the Electrical Grid of Figure 4.20

perimposing a schematized graph structure on the real grid in Figure 4.21. Extracting only the graph structure, we obtain the network of Figure 4.22.

4.3.1 Network Reliability of IEEE 118 Bus Test Case

Using the program tool NRA, we have evaluated the reliability of the network of Figure 4.22 assuming as source node s and sink node t three different

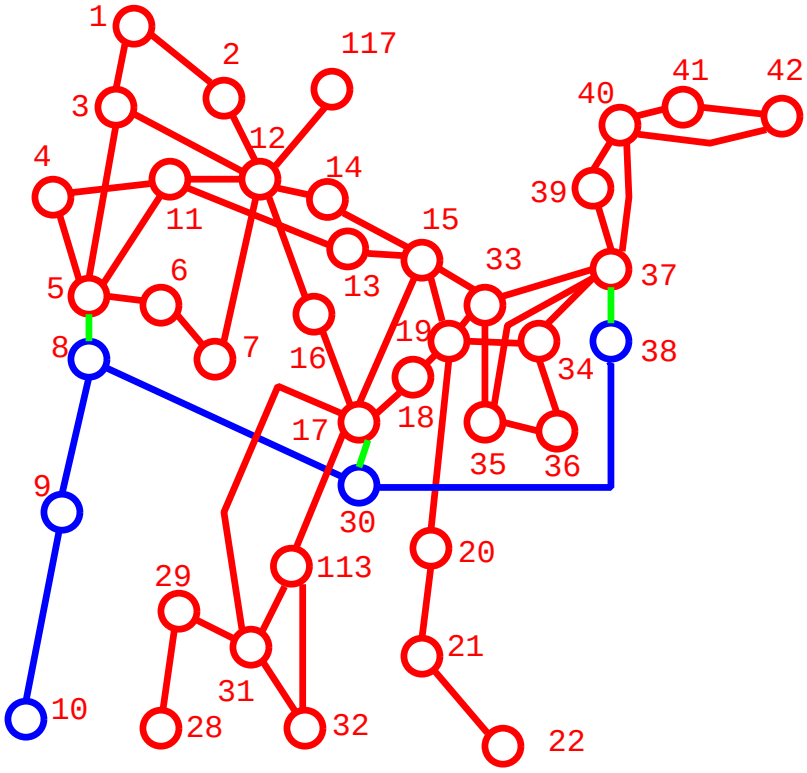

Figure 4.22: Graph layout of the Electrical Grid of Figure 4.20

couples of distant nodes[24] [26].

The results of the analysis obtained from the NRA tool are displayed in Table 4.2. The first two columns display the source node s and the terminal node t used in the computation. The subsequent columns report, respectively: the number of *minpaths*, the number of *mincuts* and the number of BDD nodes generated by using the generation algorithm based on a

Table 4.2: Reliability computations on the graph of Figure 4.22

Source Node	Terminal Node	Minpath	Mincut	No. BDD Nodes	Reliability Single Edge		
					0.9	0.95	0.99
1	35	393	3407	3933	0.9531	0.9892	0.9995
10	22	151	4162	2551	0.5810	0.7713	0.9508
5	42	496	2692	1638	0.9552	0.9895	0.9995

depth-first search on the graph [21, 23].

To have a closer look to the results we have derived the histograms of the distribution of the length of the minpaths and mincuts for the different s-t cases reported in Table 4.2. Figure 4.23 reports the histogram of the length of the minpaths for the case s=10 t=22, while Figure 4.24 reports for the same s-t case the histogram of the order of the mincuts.

Similar results are reported in the next two figures for the case s=1 t=35. Figure 4.25 reports the histogram of the length of the minpaths, while Figure 4.26 reports the histogram of the order of the mincuts.

The mincuts of lower order are the one most critical, since a lower number of simultaneous failure events are needed to disconnect the source s from the termination t. For instance, in the case of Figure 4.24 one can see that there are 4 mincuts of order two and seven mincuts of order 3.

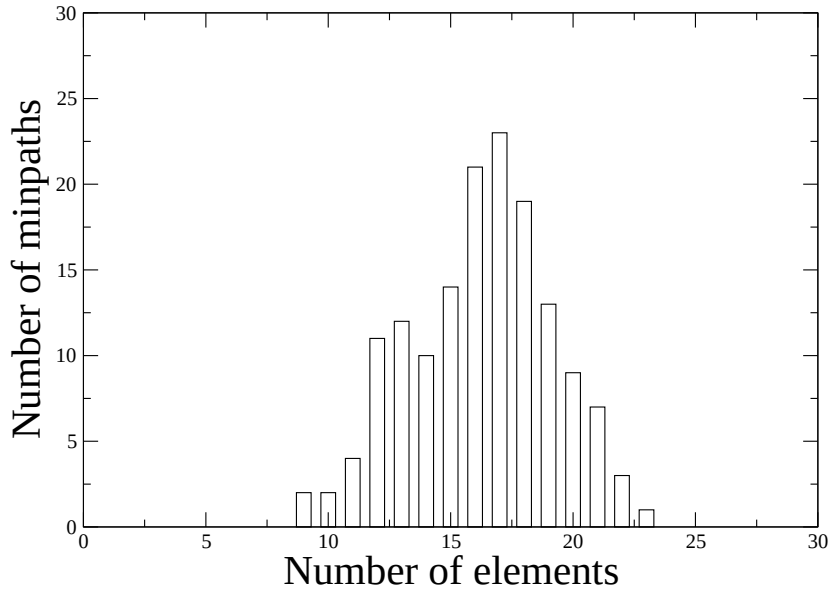

Figure 4.23: Histogram of the minpath length for the case s=10 t=22

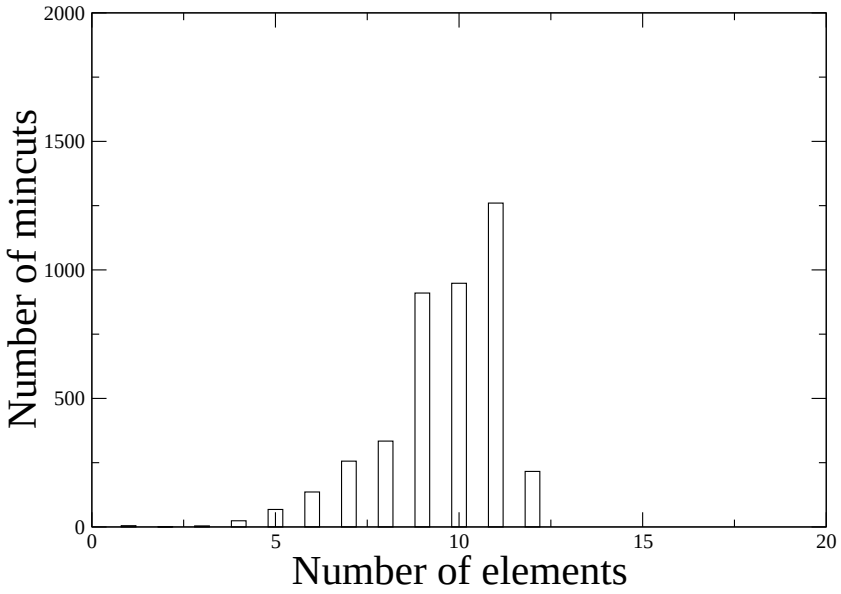

Figure 4.24: Histogram of the mincut length for the case s=10 t=22

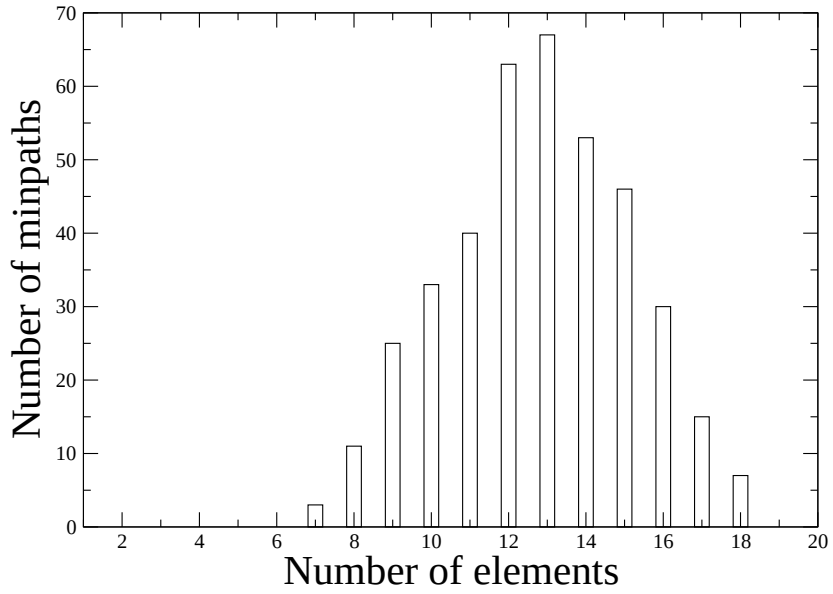

Figure 4.25: Histogram of the minpath length for the case s=1 t=35

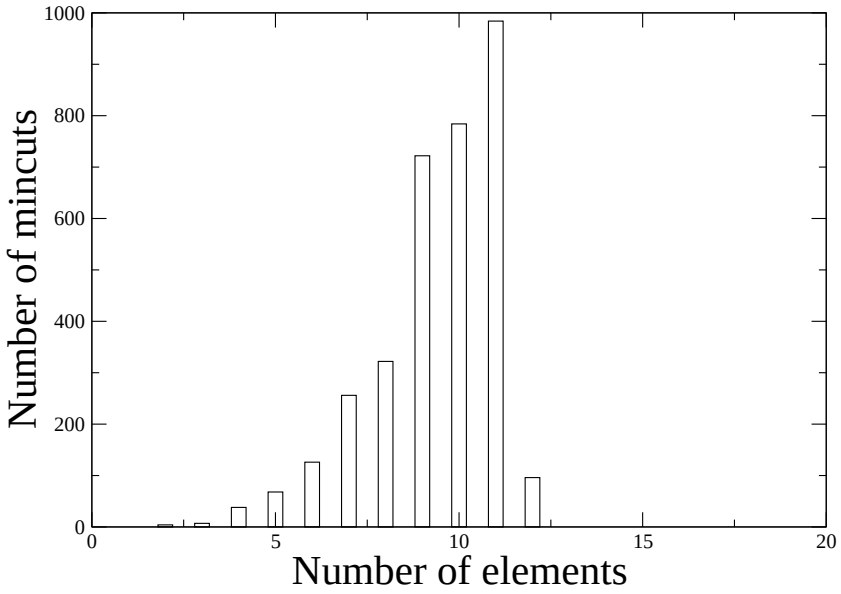

Figure 4.26: Histogram of the mincut length for the case s=1 t=35

Chapter 5

Probabilistic Weighted Networks

5.1 Introduction

This chapter starts form the unweighted networks and moves forward in the direction of enriching the system description and the measures that can be obtained, by including in the definition and characterization of the system some parameters or weights related to the services that the network can provide, and by computing the availability of the service or the Quality of Service (QoS) associated to the network.

To this end, the specification of the network must be augmented with an index or a weight that indicates a service feature of the network. We show that the weights can have many different physical meanings, and correspondingly require a different treatment. In particular, the weights can indicate a characteristics related to a cost or a distance or a resistance (i.e. a property that is additive along the paths) or a capacity (i.e. a property that

is additive along the cuts). For this step, we define formally the concept of a *"Probabilistic Weighted Network"* and we discuss the related QoS measures and how to compute them. To deal with the analysis of weighted networks we extend the data structure of the BDD's into a new and more powerful data structure called Algebraic Decision Diagrams ADD [22].

The analysis is specifically directed toward the evaluation of critical systems in the form of networks and in the interrelation among networks. To show the potentialities of the developed methodology and of the implemented tool we carry our analyses and computations on two publicly available cases study. In particular, we concentrate our attention on the Italian GARR Network and to the IEEE 118 Bus Test Case.

5.2 Weighted Networks

Traditional studies on network reliability, as those exemplified in Sections 4.2.1 and 4.3.1 , assume that nodes and links are binary entities (either up or down) and that the node-to-node reliability is computed as the probability that there exists at least one path of working links connecting the two nodes. In many cases a richer representation is useful or even needed, when some property associated to the edges (capacity, cost, length) influences the performance of the system. For example, the amount of traffic characterizing the connections in communication or transport systems, or the distance between nodes in a highway network, or the resistance of the branches in an electrical grid, are fundamental parameters for a full description of these systems. Motivated by these observations, we consider weighted probabilistic networks whose edges have been assigned a weight (indicated as a label associated to the edge). We distinguish between two types of interpretation of weights which can be used for characterizing the

properties of a network.

i) Weights are interpreted as costs, or lengths or resistances associated to the edges so that the aim of the analysis is to evaluate the probability that the connection can be established below a given cost (or distance or resistance).

ii) Weights are interpreted as capacities or bandwidths, representing the maximum flow that the edge can support when up. The aim of the analysis is addressed to the ability of the network to transport a given amount of flow.

Many analysis techniques have been explored in the literature for computing the reliability in binary probabilistic networks. In the case of dependable weighted networks, a more informative measure is the Quality of Service (QoS) defined, in the two cases above, as:

i) the probability that the source node and the sink node can be reached with a cost (or a distance) not greater than a given threshold;

ii) the probability that the network is able to transport a flow from the source to the sink not less than a given demand.

As in binary probabilistic networks the most efficient representation of the network connectivity function seem to be based by means of Binary Decision Diagrams (BDD) [47, 21, 27], the analysis of weighted probabilistic networks can be based on a data structure, derived from BDD, and called Algebraic Decision Diagram (ADD)[11]. An ADD is a binary tree whose terminal leaves can assume any positive value between 0 and the maximum flow (or distance) in the network.
A path connecting the root of the ADD with a terminal leaf with label, say

α, indicates that a flow (or distance) equal to α is transported along this path. The associated probability can be easily computed from the ADD.

5.3 Probabilistic Weighted Networks

A weighted stochastic network is a tuple $N = (G, P, W)$ where $G = (V, E)$ is a graph (with N_V vertices and N_E edges), P is the probability function that assigns to each edge $e(u, v)$ $(u, v \in V)$ a probability $p(u, v)$ of being up (and $1 - p(u, v)$ of being down) and W is a weight function that assigns to each edge $e(u, v)$ a finite real weight $w(u, v)$. $w(u, v)$ has the meaning of a reward assigned to the arc that qualifies and quantifies the service carried by the arc, or its cost, or some other attribute of the arc. Given a source node `s` and a termination node `t`, we can evaluate the Quality of Service (QoS) between (`s-t`) as a reward measure that is a function of the structure of the graph, of the probability function P and of the weight function W. Let us denote this QoS measure as $\Psi_{s,t}$. We consider two cases.

5.3.1 The Weight is a Cost

By cost we identify, generically, a parameter that is additive along the arcs of the network. From this point of view, a cost can be: a real monetary cost, the length of the arc, the resistance of the arc. The weight $w(u, v)$ of the arc $e(u, v)$ is a cost function related to the input and output node (u, v) of the arc. A simple physical idea is the length of the arc, i.e. the distance between nodes u and v. The reward function $\Psi_{s,t}$ is the minimum cost between `s` and `t` computed as the sum of the costs of the arcs of a path connecting `s` and `t`. In a stochastic network the problem can be reformulated as a QoS problem in the following terms.

Problem 1 - Given a stochastic weighted network N=(G,P,W), compute the probability that the cost $\Psi_{s,t}$ between s *and* t *is below a threshold ψ_{max}.*

To solve *Problem 1* we need some definitions.

Definition 1 - Given a stochastic network $N = (G, P, W)$ and a source node s and a sink node t, the cost of a path $H(s, t)$ is the sum of the costs $w(u, v)$ of the edges forming the path.

Corollary 1 min-distance - If a network $N = (G, P, W)$ has n minpaths H_1, H_2, ..., H_n the minimum cost between s and t is equal to the minimum cost of all its minpaths.

The min-cost corollary says that the minimum cost between any two nodes cannot be less than the minimum cost of all its paths.

5.3.2 Flow Networks

The weight $w(u, v)$ of the arc $e(u, v)$ is the nominal capacity or the bandwidth that the arc is able to carry. Networks with this attribute are usually called flow networks [52, 76] and the function $\Psi_{s,t}$ is the maximum flow that can be transmitted from s to t. The maximum flow problem has received a great attention even in the recent literature [28], [60], [42], [45], [33]. In stochastic networks the problem can be reformulated as a QoS problem in the following terms.

Problem 2 - Given a stochastic flow network N=(G,P,W), compute the probability that the flow $\Psi_{s,t}$ guaranteed between s *and* t *exceeds a minimum threshold ψ_{min}.*

Definition 2 - Given stochastic flow network $N = (G, P, W)$, and a source node s and a sink node t, the capacity of a cut $K(s, t)$ is the sum of the capacities $c(u, v)$ of all the edges forming the cut.

Corollary 2 Max-flow Min-cut- If a network $N = (G, P, W)$ has m mincuts K_1, K_2, ..., K_m the maximum flow between `s` and `t` is equal to the minimum capacity of all its mincuts.

The max-flow min-cut theorem says that the value of the maximum flow is equal to the minimal capacity carried by a mincut. In other words the theorem says that mincut that constitutes the bottleneck of a network determines its maximum flow. The quantity of flow between any two nodes cannot be greater than the weakest set of links somewhere between the two nodes.

5.3.3 Algebraic Decision Diagrams - ADD

ADD Algebraic Decision Diagrams (ADD) (also called Multi Terminal BDD - MTBDD) are an extension of BDDs [11] which allow to represent a real function of n Boolean variables as a binary tree. While BDDs have only two terminal leaves 0 and 1, ADD can have more than two terminal leaves that identify all the possible values taken by the Boolean function along the paths from the root to the terminal leaves. Like BDD, ADD provide a compact representation of a Boolean expression by means of the Shannon's decomposition principle. Each node of the ADD represents a Boolean variable and has two successors: the left branch (in solid line) represents the value of the variable 1 and the right branch (in dotted line) represents the value of the variable 0. In the present case, we assume as the n Boolean variables the arcs of the graph and the $\Psi_{s,t}$ function is defined according to *Corollary 1* or *Corollary 2* depending on the definition of the weights.

- If the weights are cost functions the terminal leaves of the ADD provide all the possible values of the costs lower than the maximum

threshold Ψ_{max} computed along the minpaths that connect s to t (Corollary 1).

- If the weights are capacity functions the terminal leaves of the ADD provide all the possible values of the flows greater than the minimum threshold Ψ_{min} that can be transmitted from s to t when s and t are connected. The flow values are computed from the mincuts following Corollary 2.

- The single terminal leaf labeled 0 is reached by the combination of variables for which the graph is disconnected or the function $\Psi_{s,t}$ does not respect the constraints.

Similarly to BDDs, the ADD is constructed by imposing an ordering on the variables and the size of the tree is extremely sensitive to the chosen ordering. An ordered ADD can be reduced by successive applications of two rules. The *deletion rule* eliminates a node with both of its outgoing edges leading to the same successor. The *merging rule* eliminates one of two nodes with the same label as well as the same pair of successors. In a probabilistic network $N = (G, P, W)$, each arc e_i is a Boolean variable with probability p_i of being up ($1 - p_i$ down). At any node V of the ADD we can compute the probability of the node $P(V)$ by the following expression:

$$\begin{aligned} P\{V\} &= p_1 P\{V_{succ-\ell}\} + (1 - p_1) P\{V_{succ-r}\} \\ &= P\{V_{succ-r}\} + p_1 (P\{V_{succ-\ell}\} - P\{V_{succ-r}\}) \end{aligned} \tag{5.1}$$

where $V_{succ-\ell}$ is the left successor node of node V in the ADD obtained by setting $e_i = 1$ and V_{succ-r} is the right successor of node V obtained by setting $e_i = 0$. Starting from a terminal leaf of the ADD where the reward function $\Psi_{s,t}$ has a value ψ we apply recursively Equation 5.1 backward up

to the root node and we sum up at each node the corresponding probability value along the path.

5.3.4 Basic operations for ADD manipulation

In order to compute the Boolean reward function $\Psi_{s,t}$, as defined in Corollaries 1 and 2, we need to define appropriate operators for manipulating and constructing the ADDs in addition to the standard Boolean operators NOT, AND and OR. The reward cost function is additive with respect to the minpaths while the reward flow function is additive with respect to the mincuts. We introduce the basic ADD operations with respect to the basic Boolean operators exemplified in Figure 5.1. Node s and t are the source

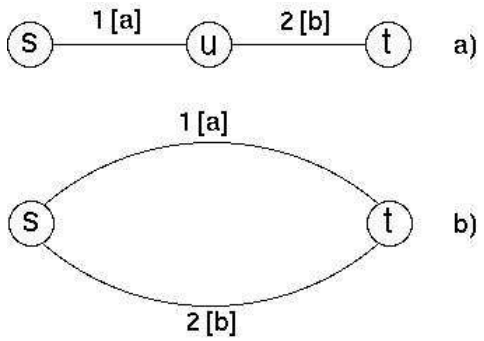

Figure 5.1: Network with two arcs: a) in series; b) in parallel

and the destination, the arc identifiers are 1 and 2 and the labels a and b in square brackets are the respective weights.

Weight as Cost

The network of Figure 5.1a) has a single minpath $H_1 = 1 \wedge 2$, the connectivity function is $C_{s,t} = H_1$ and when the network is connected ($C_{s,t} = 1$) the cost function is $\Psi_{s,t} = a + b$ otherwise $\Psi_{s,t} = 0$. In the computation of the cost function the Boolean $\wedge$ operator corresponds to a sum in the costs. We call this operation *AndSum* .
The network of Figure 5.1b) has two minpaths $H_1 = 1$ and $H_2 = 2$, the connectivity function is $C_{s,t} = H1 \vee H_2$. When the network is connected ($C_{s,t} = 1$) through arc 1 the cost is $\Psi_{s,t} = a$, when it is connected through arc 2 the cost is $\Psi_{s,t} = b$, when both arcs are up the cost is $\Psi_{s,t} = min(a, b)$. We call *OrMin* the ADD operation corresponding to the Boolean $\vee$.
Hence the generation of the cost reward function defined in Corollary 1 requires the implementation of the *AndSum* and *OrMin* operators. We summarize the definition of these operations in the truth table of Table 5.1 and the corresponding ADD construction in Figure 5.2 .

Arc 1	Arc 2	$C_{s,t}1 \wedge 2$	$AndSum$	$C_{s,t}1 \vee 2$	$OrMin$
0	0	0	0	0	0
0	1	0	0	1	b
1	0	0	0	1	a
1	1	1	$a + b$	1	$min(a, b)$

Table 5.1: Truth table of $AndSum$ and $OrMin$

Weight as flow

The network of Figure 5.1a) has two mincuts $K_1 = 1$ and $K_2 = 2$, the connectivity function is $C_{s,t} = K_1 \wedge K_2$. When the network is connected ($C_{s,t} = 1$) the flow function is $\Psi_{s,t} = min(a, b)$ otherwise $\Psi_{s,t} = 0$. In the

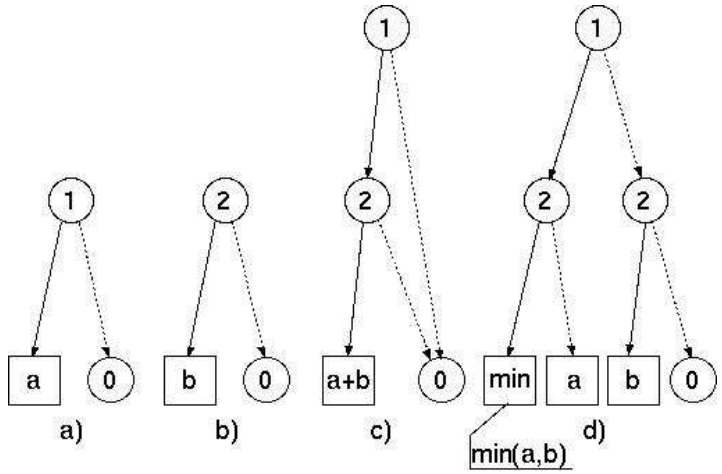

Figure 5.2: Basic ADD operations: a) ADD for arc 1 ; b) ADD for arc 2; c) 1 $AndSum$ 2 ; d) 1 $OrMin$ 2

computation of the flow function the Boolean $\wedge$ operator corresponds to a *min* in the flows. We call this operation *AndMin* .

The network of Figure 5.1b) has a single mincut $K_1 = 1 \vee 2$ and the connectivity function is $C_{s,t} = K_1$. When the network is connected ($C_{s,t} = 1$) through arc 1 the flow is $\Psi_{s,t} = a$, when it is connected through arc 2 the flow is $\Psi_{s,t} = b$, when both arcs are up the flow is corresponding to the Boolean $\vee$ and we call this operation $OrSum$.

Hence the generation of the flow reward function defined in Corollary 2 requires the implementation of the *AndMin* and *OrSum* operators. We summarize the definition of these operations in the truth table of Table 5.2 and the corresponding ADD construction in Figure 5.3.

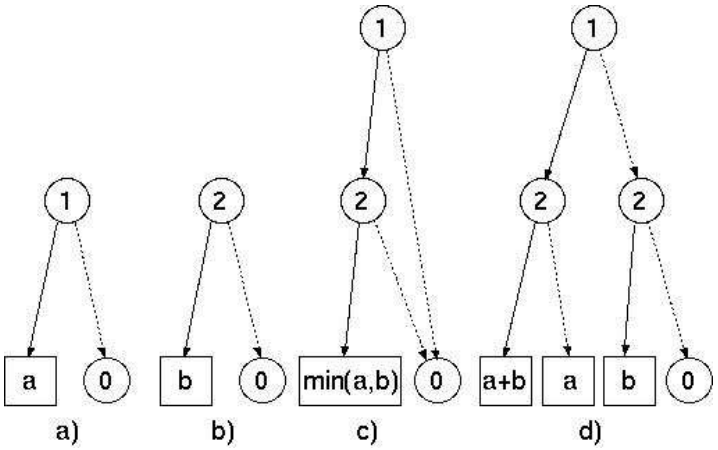

Figure 5.3: Basic ADD operations: a) ADD for arc 1 ; b) ADD for arc 2; c) 1 $AndMin$ 2 ; d) 1 $OrSum$ 2

Arc 1	Arc 2	$C_{s,t}1 \wedge 2$	$AndMin$	$C_{s,t}1 \vee 2$	$OrSum$
0	0	0	0	0	0
0	1	0	0	1	b
1	0	0	0	1	a
1	1	1	$min(a, b)$	1	$a + b$

Table 5.2: Truth table of $AndMin$ and $OrSum$

5.4 Distance Algorithm

The graph must be augmented with the length of each arc. The length may be inferred from the Euclidean distance of any couple of connected nodes if the graph is expressed by means of geographic coordinates; otherwise, the distance may appear as a label on each arc. The ADD representing the function $s - t$ distance may be constructed by a direct visit on the graph.

Algorithm 5.1 shows the pseudo-code form of the algorithm.

Algorithm 5.1 Generation of the ADD distance

```
Require: node, val
Ensure: costr_add(node, val)
 1: if val> ψ_max then
 2:    return add_constant(0)
 3: else if node == sink_node then
 4:    return add_constant(val)
 5: else
 6:    for each edge i outgoing from node do
 7:       next_node = the other end of i
 8:       add_tmp = costr_add(next_node , val + capa[node][next_node])
 9:       add_tmp =add_tmp and add_i
10:       addt = addt or_min add_tmp
11:       return addt
12:    end for
13: end if
```

We visit the graph starting from the source node s until the sink node t is reached.

For each edge we have the recursive call and we pass the other node of the edge and the value of the path until this step (variable *val*) plus the weight of the edge($capa[i]$). If this value is not less than the threshold ψ_{max} we return ADD zero. When we reach the sink node the variable *val* contains the value of the whole $s-t$ path and the terminal node is set to this value. For each edge a new ADD is built and is put in AND with the ADD produced so far.

A node can have more than one outgoing edge so from this node pass different paths. In this case, the ADD representing each path is put in OR, but the value of the terminal node is not the sum of the terminal nodes

rather the minimum value between them. or_{min} is the new function which implements this.

When the weights are interpreted as cost functions, the ADD can be built also in agreement with the expression of the network connectivity function starting from the list of the minpaths (see 5.2). We analyze one minpath at the time and we build the ADD for each single minpath by means of the AndSum operator over all the arcs of the minpath. If the distance value becomes greater than the threshold ψ_{max} the ADD is discarded and a new minpath is considered else it is combined with the OrMin operator with the ADD already generated. Algorithm 5.2 shows the pseudo-code of the algorithm.

Algorithm 5.2 Generation of the ADD for the distance function

Require: MPS, ψ_{max}
Ensure: costr_add_dist(MPS,ψ_{max})

```
 1: ADD ris
 2: for each mpsi in MPS do
 3:    ADD add_tmp= -1, double dist=0;
 4:    for each e in mpsi and dist < ψmax do
 5:       dist=dist+w[e]
 6:       add_tmp=And_sum(add_tmp, add_e)
 7:    end for
 8:    if dist < ψmax then
 9:       ris= Or_min(ris, add_tmp)
10:    end if
11: end for
12: return ris
```

Example 1: Weighted Bridge Network with Cost - A bridge network with weighted directed arcs is shown in Figure 5.4. In this first example, we assume that the labels on the edges represent the length of the arcs.

Furthermore, a failure probability equal to $p = 0.9$ is uniformly assigned to all the arcs. Referring to *Problem 1*, we want to compute the probability to reach node $t = 4$ from $s = 1$ with a distance below a given threshold.

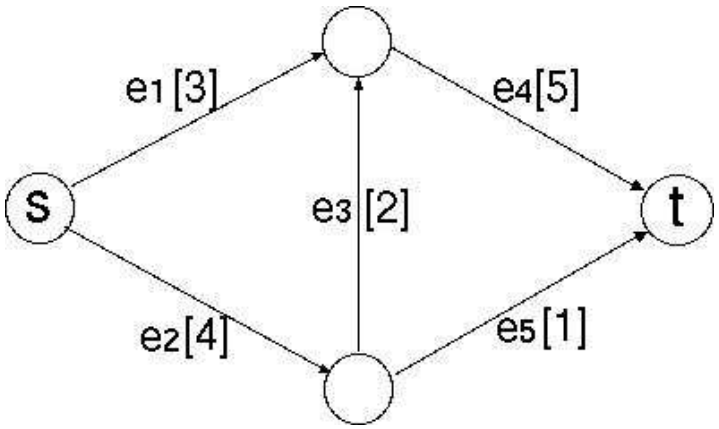

Figure 5.4: A directed bridge network

It is clear from a simple visual inspection of the network of Figure 5.4, that there are three minpaths of length, respectively:

$$\begin{array}{lll} H_1 = e_1\, e_4 & H_2 = e_2\, e_5 & H_3 = e_2\, e_3\, e_4 \\ l(H_1) = 8 & l(H_2) = 5 & l(H_3) = 11 \end{array} \tag{5.2}$$

Assuming a maximum threshold greater that the maximum length, as for instance $\psi_{max} = 15$, all the minpaths satisfy the constraint and the corresponding ADD is shown in Figure 5.5. The probability values of the different ADD terminal values are computed according to Equation (5.1) and are reported in Table 5.3. The terminal value 0 gives the probability that the constraint is not satisfied (in this case, $s - t$ are not connected).

If we change the maximum accepted distance and we fix $\psi_{max} = 10$,

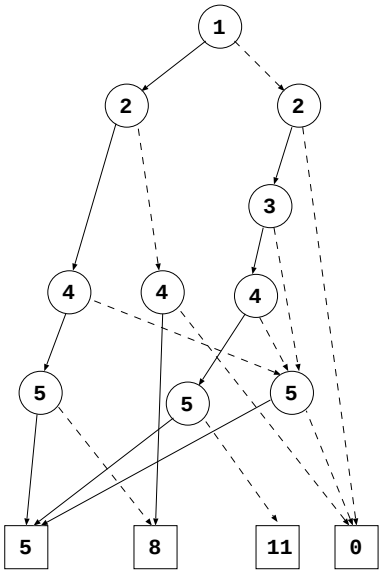

Figure 5.5: ADD bridge network, with weights interpreted as lengths

Table 5.3: Probability of the ADD terminal values (distance) for a threshold $\psi_{max} = 15$

(s-t) Distance	**Probability**
5	0.81
8	0.1539
11	0.00729
0	0.02881

we obtain the results shown in Table 5.4. The probability for the terminal value 0 in Table 5.4 is the sum of the probabilities that $s - t$ are not connected, or they are connected with a distance greater that $\psi_{max} = 10$.

This probability is the sum of the last two rows of Table 5.3.

Table 5.4: Probability of the ADD terminal values (distance) for a threshold $\psi_{max} = 10$

(s-t) Distance	**Probability**
5	0.81
8	0.1539
0	0.0361

5.5 Flow Algorithm

When the weights are interpreted as flow functions, the ADD can be built also in agreement with the expression of the network connectivity function starting from the list of the mincuts (see Section 5.3). We analyze one mincut at the time and we build the ADD for each single mincut by means of the operator over all the arcs of the mincut. We discard the terminal leaves that are lower than the threshold and the obtained ADD is combined with the operator with the ADD already generated.

Example 2: Weighted Bridge Network with Capacity - We consider the same network of Figure 5.4 with the same labels. But in this example the arc labels are interpreted as the capacities of the link, and we apply *Problem 2*. With a minimum threshold $\psi_{min} = 1$, we obtain the ADD displayed in Figure 5.6.

The values on the terminal leaves of the ADD represent the maximum flow that the network can carry along a path starting from the root node. For example:

Algorithm 5.3 Generation of the ADD for the flow function

Require: MCS, ψ_{min}
Ensure: costr_add_flow(MCS,ψ_{min})

```
1: ADD ris
2: int ck=0
3: for each mcsi in MCS and ck==0 do
4:    ADD add_tmp= 0
5:    for each e in mcsi do
6:       add_tmp=Or_sum(add_tmp, add_e)
7:       add_tmp=check_psi(add_tmp, ψ_min)
8:    end for
9:    if add_tmp==add_zero() then
10:      ck=1
11:      ris= And_min(ris, add_tmp)
12:   end if
13: end for
14: return ris
```

- The terminal leaf 6 is reached along the path $1, 2, 3, 4, 5$ representing that all the arcs are up. In this case the maximum flow is carried by the weakest cut $4, 5$.

- The terminal leaf 3 is reached along the paths $(1, 2, \overline{3}, 4, \overline{5})$ or $(1, \overline{2}, 3, 4)$ or $(1, \overline{2}, \overline{3}, 4)$ where the weakest cut is given by the arc 1 only, that carries a flow of 3. But another possible path is $(\overline{1}, 2, 3, 4, 5)$ where the weakest cut is $(3, 5)$ that carries also a maximum flow equal to the sum of the capacities of arcs 3 and 5: i.e. $2 + 1 = 3$.

- The terminal leaf 1 can be reached along many paths, as for instance $(\overline{1}, 2, 3, \overline{4}, 5)$, that are all characterized by the fact that the weakest link is arc 5 that carries a capacity of 1.

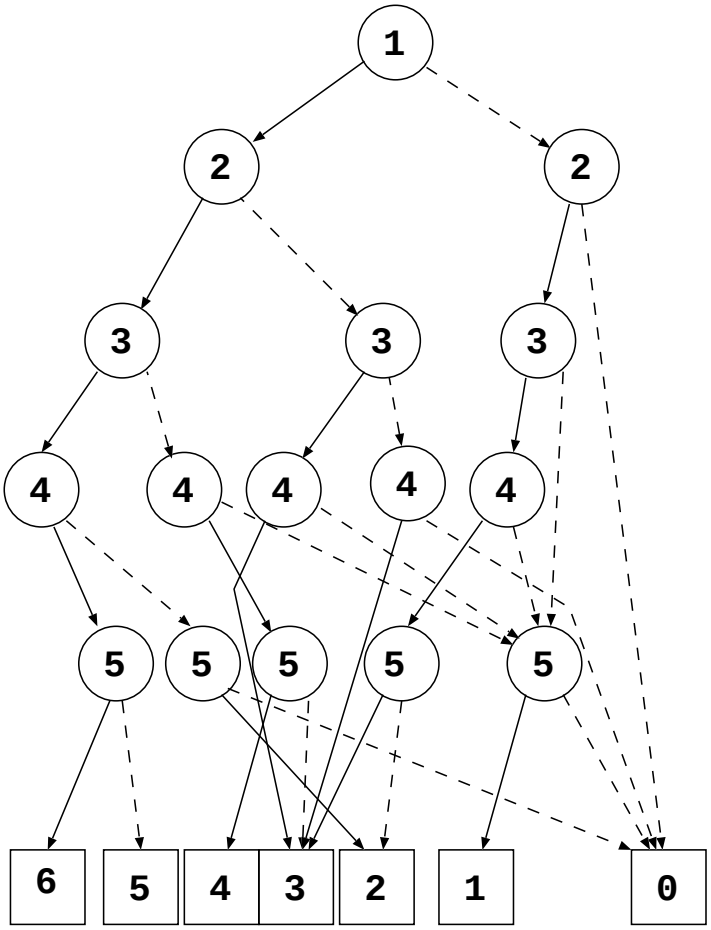

Figure 5.6: ADD bridge network, with weights interpreted as capacities

As in the previous example a failure probability equal to $p = 0.9$ is uniformly assigned to all the arcs. The values of the probability for the different terminal values of the ADD of Figure 5.6, are reported in Table

5.5.

Table 5.5: Probability of the ADD terminal values (max flow) for a threshold $\psi_{min} = 1$

(s-t) Max Flow	**Probability**
6	0.59049
1	0.08829
5	0.06561
4	0.06561
3	0.1539
2	0.00729
0	0.02881

By assuming a minimum threshold $\psi_{min} = 4$ the probability values are reported in Table 5.6.

Table 5.6: Probability of the ADD terminal values (max flow) for a threshold $\psi_{min} = 4$

(s-t) Max Flow	**Probability**
6	0.59049
5	0.06561
4	0.06561
0	0.27829

We compare the result of our algorithm with that reported in [76] . Our method generates exactly the same reliability result for network in Figure 5.7 without passing from a cutset manipulation and without utilizing external tools.

Table 5.7 shows for each possible flow the probability to obtain the value and the cumulative probability of being able to transfer a determinate amount of flow. The first column in Table 5.7 reports all the possible values of flows that can be transmitted between s and t. Each value gives origin to a different terminal leaf in the ADD construction. The second column reports the corresponding probability and the third column the cumulative probability of transmitting up the the corresponding flow value.

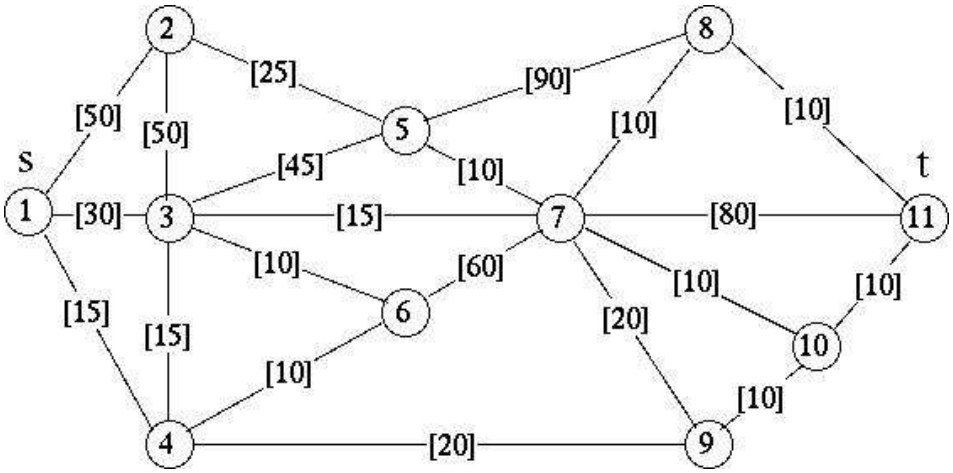

Figure 5.7: Network of paper [76]

(s-t) Max flow	Prob	Prob Cum
0	0.0023361	
10	0.0198815	0.9976639
15	0.0096750	0.9777824
20	0.0793581	0.9681074
25	0.0030942	0.8887493
30	0.0125073	0.8856552
35	0.0106024	0.8731479
40	0.0186152	0.8625455
45	0.1065640	0.8439303
50	0.0359151	0.7373663
55	0.0719313	0.7014512
60	0.0481033	0.6295199
65	0.1803750	0.5814166
70	0.1028710	0.4010416
75	0.1147200	0.2981706
80	0.0166772	0.1834506
85	0.1667720	0.1667734

Table 5.7: Probability Example $RW_{min} = 1$

5.6 Reliability Computation

After the creation of the ADD representing the weighted network it is possible to compute the probability of reach each ADD leaf.
As said in previous section each path in the ADD from the root node to a specific leaf represent the assignment of the variables that allow to obtain a specific flow or distance.
In particular we have implemented three different algorithms:

The first algorithm (algorithm 5.4) visits of the ADD through a depth

Algorithm 5.4 Compute reliability by DFS

```
1: prob_dfs(add F, double prob)
2: if F is Terminal Node then
3:    leaf_array[F] += prob;
4: else
5:    prob_dfs(F → 1 , (prob * prob_x) )
6:    prob_dfs(F → 0 , prob * (1 - prob_x) )
7: end if
```

first search (DFS) and computes the probability for each terminal node. The algorithm is similar to the algorithm used for reliability computation in case of unweighted networks using BDD. In this case it is not possible to record intermediate results because the probability value of each internal node is different for each terminal node. In this case the computation complexity depends on the number of paths inside the ADD and not on the number of nodes of the ADD.

Algorithm 5.5 Compute reliability by split

```
1: prob_split_dfs(add F)
2: set visited
3: set term
4: find_terminals(visited, terminals, F)
5: for each terminal build BDD representing the terminal do
6:    termNode is a terminal
7:    double result = prob_bdd(termNode, F);
8:    term_prob[termNode] = result;
9: end for
```

The key idea of the second algorithm (algorithm 5.5) is to split the ADD in different BDDs in order to compute the reliability over a smaller data

structure. The probability is computed considering only a terminal value at the time: for each value all the paths connecing the root node with the terminal leaf are considered. In this case some intermediate results can be reused as in the case of the unweighted networks.

Algorithm 5.6 Compute reliability by BFS

```
1: visited → set of visited add node
2: no_term_prob → map of internal node probability value
3: visited.insert( F)
4: no_term_prob[F] = 1.0
5: while !visited.empty() do
6:    node= visited.pop
7:    F1 = F → 1
8:    if  F1 is Terminal Node  then
9:       leaf_array[F1] += prob_x * no_term_prob[node];
10:   else
11:      no_term_prob[F1] += prob_x * no_term_prob[node];
12:      visited.insert(F1)
13:   end if
14:   F0 = F → 0
15:   if  F0 is Terminal Node  then
16:      leaf_array[F1] +=(1 - prob_x) * no_term_prob[node];
17:   else
18:      no_term_prob[F1] += (1 - prob_x) * no_term_prob[node];
19:      visited.insert(F0)
20:   end if
21: end while
```

The third algorithm uses a breadth first search (BFS) in order to visit the ADD and to compute the probability.
The probability can be computed starting from the root node, the prob-

	DFS	SPLIT	BFS
2x2	0.006	0.033	0.021
3x3	0.031	0.141	0.086
4x4	2.740	0.541	0.321
5x5	3486.548	3.464	1.951

Table 5.8: Comparison of reliability algorithm results

ability value of the parent node is multiplied with the probability of the node itself and assigned to each child node. We traverse each node only one time so the complexity of the algorithm depends on the number of nodes.

In Table 5.8 the results for the different algorithms are shown in order to compare the different performance.
As benchmark we use different grids of $n \times n$ nodes. The first column of Table 5.8 reports the network size, the second column reports the running time, in seconds, for the *DFS* algorithm, the third regards the *SPLIT* algorithm and the fourth the *BFS* algorithm.
As we can note for small networks (2×2 and 3×3 networks) the fastest algorithm is *DFS*, but if the network grows the fastest becomes *BFS* (in 5×5 network the running time is 1.951 seconds with BFS and 3486.548 with DFS). This is due to the fact that in the BFS algorithm we visit each node only one time (the complexity depends on the number of nodes), while in the DFS algorithm some node can be traversed several times (the complexity depends on the number of paths). In $SPLIT$ algorithm we need to build a BDD for each terminal so it is less efficient than BFS algorithm.

5.7 Service Availability of the GARR Network

With reference to Figure 4.15, we have assigned possible weights to the arcs of the network in two different ways:

- Weights are the bandwidths as derived from the legend of Figure 4.15.
- Weights are the geographical distances between any two nodes. The distances are measured in kilometers from a corresponding geographical map. In this case, we make the assumption that the cost of the connection between any two nodes is proportional to the length of the connection itself.

As in the previous case we consider two couples of source and sink nodes: $TO - CT$ and $TS1 - NA$. Furthermore, we assume that all the links have a probability $p = 0.9$ of being up and a probability $1 - p = 0.1$ of being down. The network is considered as non directed so that the various links can be traversed in both directions.

5.7.1 Weighted GARR Network: Bandwidth Availability

Resorting to the GARR Web site the bandwidth of the major network connections can be obtained. In this way, we can compute the probability of assuring a given capacity level between any two nodes. If we don't fix a minimum bandwidth, the analysis tool provides all the capacities that can be established between the source and the sink node. For the connection $TO - CT$ we have assumed a minimum threshold $\psi_{min} = 2500\ Mbps$, so that only the connections that carry a capacity greater than the minimum value ψ_{min} are reported in Table 5.10. The table is ordered from the maximum achievable bandwidth ($5000\,Mbps$) to the minimal capacity exceeding

$\psi_{min} = 2500$. The row with zero entry corresponds to the probability that the two nodes are not connected or are connected with a capacity lower than the fixed minimum.

Flow (Mbps)	**Probability**
5000	0.5795
2830	0.0014
2820	0.0006
2810	0.0002
2685	0.0069
2675	0.0055
2665	0.0036
2655	0.0096
2530	0.0027
2520	0.0020
2510	0.0201
2500	0.3292
0	0.0382

Table 5.9: Maximum flow and corresponding probability between $s = TO$ and $t = CT$

From Table 5.9, the probability that a capacity greater than $\psi_{min} = 2500$ $Mbps$ is carried between TO and CT is obtained by summing all the rows satisfying the constraint and is equal to 96% (also equal to 1 minus the probability of the row with zero value).

For the connection $TS1 - NA$ we have assumed a minimum threshold $\psi_{min} = 1$, so that all the connecting paths are included in the analysis. The results for the connection between $TS1$ and NA are reported in Table 5.10. The row corresponding to the zero value represents the probability that there is no connection between source and sink.

If we set in Table 5.10 a minimum threshold $\psi_{min} = 2500\ Mbps$ we get that the probability that a capacity greater than $\psi_{min} = 2500\ Mbps$ is carried between $TS1$ and NA is equal to 87%.

Flow (Mbps)	**Probability**
2655	0.6366
2520	0.0002
2510	0.0188
2500	0.2176
340	0.0004
330	0.0038
320	0.0017
310	0.0007
185	0.0016
175	0.0028
165	0.0015
155	0.0808
30	0.0003
20	0.0005
10	0.0060
0	0.0263

Table 5.10: Maximum flow and corresponding probability between $s = TS1$ and $t = NA$

5.7.2 Weighted GARR Network: Distance Availability

In this second experiment, we suppose that the cost of the connection is proportional to its distance. To any edge we assign a weight corresponding to the distance (in km) between the pair of nodes and we calculate the probability to reach the sink node with a distance less than a preassigned

threshold.

Assuming as source node TO and as sink node CT the obtained results are reported in Table 5.11. Since the minimal distance of a connection $TO - CT$ is 1522 Km, in generating Table 5.11 we have assumed a threshold $\psi_{max} = 1700\ km$ so that in Table 5.11 only the connections with a distance lower than ψ_{max} are quoted. The row with value zero reports the probability that the connection is not established or it is established with a total length greater than ψ_{max}.

Distance (Km)	**Probability**
0	4.43418e-02
1522	6.56100e-01
1525	1.24659e-01
1536	5.90490e-02
1539	1.12193e-02
1561	6.91464e-02
1564	1.60023e-02
1567	9.34540e-03
1575	6.22317e-03
1578	1.44021e-03
1581	8.41086e-04
1635	1.09153e-03
1649	9.82374e-05
1658.9	3.13806e-04
1672.9	2.82426e-05
1677	9.22750e-05
1691	8.30475e-06

Table 5.11: Connection distance between TO and CT lower than $\psi_{max} = 1700\ km$

To give a more detailed picture of the distance values through which the

source node TO can reach the sink node CT, we have reported in Figure 5.8 the histogram and the corresponding empirical cumulative distribution function of the distances of all the possible paths connecting TO to CT.

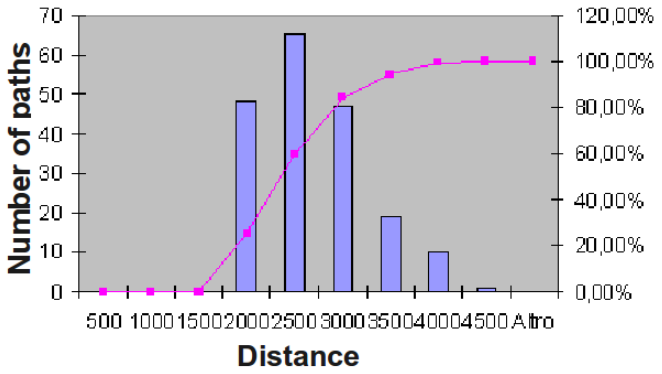

Figure 5.8: Histogram of the distance values for $s = TO$, $t = CT$

We have repeated the distance computations between $TS1$ and NA. In this case, we have assumed as a maximum acceptable distance $\psi_{max} = 1300\ km$. Table 5.12 reports the obtained valued ordered vs increasing distance. The row with value zero reports the probability that the connection is not established or it is established with a total length greater that ψ_{max}.

Similarly to the previous case, we have reported in Figure 5.9 the histogram and the corresponding empirical cumulative distribution function of the distances of all the possible paths connecting $TS1$ to NA.

Distance (Km)	Probability
0	6.4723e-02
923	5.9049e-01
1145	4.3046e-02
1187	3.8742e-03
1211	1.9076e-01
1229	6.3688e-02
1231	5.0046e-03
1237	4.5041e-04
1250	3.0404e-02
1253	4.5582e-03
1256	1.6864e-03
1264	1.1979e-03
1273	1.0173e-04
1279	9.1563e-06

Table 5.12: Connection distance between $TS1$ and NA lower than $\psi_{max} = 1300\ km$

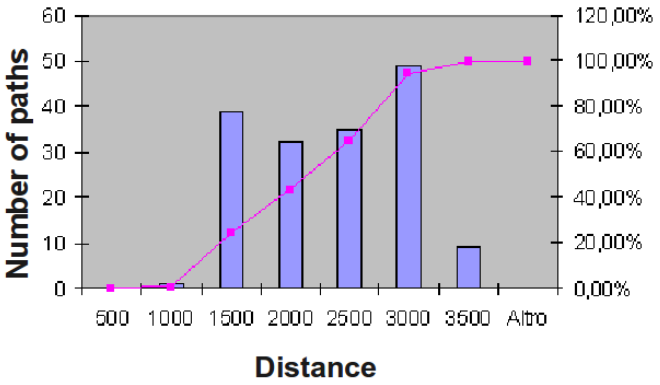

Figure 5.9: Histogram of the distance values for $s = TS1$, $t = NA$

5.8 IEEE 118 Bus Test Case

We have applied the concepts and tools of weighted networks to the IEEE 118 Bus Test Case. The weights have been obtained by resorting to the data available on the web site *www.ee.washington.edu/research/pstca*. In particular, we have derived for each arc the *resistance* and the *capacity*. The actual value of the electrical resistance is obtained from the available data, that are given as resistance *per unit*, by resorting to the base voltage. For the capacity we have assumed the apparent power. The value of resistance and capacity for each link are reported in Table 5.13. The link are identified (see Figure 4.22) by means of the two terminal nodes.

Table 5.13: Data for IEEE118 bus network

From	To	Capacity	Resistance
2	1	12.35	5.7703320
3	1	38.65	2.4566760
5	4	103.23	0.3351744
5	3	68.11	4.5896040
5	6	88.47	2.2662360
6	7	35.54	0.8741196
9	8	440.64	2.9042100
8	5	338.47	0.0000000
10	9	445.25	3.0708450
4	11	64.23	3.9801960
5	11	77.22	3.8659320
11	12	34.29	1.1331180
12	2	32.45	3.5612280
12	3	9.79	9.2172960

Table 5.13: Data for IEEE118 bus network

From	To	Capacity	Resistance
7	12	16.48	1.6415928
11	13	35.09	4.2372900
12	14	18.31	4.0944600
13	15	0.77	14.1687360
14	15	4.24	11.3311800
12	16	7.51	4.0373280
17	15	103.86	2.5138080
17	16	17.51	8.6459760
17	18	80.27	2.3424120
18	19	19.39	2.1310236
20	19	10.62	4.7990880
15	19	11.53	2.2852800
21	20	28.67	3.4850520
22	21	42.84	3.9801960
28	29	15.66	4.5134280
30	17	231.19	0.0000000
8	30	74.16	5.1299775
17	31	14.77	9.0268560
31	29	8.42	2.0567520
32	31	29.86	5.6751120
15	33	7.31	7.2367200
34	19	3.59	14.3210880
35	36	0.84	0.4265856
37	35	33.84	2.0948400
37	33	15.72	7.9032600

Table 5.13: Data for IEEE118 bus network

From	To	Capacity	Resistance
34	36	30.25	1.6587324
37	34	94.31	0.4875264
38	37	243.37	0.0000000
37	39	54.91	6.1131240
37	27	44.02	11.2930920
30	38	62.35	5.5227600
39	27	26.92	3.5040960
27	26	15.45	2.7613800
25	27	11.84	10.5694200
25	26	21.59	7.8080400
17	24	2.06	1.7387172
32	24	4.12	11.7120600
12	23	20.15	6.2654760

Assuming the couple of source and sink node $s = 10$ and $t = 22$ and $s = 1$ and $t = 35$, as in Table 4.2, we have applied the ADD computations of Section 5.3. In the absence of available significant data about the probability of each arc of being up or down, we have assumed, for the sake of illustration, a common value $p = 0.9$.

5.8.1 Weights as Resistances

We have assigned as a weight to each arc its electrical resistance derived from Table 5.13. Assuming a maximum threshold $\psi = 50$ we have derived the ADD and calculated the probability of having a connection with a resistance less than the threshold. For the couple of nodes $s = 10$, $t =$

22, the results are displayed in Table 5.14. The first column provides the resistance of the connection and the second one the probability of that connection. The first row with value 0 provides the probability that the $s - t$ connection cannot be established with a resistance lower than ψ_{max}.

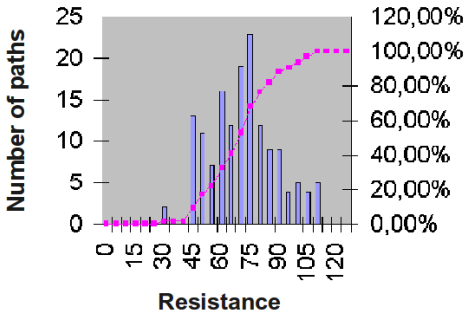

Figure 5.10: Histogram of the resistance values for $s = 10$, $t = 22$

By assuming a maximum threshold ψ_{max} large enough to include all the possible values of the resistance, we have reported in Figure 5.10 the histogram (and the related empirical cumulative distribution function) of the number of occurrences of a path of a given resistance.

For the case $s = 1$, $t = 35$, and with $\psi_{max} = 35$, the results are displayed in Table 5.15. By assuming a maximum threshold ψ_{max} large enough to include all the possible values of the resistance, we have reported in Figure 5.11 the histogram (and the related empirical cumulative distribution function) of the number of occurrences of a path of a given resistance.

Table 5.14: Probability resistance for s=10 t=22 and $\psi_{max} = 50$

Resistance	Probability
0	0.4214117
27.84	0.3874200
28.17	0.0736099
40.18	0.0482955
40.4	0.0145423
40.5	0.0091761
40.72	0.0027630
40.73	0.0153681
40.84	0.0013088
40.95	0.0046275
41.17	0.0002487
41.4	0.0004165
42.8	0.0054480
43.25	0.0004903
43.7	0.0118619
44.85	0.0000295
45.43	0.0005692
45.65	0.0001714
46.1	0.0000154
46.32	0.0008333
47.39	0.0006438
47.5	0.0002217
47.95	0.0000200
49.2	0.0003521
49.53	0.0000650
49.55	0.0000018
49.76	0.0000884

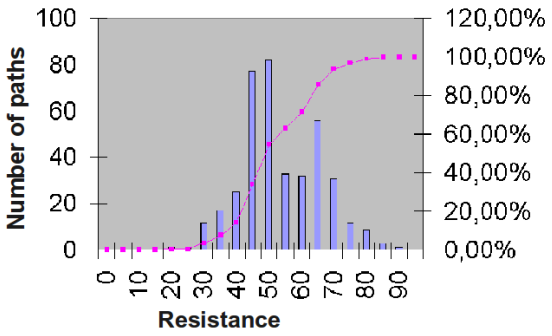

Figure 5.11: Histogram of the resistance values for $s = 1$, $t = 35$

5.8.2 Weights as Capacities

As a second example we have assigned to each arc the capacity value displayed in Table 5.13, which corresponds to the apparent power carried by the branch. The power values listed in the table have been arbitrarily assumed to correspond to the maximum capacity that the branch is able to transmit.

For the couple of nodes $s = 10$, $t = 22$, and with a minimum capacity threshold of $\psi_{min} = 9$, the results are displayed in Table 5.16. The first column provides the capacity of the connection and the second one the probability of that connection. The first row with value 0 provides the probability that the $s - t$ connection cannot be established with a capacity greater than ψ_{min}.

By assuming a minimum threshold ψ_{min} small enough not to exclude any possible value of the capacity, we have reported in Figure 5.12 the histogram (and the related empirical cumulative distribution function) of the number of occurrences of a path of a given capacitance.

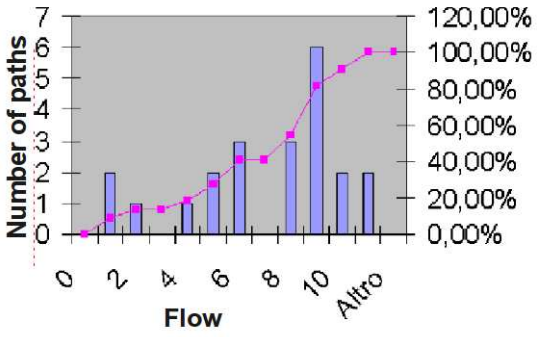

Figure 5.12: Histogram of the flow values for $s = 10$, $t = 22$

For the couple of nodes $s = 1$, $t = 35$, and with $\psi_{min} = 20$, the results are displayed in Table 5.17. The histogram of the number of occurrences of paths with a given capacity is reported in Figure 5.13.

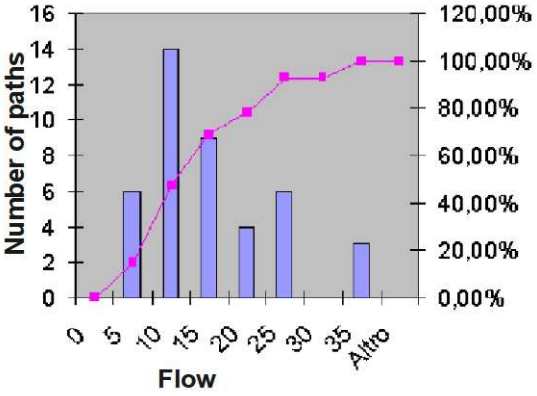

Figure 5.13: Histogram of the flow values for $s = 1$, $t = 35$

Table 5.15: Probability resistance for s=1 t=35 and $\psi_{max} = 35$

Resistance	Probability
0	0.1469162
19.71	0.4782970
20.17	0.0387420
26.83	0.0662489
27.05	0.0199483
27.29	0.0053662
27.50	0.0017954
27.51	0.0016158
27.96	0.0001454
29.16	0.0066249
29.38	0.0019948
29.61	0.0838911
29.62	0.0005366
29.83	0.0001795
29.84	0.0001616
30.07	0.0067952
30.29	0.0000145
31.83	0.0736099
31.94	0.0157501
32.03	0.0011329
32.25	0.0003411
32.29	0.0059624
32.40	0.0012758
32.49	0.0000918
32.70	0.0000307
32.71	0.0000276
32.96	0.0212607
33.16	0.0000025
33.28	0.0030302
33.46	0.0034112
33.78	0.0004544
34.87	0.0143454

Table 5.16: Probability flow for s=10 t=22 and with $\psi_{min} = 9$

Flow	Probability
0	0,4555787
9,12	0.0000572
9.79	0.0000901
10.56	0.0001900
10.62	0.5440840

Table 5.17: Probability flow s=1 t=35 $\psi_{min} = 20$

Flow	Probability
0	0.4843242
20.07	0.0000161
20.72	0.0004476
20.84	0.0000270
21.49	0.0000592
21.56	0.0000001
22.14	0.0368286
33.84	0.0954271
34.61	0.0000361
34.68	0.3828340

Chapter 6

Complex Networks

6.1 Introduction

In the last decades, due to the progress in computer power and the growing consciousness that both man-made structures and human interactions can be viewed as networks, the interest in this field has been enhanced to a wider scientific audience [15]. The increased complexity of real networked systems requires new analytic approaches to afford their new scale and predict their behavior. There is a wide literature concerning complex networked system analysis [7, 37, 62, 25] that looks at the structural properties of the graph and the rules governing the aggregation of its nodes, with the aim to predict networked system behavior. Among the main structural properties of a graph is its classification as belonging to a specific topological class. Two topological classes are relevant for representing *real network*: Random Graph (RG) networks and Scale Free (SF) networks [7, 62].

However, the complexity of real world present networks (the internet, the www, the public telecommunication networks), can reach millions or

even billions of vertices. This change of scale forces a corresponding change in the analytic approach. Many of the approaches that have been applied in small scale networks, and many of the questions that have been answered are simply not feasible in much larger networks. Recent years have witnessed a substantial new movement in network research [7, 15, 37, 62], with the focus shifting away from the analysis of small graphs to consideration of large-scale statistical properties of graphs and with the aim of predicting what the behavior of complex networked systems will be on the basis of measured structural properties and the local rules governing individual vertices.

6.2 Structure and Characterization of Complex Networks

Many systems that we encounter daily (internet, communication, transport, social etc..) have millions or even billions of nodes and are very difficult to describe and to analyse and are even quite impossible to be represented or plotted. Hence, to characterize some properties of these networks various measures have been defined (see for instance [36]) that can be computed with reasonable memory occupancy and running time even for graphs with millions of nodes. We introduce four of such measures that are probably the most significant ones and we discuss how some features of the graphs emerge through their values. The proposed measures are defined locally (node by node) but take into account progressively more extensive properties of the network.

Degree Distribution - is influenced only by the number of arcs connected to each single node;

Clustering coefficient - looks at the first neighbors of a node and how they are connected to each other;

Length of the shortest path - searches for the minimal length of a path connecting any pairs of nodes of the network;

Betweenness Centrality - Counts the percentage of shortest paths traversing a given node.

6.2.1 Degree Distribution

Given a network the total number of connections of a vertex is called its **degree** k.
In a directed network, at any node n we need to distinguish between the *input degree* k_{in} (counting the number of arcs entering into the node), and the *output degree* k_{out} (counting the number of arcs exiting from the node). So $k = k_{in} + k_{out}$ as shown in figure 6.1. In an undirected network, the degree k of a node n is equal to the number of arcs emerging from the node, or equivalently the number of nodes directly reachable from the node i with a single hop, some times called the first neighbouring nodes.
If in a graph each vertex has the same degree, this graph is said to be *regular*.

The *Degree* is the most used parameter since its distribution discriminates easily the structure of a network. In particular, a frequently utilized measure is the degree distribution as a function of the number of nodes. Generally the degree of nodes in random networks is statistically distribuited. For each node we can compute the degree distribution $p_x(k)$, the probability that the node x has k connections.
The total degree distribution can be simply computed starting from the

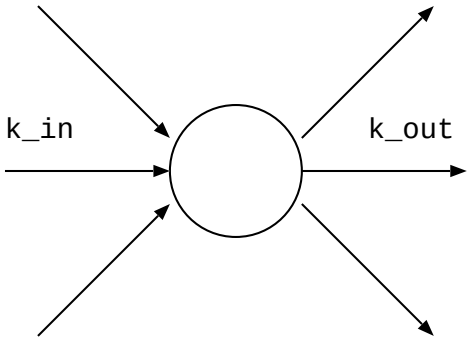

Figure 6.1: Degree of a node in a directed network

degree distribution of each node.

$$P(k) = \frac{1}{N} \sum_{x=1}^{N} p_x(k) \tag{6.1}$$

$P(k)$ represents the percentage of nodes with degree k.
Even if the degree is a local property of the node, nevertheless the distribution $P(k)$ provides fundamental information about the overall structure of the network as we see in the next sections.

6.2.2 Clustering

The clustering coefficient C_i is a measure associated to a single node i. It is still a local measure but that looks at the connectivity of the first neighbors of the node i with each other. Given a node i we define with G_i the subgraph obtained from G by taking the first neighbors of i, removing node i together with all the arcs emerging from i. If node i has degree k_i, the subgraph G_i has k_i nodes and at most $k_i(k_i - 1)/2$ arcs. The clustering

coefficient C_i measures the number of arcs really existing in subgraph G_i with respect to all the possible arcs, hence it is defined as a positive number between 0 and 1. A value of C_i close to 1 means that all the neighbors of i are connected each other.

The clustering coefficient C is the mean value of the C_i averaged over all the nodes.

6.2.3 Length of the Shortest Path

Given two nodes i and j, we define d_{ij} the shortest path (or the minimal distance) between the two nodes; i.e. the minimal number of arcs that must be traversed to reach j from i. It is possible to introduce the distribution of the shortest-path lengths between pairs of vertices of a network and the average shortest-path length L of a network. The average here is over all pairs of vertices between which a path exists. L is often called the *diameter* or *the characteristic length* of a network. It determines the effective *linear size* of a network, the average separation of pairs of vertices.

If in a network all the N nodes are connected to all the $N-1$ remaining nodes, we have of course $L = 1$, the total number of arcs is $N(N-1)/2$ and the degree is constant and equal to $k = (N-1)/2$. If k is the average node degree, from any node we can reach on the average k nodes in one step, k^2 nodes in two steps and so on, so that in L steps all the nodes are reached. Hence approximately $N \simeq k^L$ or $L \simeq \ln N / \ln k$. Usually L is a small number that increases slightly with the increasing number of nodes. This fact implies that the distance between two nodes is usually a small number, and this property is known as *small world* property [81].

One can also introduce the maximal shortest-path length over all the pairs of vertices between which a path exists. This characteristic determines

the maximal extent of a network.

6.2.4 Betweenness Centrality

In large complex networks, not all nodes are equivalent, for what concerns the effect of their removal on the connectivity property of a network. The question of the importance of nodes in a network is thus of primary interest since it concerns crucial subjects such as networks resilience to attacks [8, 36] and also immunization against epidemics [67]. Different measures have been proposed in order to quantify the concept of importance or centrality of a node. One may have the idea that the connectivity degree is a good candidate as a centrality measure. But this is not true since a node may have a very low degree but may be traversed by all the paths that connect two subgraphs. The reason is that connectivity is a local quantity which does not inform about the importance of the node in the network [17].

A good measure of the centrality of a node has thus to incorporate a more global information such as its role played in the existence of paths between any two given nodes in the network. This measure is called *betweenness centrality* and is defined as the fraction of shortest paths going through a given node [17]. More formally:

$$B(i) = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{6.2}$$

where i is a generic node, σ_{st} is the number of the shortest paths from node s to node t, and $\sigma_{st}(i)$ is the number of the shortest paths from node s to node t passing through node i. High values of the centrality indicate that a node can reach the others on shortest paths or that this vertex lies on many shortest paths. If one removes a node with large centrality it will

lengthen the paths between many pairs of nodes, or it will disconnect the network in new components.
Betweenness centrality is a measure of the influence a node has over the spread of information through the network. By considering only shortest paths, however, we implicitly assume that information spreads only along those shortest paths. In [61] Newman proposes a betweenness measure that relaxes this assumption, including contributions from essentially all paths between nodes, not just the shortest, although it still gives more weight to short paths.
Also our analysis, described in next sections, follow the idea that all the paths contribute to the network connectivity.

6.3 Regular Networks

The most relevant topological classes of networks are Regular, Random and Scale Free networks.

Regular networks are represented by graphs that can be described by means of defined geometrical properties and can be very well suited for scalable benchmarks. We have investigated two types of regular networks:

1. *Symmetric networks*: all the nodes have the same characteristics. The nodes are usually represented on a circle and are connected to the same number of adjacent nodes. Hence, a network of this type can be described by two parameters: the number of nodes N and the degree k. Figure 6.2 shows the symmetric network with $N = 5$ and $k = 2$ (note that, in the figure, the arcs are oriented counterclockwise).

2. *Lattice networks*: vertices are located on a regular square grid of $N \times N$ nodes [75, 46]. An example is shown in Figure 6.3

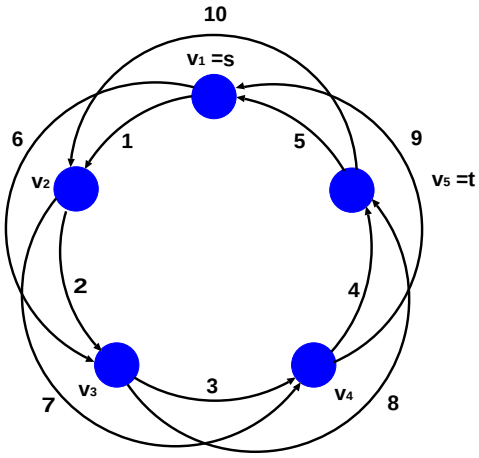

Figure 6.2: A regular network with $N = 5$ and $k = 2$

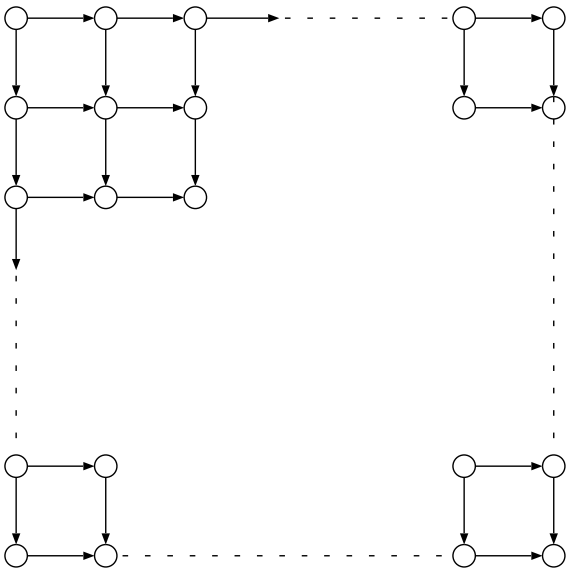

Figure 6.3: Lattice network

6.4 Random Network

Paul Erdős and Alfréd Rényi are the first that in their studies [39] define a Random Graph (RG) as N labeled nodes connected by n edges which are chosen randomly from the $\frac{N(N-1)}{2}$ possible edges.
In total there are $\left(\begin{array}{c} n \\ \frac{N(N-1)}{2} \end{array}\right)$ different graphs with N nodes and n arcs possible.
An alternative way in order to obain a *RG* is to start with N nodes and connect each pair of the nodes with probability p. The total number of edges is a random variable with the expectation value $E(n) = p\frac{N(N-1)}{2}$.

In RG networks, the degree distribution $P(k)$ follows a Poisson distri-

bution;

$$P(k) \approx e^{-pN} \frac{(pN)^k}{k!} = e^{-<k>} \frac{< k >^k}{k!} \tag{6.3}$$

with a peak in $P(< k >)$. The network degree is characterized by an average value $< k >$ with a given standard deviation.

6.5 Scale Free Network

In SF networks, the degree distribution $P(k)$ follows a power-law [63]:

$$P(k) \sim k^{-\gamma} \tag{6.4}$$

where γ is a real positive constant. SF networks manifest when nodes are highly non-equivalent. Such networks have been named *Scale Free* because a power-law has the property of having the same functional form at all scales. In fact, power-laws are the only functional forms that remain unchanged, apart from multiplicative factors, under a rescaling of the independent variable k. SF networks result in the simultaneous presence of a small number of very highly connected nodes (the hubs) linked to a large number of poorly connected nodes (the leaves). RG networks can be thought as resulting from un-supervised growth processes and, as such, are believed to be produced by a growth mechanism where new nodes stuck randomly to existing nodes (random growth mechanism). On the other hand, SF networks grow under the action of some selective pressure that determines that a new node sticks more likely to an already highly connected node. This growth model, known as *preferential attachment* [7], is realized by assuming that the probability $P_i(n+1)$ of the $(n+1)$-th generated node sticks to node i, increases linearly with the degree of i. Higher

degrees favor higher attachment probability. $P_i(n+1)$ obeys the rule:

$$P_i(n+1) \sim \frac{k_i}{\sum_{j=1}^{n} k_j} \tag{6.5}$$

For what concerns reliability studies on networks, the most relevant issue is that the structure of the network influences its reliability. SF networks are more robust to random arc or node removal than RG networks but are more prone to malicious attacks, where the nodes with highest degrees are first removed. A survey on these results is in [8, 36].

6.6 Complex Network Analysis

The dependability of systems in the form of networks is a very important issue in network analysis, since one of the main feature of networks is to have usually many redundant paths between any pair of connected nodes. In order to study the dependability of networks, the probability that the elements of the networks are correctly functioning (or non functioning) should be known as an input parameter. We assume that the arcs can be modelled as binary entities and failure means a complete disruption of the link and of the interruption of the carried service.

In a probabilistic network given two nodes, a source node s and a termination node t, we define the $s-t$ reliability as the probability that the two nodes are connected by at least one functioning path. For small networks an exhaustive analysis is possible that searches for all the possible paths between s and t and computes exactly and analytically the probability that at least one path is up. The limits of the exhaustive approach have been checked in [23] and compared with similar results in the literature [47]. For general networks exhaustive reliability analysis is limited to graphs with

few tenths of node.

6.7 Exact algorithms

In this section we analyse some networks using exact algorithms.

6.7.1 Network generating algorithm

In order to launch a benchmark to examine measures of different topology of networks we need to have a preliminary algorithm to generate networks of the desired dimension and topology. To this end, we have implemented two algorithms to respectively grow a RG and a SF network. The growth mechanism of RG requires three input parameters: the number of connections (or degree k) that a newly generated node may establish with the already generated nodes; the probability of attachment p; and the final dimension of the network N. Calling n the number of current nodes during the generation process, the network growth starts with $n = k$ nodes and a new node at the time is generated until $n+1 \leq N$. Two random generators are used to establish the attachment of the $n + 1$ node. A first generator sorts a node i among the n nodes of the network ($i \leq n$) at which the $n+1$ node could be attached. Node i will be considered for the attachment if it has not been already sorted. To generate a RG, a second generator sorts a random number p_1. If $p_1 \leq p$, the $n + 1$ node will be attached to node i, otherwise the first random generator sorts a new node $j \neq i$ and the attachment process is repeated k times.

To generate a SF network , what changes is the attachment process (see Barabasi Scale free model [7]). A first generator sorts a node i among the n nodes of the network ($i \leq n$) at which the $n + 1$ node could be attached.

Node i will be considered for the attachment if it has not been already sorted; in that case we compute $P_i(n+1)$ according to Equation 6.5. Then a second generator sorts a random number p_1. If $p_1 \leq P_i(n+1)$ then the new $n+1$ node is attached to node i, otherwise the first random generator sorts a new node $j \neq i$.

The growth algorithm is incremental, in the sense that once we have generated a graph with N nodes of any topology, we can add to the same structure an arbitrary number of new nodes.

6.8 Scalable Benchmark

A scalable benchmark has been carried out to evaluate and to compare property against the increasing complexity of different network topologies: namely Symmetric, Lattice, RG and SF networks. In all the experiments, a failure probability equal to $p = 0.9$ is uniformly assigned to all the arcs, only. The experiments have been run on a SUN machine with a dual AMD Opteron 2.3 GHz processor, and with 4 GB of memory.

6.8.1 Regular Networks

Symmetric networks depend on two parameters, the number of node N and the degree k. In Figure 6.2 is depicted a regular network with $N = 5$ and $k = 2$. In all the experiments `s` and `t` are two adjacent nodes. The results for increasing values of N and k are reported in Table 6.1.

The results obtained from the two algorithms differ only in the number of generated BDD nodes, since they use a different ordering for the variables. In Table 6.1, the column `(# BDD nodes VA)` reports the number of BDD nodes generated in the visit algorithm (see Section 4.1.4), while the

Table 6.1: Benchmark Results on symmetric networks

N	k	# arcs	# minpath	# BDD nodes VA	# BDD nodes MPA	`(s,t)` reliability
10	2	20	55	33	274	0.97137
10	4	40	755	2156	755	0.99980
10	7	70	19900	355650	62764	1
20	2	40	6765	73	43041	0.96096
20	3	60	83929	594	*n.a.*	0.99793
30	2	60	832040	113	*n.a.*	0.95065
35	2	70	n.a.	133	*n.a.*	0.94554

`(# BDD nodes MPA)` reports the number of BDD nodes generated in the minpath algorithm (see Section 4.1.3) and the value of the reliability.

The results for scalable (directed) lattice graphs [75, 46], network in Figure 6.3 with different values of N, are reported in Table 6.2. The table displays the dimension of the lattice, the number of nodes and arcs of the graph, the number of minpaths, the number of nodes of the BDD resulting from the application of the VA and MPA algorithms, respectively, and the value of the reliability. The large variability in the number of BDD nodes obtained from the application of the two algorithms reflects the well known assertion that the complexity of the BDD strongly depends on the variable ordering. The actual experiments support the conjecture that with low average degree $< k >$ the VA ordering is more efficient than the MPA ordering.

The number of minpaths versus the lattice dimension is plotted in Figure 6.4. An exponentially rising curve is evident from the plot, as theoretically argued in [75].

Table 6.2: Benchmark Results on Directed Lattice Graph

Dimension	#nodes	# arcs	# minpath	# BDD nodes VA	# BDD nodes MPA	(s,t) reliability
2×2	4	4	2	6	6	0.96390
4×4	16	24	20	94	117	0.97308
6×6	36	60	252	1034	2012	0.97485
8×8	64	112	3432	8348	24219	0.97511
10×10	100	180	48620	56338	*n.a.*	0.97514
12×12	144	264	705432	342038	*n.a.*	0.97515

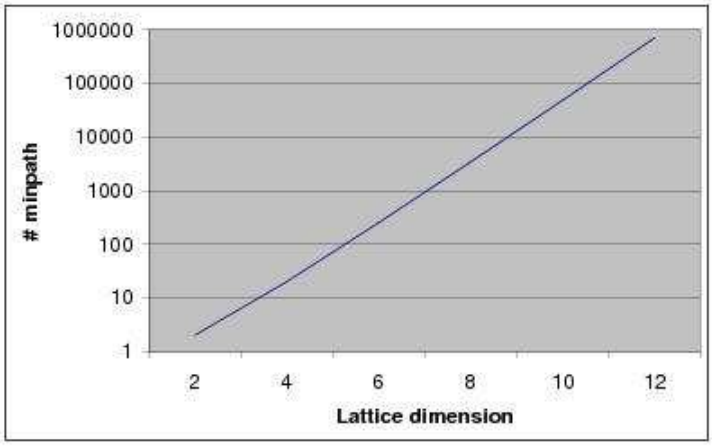

Figure 6.4: BDD complexity of the connectivity function of a lattice graph

Table 6.3: Benchmark Results on Random Graphs

#nodes	#arcs	clustering coefficient	# minpath	# BDD nodes VA	# BDD nodes MPA	(s,t) reliability
20	72	0.1632	2046	316360	10032	0.98885
25	92	0.1267	21040	3333930	32653	0.98886
30	112	0.1502	119033	*n.a.*	708801	0.98906
40	152	0.1379	6757683	*n.a.*	*n.a.*	*n.a.*
50	192	0.0660	*n.a.*	*n.a.*	*n.a.*	*n.a.*

6.8.2 Random Graphs and Scale Free Networks

Tables 6.3 and 6.4 display the results obtained on RG and SF networks, respectively. Networks are grown according to the algorithm of Section 6.7.1, with an increasing number of final nodes N, while keeping constant the number of connections ($k = 2$), and, for RG networks only, the probability of attachment ($p = 1$). The first generated node is assumed as the source `s` and the last generated node as the sink `t`.

The first three columns of each table Table 6.3 - 6.5 report the final number of nodes, the final number of edges and the clustering coefficient. Columns 4, 5, 6 report the number of minpaths, the number of the nodes of the BDD with VA algorithm, the number of the nodes of the BDD with MPA algorithm and the reliability value respectively.

All the results in Tables 6.3 and 6.4 have been obtained on incremental networks, in the sense that the network with $M > N$ nodes has been obtained by starting from the network with N nodes and adding the remaining $M - N$ nodes. This means that the network in any line of Tables 6.3 and 6.4 contains all the nodes and arcs of the networks of the preceding lines. Also in this case the results show a large variability and the minpath search seems to be more efficient than the direct BDD construction.

Table 6.4: Benchmark Results on Scale Free networks

#nodes	#arcs	clustering coefficient	# minpath	# BDD nodes VA	# BDD nodes MPA	(s,t) reliability
20	74	0.5869	263	10409	266	0.98010
25	94	0.4388	2651	186350	5019	0.97815
30	114	0.3927	4707	*n.a.*	239788	0.98001
40	154	0.3888	73680	*n.a.*	*n.a.*	*n.a.*
50	194	0.3499	*n.a.*	*n.a.*	*n.a.*	*n.a.*

Table 6.5: Comparative repeated experiments on different RG and SF networks

Network Type	#nodes	#arcs	clustering coeff (mean value)	# minpath Min	# minpath Max	# minpath mean value
RG	25	92	0.1477	3566	36661	17568
SF	25	94	0.2819	750	12135	6117

The results in Tables 6.3 and 6.4 reflect the stochasticity of the generated networks and the choice of `s` and `t`. To mitigate these effects, we have repeated different tests to confirm the results and their general trends. In the first set of experiments we have generated 10 different RG and SF networks (starting with different seeds in the random generators) with the same number of nodes and arcs. In the second set of experiments we have run the algorithms on the same network but changing randomly the source `s` and the sink `t`. The results of the first set of experiments are reported in Table 6.5

The results of the second set of experiments are reported in Table 6.6. By comparing the whole set of results for RG and SF it appears clear that RG's display a lower clustering coefficient (as known from the literature since a pure random connection does not favor a local coalescence) and a

Table 6.6: Comparative repeated experiments modifying `s` and `t`

Network Type	#nodes	#arcs	# minpath Min	# minpath Max	# minpath mean value
RG	25	92	7920	23392	16850
SF	25	94	794	2764	1882

higher number of minpaths than SF. The presence of the hubs reduces the total number of possible connections among any two nodes.

6.9 Simulative Analysis of Complex Network Reliability

If the dimensions of the network exceed the capability of exact algorithms the possible solution is to resort to simulation. Simulation is still the principal and most diffused investigation tool for complex networks [8, 36].

In this section, we discuss a simulative approach to the computation of the point to point reliability in a complex network.
We first describe the implemented simulative approach and then the results.

6.9.1 Simulation

The Monte Carlo method of reliability prediction is useful when system complexity makes the formulation of exact models essentially impossible. The characteristics of the Monte Carlo method makes it ideal for estimating the reliability of systems. Instead, system components need only be tested for failure during operation, which ensures that components which are used more often contribute proportionally more to the overall reliability estimate

6.9.2 Reliability Simulation

Because the reliability of each component is based on probability distributions, the reliability of each component in a system can be modelled by a set of random numbers. By generating random numbers, it is possible to simulate the state of each component. These component states can then be combined using the structure function to determine the state of the system. Since each execution of a simulation tells only whether a particular set of conditions did or did not exist, the Monte Carlo method is an experimental problem-solving technique such that many simulation runs have to be made to understand the relationships involved in the system. Each repetition of the simulation results in another independent estimate of the reliability of the system. As the number of simulations increases, the sample mean of these independent estimates approaches the actual characteristics of the system.
In order to obtain an estimation of the point to point reliability of the different networks we apply a simple simulation algorithm.
The algorithm for the computation of the point to point reliability via simulation can be decomposed in the following steps.

1. Read the data of a probabilistic network $G = (V, E, P)$ and fix a source node s and a termination node t.

2. Define a global counter of the number of executions ν_g and a counter for the executions that provide a favorable result ν_f.

3. For each execution and for each arc i of the graph, extract a random number r between 0 and 1 and compare this number with the probability p_i. If $r \leq p_i$ arc i is assumed up, while if $r > p_i$ arc i is assumed down.

4. Once this assignment is completed we verify if s and t are connected by the remaining up links: if they are connected we increase by one the counter ν_f.

5. We increase by one the counter ν_g.

   (a) The reliability as a function of the number of execution ν_g is evaluated by means of the estimator $R_{\nu_g} = \nu_f/\nu_g$

   (b) the estimation error is computed as a function of the current value of ν_g and R_{ν_g}.

6. We start a new execution iterating from step 3. As termination criteria, we can adopt a criterium based on the total number of experiments (by fixing a large value) or on the approximation error.

6.9.3 System Reliability Estimation

Now we use Monte Carlo methods to simulate the behavior of the system and thus to estimate its reliability.
Two special nodes are identified, the source node s and the sink node t; we want to compute the probability that node s is connected with node t knowing the failure probability of each edge.
The simulation algorithm extracts a random value r between 0 and 1 for each arc: if the extracted value exceeds the probability value, the arc is not included into the new network (see step 3 of the algorithm) . After that all edges are examined, we analyse the new network in order to know whether the source node and the sink node are connected (step 4).
We execute different runs with different instances of the network obtained from the original network using different random values r (step 6).
If we compare the number of networks in which the source and the sink

node are connected over the total number of generated networks, we can obtain an estimation of the $s-t$ reliability and an estimation of the error. Assessing the accuracy of the estimates is of extreme importance. Suppose that the (unknown) exact value of the reliability is R and that the simulation yields an estimate R_{ν_g}(step 5a) for it. Since the procedure involves the use of random numbers, R_{ν_g} itself is a random variable. It is not important to obtain a particular value for R_{ν_g} if we have not idea how close that value is to R.
A probabilistic measure of the distance between R_{ν_g} and R should be derived, of the following type:

$$P(\mid R - R_{\nu_g} \mid < \sigma) \geq \alpha, \quad 0 \geq \alpha \geq 1. \tag{6.6}$$

The interval $(R_{\nu_g} - \sigma, R_{\nu_g} + \sigma)$ is called a *confidence interval* for R; the probability α is the *level of confidence* [51].
Assuming that each simulation run provides an indipendent estimate R_{ν_g}, the distribution of n samples is a binomial distribution of parameter $p = R_{\nu_g}$, whose mean value and variance value are given by:

$$p = R_{\nu_g} = \frac{\nu_f}{\nu_g} \quad ; \quad \sigma_{R_{\nu_g}} = \sqrt{\frac{p(1-p)}{\nu_g}} = \sqrt{\frac{R_{\nu_g}(1-R_{\nu_g})}{\nu_g}} \tag{6.7}$$

Hence, assuming a normal approximation for n large, eq. 6.6 becomes:

$$R = R_{\nu_g} \pm z_{\alpha/2}\sqrt{\frac{R_{\nu_g}(1-R_{\nu_g})}{\nu_g}} \tag{6.8}$$

The pseudocode of the implemented algorithm is explained in algorithm 6.1. We analyse two specific networks with different topological features:

Algorithm 6.1 Monte Carlo Simulation

1: $\nu_f \Leftarrow 0$
2: $\nu_g \Leftarrow 1$
3: **while** $\nu_g \leqslant max_{run}$ **do**
4: **for all** link i of N **do**
5: generate random number $r \in P(0,1)$
6: **if** $r \leqslant p_i$ **then**
7: add new edge to G
8: **end if**
9: **end for**
10: **if** G is connected **then**
11: $\nu_f \Leftarrow \nu_f + 1$
12: **end if**
13: $\nu_g \Leftarrow \nu_g + 1$
14: **end while**
15: $R_{\nu_g} \Leftarrow \dfrac{\nu_f}{\nu_g}$
16: Confidence interval$=\pm z_{\alpha/2} \sqrt{\dfrac{R_{\nu_g}(1 - R_{\nu_g})}{\nu_g}}$

both networks have 1000 nodes and 2000 arcs (average degree $< k_i >= 2$) but the first N_r is a Random network whereas the other network, N_{sf}, is generated using Barabasi Scale free model [7] (see section 6.5).
We generate several source node - sink node pairs and we compute the $s-t$ reliability of the networks as the average of the value of reliability of the different pairs.
We compute the reliability for increasing value of p_i as shown in table 6.7.

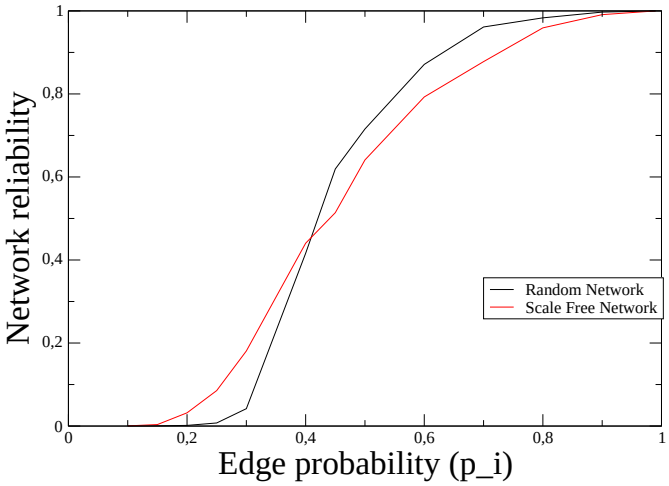

Figure 6.5: Graph networks with $< k_i >= 2$

In order to help to interpret the results we plot the graph of the reliability (Figure 6.5). The reliability is plotted as a function of p_i.

We can observe that Scale Free Network N_{sf} is more reliable than Random Network N_r for value of $p_i < 0.4$. For value around $p_i = 0.4$ one line crosses the other and for $p_i > 0.4$ we have an inversion: the Random

p	Rel Bara	Rel Rand
0.1	0.000417000	0.000116000
0.15	0.003587000	0.000624000
0.2	0.0318760000	0.0016220000
0.25	0.0854360000	0.0074880000
0.3	0.1807980000	0.0418990000
0.4	0.4403580000	0.4160740000
0.45	0.5139360000	0.6193020000
0.5	0.6409520000	0.7156280000
0.6	0.792743000	0.871165000
0.7	0.877918000	0.961339000
0.8	0.959214000	0.983309000
0.9	0.99097700	0.99714000
0.99	0.99993111111	0.99998000

Table 6.7: Relialility of networks with $< k_i >= 2$

Network appear more reliable than Scale Free Network.

Now we increase the degree of networks: in the previous example the average degree $< k_i >$ is 2, now we analyse networks with $< k_i >= 5$ namely with 1000 nodes and 5000 arcs. Results are plotted in Figure 6.6. We can observe that the switch value is moved around $p = 0.15$, but the general behaviour is the same of the previous networks.

A similar behaviour is described in [65] where Hiroyuki Ohsaki et al. have investigated the effect of scale-free structure of a network on its end-to-end performance. They have found that when the average degree of a network is small (i.e., there are not many links in a network), a scale free network shows higher end-to-end performance than a random network, and conversely, when the average degree of a network is large (i.e., many links in

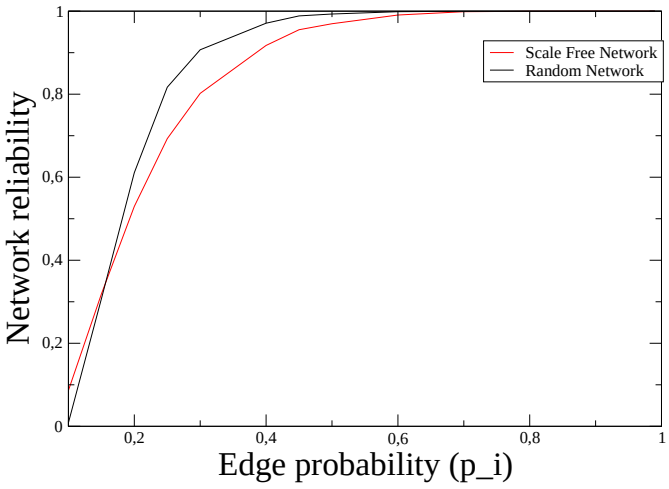

Figure 6.6: Graph networks with $< k_i >= 5$

a network), a random network shows higher end-to-end performance than a scale-free network.

These results validate our observation: in case of small value of p_i network owns a small number of links working so scale free networks appears more robust, on the contrary with hight values of p_i random networks are more reliable.

This can be explained also thinking to number and length of paths: it can be proved that scale free have less and shorter minimal paths in opposite to random networks that have more and longer minimal path.

In scale free networks minpaths are less and shortest because a lot of minpaths pass through the hubs, this is due to the small world phenomena: most nodes can be reached from every other by a small number of hops or steps (see results in section 6.8.2 Table 6.5).

Albert et al. in [8] and Crucitti et al. in [36] have studied the robustness of complex systems. The robustness of a complex system can be defined by its behavior under stress. In particular, they have analysed two general categories of such stress: errors-failures of randomly chosen components, and attacks-failures of components that play a vital role in the system. They compare random networks vs scale free networks: these graphs may differ greatly in their response to failures. For instance, scale free systems exhibit remarkable robustness to errors but, at the same time, they are very vulnerable to attacks such as the removal of the most highly connected nodes. These studies of network tolerance show that scale free networks are more robust to the occurrence of random failures with respect to random networks. This conclusion is based on a simulative experiment in which one arc at the time is randomly removed from the graph until the connectivity of the network breaks down.

We consider a network in which each time a certain fraction of the arcs has been removed. Such damage may break an initially connected graph into various disconnected parts. The simplest quantitative measure of damage is therefore given by the relative size of the largest connected component of the remaining network. The network will indeed keep its ability to perform its tasks as long as a "giant cluster" of size comparable to the original network exists.

When the fraction of deleted arcs exceeds a threshold f, the network has been broken into many small disconnected parts and is no longer functional. The study of the evolution of the size of the largest connected component as a function of the fraction of removed arcs therefore characterizes the network response to damage. Random graphs exhibit a definite level of damage over which size of the connected component abruptly drops to zero, signaling the total fragmentation of the network. Scale free networks,

on the contrary, appear to lack a definite threshold f and display higher tolerance to large random damage.
If we compare our results (see Figure 6.6) with the studies in [8] and [36] we can note the same behaviour of the networks: when the value of p_i is below a given value, scale free networks appear more reliable than random networks, this because in scale free networks a threshold f doesn't exist but it exists in random networks.
So random networks are more robust if we remove a small number of arcs, while the scale free networks appear more fault tollerant in case of big deletion.

6.10 Wireless Sensor Networks

In this section we analyse Wireless Sensor Networks (WSN) by means of simulation techniques. WSNs are characterized by a high number of sensors that collect information.
We suppose to scatter 1000 sensors in a 100×100 area ($l = 100$), these sensors take position in random coordinates. Each sensor can transmit within a limited radius rad, so only the sensors inside rad are able to receive the message.
For sake of semplicity in Figure 6.7 only 10 sensors in a 10×10 area ($l = 10$) are shown. Each sensor is able to transmit with radius $rad = 2$.
We suppose than the link is created only if the distance between sensors is less than rad (in this case $rad = 10$) so we obtain a network with 1000 nodes and 2000 edges. Now we want to compute the probability that a single sensor is able to communicate with the sink node, a special node placed in the centre of the 100×100 area.
Each link between two sensors is established with a fixed probability p_i if

we vary the value of p_i we obtain the graph plotted in Figure 6.8.

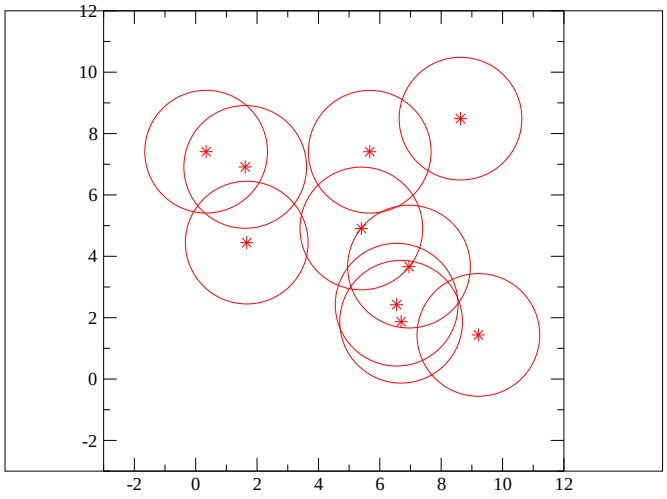

Figure 6.7: Example of sensor network

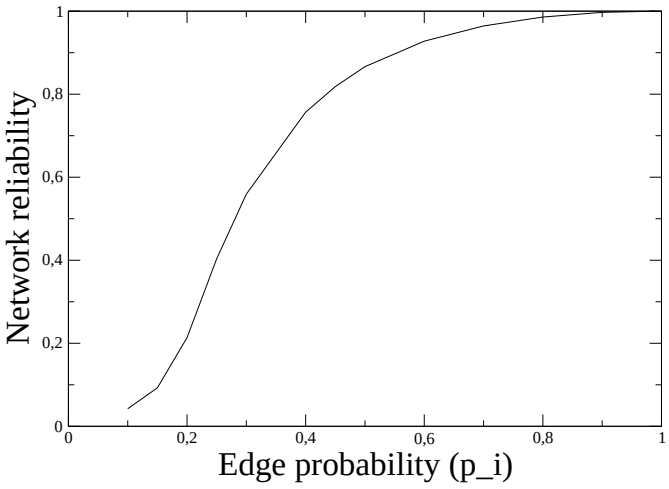

Figure 6.8: Graph with fixed probability value

6.11 Change Probability value

In the previous section we consider the probability of failure of the link as a fixed value, now we assume that the probability value is dependent on the distance d between two sensors: greater is the distance between them, smaller is the probability that the message can reach the destination because the signal is attenuated.
We suppose two different types of decay: the first based on the following expression:

$$p_1 = p_i * e^{-\frac{d^2}{2l^2}} \tag{6.9}$$

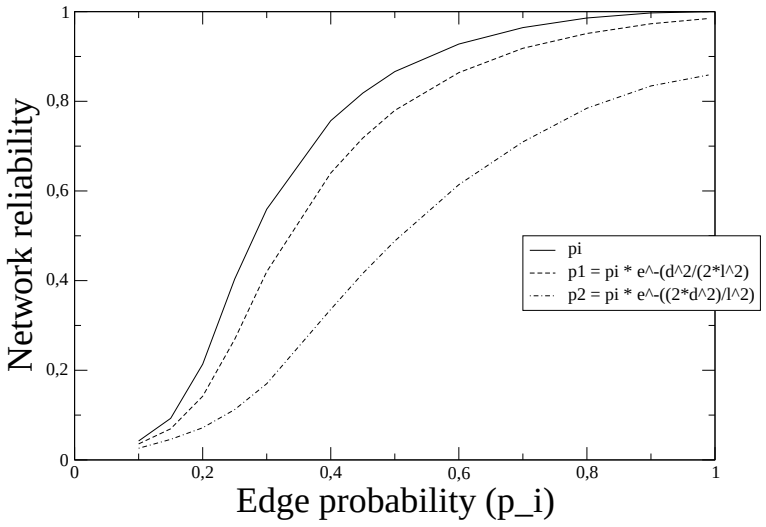

Figure 6.9: Graph of sensor network with different probability values

The second based on the following expression:

$$p_2 = p_i * e^{-\frac{2d^2}{l^2}} \tag{6.10}$$

We assign a link failure probability p_i and we compute the value of p_1 and p_2 following formula 6.9 and formula 6.10.
In Figure 6.9 the results are plotted: we compare the different curves and we note that the values computed using as link probability only the original probability value p_i appear higher than values involving the distances. Curves computed using equation 6.10 appear right shifted than values computed using equation 6.9. The distance adversely affects the reliability value.

Chapter 7

A Case Study

In this chapter we present a possible application of the theoretical results presented in previous chapters of this thesis ([26], [24],[27], [18]).
The automation of Power Grids by means of Supervisory Control and Data Acquisition (SCADA) systems has led to an improvement of Power Grid operations and functionalities but also to pervasive cyber interdependencies between Power Grids and Telecommunication Networks. Such cyber interdependencies mainly emerge from the presence of a Communication Infrastructure which connects the Supervisory Control Room of the SCADA to both its Remote Terminal Units (RTU) and to the "external cyber world". Many power grid services are increasingly depending upon the adequate functionality of SCADA system which in turn strictly depends on the adequate functionality of its Communication infrastructure.
In order to show possible applications we describe a case study on an actual SCADA system for an electrical power distribution grid.

7.1 Description of the scenario

SCADA systems constitute the nervous systems of Electrical infrastructures. They relies on SCADA communication infrastructures, which in turn are more and more depending upon Telco networks. For such a reason SCADA systems also represent one of the major means of mutual propagation of disturbances and adverse events between Electrical infrastructures and Telco networks. Many Power Grid services (i.e. availability of supply of critical users/large urban areas, grid reconfiguration after failures, telemetry) are increasingly depending upon the adequate functionality of SCADA system which in turn strictly depend on the adequate functionality of Telco network. On the other hand both SCADA system and Telco network need to be adequately fed by Power Grid. As evidenced by the occurrence of several catastrophic events, the link between SCADA systems and the reliability of the Electrical infrastructures is well established. Nowadays, SCADA communication infrastructure is typically composed by a proprietary network and a public telecommunication network. Such a solution grants no limits for transmission performances, but introduces a number of potential failure points that did not exist previously. Questions arise about security, making SCADA systems also vulnerable to cyber-warfare and cyber terrorism attacks. Public IP networks are, in fact, sensible to random failures, but also are highly sensible to security holes, because information is transferred on wired public trunks and even on wireless trunks that are more "open" channels than cables. As a consequence, networks may be subjected to various kinds of malfunctions, like logical misconfigurations, compromised redundancy, possible security breaches, loss of application data, and degraded services. Real-time and bandwidth requirements of SCADA communication infrastructure need to be considered. For exam-

ple, status and command data of field points are typically of a real time nature; regional control signals are typically of a near real-time nature, and historical status data is of a loose real-time to non-realtime characteristic. Care must be taken when utilizing the same communication channel for real-time communication and transfers of non-real-time data because large transfers of data could disrupt the transmission of the critical real-time data.

7.1.1 SCADA System

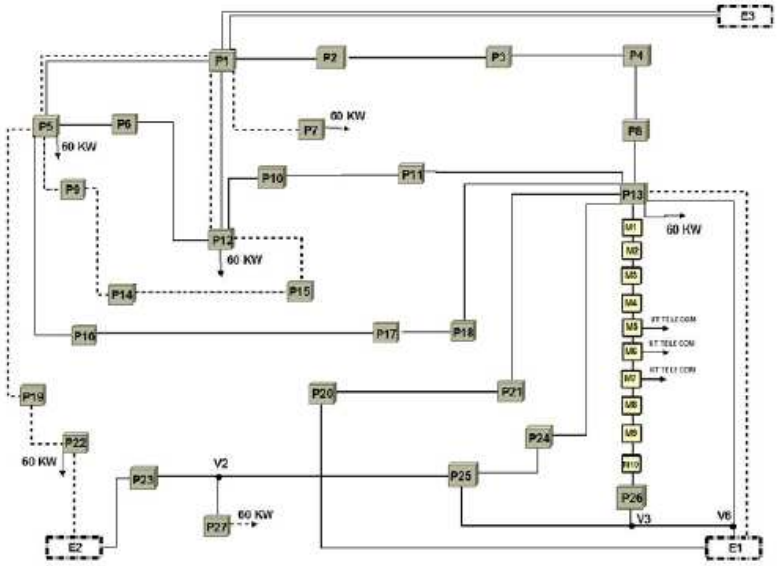

Figure 7.1: Schema of a portion of the power distribution grid

Figure 7.1 shows the Power Grid under consideration. It includes a portion of the HV (High Voltage) grid at 150 kV and the backbone of the MV (Medium Voltage) grid at 20 kV. Each node represents a primary substation (Pi, large rectangle), in case of HV network, or a secondary substation (Mi, small rectangle), in case of MV network. Nodes, named Ei, represent the substations of the national power transmission grid. They feed the power distribution grid. The physical link between any two nodes is an electrical trunk. Figure 7.2 reports the mapping of the SCADA system on the whole power distribution grid which include the Power Grid of Figure 7.1.
Differently from Figure 7.1, the links among substations do not represent the electrical trunks, but the SCADA communication link. A Main SCADA Control Centre (MSC) directly controls and supervises the portion of the power grid of Figure 7.1. A Disaster Recovery SCADA centre (DRS), directly controls and supervises a complementary portion of the power distribution grid. There are two types of Remote Terminal Units (RTUs), which interface the SCADA with power distribution grid: HV RTUs, located at HV substations, and MV RTUs, located at MV substations. MSC (directly and through the DRS) controls and supervises the power grid, while DRS assumes the control and the supervision of all the grid, in case of MSC failure. MSC and DRS are connected, via firewalls, by two redundant, public, high speed Telco links. Information and data concerning the status of the power grid (i.e. power flows, voltages, frequencies, loads, breakers positions, power transformers status) are transmitted from the RTUs to the control centres, while commands (i.e. the remote control of switchgears for energizing/de energizing power transformers and distribution feeders under normal/disturbance/fault conditions) are transmitted from control centres to RTUs by means of communication channels. SCADA system can be decomposed in three interacting subnets, highlighted in Figure 7.2 by dif-

ferent icons and lines. The three networks transport digital information measured in Mega-bit per second (Mbps) units. The Default Proprietary Network (DPN) serially connects the SCADA Control centres to HV RTUs. The DPN nodes (HV RTUs) are depicted as grey rectangles and are labeled on the graph Pi (i= 1, 44). DPN nodes can also communicate each other through the public PSTN (Public Switched Telephone Network) subnetwork since any DPN node is connected with a single link to one PSTN node. The PSTN network represents the back up public Telco network which connects SCADA control centres and HV RTUs to MSC and to DRS. The PSTN nodes are numbered on the graph from 55 to 61 and from 62 to 66. They are connected together (dotted lines) and cannot communicate each other through the DPN nodes. Two Virtual Private Networks (VPN) are established between the two SCADA Control Centres, via two HDSL (High data rate Digital Subscriber Line) connections, throughout two routers located in two PoP (Point of Presence), named PoP1(node 65) and PoP2 (node 66). MV RTUs, differently from HV RTUs, are connected to SCADA centres just by means of public Global System Mobile (GSM) connections (point/dashed lines). Particularly, MV_RTU are connected to their SCADA Control Centre throughout a Base Trans-receiver sub-System (BTS-node 62) and a Transit Exchange (Tex). MV_RTU nodes, labeled from M1 to M10, are terminal nodes that can be reached through the PSTN subnetwork only.

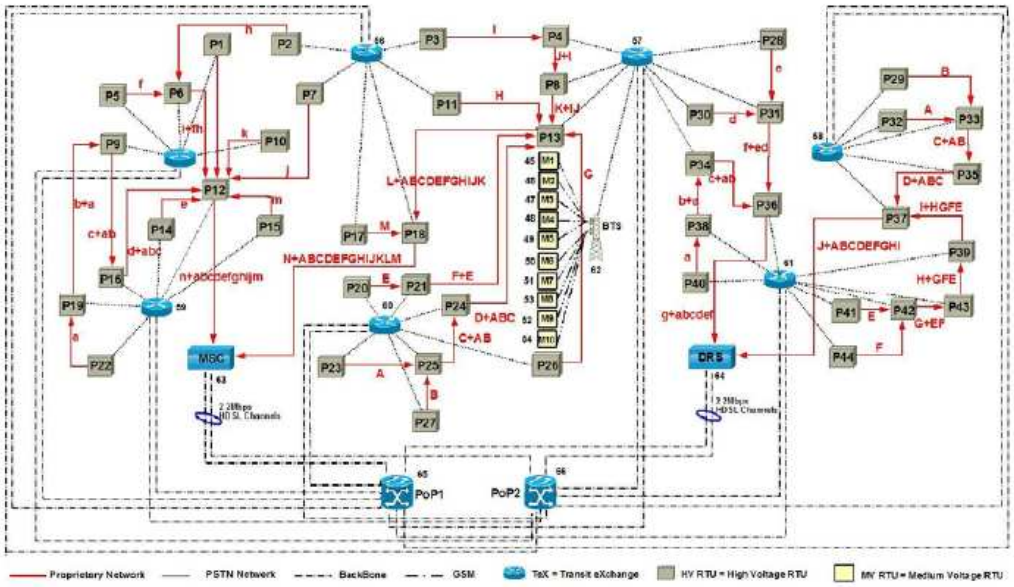

Figure 7.2: Schema of SCADA system

7.2 The Case Study

In this section we analyse more in detail the power distribution grid case study whose layout is displayed in Figure 7.3. The network of Figure 7.3 is composed by $N = 66$ nodes and $N_E = 118$ undirected arcs.

In a first instance, the network can be as a single flat graph where all the nodes and arcs have the same meaning and the same property. The connection between any pair of nodes is indifferently established by any path linking the two nodes.

However, in some cases, a network may be composed by structurally and

functionally different subnetworks: e.g. transport and control networks; primary and back-up network; electrical grid and information network. In these cases, the network can be decomposed in interacting parts that can be analysed separately.

In the case study of Figure 7.3, we have considered three interacting subnetworks, highlighted in the figure by different colors (or levels of grey in a w/b printout): a red part (in dark grey), a blue part (in medium gray) and a green part (in light grey). The three networks transport digital information measured in Mega-bit per second (Mbs) units. The connection between nodes and the flow of information among nodes of different colors is governed by the following rules.

- *Red subnetwork (in dark grey).* The red nodes (numbered on the graph from 1 to 44) are serially connected together in four distinct subgroups through the red arcs. But the red nodes can also communicate each other through the blue subnetwork since any red node in connected with a single link to one blue node. For the red subnetwork special blue nodes are nodes 63 and 64 where the four red groups converge.

- *Blue subnetwork (in medium grey).* The blue nodes are numbered on the graph from 55 to 61 and from 62 to 66. They are connected through blue arcs and cannot communicate each other through the red links.

- *Green subnetwork (in light grey).* The green nodes are numbered from 45 to 54 and 62; unless 62, they are terminal nodes that can be reached through the blue subnetwork only.

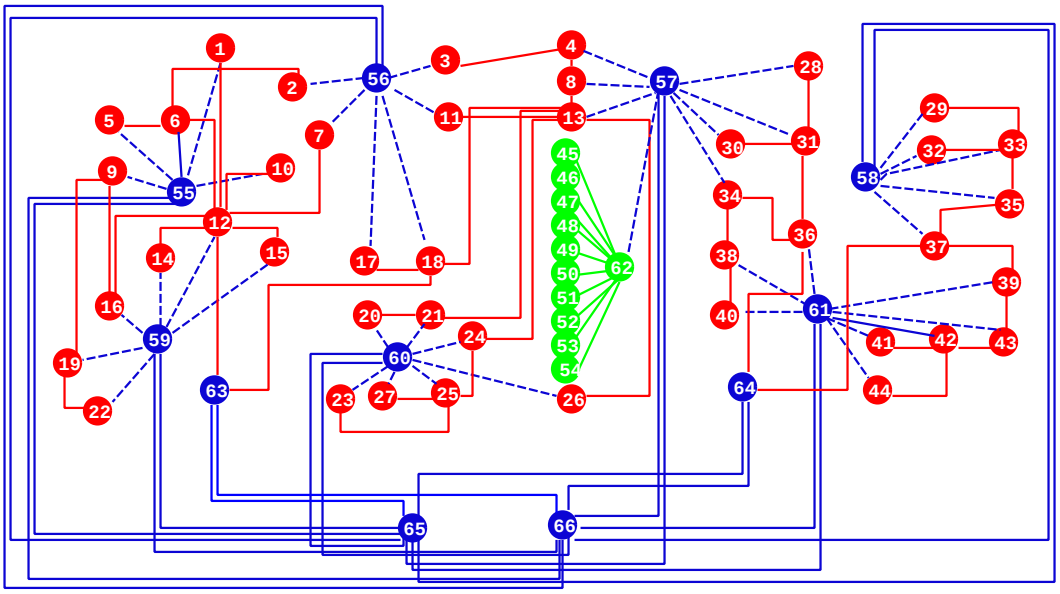

Figure 7.3: The layout of the benchmark network

7.3 Structure and Characterization of Complex Networks

7.3.1 Distribution of the Connectivity Degree

The most used parameter in order to discriminate the structure of a network is the Connectivity Degree (or simply the *Degree*): frequently is computed the degree distribution as a function of the number of nodes. In an undirected network, the degree of a node N is equal to the number of arcs emerging form the node, or equivalently the number of nodes directly reachable form the node i with a single hop, some times called the first

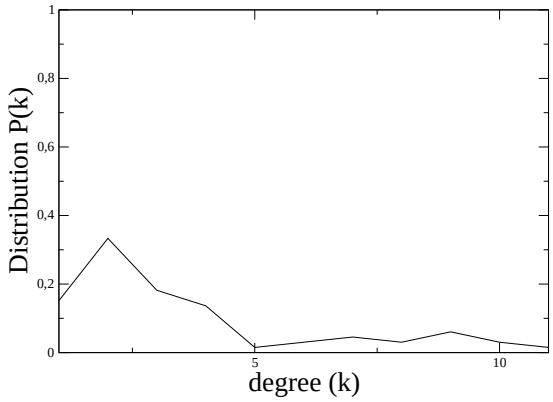

Figure 7.4: Connectivity distribution of the benchmark network: in linear scale

neighboring nodes. In an oriented network, at any node N we need to distinguish between the *input degree* (counting the number of arcs entering into the node), and the *output degree* (counting the number of arcs exiting from the node).

Considering the case study of the undirected network of Figure 7.3, we denote with k_i the degree of node i ($i = 1, 2, \ldots, N$); the degree distribution of the network is $P(k)$, where $P(k)$ is the percentage of nodes with degree k. The distribution $P(k)$ provides fundamental information about the overall structure of the network even if the degree is a local property of the node.

The degree distribution of the case study is given in in Figure 7.4 in linear scale and in Figure 7.5 in log-log scale. The average connectivity degree for the case study is $\overline{k} = 3.56$.

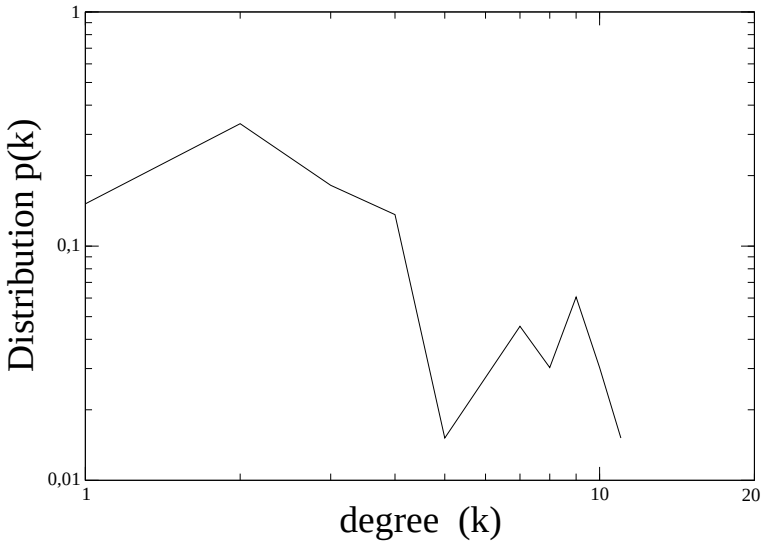

Figure 7.5: Connectivity distribution of the benchmark network: in log-log scale

Discussion and interpretation of these results are deferred to Section 7.3.5.

7.3.2 Clustering Coefficient

The clustering coefficient is one of the most important properties in classical networks. The clustering coefficient of a vertex in a graph quantifies how close the vertex and its neighbors are to being a clique (complete graph). For a node i, the clustering coefficient $C(i)$ is the fraction between the number of observed triangles to all possible triangles in one network.
The clustering coefficient for a vertex is then given by the proportion of links between the vertices within its neighbourhood divided by the number of

links that could possibly exist between them. It can be used to characterize the small-world properties [81], understand the synchronization in scale-free networks [68] and analyze networks of social relationships [64]

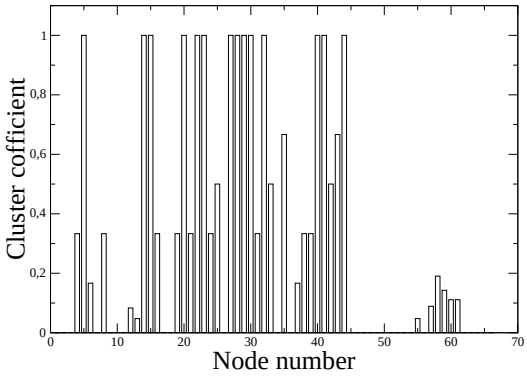

Figure 7.6: Clustering coefficient: the Clustering coefficient C_i as a function of the node number i

In the case study of Figure 7.3, the value of the clustering coefficient C_i as a function of the node number is reported in Figure 7.6. Note that a value $C_i = 0$ means that none of the first neighbors of node i are connected together, while a value $C_i = 1$ means that all the first neighbors of node i are connected together. An example of the first case ($C_i = 0$) is node $i = 10$, an example of the second case ($C_i = 1$) is node $i = 15$.

Figure 7.7) reports, in the form of an histogram, the distribution of the values of the clustering coefficient C_i. The average clustering coefficient is $C = 0.3180$.

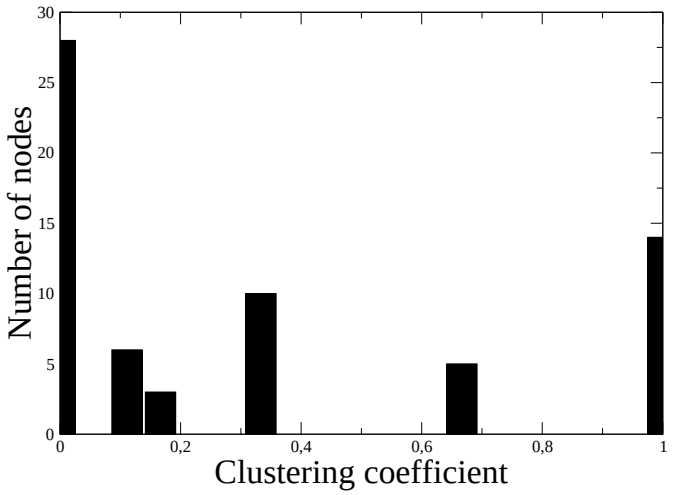

Figure 7.7: Clustering coefficient: Histogram of the distribution of the Clustering coefficient C_i

7.3.3 Length of the Shortest Path

The shortest path (or the minimal distance) between the two nodes is defined as the minimal number of arcs that must be traversed to reach a node from the other.
Usually it is studied the distribution of the shortest-path lengths between pairs of vertices of a network and the average shortest-path length. Average Shortest Path is the average of the shortest distance between every pairs of nodes between which a path exists.
Diameter is defined as the longest shortest path between any pair of nodes of a distributed network.

In our case study, the distribution of the length of the shortest path is

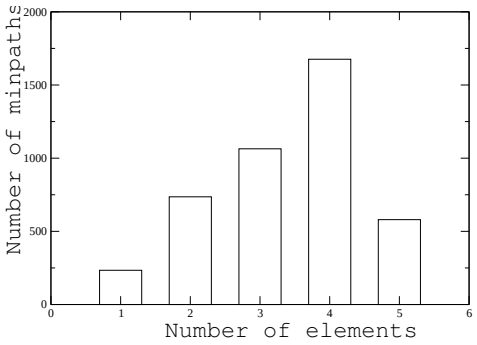

Figure 7.8: Histogram of the length of the shortest path

reported in the histogram of Figure 7.8. Note that in our case the number of values on which the histogram is built is equal to the pairs of nodes that are $N * (N - 1) = 4290$. The average distance among reachable pairs is $L = 3.380$. The most distant vertices separated by a maximum distance of $L_{max} = 5$ (as, for instance, the pair $1 - 54$). This result means that on the network of Figure 7.3 every couple of nodes can be reached on the average in L hops but in any case the maximal separation among nodes is $L_{max} = 5$.

7.3.4 Betweenness Centrality

Betweenness centrality is a parameter that measuers the importance of elements in a network based on their position in the shortes paths connecting a pair of non-adjacent nodes.

In our case study the histogram of the betweenness centrality is reported in Figure 7.9.

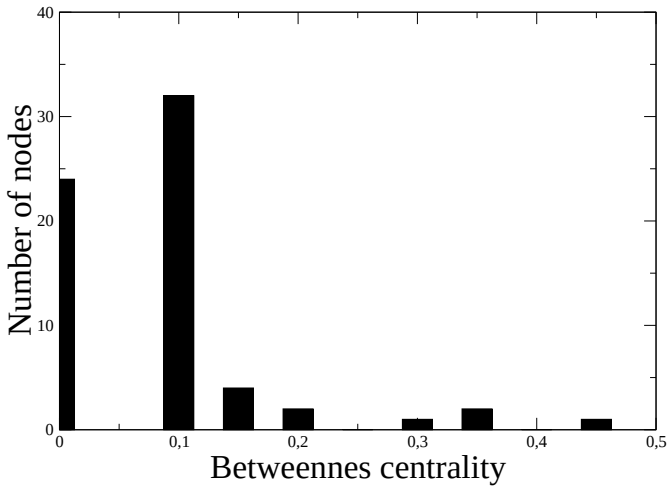

Figure 7.9: Histogram of the betweennes centrality

The average betweenness centrality is 0.37927. There are several nodes with betweenness centrality equal to 0 (there non shortest pthas passing through that node), and the node with the highest betweenness centrality is node 57.

7.3.5 Typical Network Structure

The most typical network structures are the *random networks (RN)* and the *Scale Free Networks (SF)*. They have been extensively considered in many survey papers [7, 15, 37, 62, 25].

In this paragraph we recall some of the main features and properties related to the network structure that are relevant in the analysis of the case study of Figure 7.3.

The main characteristics that has been considered to discriminate be-

tween RN and SF is the degree distribution. In RN, as the number of nodes N tends to infinity the degree distribution assumes the form of a Poisson distribution, which is bell shaped around a mean values with two exponentially decaying tails. Furthermore, in a RN the clustering coefficient is small (the nodes are connected randomly and there are not clustering effects) and tends to a value $\overline{C} = \overline{k}/N$ where $\overline{k}$ is the average degree. Applying the above relation to the case study, we get $\overline{C} = 3.56/66 = 0.0539$. The average clustering coefficient calculated in Section 7.3.2 has a value much higher than the theoretical one for RG, thus evidencing the non conformity of the case study to RG.

In SF networks the tail of the degree distribution decays as as the polynomial $P(k) \sim k^{-\gamma}$, so that in a log-log scale the tail appears as a straight line with angular coefficient $-\gamma$. The network of Figure 7.3 is too small dimension to approach a limiting behaviour and to show a definite behavioral structure. Nevertheless, the parameters of the network are more oriented toward a scale free structure. Furthermore the SF networks have higher clustering properties than RG.

7.4 Probabilistic Networks

The dependability of systems in the form of network is a very important issue in network analysis, since one of the main feature of networks is to have usually many redundant paths between any pair of connected nodes. In order to study the dependability of network, the probability that the elements of the networks are correctly functioning (or non functioning) should be known as an input parameter. We assume that the arcs can be modelled as binary entities and failure means a complete disruption of the link and of the interruption of the carried service.

In a probabilistic network given two nodes, a source node s and a termination node t, we define the $s-t$ reliability as the probability that the two nodes are connected by at least one functioning path. For small networks an exhaustive analysis is possible that searches for all the possible paths between s and t and computes exactly and analytically the probability that at least one path is up. The limits of the exhaustive approach have been checked in [23] and compared with similar results in the literature [47]. For general networks exhaustive reliability analysis is limited to graphs with few tenths of node.

If the dimensions of the network exceed the capability of exact algorithms the possible solution is to resort to simulation. Simulation is still the principal and most diffused investigation tool for complex networks. Simulation studies for probabilistic networks have been already appeared in the literature [8, 36], but their assumptions are not very convincing for the estimation of the point to point reliability. In this section we present first a simulation study, and then possible ways of attaching the problem via the exact tool WNRA taking into account the functional properties of the case study network and the partition in subnetworks.

7.4.1 Simulative Analysis of Complex Network Reliability

In Section 6.9, we discussed a simulative approach to the computation of the point to point reliability in a complex network. We have analyzed the graph of Figure 7.3 using the simulation algorithm. In this experiment we have considered the graph as a *flat* graph, i.e. a graph in which all the nodes and arcs have the same properties. We have thus neglected the colors and we have assumed that any node can communicate with any other node through paths joining arcs of any color.

In the described experiments we have assumed arbitrarily $p_i = 0.9$ for all i (all the arcs have the same probability of being up). We have assume as a source node $s = 63$ and termination node $t = 64$ and as termination criterium a total number of executions equal to $\nu_g = 1 \cdot 10^7$.

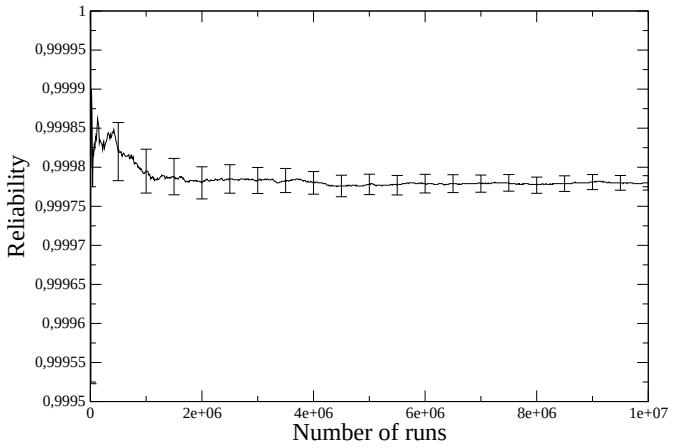

Figure 7.10: Network reliability between nodes 63 and 64 computed using a simulative approach

The results are reported in Figure 7.10. The horizontal axis reports the number of execution while the vertical axis reports the reliability value and the simulation error. As the figure shows, the value of the reliability stabilizes rather quickly remaining well inside the estimated error. The final value after $\nu_g = 1 \cdot 10^7$ executions is:

$$R = 0.9997803 \pm 1.83719 \cdot 10^{-05}$$

The described simulation procedure has been implemented in a com-

puter program written in C, and runs on a standard laptop. The simulation can be replicated for any pair of nodes and for any distribution function of the probability over the arcs.

7.4.2 Exhaustive Analysis of Weighted Networks

To characterize the benchmark example of Figure 7.3 as a probabilistic weighted network, we have quantified the probability function P and the weight function W in the following way:

- In the lack of sound experimental data about the reliability of any single link, we have assumed $p_i = 0.9$ for any i (all the arcs have the same probability of being up). However, we remind the the analysis tool WNRA is able to accept any value for the probability of each node.
- The weights are interpreted as capacities, and the following data have been assigned:
  - links 1-44 : 0.5 Mbps (solid red links)
  - links form 1-44 to 55-61 and from 62 to 57: 0.5 Mbps (dashed blue links)
  - links from 45-54 to 62 : 0.35 Mbps (green links)
  - links from 55-61 to 65 or 66 : 1.0 Mbps (solid blue links)
  - links from 63-64 to 65-66: 2.0 Mbps (solid blue links)

The exhaustive analysis through the tool WNRA cannot be performed on the whole *flat* network as explained in the previous Section 6.9. In this case we can exploit the fact that the whole network is composed by different

Total Capacity	*Probability*
4	0.6561
2	0.323996
1	4.32135e-06
0	0.0199001

Table 7.1: Capacities that can be transmitted from s to t and their probabilities

(colored) interacting sub-networks and that the node of some colors ar inhibited to communicate with other nodes via links of different colors.

In the following we present the results of the analysis of parts or interacting parts of the complete weighted probabilistic network of Figure 7.3.

The Blue Net

Since the blue nodes communicates to other blue nodes only via blue arcs, we can isolate the blue subnet and analyze it in isolation. Extracting from the complete network of Figure 7.3 only the blue part, we obtain the subnetwork of Figure7.11.

We have assumed as source node $s = 63$ and as terminal node $t = 64$. The following results have been obtained by running the tool WNRA. Table 7.1 reports the capacities that can be transmitted from s to t and the probabilities of these values.

The data reported in Table 7.1 have the following meaning. The first column reports all the possible capacity values that can be transmitted from

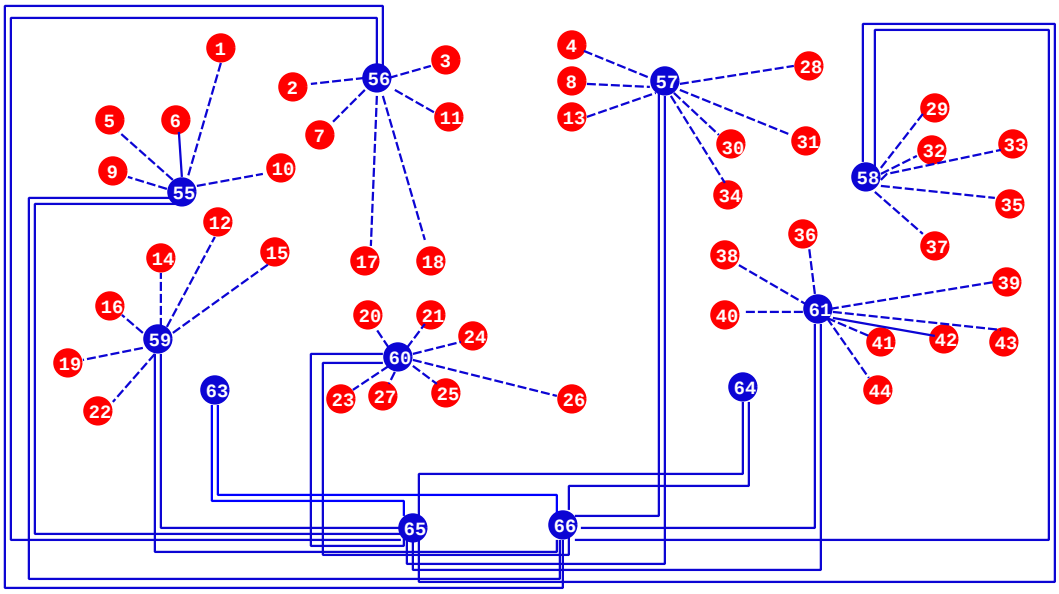

Figure 7.11: The layout of the blue sub-network

s to t, given the maximum capacity associated to each link, as reported in Section 7.2. The value 0 in the last row means that s and t are disconnected. The second column reports the probability that the network carries the corresponding capacity. The values in the second column sum to 1.

For what concerns the structural analysis, the $s-t$ connection of Figure 7.11 has 16 minpaths: 2 are of length 2, and 14 are of length 4. The $s-t$ connection has 258 mincuts: 2 are of order 2, and 256 are of order 9.

Finally the reliability is : $R = 0.9800998$. The BDD generated from the tool WNRA to compute the above figures has a size of 519 nodes.

Interaction of the Blue and Green Nets

The green part of the net is only connected to the blue part: hence we can join the subnets together and examine how the green nodes are connected to the blue ones. The layout of the union of the green and blue sub-networks is reported in Figure 7.12.

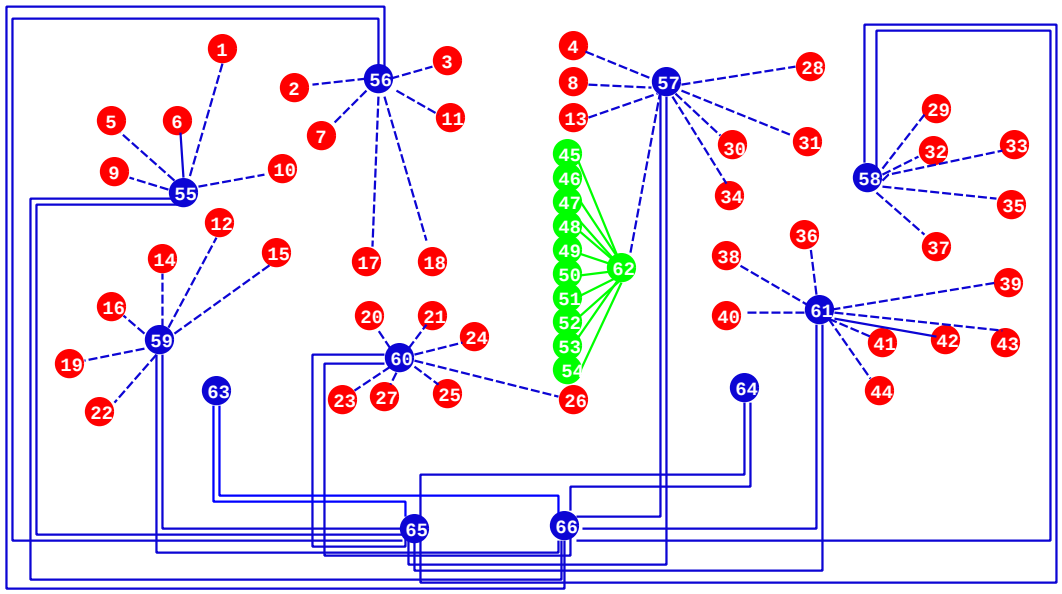

Figure 7.12: The layout of the union of the green and blue sub-networks

In the analysis we have assumed as source node $s = 54$ and as sink node $t = 63$. The following results have been obtained. The capacity analysis is reported in Table 7.2, but it is not very informative, since the connection $s - t$ can carry only a capacity 0.35, due to the bottleneck link from any

Total Capacity	*Probability*
0.35	0.793881
0	0.206119

Table 7.2: Capacities that can be transmitted from s to t and their probabilities in the network of Figure 7.12

green node to node 57. Hence, the first row of Table 7.2 coincides with the $s-t$ reliability (s and t are connected), and the last row of Table 7.2 coincides with the unreliabilty (s and t are not connected).

The number of minpaths is 16 as in the previous case, while the number of mincuts is 260. The reliability is $R = 0.7938808$ and the size of the BDD is of 555 nodes. Of course, the reliability value reported above and computed independently by the tool WNRA is equal to the value reported in the first row of Table 7.2.

The Red Net

If we isolate the red part of the net we obtain the graph of Figure 7.13. The red net is constituted by four non-interacting sets of nodes that are serially connected and terminates in the blue nodes 63 and 64 (that are maintained on the graph of Figure 7.13).

To exemplify possible analysis and the obtained results, we have assumed as source node $s = 22$ and as terminal node $t = 63$. The tool WNRA has provided the following results.

The capacity analysis (Table 7.3) is again not very informative, since the connection $s-t$ can carry only a capacity 0.5, due to the serial connection

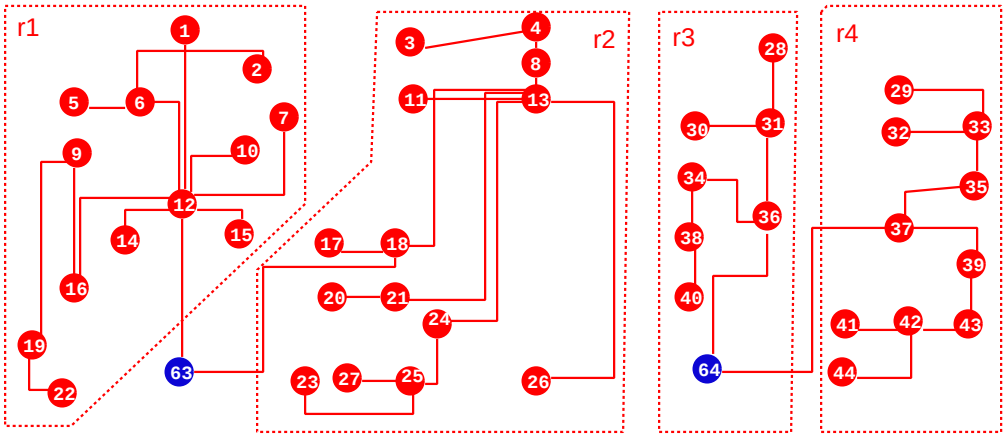

Figure 7.13: The layout of the red sub-network

of the red arcs. The serial connection involves also that there is a single minpath of length 5 and 5 mincuts of order 1. The reliability is $R = 0.5904899$ and coincides again with the first row of Table 7.3.

Interaction of the Red Subgraph *r3* and the Blue Net

In the following two subsections we examine the interactions between the red and the blue subnetworks of Figure 7.3. The red nodes are connected together in groups through the red arcs as displayed in Figure 7.13, however they can also communicate through the blue arcs. Hence, if we consider the red and the blue connections together we can evaluate the structural and quantitative parameters of the joined networks. In Figure 7.14 we have joined the $r3$ subnet of Figure 7.13 with the blue subnet of Figure 7.11.

Total Capacity	*Probability*
0.5	0.59049
0	0.40951

Table 7.3: Capacities that can be transmitted from $s = 22$ to $t = 63$ and their probabilities in the red network of Figure 7.13

The resulting network has 18 nodes and 64 arcs.

In particular, we have assumed as source node $s = 40$ and as a sink node $t = 63$, and we have submitted the resulting network to the tool WNRA. The first result refers to the capacities that can be transmitted between s and t and the corresponding probabilities. These values are reported in Table 7.4. While the red network can carry a capacity of 0.5 only (see also Table 7.3), the communication through the blue network can carry a capacity of 1.

Total Capacity	*Probability*
1	0.788982
0.5	0.189326
0	0.0216924

Table 7.4: Capacities that can be transmitted from $s = 40$ to $t = 63$ and their probabilities in the r3-blue network of Figure 7.14

The total number of the minpaths is 494 and the distribution of their length is reported in the histogram of Figure 7.15). There are 2 shortest minpaths of length 3 characterized by the following sequence of nodes:

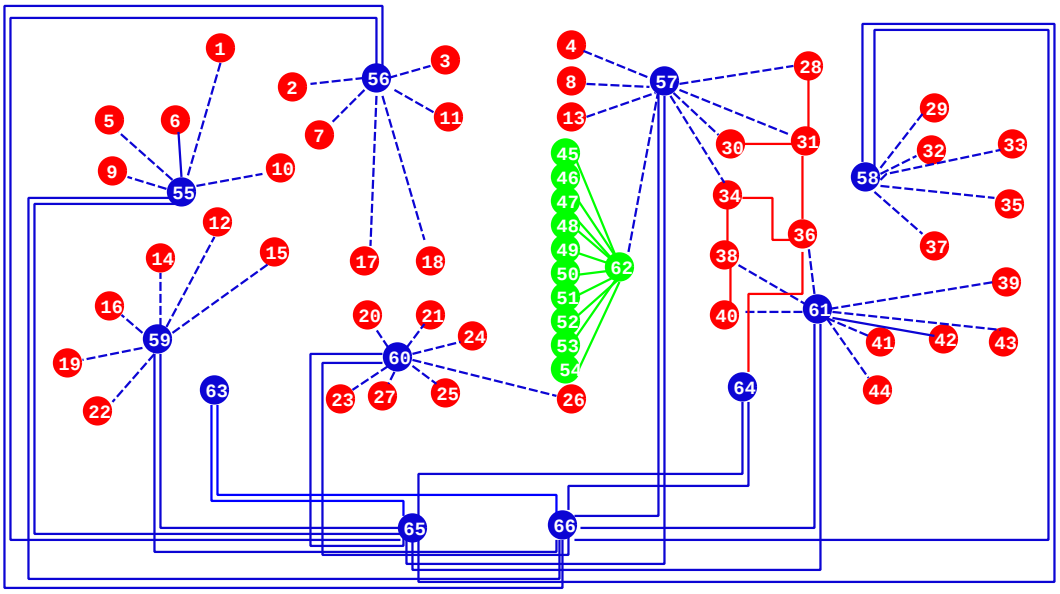

Figure 7.14: Interaction of the red3 and the blue sub-networks

$$40 \to 51 \to 65 \to 63$$
$$40 \to 51 \to 66 \to 63$$

The longest minpaths have length 12 and in the number of 20.

The total number of the mincuts is 4541 and the distribution of their order is reported in the histogram of Figure 7.16). There are 2 shortest

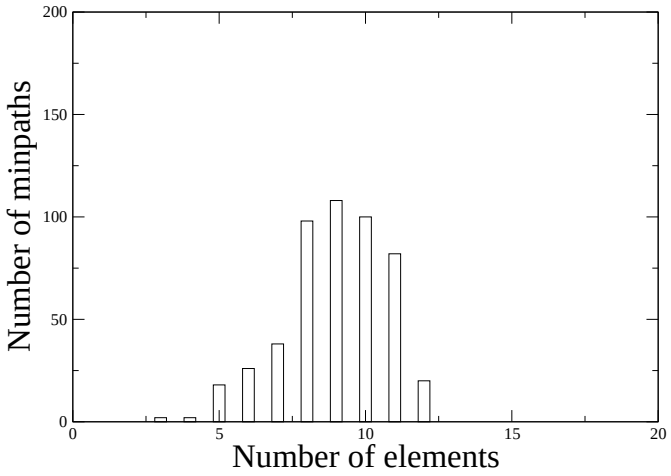

Figure 7.15: Results for the network of Figure 7.14: Histogram of the length of the minpaths;

mincuts of order 2 ($40 \rightarrow 38 \wedge 40 \rightarrow 61$) and ($61 \rightarrow 65 \wedge 61 \rightarrow 66$), and 1024 mincuts with maximum order 16. The size of the BDD on which the above figures (distribution of minpaths and mincuts) have been obtained is of 735 nodes, and the reliability is $R = 0.9789903$.

Interaction of the Red Subgraph *r1* and the Blue Net

In this last example we consider the interaction between the red subnet $r1$ and the blue net as depicted in Figure 7.17. The resulting network has 24 nodes and 88 edges. We assume the same source node $s = 22$ and the same sink node $t = 63$ considered in Section 7.4.2, where the red network only was examined.

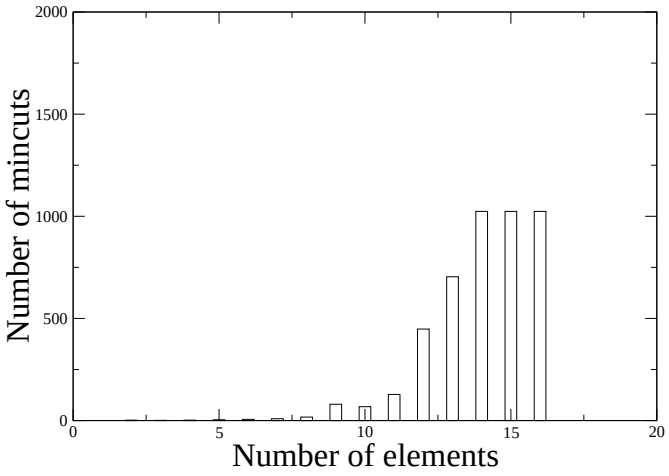

Figure 7.16: Results for the network of Figure 7.14: Histogram of the order of the mincuts.

As usual, the first result refers to the capacities that can be transmitted between s and t and the corresponding probabilities. These values are reported in Table 7.5.

The total number of the minpaths is 3305 and the distribution of their length is reported in the histogram of Figure 7.18). There are 3 shortest minpaths of length 3 represented by the following sequences of nodes:

$$22 \to 59 \to 65 \to 63$$
$$22 \to 59 \to 66 \to 63$$
$$22 \to 59 \to 12 \to 63$$

There are 60 minpaths of maximal length 16.

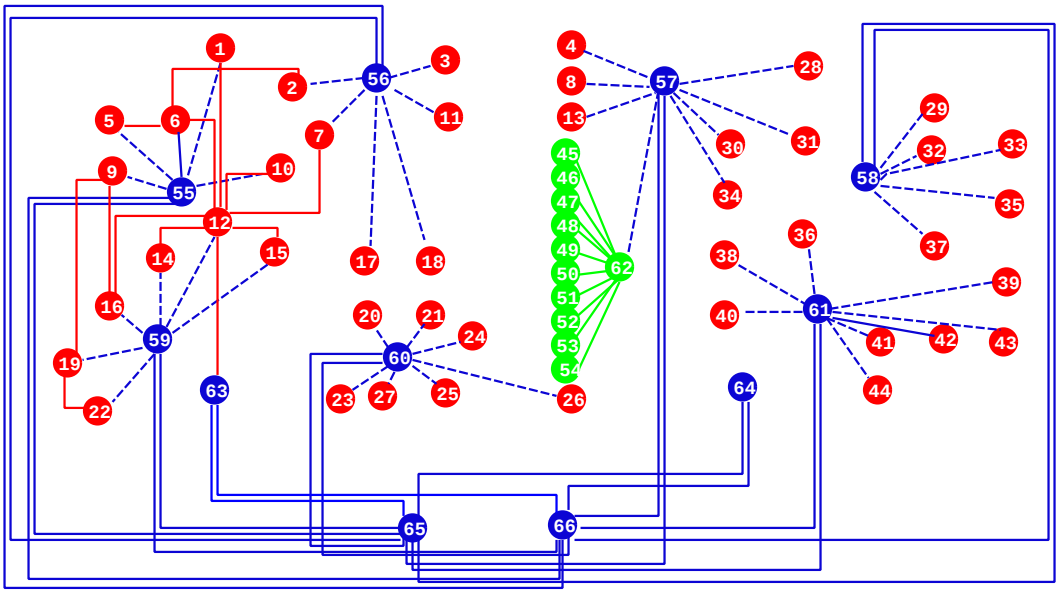

Figure 7.17: Interaction of the red1 and the blue sub-networks

The total number of the mincuts is 57174 and the distribution of their order is reported in the histogram of Figure 7.19). The shortest mincuts are: one single mincut of order 2 ($22 \rightarrow 19 \wedge 22 \rightarrow 59$) and two mincuts of order 3 ($12 \rightarrow 63 \wedge 59 \rightarrow 65 \wedge 59 \rightarrow 66$) ; ($12 \rightarrow 63 \wedge 63 \rightarrow 65 \wedge 63 \rightarrow 66$).

The size of the BDD on which the above figures (distribution of min-paths and mincuts) have been obtained is of 54095 nodes, and the reliability is $R = 0.9879523$. This last result should be compared with the value $R = 0.5904899$ obtained in Section 7.4.2, by considering the red links only. The difference between the two values shows the contribution in reliability

Total Capacity	*Probability*
1	0.791882
0.5	0.19604
0	0.012077

Table 7.5: Capacities that can be transmitted from $s = 22$ to $t = 63$ and their probabilities in the r1-blue network of Figure 7.14

increase that can be get by linking the red nodes through the blue network also.

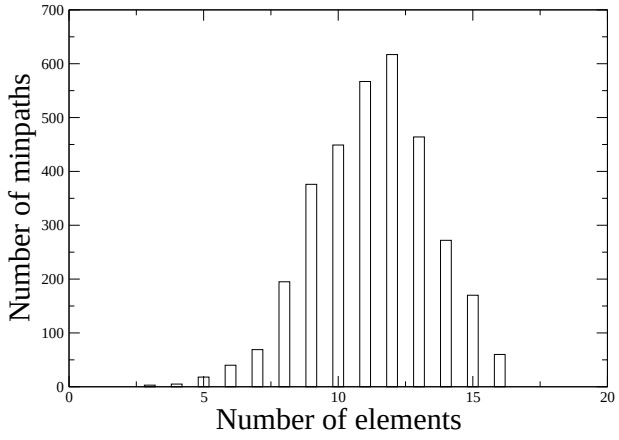

Figure 7.18: Results for the network of Figure 7.17: Histogram of the length of the minpaths;

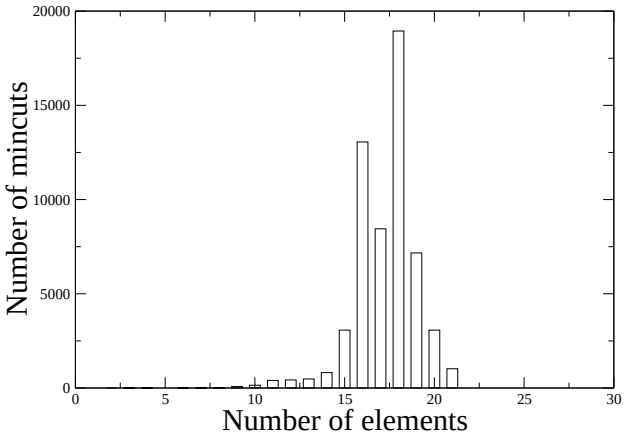

Figure 7.19: Results for the network of Figure 7.17: Histogram of the order of the mincuts.

Chapter 8

Probabilistic Weighted Multi-State Networks

Most works in reliability theory are based on the traditional binary concept of reliability models allowing only two possible states for a system and its components, perfect functionality or complete failure. However, many real-world systems are composed of multi-state components, with different performance levels and several failure modes with various effects on the system's entire performance. Such systems are called multi-state systems (MSS). For MSS the outage effect will be essentially different for units with different performance rates. Therefore, the reliability analysis of MSS is much more complex when compared with binary-state systems [55] [82].

The classical system reliability theory assumes that the system and its components are allowed to be in one of two possible states: either working or failed. The states of the system and its components each can be represented by a binary variable. In the previous chapters we have analysed binary networks: we have described different algorithms for system

reliability evaluation, but more specific analysis is needed.

The multi-state system reliability theory allows the system and its components to experience more than two possible states. The state of a component or the system may be represented by a non-negative integer taking values between 0 and M (discrete models) or by a continuous variable taking values in the interval [0, 1] (continuous models). In many real-life situations, multi-state system models provide more realistic descriptions of the actual systems. The term "reliability" is no longer suitable for describing the performance of the system or a component. Instead, the corresponding state distribution is used. Under the multi-state assumption, the relationship between component states and system state is much more complicated.

A distinct characteristic of a MSS is that the system and/or its components may exhibit multiple performance levels (or states) varying from perfect operation to complete failure. A MSS can model dependencies such as shared loads, performance degradation, imperfect fault coverage, standby redundancy, limited repair resources, and common-cause failures [50] [82]. The non-binary state property of a MSS and its components makes the analysis of such a system difficult.
We study the probabilistic weighted multi state networks by means of Multi-valued decision diagrams data structure. We propose a novel way in order to represent network structure considering also the possible component states and some propriety as cost or capacity. We compute the reliability of these networks considering the various peculiarity.

8.1 Multivalued Decision Diagrams

Multi-valued decision diagrams (MDDs) [53], [41] are directed, acyclic graphs used to represent K-variable integer functions of the form

$$f : \{0, \ldots, N_{K-1}\} \times \ldots \times \{0, \ldots, N_1 - 1\} \rightarrow \{0, \ldots, M - 1\}$$

.

Nodes in the MDD are either terminal or non-terminal. The terminal nodes correspond to the return values of the function and are labeled with an integer $0, ..., M-1$. Nonterminal nodes are labeled with a variable x_k , and contain N_k pointers to other nodes.

These pointers correspond to the cofactors of f , where a cofactor is defined as

$$f_{x_k=c} \equiv f(x_K, \ldots, x_{k+1}, c, x_{k-1}, \ldots, x_1)$$

for variable x_k and constant c. A non-terminal node representing function f is then written as the (N_{k+1})-tuple $(x_k, f_{x_k=0}, ..., f_{x_k=N_k-1})$. Since a terminal node labeled with m represents the integer constant m, every MDD node represents some integer function.

As for BDD the paths in an ordered MDD (OMDD) visit non-terminal nodes according to some total ordering on the variables $x_K, ..., x_1$. That is, if a path through the OMDD visits non-terminal nodes labeled with x_i and x_j, then x_i is visited before x_j if and only if $i > j$.

We say a non-terminal OMDD node labeled with x_k is a level-k node, and a terminal node is a level-0 node. Note that the ordering property implies that all downward pointers from level-k nodes are to level-j nodes, where $j < k$.

A reduced OMDD (ROMDD) has the following additional properties.

- There are no duplicate terminal nodes. That is, at most one terminal node is labeled with integer m.
- There are no duplicate non-terminal nodes. That is, given two non-terminal nodes $(x_i, f_{x_i=0}, \ldots, f_{x_i=N_i-1})$ and $(x_j, g_{x_j=0}, \ldots, g_{x_j=N_j-1})$, we must have either $x_i \neq x_j$ or $f_{x_i=n} \neq g_{x_i=n}$ for some $n \in 0, \ldots, N_i - 1$.
- All non-terminal nodes depend on the value of their variable. That is, given a non- terminal node $(x_k, f_{x_k=0}, \ldots, f_{x_k=N_k-1})$, we must have $f_{x_k=i} \neq f_{x_k=j}$ for some $i, j \in 0, \ldots, N_k - 1$.

It has been shown that ROMDDs are a canonical structure: given any integer function and a variable ordering, there is exactly one ROMDD representation for that function.
BDDs are a special case of MDDs applied to K-variable logic functions, which have the form $f : 0, 1^K \rightarrow 0, 1$. The size of ROMDD depends heavily, as in the BDD case, on the input variable ordering used to build the ROMDD.

8.2 Reliability of Multi State Systems

Given a weighted network $N = (G, P, W)$ where $G = (N, E)$, a set of M state $S = \{S_1, S_2, \ldots, S_M\}$ of the components of the system and a pair of nodes $s - t$, it is possible to evaluate the Quality of Services between s and t.
As for weighted networks consider two cases:

- Weight is a cost: we compute the probability that the connection can be established below a given cost. The single component cost value increases following the decreasing of functionality. In this case the state representing the component perfectly working takes the lowest value.
- Weight is a flow: we compute the probability that the flow is greater than a minimun threshold. In this case the state representing the component perfectly working takes the highest value.

8.3 Weight as Flow

We can suppose that the flow between two nodes is not a fixed value, but it changes as a function of the degradation of the link. If the line is perfect the maximal value of flow carried is the whole flow value, in the case of decay the value decrases.
Network can work with different distinguished levels of efficiency, but we consider a finite number of states.
For example we can suppose to analyse the bridge network of the previous chapters. In Figure 8.1 we depict the network with the assigned flow values. We can suppose that, in a determinate instant of time, each element can be in one of the following three states:

- **perfectly working** : the flow is the 100% of nominal value
- **degradated**: the flow is the 50% of the nominal value
- **failed**: the flow is 0, the link doesn't work.

Each element i is in one state z with a probability $p_{i,z}$. In this case we suppose that all elements have the same probability if they are in the same

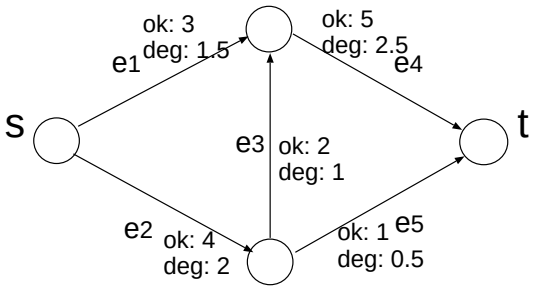

Figure 8.1: Bridge network

state. We have the probability values in Table 8.1.

Table 8.1: State probability of elements

State	**Probability**	**Link capacity**				
		e_1	e_2	e_3	e_4	e_5
perfectly working (state 2)	0.65	3	4	2	5	1
degradated (state 1)	0.25	1.5	2	1	2.5	0.5
failed (state 0)	0.1	0	0	0	0	0

We have developed an algorithm, using the Iowa State University MEDDLY library [78], that builds an MDD starting from the list of mincuts. This algorithm is similar to the algorithm developed for weighted binary networks, but in this case the nodes of the data structure are multivalue and not only binary.
Also in this case the operation *Or_sum* (see Section 5.3.4) is used in order to combine the elements of each single mincut. The different MDDs representing each mincut are combined with the *And_min* operation (see

Section 5.3.4).
In Figure 8.2 the MDD for the network in Figure 8.1 is depicted.

Starting from the MDD representing the network it is possible to compute the reliability value of each MDD terminal node, that is the probability to obtain a deteminate flow considering that each network element can be in one of the three different states with a specific probability. In Table 8.1 the different probability value are shown.
Now we apply the probability algorithm shown in Algorithm 8.1 and we obtain results in Table 8.2

Algorithm 8.1 Compute reliability by BFS

```
1: visited → set of visited add node
2: no_term_prob → map of internal node probability value
3: visited.insert(F)
4: no_term_prob[F] = 1.0
5: while !visited.empty() do
6:    node= visited.pop
7:    for each state i do
8:       F_i = F → i
9:       if  F_i is Terminal Node  then
10:         leaf_array[F_i] += prob_x_i * no_term_prob[node];
11:      else
12:         no_term_prob[F_i] += prob_x_i * no_term_prob[node];
13:         visited.insert(F_i)
14:      end if
15:   end for
16: end while
```

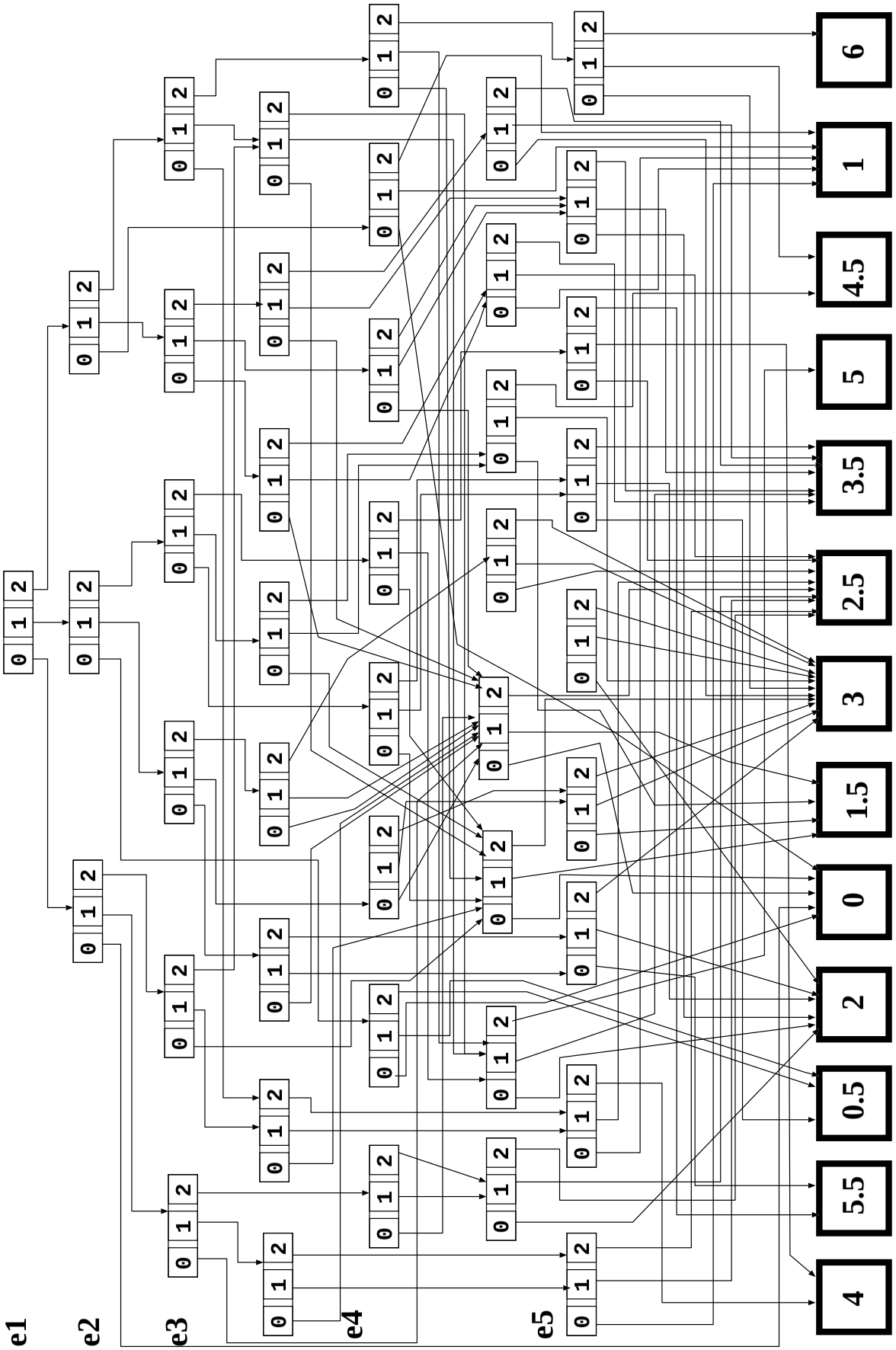

Figure 8.2: MDD of bridge network (flow)

Table 8.2: Probability of the MDD terminal values (max flow)

(s-t) Max Flow	Probability
0.5	0.024525
1	0.06579
1.5	0.0295875
2	0.0366525
2.5	0.0622625
3	0.129128
3.5	0.194513
4	0.0513866
4.5	0.0683922
5	0.148298
5.5	0.0446266
6	0.116029
0	0.02881

8.4 Weight as Cost

We consider the same bridge network as Section 8.3, but in this case we consider weights as costs. The network is depicted in Figure 8.3.
Also in this case the weight value of the link is due to the level of link degradation: if the edge works well then the cost of the link is the original value, but if the link is troubled the value cost is higher.
We can immage that, in a determinate instant of time, each element can be in one of the following three states:

- **perfectly working** : the cost is the pure cost
- **degradated**: the cost is double of the original cost

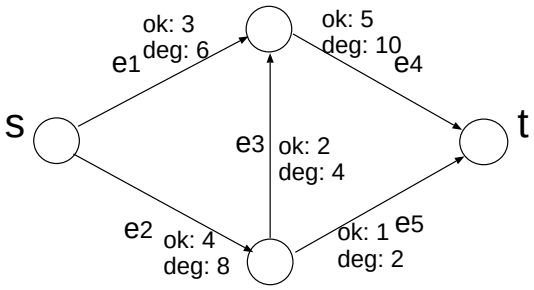

Figure 8.3: Bridge network

- **failed**: the link doesn't work.

In Table 8.3 the probability for the different states is presented.

Table 8.3: State probability of elements

State	**Probability**	**Link cost**				
		e_1	e_2	e_3	e_4	e_5
perfectly working (state 1)	0.65	3	4	2	5	1
degradated (state 2)	0.25	6	8	4	10	2
failed (state 0)	0.1	0	0	0	0	0

In Figure 8.4 the MDD representing network in Figure 8.3 is depicted. The MDD is obtained by means of an algorithm using the list of minpaths. We use the operation *And_sum* (see Section 5.3.4) in order to combine the elements of each single minpath. The different MDDs representing each minpath are combined with the *Or_min* operation (see Section 5.3.4).

The values of the probability for the different terminal nodes of the MDD of Figure 8.4 are reported in Table 8.4. The reliability values are obtained applying algorithm 8.1.

Table 8.4: Probability of the MDD terminal values (cost)

(s-t) Distance	**Probability**
5	0.4225
6	0.1625
8	0.175338
9	0.0938438
10	0.0360938
11	0.0336213
13	0.0319313
15	0.00105625
16	0.0129313
17	0.00040625
18	0.00040625
20	0.00040625
22	0.00015625
0	0.02881

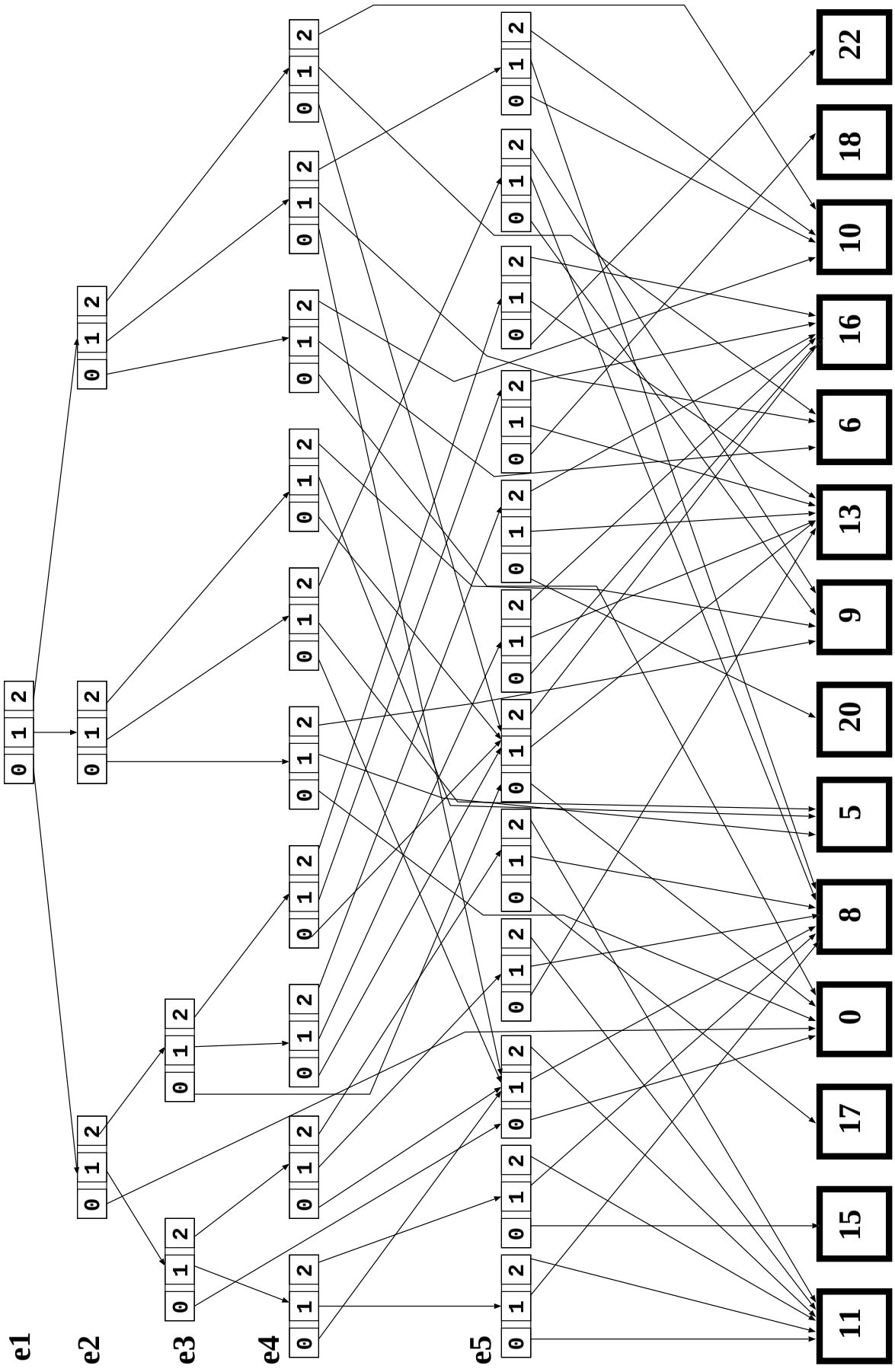

Figure 8.4: MDD of bridge network (cost)

Chapter 9

NRA and WNRA Tools

In this chapter we will describe in details two tools that we have implemented [2]: one for unweighted networks (section 9.1), the other for weighted networks (section 9.2).

9.1 Tool NRA: Network Reliability Analyzer

In this section we are going to present the framework developed in order to analyse the unweighed network. This framework combines together several theoretic concepts explained in previous chapters. The Network Reliability Analyser (NRA) is based on Binary Decision Diagrams (BDD) which provides an efficient method to represent and manipulate Boolean functions and have been also exploited to model the connectivity of Boolean networks.

NRA computes the qualitative and quantitative proprieties of a network according to two approaches:

- qualititative: the list of the minpaths and of the mincuts and its

length distribution is determined and

- quantitative: the reliability is computed by BDD. It is computed directly by network visit or passing through a preliminary search of the minpath/mincuts.

A software tool for network reliability analysis has been implemented according to the algorithms illustrated in the preceding chapters.
NRA accepts in input a unweighted network given in various standard representation formats. Beside the graph structure, the only necessary information associated to the edges is the failure probability.
The tool accepts in input a graph with a variety of possible formats:

- incidence matrix;

- adjacency list;

- formats provided by some of the most diffused tools for network generation and analysis;

- Networks generated by generating algorithm inside the tool.

The user may optionally choose which elements are failure prone; only the arcs, only the nodes or both, and consequently, assign the corresponding failure probabilities. Finally, the `s` and `t` node must be assigned so that the program can provide the two-terminal reliability and the list of minpaths and/or mincuts.

The tool embodies the two approaches:

- A preliminary search of the minpaths is executed. The algorithm relies on a recursive call of a function based on the classical Dijkstra's

algorithm (with all the edges having a unitary weight). Once the minimal paths between nodes `s` and `t` are explored (and ordered by rank), the BDD is constructed.

- In the second approach the BDD for the chosen connectivity function is directly constructed resorting to the algorithm of Figure 4.3.

The construction and manipulation of the BDD's is managed through the BDD library developed at the Carnegie Mellon University [1]. The list of the minpaths and of the mincuts may be optionally determined by applying the transformation algorithm proposed in [70].

9.1.1 Tool Input

Two versions of the tool are available:

- Command line version
  - all the input parameters must be passed by command line
- Interactive version
  - the input parameters can be specified in interactive way

In order to execute the programm in both the versions should be insert:

- graph file
- probability file (or N if no reliability value is required)
- search type (1 edge, 2 node, 3 node and edge)
  - 1 edge: we consider that only edges can fail.

  - 2 node: we consider that only nodes can fail.
  - 1 edge: we consider both edges and nodes can fail.

- file type (1 list, 2 brite, 3 pajek, 4 matrix)
  - 1 list: the graph is represented as adjacency list
  - 2 brite: the graph is created with package brite
  - 3 pajek: the graph is created with package pajek
  - 4 matix: the graph is represented as adjacency matrix

- algorithm type (1 direct bdd, 2 minpath bdd)
  - 1
    * The BDD is directly derived without passing from a preliminary search for the minpaths. Minpaths/mincuts are obtained by applying a trasformation of the BDD.
  - 2
    * The list of minpaths is obtained directly from the graph, and the BDD for the reliability computation is built from minpaths list.

- source node

- sink node

- information type (1 minpath, 2 mincut, 3 minpath and mincut)
  - This parameter is used only if algorithm type = 1

In interactive version it is possible to choose the variable order for build the BDD or it is possible to use the standard fixed order obtained by means of an heuristic algorithm, while in command line version the fixed order is used.

9.1.2 Tool Output

As output the tool returns the results screen diplayed and stored in a file. The output file is named *graphname_output.txt* and include graph information as name, structure, degree of nodes and node degree distribution.
If the algorithm type is 1 is present also bdd size and, if computed, the reliability of the network.
If required are printed the number of minpaths/mincuts, the ordered list and the distribution length of minpaths/mincuts.
If the algorithm type is 2 the file includes the ordered list of minpaths, matrix of element occurrence in the minpath, the distribution length and the average minpath length.

9.2 Tool WNRA: Weighted Network Reliability Analyzer

The extended version of the NRA tool is called WNRA and is aimed at analysing probabilistic weighted networks. In this case, an additional information is needed related to the weights to be assigned to the arcs.
Two types of analysis are possible:

- weights as costs
- weights as flows

In the first case starting from the minpath list and arc cost list it is possible to obtain the probability of working network with all possible combination of cost considering each components up with a determinated probability. In the second case we consider weights as flows it is possible compute the probability that the network provides each possible flow. In this case we start from the mincut list

9.2.1 Tool Input

- GRAPH_FILE_NAME is the name of the graph file
- PROB_FILE_NAME is the name of the probability file (or N if no reliability value is required)
- WEIGHT_FILE_NAME is the name of a file containing the weights (or N if no weight value is required)
- RIC_TYPE search type (1 edge, 2 node, 3 node and edge)
  - 1 edge: we consider that only edges can fail.
  - 2 node: we consider that only nodes can fail.
  - 3 edge: we consider both edges and nodes can fail.
- FILE_TYPE file type (1 list, 2 brite, 3 pajek, 4 matrix)
  - 1 list: the graph is represented as adjacency list
  - 2 brite: the graph is created with package brite
  - 3 pajek: the graph is created with package pajek
  - 4 matix:the graph is represented as adjacency matrix

- REL_TYPE algorithm type (1 direct bdd, 2 minpath bdd)
  - 1
    - The BDD is directly derived without passing from a preliminary search for the minpaths. Minpaths/mincuts are obtained by applying a trasformation of the BDD.
  - 2
    - The list of minpaths is obtained directly from the graph, and the BDD for the reliability computation is built from minpaths list.
- SOURCE_NODE source node
- SINK_NODE sink node
- PATH_CUT information type (1 minpath, 2 mincut, 3 minpath and mincut)
  - This parameter is used only if algorithm type = 1

The weighted network analysis is based on the previous knowledge of the complete set of mincuts / minpaths; hence, the capacity computation can start only if and after the mincuts / minpaths list is obtained.

To this end, the parameter

`PATH_CUT`

must be assigned a value 2 (for the computation of the mincuts only) / 1 (for the computation of the minpaths only) or 3 (for the computation of the mincuts and minpaths).

9.2.2 Tool Output

As output the tool returns the same output of NRA, but the reliability is computed for each possible flow (or cost).
For example the output:

```
Flow   Prob
6         0.59049
1         0.08829
5         0.06561
4         0.06561
3         0.1539
2         0.00729
0         0.02881
```

means that with probability of 0.59049 the network can carry a flow of 6.

Chapter 10

Conclusion

In this thesis some original contributions have been presented, concerning the analysis of probabilistic networks. In this section we summarize our main results.

Many social, economical and technological structures can be abstracted in the form of networks where the vertices are the entities of the system and the edges the physical or relational links among them. One relevant property of networks that make them a preferential structure both in natural and technological systems is that the connection between any two nodes of the networks can be achieved through many redundant paths, thus making the connection intrinsically reliable. Network reliability is studied in this thesis by resorting to different approaches making use of the Decison Diagrams representation of Boolean functions. The related algorithms are presented and their merits and limits are briefly discussed.
In particular starting from a binary unweighted network (the edges can be only up or down) we have extended the analysis to the weighted netwoks, by associating a weight to the edges.

We have considered this weight as a cost and as a capacity and we have implemented some new boolean operations in order to represent the network structure by means of ADD. Visiting the ADD it is possible to obtain the different reliabilty values.
Some different topologies of complex network have been analysed and compared. We have computed the reliability of networks with the same number of nodes and links, but we have varied the topology. Probabilistic weighted multi-sate networks can be used in order to represent systems where a weight is assigned to the components and each element can be in more than two states. We have supposed that the different states are due to a degradation of the weight (if we consider weight as cost the value is increased, while if we consider weight as capacity the value is decreased). New algorithms have been developed in order to compute the network reliability considering both weights and multi state.
A possible future work in this area will be to used the Probabilistic weighted Multi-Sate networks in order to represent network with delay.
Finally we are working to update the network reliability tools in order to create a single framework including all the analysis techniques described in this thesis.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] Bdd library. http://www.cs.cmu.edu/ modelcheck/bdd.html.

[2] Cresco project. http://www.cresco.enea.it/.

[3] *IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries*. IEEE Computer Society Press, New York, NY, USA, January 1991.

[4] J.A. Abraham. An improved algorithm for network reliability. *IEEE Transaction on Reliability*, 28:58–61, 1979.

[5] A. Agrawal, , and R. E. Barlow. A survey of network reliability and domination theory. *Operations Research*, 32:478–492, 1984.

[6] Avinash Agrawal and Richard E. Barlow. A survey of network reliability and domination theory. *Operations Research*, 32(3):478–492, 1984.

[7] R. Albert and A.L. Barabasi. Statistical mechanics of complex networks. *Review Modern Physics*, 74:47–97, 2002.

[8] R. Albert, H. Jeong, and A.L. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2002.

[9] Henrik Reif Andersen. An introduction to binary decision diagrams.

[10] B. Anrig and P. A. Monney. Using propositional logic to compute the probability of diagnoses in multistate systems. In *International Journal of Approximate Reasoning*, pages 113–143. 20(2), 1999.

[11] R.I. Bahar, E.A. Frohm, and C.M. Gaona. Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2-3):293–318, 1997.

[12] A.O. Balan and L. Traldi. Preprocessing minpaths for sum of disjoint products. *IEEE Transaction on Reliability*, 52(3):289–295, September 2003.

[13] M.O. Ball. Network reliability analysis: Algorithms and complexity. *Ph.D. thesis*, Departmentof OperationsResearch,CornellUniversity, 1977.

[14] M.O. Ball. Complexityof network reliability computations. In *Networks*, volume 10, pages 153–165, 1980.

[15] A.L. Barabasi. *Linked: the new science of networks*. Perseus Publishing, Cambridge Mass, 2002.

[16] R.E. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing*. Holt, Rinehart and Winston, New York, 1975.

[17] M. Barthelemy. Betweenness centrality in large complex networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):163–168, 2004.

[18] A. Bobbio, E. Ciancamerla, S. Di Blasi, A. Iacomini, F. Mari, I. Melatti, M. Minichino, A. Scarlatti, E. Tronci, R. Terruggia, and

E. Zendri. Risk analysis of scada systems interconnecting power grids and telco networks via heterogeneous models and tools. In *4th International Conference on Risks and Security of Internet and Systems*, 2009.

[19] A. Bobbio, C. Ferraris, and R. Terruggia. New challanges in network reliability analysis. In *Proceedings of International Workshop on Complex Network and Infrastructure Protection - CNIP06*, pages 554–564, 2006.

[20] A. Bobbio and A. Premoli. Fast algorithm for unavailability and sensitivity analysis of series-parallel systems. *IEEE Transactions on Reliability*, R-31:359–361, 1982.

[21] A. Bobbio and R. Terruggia. Binary decision diagram in network reliability analysis. In *1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS'07)*, pages 57–62, 2007.

[22] A. Bobbio and R. Terruggia. Reliability and quality of service in weighted probabilistic networks using algebraic decision diagrams. In *Proceedings IEEE-RAMS*, volume Reliability and Maintainability Symposium, 2009. RAMS 2009. Annual, pages 19–24, 2009.

[23] A. Bobbio, R. Terruggia, A. Boellis, E. Ciancamerla, and M. Minichino. A tool for network reliability analysis. In F. Saglietti and N. Oster, editors, *Int. Conference on Computer Safety, Reliability and Security, SAFECOMP2007*, pages 417–422. Springer Verlag - LNCS, Vol 4680, 2007.

[24] A. Bobbio, R. Terruggia, E. Ciancamerla, and M. Minichino. Evaluating network reliability versus topology by means of bdd algorithms. In *In: PSAM-9, Hong Kong*, May 2008.

[25] S. Boccaletti, V. Latora, M. Chavez, and D. Hwang. Complex networks: structure and dynamics. *Physics Reports*, 424:175–308, 2006.

[26] G. Bonanni, E. Ciancamerla, M. Minichino, R. Clemente, A.Iacomini, A. Scarlatti, E.Zendri, A. Bobbio, and R. Terruggia. Availability and qos analysis of interconnected networks. In *In 5th International Service Availability Symposium, ISAS 2008 Tokyo, Japan*, May 19-21 2008.

[27] G. Bonanni, E. Ciancamerla, M. Minichino, R. Clemente, A. Iacomini, A. Scarlatti, E. Zendri, and R. Terruggia. Exploiting stochastic indicators of interdependent infrastructures: the service availability of interconnected networks. In *Safety, Reliability and Risk Analysis: Theory, Methods and Applications*, volume 3. Taylor and Francis, 2009.

[28] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[29] Karl S. Brace, Richard L. Rudell, and Randal E. Bryant. Efficient implementation of a bdd package. In *DAC '90: Proceedings of the 27th ACM/IEEE conference on Design automation*, pages 40–45. ACM, 1990.

[30] R.E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[31] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98:142–170, 1992.

[32] M. K. Chari and C. J. Colbourn. Reliability polynomials: a survey. In *Journal of Combinatorics, Information and System Sciences*, pages 177–193. 22, 1998.

[33] S. K. Chaturvedi. Irredundant subset cut generation to compute capacity related reliability. *International Journal of Performability Engineering*, 3:243–256, 2007.

[34] C. J. Colbourn. Some open problems on reliability polynomials. Technical Report 93-28, University of Waterloo, Canada., 1993.

[35] Charles J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, Inc., New York, NY, USA, 1987.

[36] P. Crucitti, V. Latora, M. Marchiori, and A. Rapisarda. Efficiency of scale free networks: error and attack tolerance. *Physica A*, 320:622–642, 2003.

[37] S.N. Dorogovtsev and J.F.F. Mendes. Evolution of networks. *Advances in Physics*, 51:1079–1187, 2002.

[38] Manzi E., M. Labbe, G. Latouche, , and F. Maffioli. Fishman's sampling plan for computing network reliability. In *IEEE Transactions on Reliability*, pages 41–46. 50(1), 2001.

[39] P. Erdös and A. Rényi. *On Random Graphs. I.* Publicationes Mathematicae, 1959.

[40] W. Feller. *An Introduction to Probability Theory and its Applications - Vol. I and II.* Wiley, New York, 1968.

[41] Ciardo G., Luttgen G., and Miner A. S. Exploiting interleaving semantics in symbolic state-space generation. *Formal Methods in System Design*, 2007.

[42] Deqiang Gan, Xiaochuan Luo, Donald V. Bourcier, and Robert J. Thomas. Min-max transfer capabilities of transmission interfaces. *International Journal of Electrical Power & Energy Systems*, 25(5):347 – 353, 2003.

[43] N.K. Goyal. Network reliability evaluation: a new modeling approach. In *Int Conference on Reliability and Safety Engineering*, pages 473–488, 2005.

[44] J. Graver and M. Sobel. You may rely on the reliability polynomial for much more than you might think. In *Communications in Statistics - Theory and Methods*, pages 1411–1422. 34(6), 2005.

[45] M.T Hajiaghayi and Tom Leighton. On the max-flow min-cut ratio for directed multicommodity flows. *Theoretical Computer Science*, 352(1-3):318 – 321, 2006.

[46] G. Hardy, C. Lucet, and N. Limnios. Computing all-terminal reliability of stochastic networks by binary decision diagrams. In *Proceedings Applied Stochastic Modeling and Data Analysis*, 2005.

[47] G. Hardy, C. Lucet, and N. Limnios. K-terminal network reliability measures with binary decision diagrams. *IEEE Transactions on Reliability*, 56:506–515, 2007.

[48] K. Heidtmann. Smaller sums of disjoint products by subproducts inversion. In *IEEE Transactions on Reliability*, pages 305–311. 38(4), 1989.

[49] K. Heidtmann. Statistical comparison of two sum-of-disjoint product algorithms for reliability and safety evaluation. In *Proceedings 21st Intern. Conference SAFECOMP 2002*, pages 70–81. Springer Verlag, LNCS Vol 2434, 2002.

[50] J.C. Hudson and K.C. Kapur. Reliability analysis of multistate systems with multistate components. *IIE Transactions*, 15:127–135, 1983.

[51] Mitrani I., Reynolds B., Durham M., Perry E., Miles S., Poll M., and Sarafian R. *Simulation techniques for discrete event systems*. Cambridge University Press, Cambridge ; New York :, 1982.

[52] Chin-Chia Jane and John Yuan. A sum of disjoint products algorithm for reliability evaluation flow of flow networks. *European Journal of Operational Research*, 127(3):664–675, June 2001.

[53] T. Kam, T. Villa, and R. Braytonand A. Sangiovanni-Vincentelli. Multi-valued decision diagrams : Theory and applications. *Multiple-Valued Logic*, 4(1):9–62, 1998.

[54] A. Kaufmann, D. Grouchko, and R. Cruon. *Mathematical Models for the Study of the Reliability of Systems*. Academic Press, 1977.

[55] Krzysztof Kolowrocki. On limit reliability functions of large multi-state systems with ageing components. *Appl. Math. Comput.*, 121(2-3):313–361, 2001.

[56] S.Y. Kuo, S.K. Lu, and F.M. Yeh. Determining terminal pair reliability based on Edge Expansion Diagrams using OBDD. *IEEE Transactions on Reliability*, 48:234–246, 1999.

[57] C. Lucet and J. Manouvrier. Exact methods to compute network reliability. In *MMR'97, 1st International Conference on Mathematical Methods in Reliability*. Bucharest, Romania, 1997.

[58] T. Luo and K.S. Trivedi. An improved algorithm for coherent-system reliability. *IEEE Transaction on Reliability*, 47:73–78, 1998.

[59] E. Moskowitz. The analysis of redundancy networks. *AIEE Trans. Commun. Electron*, 39:627–632, 1958.

[60] Hiroshi Nagamochi and Toshihide Ibaraki. On max-flow min-cut and integral flow properties for multicommodity flows in directed networks. *Information Processing Letters*, 31(6):279 – 285, 1989.

[61] M. E. J. Newman. A measure of betweenness centrality based on random walks, 2003.

[62] M.E. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[63] M.E. Newman. Power laws, Pareto distributions and Zipfs laws. *Contemporary Physics*, 46:323–351, 2005.

[64] M.E.J. Newman. Ego-centered networks and the ripple effect. *Social Networks*, 25(1):83–95, 2003.

[65] Hiroyuki Ohsaki, Koutaro Yagi, and Makoto Imase. On the effect of scale-free structure of network topology on end-to-end performance.

In *SAINT '07: Proceedings of the 2007 International Symposium on Applications and the Internet*, page 12. IEEE Computer Society, 2007.

[66] L.B. Page and J.E. Perry. A practical implementation of the factoring theorem for network reliability. *IEEE Transaction on Reliability*, R-37:259–267, 1988.

[67] R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. *Physical Review E*, 65:036104, 2002.

[68] McGraw P.N. and Menzinger M. Clustering and the synchronization of oscillator networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 72(1):1–4, 2005.

[69] S. Rai, M. Veeraraghavan, and K. S. Trivedi. A survey of efficient reliability computation using disjoint products approach. *Networks Journal*, 25(3):147–163, 1995.

[70] A. Rauzy. New algorithms for fault tree analysis. *Reliability Engineering and System Safety*, 40:203–211, 1993.

[71] A. Rauzy. A brief introduction to binary decision diagrams. *Journal Européen des Systèmes Automatisés (RAIRO-APII-JESA)*, 30(8):1033–1051, 1996.

[72] A. Rosenthal. Computing the reliability of complex networks. *SIAM Journal on Applied Mathematics*, 32:384–393, 1977.

[73] A. Satyanarayana. Unified formula for analysis of some network reliability problems. *IEEE Transactions on Reliability*, R-31(1):23–32, 1982.

[74] A. Satyanarayana and A. Prabhakar. New topological formula and rapid algorithm for reliability analysis of complex networks. *IEEE Transactions on Reliability*, 27:82–100, 1978.

[75] K. Sekine and H. Imai. A unified approach via bdd to the network reliability and path number. Technical Report TR-95-09, Department of Information Science, University of Tokyo, 1995.

[76] Sieteng Soh and Suresh Rai. An efficient cutset approach for evaluating communication-network reliability with heterogeneous link-capacities. *IEEE Transactions on Reliability*, 54(1):133–144, 2005.

[77] K. Trivedi. *Probability & Statistics with Reliability, Queueing & Computer Science applications*. Wiley, II Edition, 2001.

[78] Iowa State University Ames (IA) USA. Meddly decision diagram library. http://sourceforge.net/projects/meddly/.

[79] L. G. Valiant. The complexity of enumeration and reliability problems. In *SIAM Journal on Computing*, pages 410–421. 8(3), 1979.

[80] M. Veeraraghavan and K. Trivedi. An improved algorithm for the symbolic reliability analysis of networks. *IEEE Transactions on Reliability*, 40:347–358, 1991.

[81] D.J. Watts and S.H. Strogatz. Collective dynamics of small world networks. *Nature*, 393:493–592, 1998.

[82] Elena Zaitseva and Vitaly Levashenko. Investigation multi-state system reliability by structure function. In *DEPCOS-RELCOMEX '07: Proceedings of the 2nd International Conference on Dependability of*

Computer Systems, pages 81–90, Washington, DC, USA, 2007. IEEE Computer Society.

[83] X. Zang and H. Sunand K. Trivedi. A bdd-based algorithm for reliability graph analysis. Technical report, Department of Electrical Engineering, Duke University, 2000.